Jamie Brindle

Student ID: 06352322

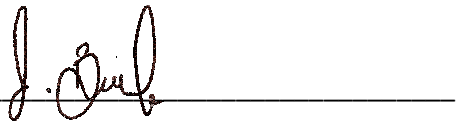
Course: Msc Advanced Computing

Project ID: NW.9

Title: Project Report:
Mobile Device Based Student Information System

Project Supervisor: Dr. Nick Whittaker

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed 

Abstract

This project will guide a reader through the development life cycle of what has been named a 'Student Information Kiosk'. Imagine a student walking into a university building and attached to the walls in specific places around the university is a relatively small, touch screen tablet computer that students can use to view their timetable, assignment results, information messages from lecturers or departments of the university and a campus map in case the student is lost.

From experience of being student I have at times forgotten which lecture room I'm, supposed to be in and the only way I can find out is to go to the library or computer lab and look on the department website on the internet for my time table, which is timely and frustrating. The proposed system eliminates this annoyance can be done relatively easily and cheaply using modern technology mobile technology.

My task is to design and implement the front and back end software for this Student Information Kiosk, which is to include quick and alternate means to access the kiosk, such as the use of RFID (radio frequency identification) tags to log into the system and utilise other modern technologies, in which I have utilised the popular Bluetooth technology, which can be used for a variety of application.

This report also aims at demonstrating a level of professionalism by illustrating good project planning, time management and writing skills, competent programming skills, graphical user interface design and the ability to thoroughly test and critically evaluate the created software system and project report to that of degree and industry standard.

Acknowledgement

I would like to give a special thanks to my project supervisor, Dr Nick Whittaker, who has aided in the developmental path of both my system implementation and report content and has also shown keen enthusiasm to myself and my work.

Attachments

Attached to the end of this report is:

- The 'User Guide – Distributing and Running the Student Information Kisok'.
- Attached to the inside of the back cover is a plastic wallet containing a CD which contains the implemented Student Information Kiosk System including source files, and required installation files and libraries, implementation documentation such as ESS models, UML diagrams and Javadocs and this report in .pdf format.

Key Directories and File Locations on the CD

This Report In PDF format:
- BrindleJ – 06352322 – NW.9 – Project Report.pdf

User GUI – Distributing and Running the Student Information Kiosk Interface in PDF Format:
- SIK – User Guide.pdf

Development Project Folders:
Implementation/Workspace/

ESS Models and Class UML Diagrams:
- Implementation/Doc/ESS Models & Class UML Diagrams

Javadoc:
- Implementation/Doc/JavaDoc

Completed Questionnaire Scanned Jpegs:
- Misc/Completed Questionnaire Scans

Any Diagrams Created For This Report:
- Misc/Publisher Diagrams

Sun Microsytems Inc Code Conventions Document 1999:
- Misc/Code Conventions.pdf

Installation Files for the Phidgets (RFID Scanner API) C++ Files:
- Implementation\Phidgets \Phidget-x86_2.1.7.20100803 (if using 32-bit Windows)
- Implementation\Phidgets \Phidget-x64_2.1.7.20100803 (if using 64-bit Windows
- Phidgets\Phidgets-LinuxSource\configure (if using Linux)

Executable Files for the Student Information Kiosk RMI Server:
- Implementation\Workspace\SIK-Server\SIK-Server.jar
- Implementation\Workspace\SIK-Server\SIK-Server.exe
- Implementation\Workspace\SIK-Server\SIK-Server.bat (will also show a consol window)

Executable Files for the Student Information Kiosk Student Records Administration GUI:
- Implementation\Workspace\SIK-Client-Admin\SIK-Client-Admin.jar
- Implementation\Workspace\SIK-Client-Admin \SIK-Client-Admin.exe
- Implementation\Workspace\SIK-Client-Admin \SIK-Client-Admin.bat (will also show a consol window)

Executable Files for the Student Information Kiosk GUI Interface (which a student user would use):
- Implementation\Workspace\SIK-Client-User\SIK-Client-User.jar
- Implementation\Workspace\SIK-Client-User \SIK-Client-User.exe
- Implementation\Workspace\SIK-Client-User \SIK-Client-User.bat (will also show a consol window)

Contents

Project Plan

			June					July				August					September				
Task	**Start Date**	**End Date**	03	08	15	22	29	06	13	20	27	03	10	17	24	31	07	14	21	28	**Duration**
Introduction & ToR	03-06-10	08-06-10																			1 Week
Research	03-06-10	29-06-10																			4 Weeks
Design	29-06-10	13-07-10																			2 Weeks
Implementation (Draft)	13-07-10	10-08-10																			4 Weeks
Interim Report	03-08-10	10-08-10																			1 Week
Implementation (Finalise)	10-08-10	31-08-10																			3 Weeks
Software and Device Testing	31-08-10	07-09-10																			1 Week
User Testing	07-09-10	14-09-10																			1 Week
Implementation (Adjustment)	14-09-10	21-09-10																			1 Week
Evaluation	14-09-10	25-09-10																			1.5 Week
Project Review and Hand-In	25-09-09	**30-09-10**																			1 Week

Introduction

Computing power has come a long way in the past two decades. In the 1980's computers were expensive, slow and rare; you wouldn't find one in a typical family home and they offered little in terms of functionality compared to the virtually limitless uses of today. Nowadays it would be difficult to go one day without the use of this exponentially growing technology, from the use of a mobile phone to watching cable television. Also the ability for devices to communicate and interface with one another has increased considerably.

A possible demand for the use of particular computer technologies might be in the form of a 'information kiosk', which would be in the form of a touch screen display panel situated at multiple locations of a company building or a university that provide a quick means of retrieving certain required or useful information. Rather than having to find and log onto a computer to retrieve a lecture time table or what time a meeting is held, a person could simply walk up to the nearest situated kiosk and quickly retrieve the information from there.

The purpose of this project is to design and implement the front and back-end of this 'information kiosk', and to actually create a physical prototype, in which the interface can be accessed via a number of methods such as a bar code and bar code reader, swipe card reader, RFID (radio frequency identification), mobile Bluetooth pairing or even fingerprint recognition, which can be used to authorise the user. The authorisation/access methods are yet to be explored however the interface must have alternative means of access as opposed to just the one.

The target audience for the information kiosk that will be designed and implemented will be aimed at students i.e. as if the kiosk is situated within a university building.

Project Specification

The structure of this project and project report will involve:

Research

It is necessary to do an adequate amount of research before designing and implementing such an information system. It will be necessary to investigate which technologies (hardware and software) are currently available to implement such a system. In respect to the hardware, such attributes such as cost, maintenance requirements, reliability and the ease of integration will need to be explored. In respect to the software, such attributes like ease of programming, level of functionality offered, integrity and again reliability will need to be explored.

Examples of hardware include input/ID devices, such as bar code readers, swipe card readers RFID readers etc and examples of software include the programming languages, databases, user interfaces etc.

It will also be necessary to research computer project creation techniques in order to draw up more comprehensive technical designs, such as the use proper state transition diagrams and to develop better implementation skills such as the use of design patterns and to also provide a generic structure in writing the report.

Design

The purpose of the information system is to provide someone with required or useful information as quickly and as easily as possible. Therefore the interface shouldn't be too complex or provide too much (or useless) information and speed of access will be of importance, hens the investigation of authorisation/access methods.

The system will focus more on function rather than how it looks, therefore time won't be wasted on fancy graphics and animation however the interface must be generally appealing to users so they want to use the kiosk, therefore it should be user friendly and graphically appealing.

The interface should utilise modern, popular technologies, such as Bluetooth and network connectivity, for the use of particular technologies such as database connectivity. As well as designing and implementing the user interface for the use of student of this system, it should also provide a user interface for lecturers or technicians so they can easily add, edit and remove student information to / from the system. The system should also be easily integrated into existing hardware / software systems. I.e. The system should be able to run on various computers and operating systems.

The complete type of information that a student will have access to and a complete functionality specification will be presented in the design state of this report after enough research gas been gathered.

Implementation

The most obvious choices for programming and implementing the system will include higher level object oriented programming language such as Java, C++ or C sharp as they provide a means to create a user interface required for users to use the information kiosk; they provide a great range of functionality and a means to interface and communicate with 3rd party software such as a databases and to devices such as bar code readers, swipe card readers and other input devices. Whichever programming language is used, it's important to program the system to conform to practice standards and design patterns in order to generate high quality, object based, and easily maintainable code to that of degree and industry standard.

The database to be used needs to be decided upon, which could be a SQL database or simply a text or data file, or even a file containing java objects. The network communication means will also need to be decided upon. Obvious choices include the use of Sockets (platform independent), RMI (remote method invocation – Java) or the use of Java servlets, which will be an ideal method if the use of an internet browser is to be used as the user interface, which is a likely option and has become very popular over the past few years as they're easy to implement and provide reliable connectivity, as most of the time communication takes place over the HTTP protocol, which is usually 'fire-wall' friendly and there are a number of packages and tools readily available to aid web-service development, such as Visual Studio, Eclipse, Netbeans, all which can incorporate web-servers and the back-end code and the markup / front-end all in one development suite.

The hardware for the interface will need to be decided upon. The system will be implemented to be used on a touch screen 'panel' that will sit on a wall in a university building, however access to this type of hardware may be limited due to budgeting costs. The university may offer a limited number of devices such as the Samsung Q1 Tablet PC or Samsung DM 200, which are small, touch screen, Microsoft Windows based tablet computers that should be enough to implement the system. It is not mandatory however

that this hardware must be used. Implementing the system on a standard PC may suffice, which would act as a 'virtual simulation' for the prototype information system.

Testing

As well as using the obvious 'test as you go' technique, a thorough testing plan will be created to test for any hardware or software problems. Once complete the kiosk will be user tested to see how they liked/disliked the system as a whole and particular parts and where the system could be improved, in which will be in the form of a questionnaire. This will serve as part of the evaluation of the system and will also provide a means to go back and alter parts of the system to appeal better to the user if time allows.

Evaluation

The testing will provide for part of the evaluation including problems that are found with the software or hardware and from the user testing. Also any problems found while implementing the system will be discussed and if necessary why and how the implementation deviated from the proposed design.

The evaluation will also include my personal opinion of the final system including where the system could be improved if I were to do the project again and also my personal opinion of completing the project as a whole and again what I would have done differently if I were to do the project again. This section will also give me the opportunity to show how the system could be expanded to offer more functionality and include a number of scenarios for the uses of the system.

Research

Available Authorisation/Access Technologies

User ID + Password

The system could be implemented so the user enters an ID number and password to access the system.

Magnetic Stripe Card and Reader

There are a vast number of stripe card readers available on the market. Involved is a plastic card that has a magnetic stripe on one side. The stripe is capable of storing data by modifying the magnetism of tiny iron-based magnetic particles on a band of magnetic material on the card. The stripe is read by physical contact and manually swiping past a reading head.

The stripe can typically hold either 7-bit alphanumeric characters or 5-bit numeric characters; however it wouldn't be necessary to program the card, only to have cards with a number on it already which can be matched to a user, only the card and reader will be necessary.

(*WebPage:* Magnetic Stripe Card – wikipedia.org)

Magnetic Stripe Coercivity

Magstripes come in two main varieties: high-coercivity (HiCo) at 4000 Oe and low-coercivity (LoCo) at 300 Oe. Higher-coercivity magstripes are harder to erase and are generally black or nearly black in colour and it would be hard to corrupt the magnetism using any magnets owned in the home. The low coercivity magstripes (generally brown in colour) are much easier to erase and could be corrupted using a low power magnet such as a hand-bag clip, which is why most of today's stripe cards will use high coercivity magstripes. (*WebPage:* Magnetic Stripe Card – wikipedia.org)

There are a vast number of stripe card and reader manufacturers. Typical low cost card readers start at around £50. A Typical specification follows:

MiniMag Duo Specification: (*Webpage:* MiniMag Duo – kestronics.co.uk)
A unique dual-head design reduces the confusion at the point of swipe by accepting cards being swiped regardless of the magnetic stripe orientation. Two magnetic heads provides the most convenience to the user and ensures the data is captured on every swipe. The MiniMag Duo delivers exceptional performance in a compact 90mm long solution.

Available in USB Keyboard and USB HID interfaces, the MiniMag Duo offers one of the smallest form factors for the industry. It reads 3 tracks of magnetic stripe card data regardless of swiping direction and over a large speed range.
The MiniMag Duo is a strong fit for kiosk applications as well as POS, loyalty and hospitality applications.

The MiniMag Duo provides embedded metal threaded inserts in its base to allow the reader to be mounted.

The MiniMag Duo is programmable so that the data format and intelligent interface output can be programmed & configured to match application and communication requirements. Data output format can be customized with the ID TECH provided windows based MagSwipe Configuration Utility. The MiniMag Duo design ensures over 1,000,000 card cycles and is one of the industry's most reliable card readers.

Features:

- Dual-head design enhances usability and adds flexibility
- USB Keyboard / USB HID interface
- Reads up to 3 tracks of information, bi-directionally
- Compact size allows use in many applications
- User-friendly configuration software utility
- Signifies successful reads with a beeper and LED indicator
- Reliable for a minimum of 1 million cycles
- Programmable for easy integration with existing environmentso Specifications

Electrical	
Power Requirement	+5 VDC from USB port, 40mA Max.
Interfaces	USB/KB, USB/HID.
Certifications	CE & FCC Class A
Environmental	
Operating Temperature	32°F to 131°F (0°C to 55°C)
Storage Temperature	-22°F to 158°F (-30°C to 70°C) non-condensing
Humidity	Maximum 95% non-condensing
Reliability	
Magnetic Head Life	1,000,000 passes minimum.
Mechanical	
Media Thickness	0.015 inches (0.127mm)to 0.038 inches (1.14mm)
Media Formats	ISO 7811, AAMVA, and other F2F formats
Slot Width	0.040 inches (1.37mm)
Indicators	Tri-colored LED and beeper
Swipe speed	3 to 60 inches per second, bi-directional
Dimensions	Length: 3.54 inches (90mm) Width: 1.34 inches (34mm) Height: 1.10 inches (28mm)

RFID Reader

RFID stands for Radio Frequency Identification. RFID is a member of the automatic identification and data capture (AIDC) family of technologies and is a fast and reliable means of identifying objects.

Two Main Components Include:

- The interrogator (RFID reader) which transmits and receives the radio signals
- The transponder, in which is a 'tag' or fob that is attached to an object. The tag is composed of a miniscule microchip and antenna. RFID tags can be passive or active and come in a wide variety of shapes and sizes.

Communication between the tag and the reader occurs wirelessly and therefore does need a line of site between the devices. The RFID reader emits a low-power radio wave field which is used to power up the tag so as to pass on any information that is contained on the chip.

Passive tags are generally smaller, lighter and inexpensive compared to those that are active, which can transmit a lot further. Active tags however require their own power source

(*Webpage:* What is RFID – rfidreader.com)

The range of applications of these access devices are becoming more popular however slightly more expensive, prices start around £100 for low priced basic RFID reader's and kits. A typical specification follows:

pcProx USB RFID Reader (*Webpage:* pcProx Proximity Card Reader – USB – gridconnect.com)

The pcProx family of RFID Proximity Card Readers allow you to use your access proximity cards for access to PC based applications or other assets such as PLCs and industrial equipment.

Using the pcProx reader, end users can read the data stored on their access cards, and use that identification information for any PC application. For example, employees can identify themselves to programs such as Excel, Access or any in-house created application.

pcProx is compatible with Windows CE, 98SE, 2000, XP, Vista, Macintosh, Linux. No client side software is required.

- Applications PC/LAN access control
- Application log-on
- Employee identification (multi-function printers MFP, embedded devices, PCs etc)
- Time and attendance
- Form filler to existing software applications
- PLC and embedded controllers
- Hoteling, meeting attendance, vistor management (using wall-switch housing)

The standard housing can be placed anywhere on the desktop. Featuring an articulated cable it can easily be mounted on kiosks, monitors, time clocks, and more. Optional base and mounting brackets expand placement options. Other form factors allow for easy, unobtrusive placement.

- Typical Maximum read range: 1.0” – 3.0” (2.5 – 7.6cm) dependent upon proximity card type and environmental conditions
- Dimensions: 3 3/8” x 2” x 0.6” (Models with RDR in part number only) 4.2” x 2.5” x 0.875” (10.6 x 6.35 x 2.2 cm)
- Weight: 0.45 lbs (12.7g)
- Power supply and interface: USB self-powered; RS-232 model; 5V supplied by PS2. Keyboard pass-thru connector
- Indicators: Tri-state LED, beeper
- Transmit frequency: 125 kHz
- Operating temperature range: -22° to 150°F (-30° to 65°C)
- Operating humidity range: 5% to 95% relative humidity, non-condensing
- Storage temperature range: -40° to 185°F (-40° to 85°C)
- Interface: RS-232 DB9 Connector or USB or Ethernet
- Certifications: FCC, United States; CE Mark Europe, C-tic Australia, RoHS

Fingerprint Recognition/Authentication

Fingerprint authentication refers to the automated method of verifying a match between two human fingerprints and is a very secure method of verifying ones identify, since everyone’s fingerprints are unique.

Fingerprint recognition has come so far in the past decade that they are very reliable and very cheap. You can get a typical Microsoft Fingerprint reader from as little as £25.

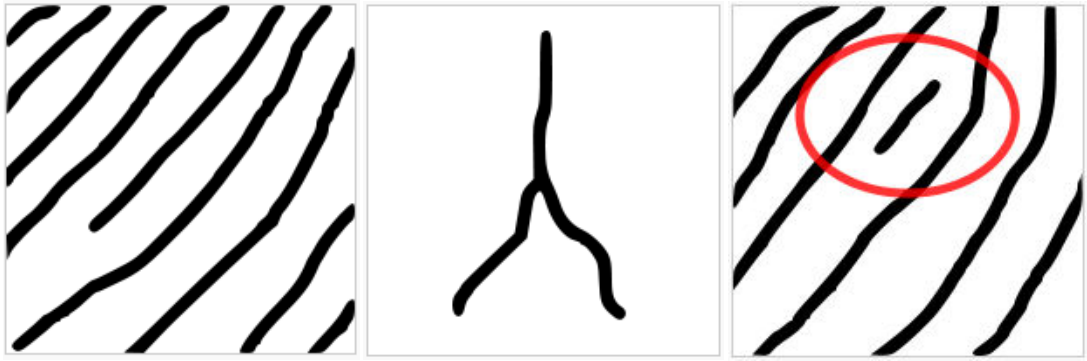
There are four sensor designs for fingerprint recognition, which include optical, ultrasonic, passive capacitance and active capacitance and the two typical algorithms used to recognise the fingerprint include ‘minutia’ and ‘pattern’.

Patterns

Involves working an algorithm on the type of pattern of the fingerprint

The arch pattern | The loop pattern | The whorl pattern

Minutia Features



Ridge Ending | Bifurcation | Short Ridge (dot)

(*Webpage:* Fingerprint Recognition – Wikipedia.org)

Sensors from leading manufactures such as AuthenTec don't actually read the fingerprint image, so the software does not store any image in memory. Instead they use capacitance sensors, which scan the minute radio frequency pattern beneath the live skin of the finger pad. Storing an image weakens the security of biometric access. The pattern creates a unique algorithm to identify the user. RD strip sensors are also low prices and can be very small.

A possible disadvantage to finger print recognition technology is the limited available development software. With stripe card readers and RFID readers it's easy to get a binary or numeric input to a computer, with fingerprint recognition software this is not that case and you need to use specialist SDK's which aren't generally free but are available with some fingerprint reader kits, for example the Verifi organisation, who provide developmental SDK's with their fingerprint recognition software.

(*Webpage:* P4000 Fingerprint Device –zvetcobiometrics.com)

Barcode Readers

A barcode reader is a device that scans or reads a printed barcode. It acts as a flatbed scanner, using a light source and sensor to translate optical impulses to electrical ones. Most barcode reader also contain a decoder that analyses the barcode image and translates it into numbers or letters and sends it to a computer or device as an output of numbers or letters, not an image.

There are a number of different styles of barcode readers, the most common being hand held or fix mounted. For this project the likely option would be the fix mounted barcode reader if a barcode reader is to be used and the user would simply place a plastic card with a bar code on it near the reader.

Bar codes can be of a variety of sizes and could be printed on a small piece of plastic that could be attached to a keychain.

Types of Reading

There are a number of types of reading a bar code, such as the use or lasers, CCD readers and camera based readers. CCD readers use an array of hundreds of tiny light sensors lined up in a row. Each sensor measures the intensity of the light in front of it, which generates a voltage pattern which can be sampled and converted into a digital representation of numbers of letters. Camera-based readers work in a similar way, as a camera has many thousands of tiny light sensors arranged in 2D array. They're becoming more popular these days due to their reliability however this is due to the higher level of software and algorithms used to decode the bar code rather than the technology. Laser scanners are still the most widely used and can consist of a single laser beam, such as in a hand held barcode reader. The beam is shined across the bar code and a photodiode is used to measure the intensity of the light reflecting back from the bar code. There are also omni-directional barcode scanners, which are often used in fix-mounted barcode scanners. They use a series of straight or curved scanning laser lines of varying directions so that the barcode can be read from different orientations. You'd typically find these type of scanners in supermarket checkouts.

(*webpage:* Barcode Reader – wikipedia.org)

Barcodes and readers are widely used in a variety of application in everyday business. Some of which include:

- Keeping track of equipment within a company or warehouse
- In retail, to identify the price of items and to keep track of stock, and generate sales data
- In libraries to keep track of outgoing and incoming books
- By postal and delivery companies to identify where a parcel is supposed to go without having to read an address, and also for tracking services. And to mark if a parcel has been successfully delivered.

(*Webpage:* Use of Barcode Scanners – nationalbarcode.com)

The typical disadvantages of the use of barcodes are that barcode readers can't read labels that are wrinkled, dirty or smudged and also the cost of the equipment. They're very expensive and the costs grow exponentially with the size of the business. A low budget barcode scanner for the means of this project starts from around £250.

Bluetooth Technology

Bluetooth is a short range (around 10 to 30 metres) open wireless communications technology that uses short length radio waves, intended to replace cable connecting portable or fixed devices, with key features such as robustness, low power and low cost. Data transfer rates start at around 1 Mb/second for low energy devices, such as mobile devices and can go up to about 24 Mb/second for more specialised devices. A master Bluetooth device can communicate with up to eight devices in a wireless group. *(Webpage:* Bluetooth Basics – Bluetooth.com)

Typical Applications of Bluetooth

- Wireless control of and communication between a mobile phone and a hands-free headset.
- Wireless networking between PCs in a confined space and where little bandwidth is required.
- Wireless communication with PC input and output devices, the most common being the mouse, keyboard and printer.
- Transfer of files, calendar appointments, and reminders between devices with OBEX.
- Replacement of traditional wired serial communications in test equipment, GPS receivers, medical equipment, bar code scanners, and traffic control devices.
- For controls where infrared was traditionally used.
- For low bandwidth applications where higher USB bandwidth is not required and cable-free connection desired.
- Sending small advertisements from Bluetooth-enabled advertising hoardings to other, discoverable, Bluetooth devices.
- Wireless bridge between two Industrial Ethernet (e.g., PROFINET) networks.
- Three seventh-generation game consoles, Nintendo's Wii and Sony's PlayStation 3 and PSP Go, use Bluetooth for their respective wireless controllers.
- Dial-up internet access on personal computers or PDAs using a data-capable mobile phone as a wireless modem like Novatel mifi.

(*Webpage:* Bluetooth – Wikipedia.org)

Relevance

Bluetooth technology has become very cheap and easy to get a hold of in a variety of forms, such as a Bluetooth dongle which can be connected via USB to a computer. Nearly all modern mobile devices such as mobile phones come with Bluetooth technology as standard and Bluetooth technology offers a wide range of functionality. Also many higher level programming languages such as Java offer APIs to utilise Bluetooth technology. Therefore it would be an ideal technology to utilise in this project. For example, as well as being able to walk up to an 'information kiosk' in a university for a student to access their lecture timetable, they could also download their timetable to their mobile phone. Once a student's mobile phone is paired with the kiosk, the kiosk could then broadcast 'advertisements' or messages to that phone once it's in range of the kiosk. Messages could include remainders of their next lecture or inform them of assignment results or even just general news from the university. It may also be possible to use a Bluetooth enabled mobile device as a form of access to the information kiosk, rather than using a stripe card or RFID as previously explored.

Pairing and Security

There are many service offered over Bluetooth that can expose private data or allow a connecting party to control the Bluetooth enabled device. Therefore security precautions have been developed, namely 'pairing'. It involves manually allowing a device to connect to it. The pairing process is triggered automatically the first time a device receives a connection request. Once pairing has been established it is remembered by the devices, so that it can allow connection from then on without user intervention of desired, which can be sometime useful for Bluetooth devices to automatically establish a connection when a device is in range.

During the pairing process, the devices involved establish a 'bond' by creating a secret shared key. If the key is stored by both devices they are said to be 'bonded'. A device that wishes to communicate only with a bonded device can cryptographically authenticate the identity of the other device, and so can be sure that the device was previously paired with it. Once a key has been generated, an authenticated ACL link between the devices may be encrypted so that the data they exchange wirelessly is protected from eavesdroppers. These shared keys can be deleted at any time by either device.

(*Webpage:* Bluetooth – Wikipedia.org)

Bluetooth Architecture

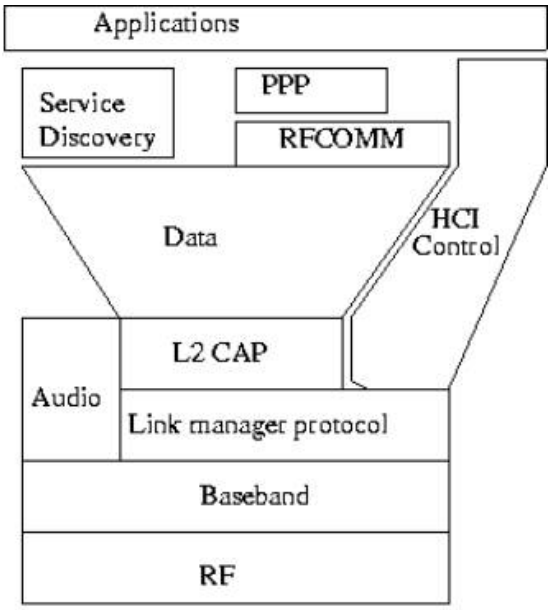


Figure 1: Bluetooth architecture

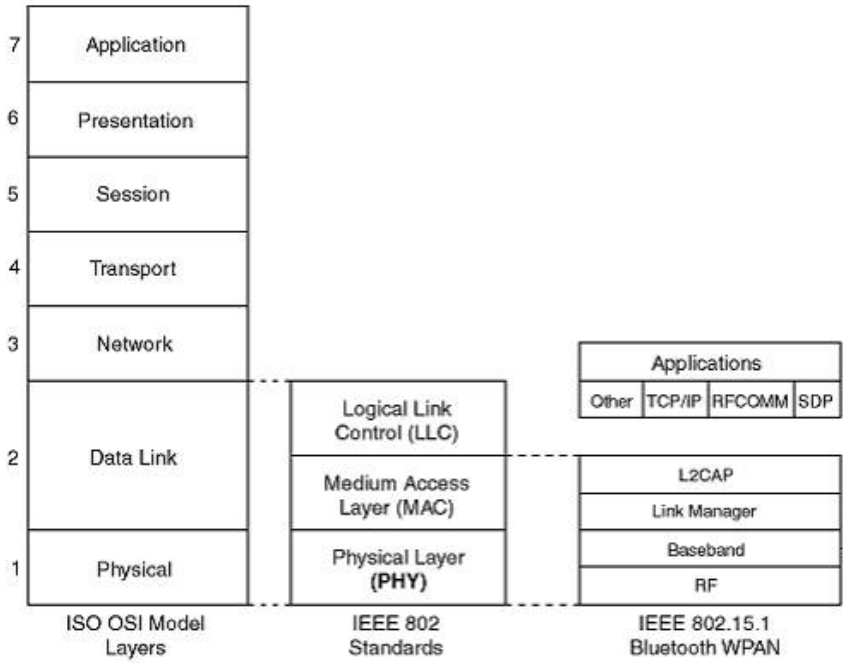


Figure 2: Mapping between Bluetooth, OSI model and IEEE802

(Figure 1 and 2 from Silian Liu, 2009, *Webpage:* Bluetooth Technology)

Bluetooth Topology

There can only be between 2 and 8 devices connected and communicating with one-another (this is called a piconet). In which there is only one master and the rest of slaves. A device however can belong to two piconets, serving as slaves in both piconets or a master in one piconet and a slave in another. This is called a bridging device. Bridging devices connect piconets together to form a scatternet.

(Lui, 2009, *webpage:* Bluetooth Technology)

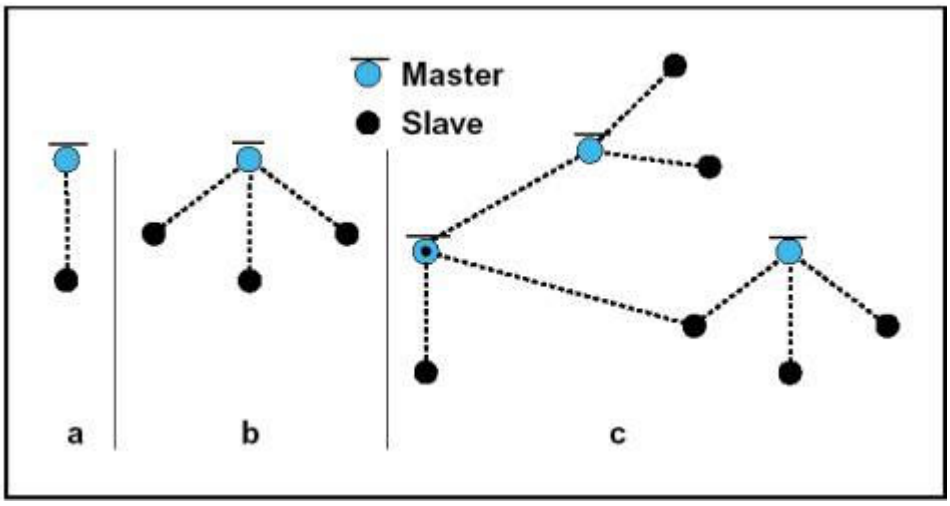


Figure 3: Single-slave piconet (a), multiple-slave piconet (b) and scatternet (c)

(Figure 3 from Silian Liu, 2009, *Webpage:* Bluetooth Technology)

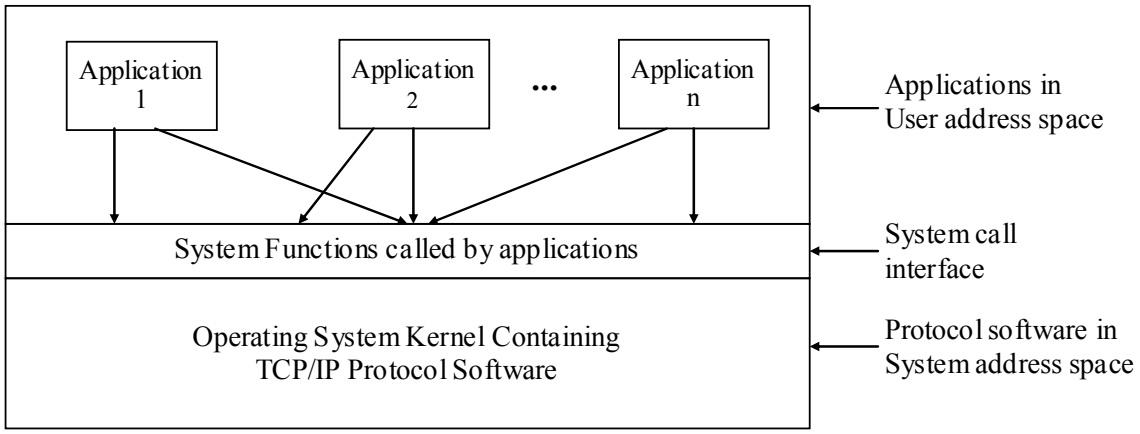
Network Communication

There are a number of means to communicate over a network, such as the use of sockets, RMI (remote method invocation) and the use of servlets. These are the likely choices for this project. Deciding which one(s) to use will depend on their offered functionality, reliability, ease of programming and the method used to create the user interface. For example, servlets use the HTTP request/response to send data across a network and are generally used to present data on a web browser. This method would be ideal if the user interface were to take place within a web browser as if it was a dynamic web site. RMI could also be and can also be used if the interface was java applet based. RMI is exclusive only to Java, however it offers a great deal of functionality due to Java's extensive APIs and it allows the invocation of methods as if they were local thus allowing further functionality. RMI provides an extra layer on top of typical sockets, which is generally the technology used for most end-point to end-point communication. It is possible to just use sockets however. The use of sockets are platform independent and involves sending raw data across a network however it offers only limited functionality (unless I write an extensive number of APIs and develop my own web-service protocols), and it is a less modern approach to sending data across a network. All three means of network communication have their advantages and disadvantages.

Sockets

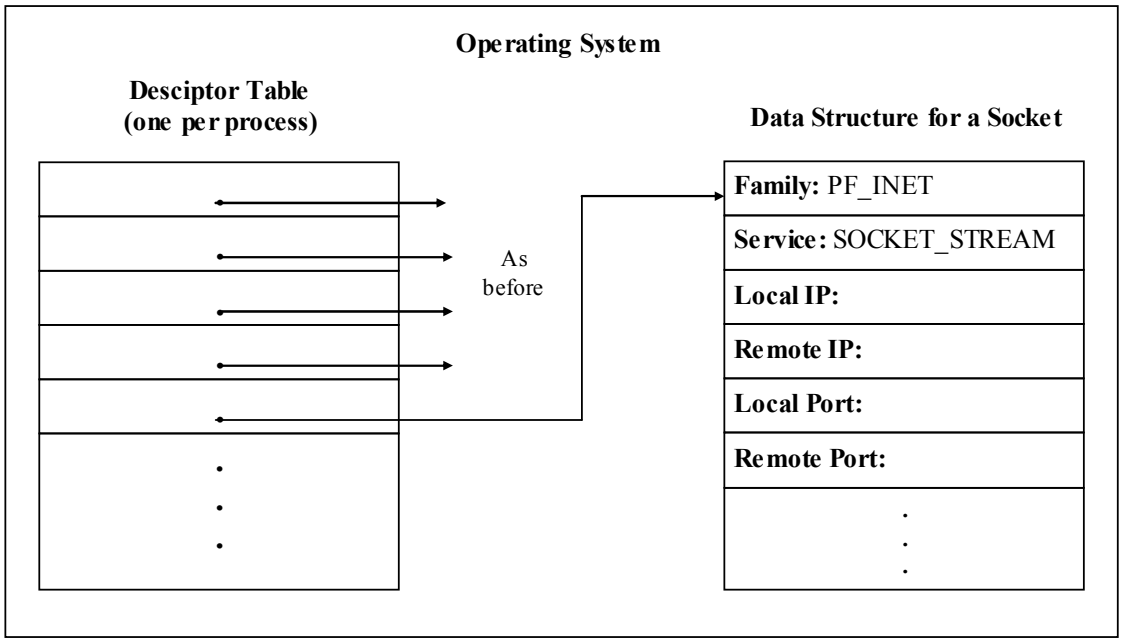
System Calls

System calls are tightly coupled with Sockets. The diagram below illustrates the system call mechanism that most operating systems use to transfer control between an application program and the operating system procedures that supply services. When an application 'invokes' a system call, control is then passed from the application to the system call interface. This will then transfer control to the operating system. It is the operating system's job to direct the incoming call to an internal procedure that will perform the requested operation. Once the internal procedure completes, control returns through the system call interface to the application, which then continues to execute. In a way, when the application needs to use a service, the process that is executing the application will climb into the operating system, acquire privileges that it needs to allow it to read or modify data in the operating system, then climbs back out (Comer, 1996, p. 37).



(From figure 4.1 of Comer, 1996, p.38, *Internetworking with TCP/IP Client Server Programming and Applications)*

When an application calls socket, the operating system then allocates a new data structure to hold the information needed for communication, and fills in a new descriptor table entry to contain a pointer to the data structure, an example is given below:



(From figure 5.2 of Comer, 1996, p.46, *Internetworking with TCP/IP Client Server Programming and Applications)*

Although the internal data structure for a socket contains many fields, the system leaves most of them unfilled when it creates the socket. The application that created the socket must make additional system calls to fill in the information in the socket data structure before the socket can be used.
Once a socket has been created, it can be used to wait for an incoming connection or to initiate a connection. A socket used by a server to wait for an incoming connection is called a 'passive socket', while a socket used by a client to initiate a connection is called an 'active socket' (Comer, 1996, p. 47).

Initially, when a socket is first created it doesn't contain much information about how it will be used. The socket doesn't contain information about the protocol port or any IP addresses (local or remote). Before the application uses a socket, it must specify one or both of these addresses (Comer, 1996, p. 47).

Major System Calls Used with Sockets

- **The Socket Call:**
  An application calls 'socket' to create a new socket that can be used for network communication. The call returns a descriptor for the newly created socket. For a socket that used the Internet protocol family, the protocol or type of service argument determines whether the socket will use TCP or UDP.
- **The Connect Call:**
  After creating a socket, a client calls 'connect' to establish an active connection to a remote server. An argument to 'connect' allows the client to specify the remote endpoint which includes

the remote machine's IP address and port number. Once the connection is made, a client can transfer data across it.

- **The Write Call:**
  Both clients and servers use 'write' to send data across a TCP connection. Clients usually use 'write' to send requests, while servers use it to send replies.
- **The Read Call:**
  Both clients and servers use 'read' to receive data from a TCP connection. Usually after a connection has been established, the server uses 'read' to receive a request that the client sends by calling 'write'. After sending the request, the client uses 'read' to receive a reply.
- **The Close Call:**
  Once a client or server finishes using a socket, it calls 'close' to de-allocate it. If only one process is using the socket, 'close' immediately terminates the connection and de-allocates the socket. If several process share a socket, 'close' decrements a reference count and de-allocates the socket when the reference count reaches zero.
- **The Bind Call:**
  When a socket is created, it does not have any notation of endpoint addresses (local or remote). An application calls 'bind' to specify the local endpoint address for a socket. The call takes arguments that specify a socket descriptor and an endpoint address.
- **The Listen Call:**
  When a socket is created, the socket is neither active (use by client) nor passive (use by server) until the application takes further action. Connection-oriented servers call 'listen' to place a socket in 'passive mode' and make it ready to accept incoming connections. Most servers consist of an infinite loop that accepts the next incoming connection, handles it, and then returns to accept the next connection.
- **The Accept Call:**
  For TCP sockets, after a server calls 'socket' to create a socket, 'bind' to specify a local endpoint address and 'listen' to place it in passive mode, the server calls 'accept' to extract the next incoming connection request. An argument to 'accept' specifies the socket from which a connection should be accepted. 'Accept' creates a new socket for each new connection request and returns the descriptor of the new socket to its caller. The server uses the new socket only for the new connection; it uses the original socket to accept additional request, once it has accepted a connection, the server can transfer data on the new socket.

(Comer, 1996, pp. 49-51)

Advantages

Sockets are flexible and sufficient. Efficient socket based programming can be easily implemented for general communications and cause low network traffic compared to HTML forms and CGI scripts that generate and transfer new web pages for each new request.

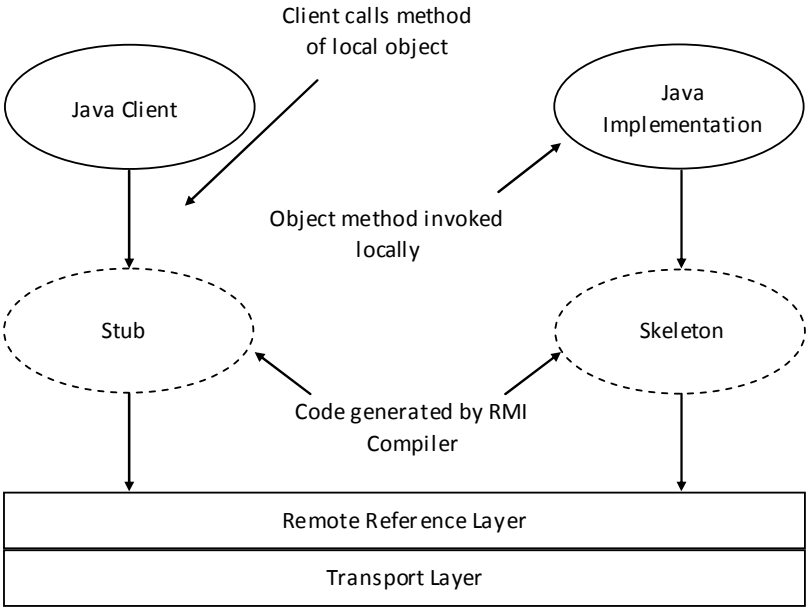
Disadvantages

- Sockets based communication allows only to sends packets of raw data between applications. Both the client-side and the server-side have to provide mechanisms to make the data useful.
- Since the data formats and protocols remain application specific, the re-use of socket based implementations are limited.

RMI

RMI (remote method invocation) allows one to invoke methods on objects that reside on another machine in another virtual machine and treats them as if they were local. At the server side, an RMI service is created. This service is an object with a *main* class that does nothing other than creating the remote object with *new* and *binding* it into an RMI registry with a unique name. The client then needs to know this name to ask the registry to a get a reference to the service. Once the client has the references it can make remote method calls with parameters and return values of if the objects reside on the local host. Objects are transmitted through serialisation (process of converting a data structure or object into a sequence of bits so it can be transmitted across a network). (Downing, 1998, pp. 12-14)

Architecture:



Stubs Layer

The stub is a proxy for representing the interface to remote objects for the client and defines a complete interface of the remote implementation. It communicates with the skeleton layer via the remote reference layer

Skeleton Layer

The skeleton is an interface between the remote implementation and the remote reference layer. It communicates with the sub via the remote reference layer and marshals any return values or exceptions.

The stubs and skeletons are generated by the RMIC (remote method invocation compiler).

Remote Reference Layer

The remote reference layer is responsible for the management of remote objects and is responsible for the invocation of remote object methods.

Transport Layer

The transport layer handles all lower-level network issues. It sets up a connection over a physical socket, it serialises objects as required and it monitors the connection for signs of faults such as unresponsiveness.

(McCacken, 2000, pp. 1-12)

Advantages

- Simple and clean to implement that leads to more robust, maintainable and flexible applications
- Dynamic loading of classes is available
- Can make changes to the server end, that might not mean you need to change anything on the client side
- Efficient handling of:
    - Sockets
    - Threads
    - The marshal of object

Disadvantages

- Less efficient than sockets objects
- Cannot use the code outside the scope of Java
- Security issues need to be monitored more closely as opposed to other web service methods
- Firewalls can sometimes get in the way, as not using HTTP protocols

(Satpute, 2008, *Webpage:* RMI – Remote Method Invocation)

Servlets

A servlet is merely modules of Java Code or classes that run in a server application and in which conforms to the Java Servlet API, a protocol in which responds (or may respond) to HTTP requests. A servlet can be used to add dynamic content to a web server using the java platform. The content is typically HTML but can use other data such as XML. The servlet API defines the expected interactions of a web container and a servlet. A web contain is the component of a web server that interacts with the servlets and the web container is then responsible for managing the life cycle of servlets, such as mapping an URL to a servlet and making sure the URL requester has the correct access rights.

One example of a web server that uses servlets is 'Apache Tomcat', an open source servlet container. Tomcat implements the Java servlet and the JSP (Java Server Pages) specification from Sun Microsystems and provides a Java HTTP web server environment for Java code to run on.

Typical uses for Servlets include:

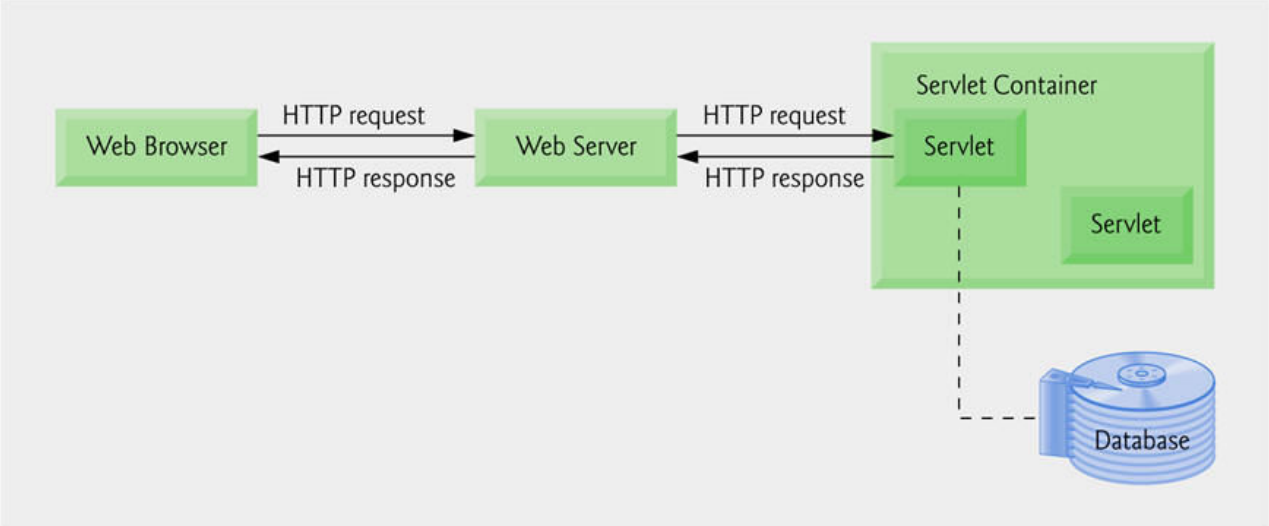
- Processing and/or storing data submitted by an HTML form.
- Providing dynamic content, e.g. returning the results of a database query to the client.
- Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

(Zeiger, 1999, *Webpage:* Servlet Essentials)

HTTP request-response

HTTP stands for HyperText Transfer Protocol). It is a protocol used by a web server via the use of generally a browser. HTTP is a request-response oriented protocol, an HTTP request typically consists of a request method, an URL (location), header fields and a body. A HTTP reponse is the result or outtome of the request.

Typical Servlet Architecture



(From Figure 26.1, *Webpage:* Servlet Overview and Architecture – faculty.inverhills.mnscu.edu)

Advantages

- One one instance of the JVM needs to be started, a new thread is created and managed by the same JVM, which is not the case for other web servers. Thus JVM only needs to keep track of one copy of the actual servlet code to create the dynmanicity.
- The user does not need to have a Java-capable browser , all they see is html, all the work is done in the background.
- Using Java for a web service allows a for an extensible range of APIs, thus offering a magnitude of functionality
- Use of HTTP protocol, which is typically allowed through most firewalls.

Disadvantages

- The need to have a special ‘servlet container’ to run servlets, which typical web servers (for public hosting) don’t include this
- Need to use the Java Runtime Environment on the server to run servlets.
- Sometimes there are no Java interfaces to particular resources or devices
- Java is quite slow, not as fast as native executables
- Since the servlet manager (in this case Tomcat) is a Java program and is separate from the web server, Apache, the communication between the two is not instantaneous. This may produce a performance hitch on heavily active web servers.

(Sorfa, 2002, p. 1)

How to Design the Interface

It is important to do adequate research about designing the user interface before beginning the design; this can be split into 2 separate categories. The first thing to look at is the basic psychology of designing a user interface. An audience is to be taken into consideration and attributes like usability, quality of service and balancing function and fashion need to be taken into consideration. The second part will briefly go into the 'GUI' aspect that the Java programming language offers.

Usability Requirements

Appreciation of an interface comes from features such as usability, universality and usefulness. These goals are achieved though thoughtful planning, sensitivity to the user's needs, devotion to requirements analysis and diligent testing. Designers generally produce multiple design alternatives for consideration. User-interface building tools enable rapid implementation and easy revision. Evaluation of designs refines the understanding of appropriateness for each choice. Successful designers go beyond the vague notion of 'user friendliness' probing deeper than simply making a checklist of subjective guidelines. They have a thorough understanding of the diverse community of users and the tasks that must be accomplished (Shneiderman, 2005, p. 12).

Setting explicit goals helps designers to achieve those goals. In getting beyond the vague quest for user-friendly systems, designers can focus on specific goals that include well-defined system engineering and measurable human-factor objectives. The U.S. Military Standard for Human Engineering Design Criteria (1999) states these purposes:

- Achieve required performance by operator, control and maintenance personnel
- Minimize skill and personnel requirements and training time
- Achieve required reliability of personnel-equipment/software combinations
- Foster design standardisation within and among systems.

These functional purposes are a good starting point, but effective interfaces might also enhance the quality of life for users or improve their communities (Shneiderman, 2005, p. 12).

Practical goals generally include:

1. Time to Learn
2. Speed of performance
3. Rate of errors by users
4. Retention over time (how well do users maintain their knowledge after a certain period of time?)
5. Subjective satisfaction (how much did the users like using various aspects of the interface?)

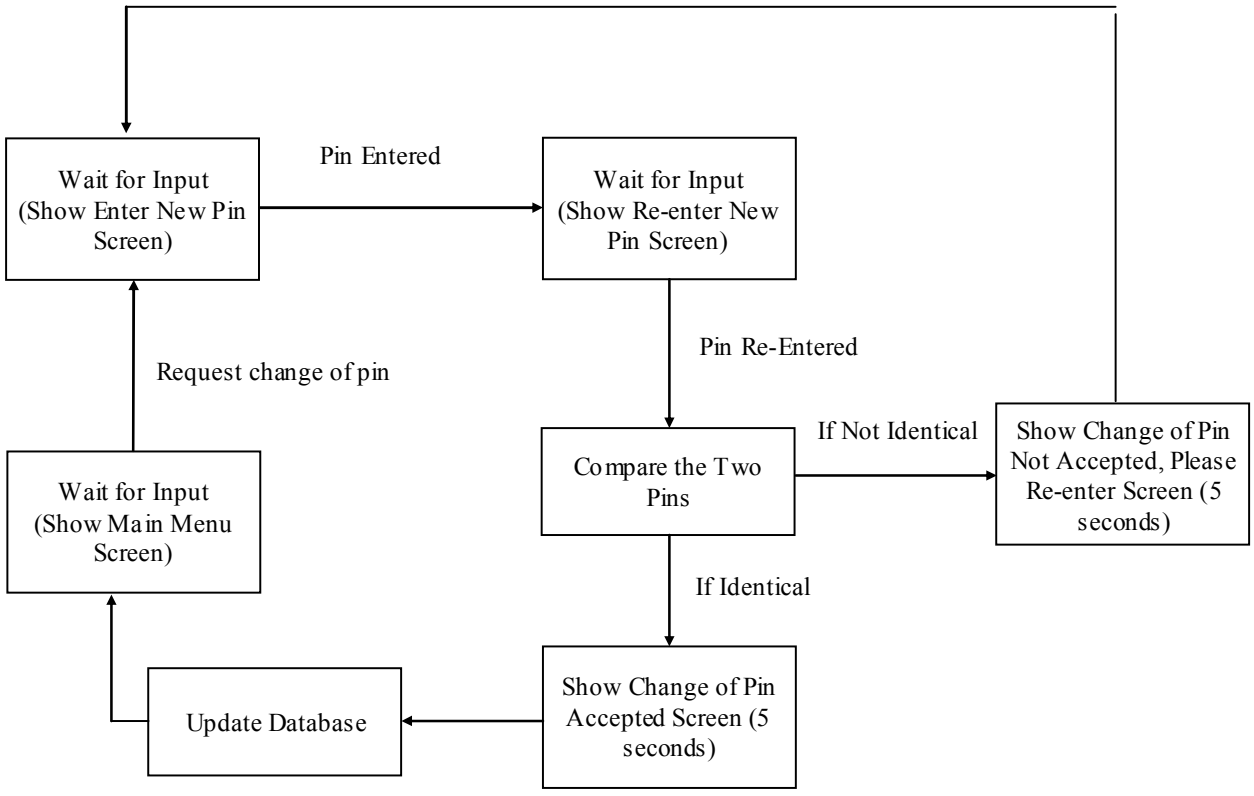
(Shneiderman, 2005, p. 16)

The Use of State Transition Diagrams

A 'Finite-State Machine' is one which has a 'finite' or 'set number' of internal states or state transitions. An ATM machine is a good example of this. Shannan, Huffmann, Kleene and Moore applies it in their fundamental works of computer theory and information theory. The intuitive definition of a state is: "The state contains all information characterising the history of the system and enabling it to be computed in the future. The history of the past can influence the future only within finite steps" (Tarnay, 1991, p. 110).

A key concept used in many protocol models is the 'finite state machine'. Each protocol machine is always in a specific state at every moment in time. Its state consists of all the values of its variables, including the program counter (Tanenbaum, 2002, p. 229). From each state, there are zero or more possible transitions to other states. Transition occurs when some even happens. For a protocol machine, a transition may occur when a frame is set (a sequence of data), when a frame arrives, when a timer expires or when an interrupt occurs.

An application will have a number of states, in which, in designing an application, a good starting point would be to identify these states; A UML example of the states of an ATM's (cash point) program is given on the next page. Of course typically an ATM may offer other services. These services include printed receipts, depositing cash, a mobile phone top up service, advertisements and change of pin, but to name a few. An example UML is given on the next page of the states the ATM's program would take in changing a pin. These sorts of diagrams will be created and finalised in the design section of this report which will ultimately aid the implementation process and demonstrate sophisticated design skills.



Research Analysis – Pre-Design

After completing a substantial amount of search surrounding the topic area of this project a number of decisions have been made which allow me to start the design and implementation phase of this project.

Programming Languages and Network Communication APIs to be used

Java is going to be the main programming language to be used. This particular language appears to have stood out in important research areas such as the use of RMI and servlets, both of which use the Java programming language. Java also offers an extensive amount of APIs offering a wide range of functionality and appears suited for this project involving the use of 3rd party input devices (bar code reader, stripe card readers), web and network services (RMI, servlets, sockets), database access APIs (mySQL) and the ability to create a sophisticated user interface (use of Swing, AWT) that can have direct interaction with all the previously mentioned APIs, rather than having to go through an intermediary. Java also offers the use of MIDlets, which would enable programs to be created that can run on a mobile device such as a mobile phone, using J2ME (Java 2 Micro Edition), which would be ideal if mobile devices are to be utilised in this project.

Java is also platform independent with the use of its Java Virtual Machine, allowing a larger scope of utilisation.

Another important reason for using Java is the current experience I have with the language. I've used Java for around 4 years now and know it better than any other program language. It is logical to keep using it as I will be familiar with the level of functionality it offers and how to utilise it. Also, it's free.

The chosen method of communication across the network is to be RMI. I have used sockets, RMI and servlets in the past and have found that RMI is the cleanest, the least hassle to use, requires far less coding and provides a network service that supports a larger range of functionality. Using servlets would have been the first choice due to the fact that it is browser based and can create dynamic web sites, in which allow a greater range of user interface styles and presentation, and it does not require the Java Runtime Environment to be installed (all that's required is a web browser). However data needs to be sent over HTTP request-response which can be a pain when having to write code to unwrap and make use of the data, and the quantity of code increases dramatically as the functionality of the interface grows. RMI, even though offers less in terms of user interface styles without a great deal of time being spent on AWT code, is sufficient for this project, and it does not require the use of web server running a temperamental servlet container such as Apache Tomcat.

Database to be used

Originally the use of a mySQL database were to be used, however it has been decided that student information is to be stored in a Java data file, which consist of Java objects. This is due to the fact that it does need require yet another server to be running (mySQL server), in which eliminates another potential fault (i.e. mySQL server being down) and it is much easier to amend a database structure using the proposed database implementation. For example, when implementing the student implementation system if I decide to change an attribute or a database field I only need to change the java implementation code, not the database as well, plus Java allows arrays to be stored within arrays and objects within

objects etc which allows for better data permutations and easier access rather than having a large amount of prepared statements and duplicate ID keys in every table as you would in a SQL database. However if I was to implement the system to be used in the real-world I would definitely use a SQL database, as it provides a fast, disk-space efficient, multi-access means of managing data, particularly when there are large amounts of information (i.e. Data for each student in an entire university), however in this case, as this a prototype, a data file will suffice.

Authorisation/Access Technologies to be Used

The technologies to be used to authorise a user access to the information kiosk will be the use of RFID readers and tag. These seem to be the most feasible technologies to use as they provide a quick means of access and they don't require a user to have to remember a password, although a password will be used as alternative means to access the interface, in the event that a user forgets or loses their card, or there card/tag simply doesn't work. The fingerprint recognition would provide a means to eliminate the problem of losing a or damaging a card, the technology is cheap enough to get a hold of and it reduces the cost of creating / buying stripe cards and RFID tag cards due to the limited available SDK's for the development involving this technology. The use of this technology will not be utilised right away in the implementation of this system, however it may be looked into in a later stage of the development cycle, it is not by any means being excluded just yet, although the decision may be made that the use of this technology is unfeasible, as it means each and every student must have their finger prints taken, which could cost time and money.

The use of a bar code scanner is definitely unfeasible. The technology is too expensive especially if this system was to be widely used around universities and the ease of damaging and dirtying a bar coded cards to the extent where the card can't be read means that other technologies would serve the system more effectively.

Bluetooth Utilisation

As Bluetooth has become so popular, with many mobile devices (practically all mobile phones) being Bluetooth enabled, and the fact that most people own a mobile phone and with the high level of functionality offered by Bluetooth technology, this will be incorporated into the student information system. A user will be able to pair with the information system and have the ability to download particular data as they desire. The user also may not even have to 'pair' their device. The system could simply use Bluetooth 'object push', which requires no pairing, instead a user simply states they wish to or don't wish to receive a file (the object).

Further Research

The bulk of the research has now been completed and has allowed enough decisions to be made to go ahead and start designing the student information system. However it will be necessary to do further research in the implementation process to learn how to implement some of the functionality that will be outlined in the design phase of this report.

Design

Overview

The project implementation will be split up into 3 parts, which include a server and two clients. Both clients will be able to communicate with the server and utilise its services. Both clients will include a user interface. One user interface will act as 'administration' interface for the student records, in which will provide an 'admin user' a means to input data of a particular student to form a 'student object' which will be utilised by the clients and the server. The other client will be the actual 'Student Information Kiosk' which is used by students, where they can retrieve the information from their unique student object.

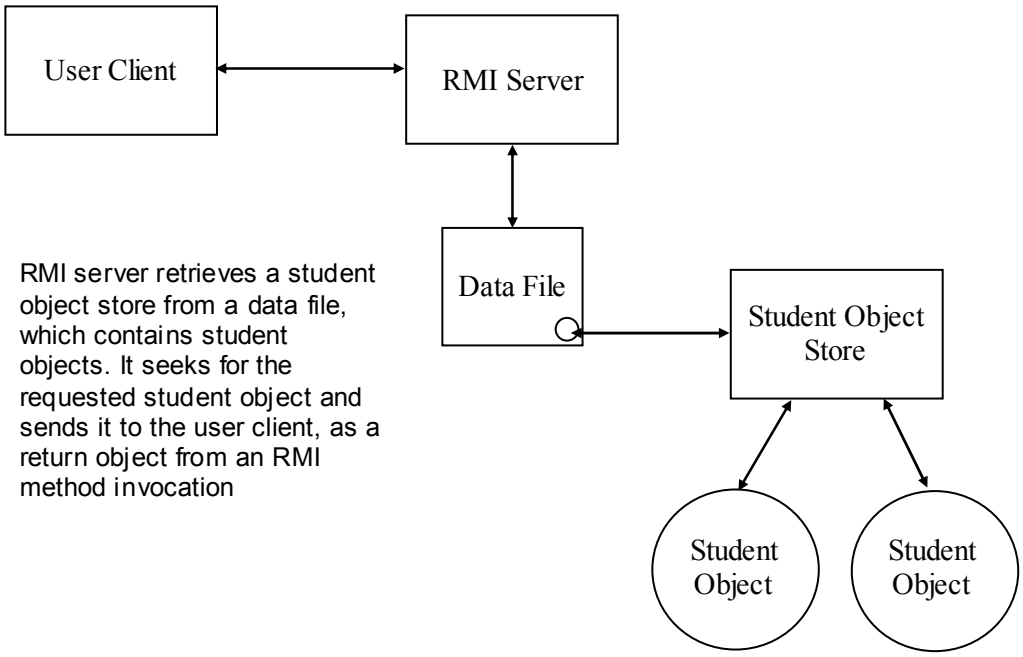
A major difference between the two different clients is that the Student Information Kiosk interface will rely on the server in order for it to work; otherwise it can't retrieve any data of a student, and thus would be useless. The student records administration interface however will have the option to be able to work with the server or act in standalone mode. The purpose of this interface is to create student record objects and store them in a file, in which case this should be able to be done locally, and then later be able to export a file to the server for the server to use. The administration interface should also allow retrieval of the student records store file from the server.

Network Communication - RMI

As previously mentioned, the proposed technology to enable client / server communication across a network will be via the use of RMI (remote method invocation). As this has been decided, the choice has to be made as to whether a *large* amount of data is to be sent from client to server or vice-versa *un-often* (less network traffic, but could result in delays when sending / receiving information due to the amount of data being sent) or a *small* amounts of data being sent *quite often* (more network traffic, but possible fewer delays due to only sending a small amount of data at a time). This of course depends on type of system that is to be implemented. There's no right and wrong way.

It has been decided that the less network traffic approach is to be used, for a number of reason:

1. The data won't be large in size anyway
2. Sending data fewer times frees up the server machine to do other things
3. Fewer methods will have to be implemented on the server side, the server would simply send a 'student object' to a client machine, which contains all the relevant information of a student, in which is using the client interface. It's then up to the client to decide what to do with the data.
4. If there are any network problems when a user is using the interface, the user can know before they do anything, rather than in between doing something, as once the 'student object' has been sent, the client is pretty much standalone.



Student Objects and Store Objects

A student object will be made up of the following properties:

Unique Identifier: Student ID (integer)
Pin Number (integer)
Course (String)
Title (String)
First Name (String)
Second Name (String)
First Line Address (String)
Second Line Address (String)
City (String)
County (String)
Telephone Number (String)
RFID Tag Serial Number (String)
Timetable (Array of type 'String')
Assignment Results (Array of type 'String')
Messages (Array of type 'String')
Photo (Image Icon)

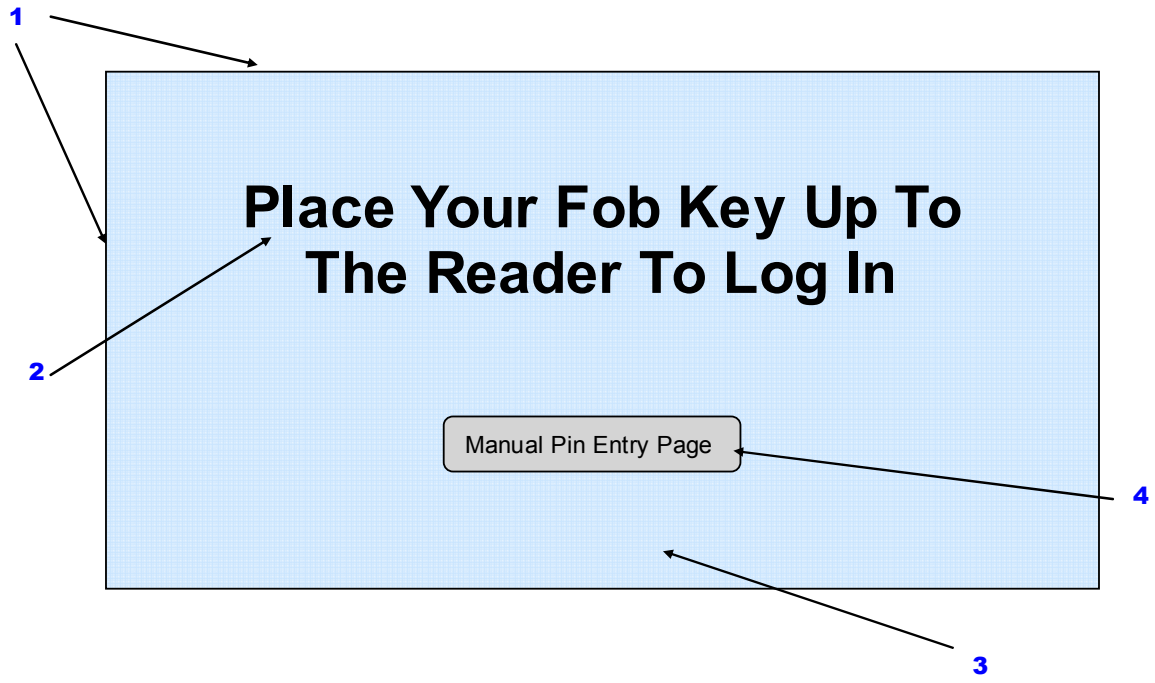
It is these properties which defines a student in this system, and it is these objects that will be passed around from client and server where there details may or may not be altered.

These student objects will be collated into a 'student object store', which is basically the database of student information, and in which this 'store object' will contain methods for retrieving, saving and navigating the student objects. The store object can then be saved as a data file for permanent storage.

Story Boards – The User Interfaces

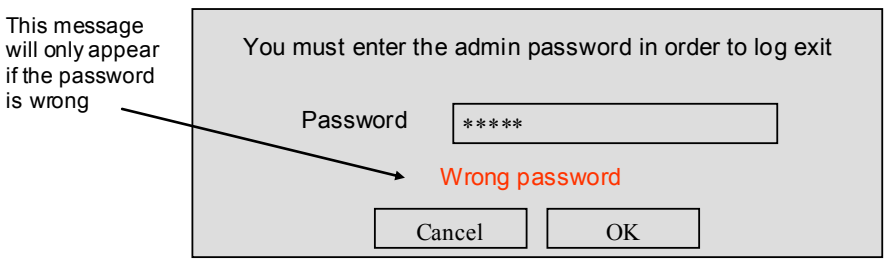
The Student Information Kiosk Interface

Introduction Panel

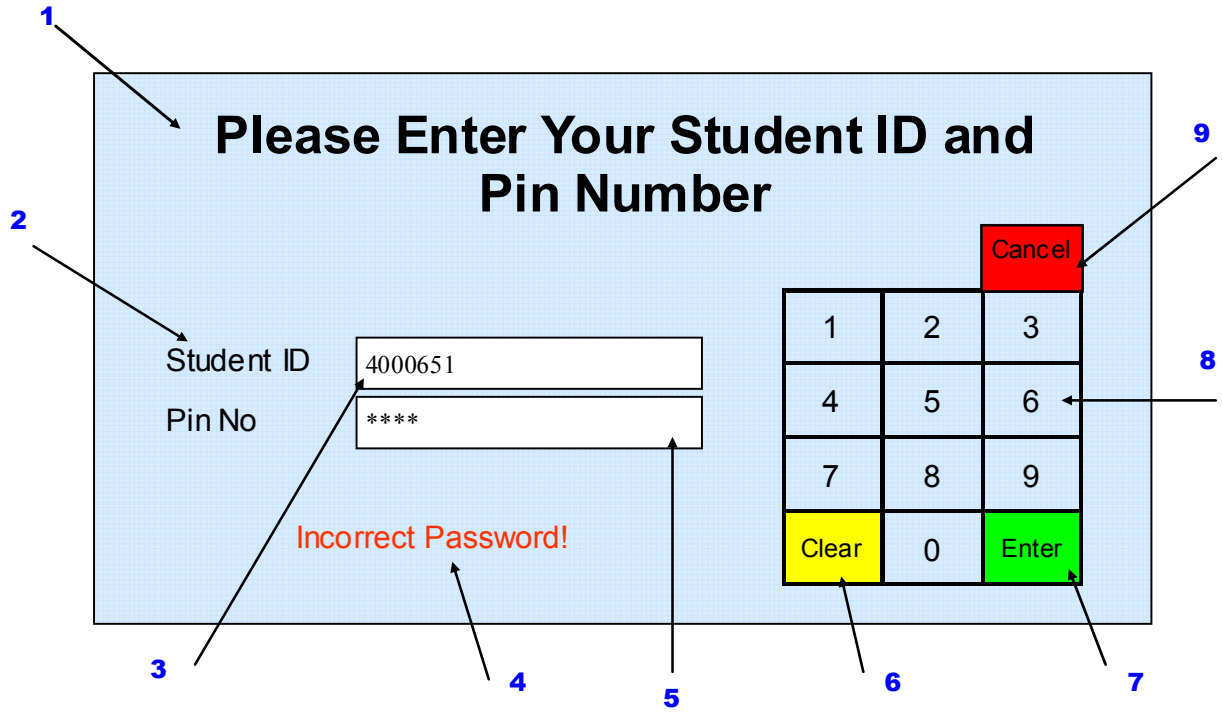


1	Panel Size	2	Introduction Message	3	Background Image
The panel size will be equal to 1024 x 600 pixel resolution to match that of the Samsung Q1 tablet PC, which is where the user interface is to be run		The message that will be shown is designed to aid users. It will also be used to display any error messages to the user, such as a technical fault		There will be a background image on the panel to make the interface look more appealing	

4	Manual Entry Button	N	Other Notes
If the user wishes to log into the student information kiosk by manually entering their student ID and pin number they can do when they click this button, which will take them to the manual pin entry panel. The button, as all button are likely to be an image icon rather than the standard grey button so that it looks more appealing		This will be the start and end panel of the state cycle, where a user starts and end using the interface. The mouse pointer will be disabled on this panel as it's to be displayed on a touch screen panel and won't include a mouse. As this is to be a run on a tablet computer on a university wall the interface must not be allowed to be exited by a student, so an administrator may press a shortcut key to bring up a dialog and then enter a password to exit (see below)	



Manual Pin Entry Panel



1	Title
Simple JLabel. This title message does not change	

2	Text Field Labels
The label of a text field	

3	Student ID Text Field
This is where the user enters their pin number, or rather where the numbers appear when a user clicks a num pad button	

4	Info Message
Here a message will be displayed informing a user of any advice or if any problems occur, such as a wrong pin entered or an RMI connection failure	

5	Pin No Password Field
The same as a text field except any characters in this field will be displayed as an 'asterix' for security purposes	

6	Clear Button
Clears the characters entered into the text fields	

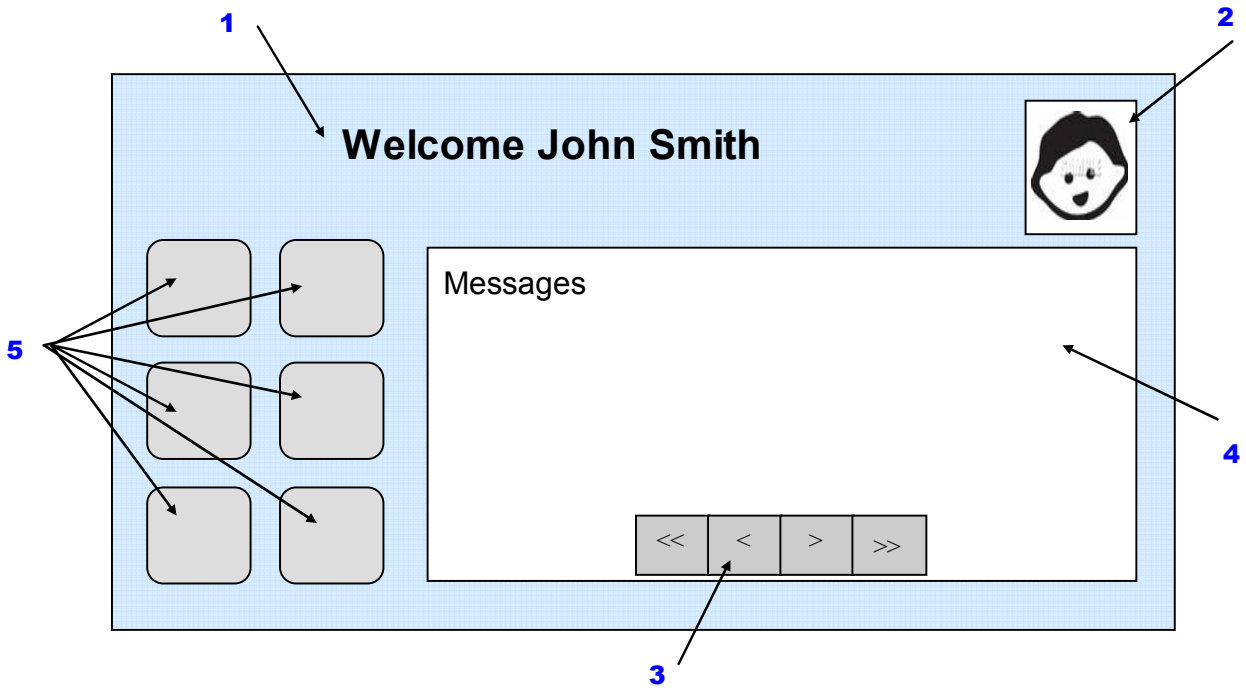
7	Enter Button
Takes the information entered into the text fields, validates them with the server and displays the 'home panel' if the entered credentials are correct	

8	Num Pad
When a user presses a num pad button, the corresponding digit will be displayed on the focused text field (or asterix if on the password text field)	

9	Cancel Button
Clears the text fields and returns to the 'introduction panel'	

N	Notes
This panel is on a timer, in which if there's no user activity, once the timer runs out will return to the 'introduction' panel.	

Home Panel

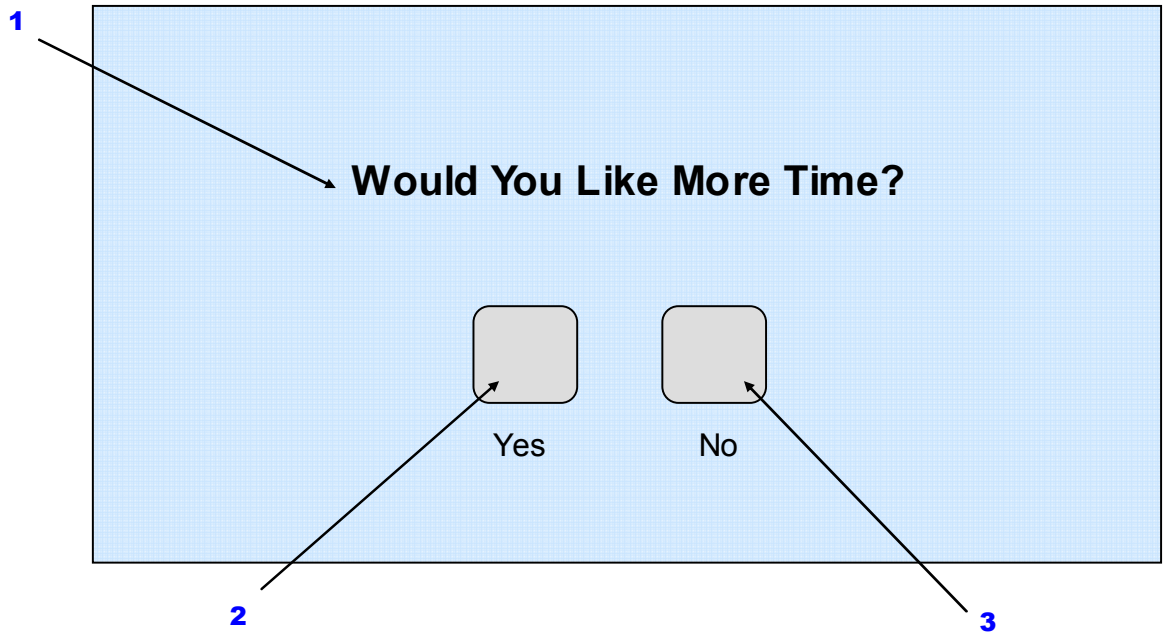


1	Title	2	Photo	3	In-Panel Navigation
Simple JLabel. This will also include the name of the user, which is brought from the student object, which was brought from the server		This is a Image Icon of the user (if one exists), which is brought from the student object, which was brought from the server		Buttons which navigate through elements of a particular kind, such as messages, timetable or results, which are in the form of an array in the student object	

4	Category Pane	5	Category Navigation Buttons
This is a panel within the home panel which displays other panels based on which area of the student information kiosk a user is in, which include messages, timetable, assignment results and Bluetooth options. These areas are selected via the category buttons (See '5'). The example shows 'Messages', which will display the message elements from the student object, which can be navigated through. The other categories mentioned are similar.		These are JButtons, which will include an image icon, they won't be the standard 'grey buttons'. When a user presses a button it will load a panel in the category panel to the right (these include messages, results, timetable, Bluetooth options), The other two buttons include 'campus map' which loads a new panel entirely and 'exit' which will load the original 'introduction panel'	

N	Notes
This panel is on a timer, in which if there's no user activity, once the timer runs out will load the 'Would you like more time panel' which give the user a choice to press a button if they require more time. This panel starts a new timer, so if there's still no user activity, the original 'introduction panel' will be loaded.	

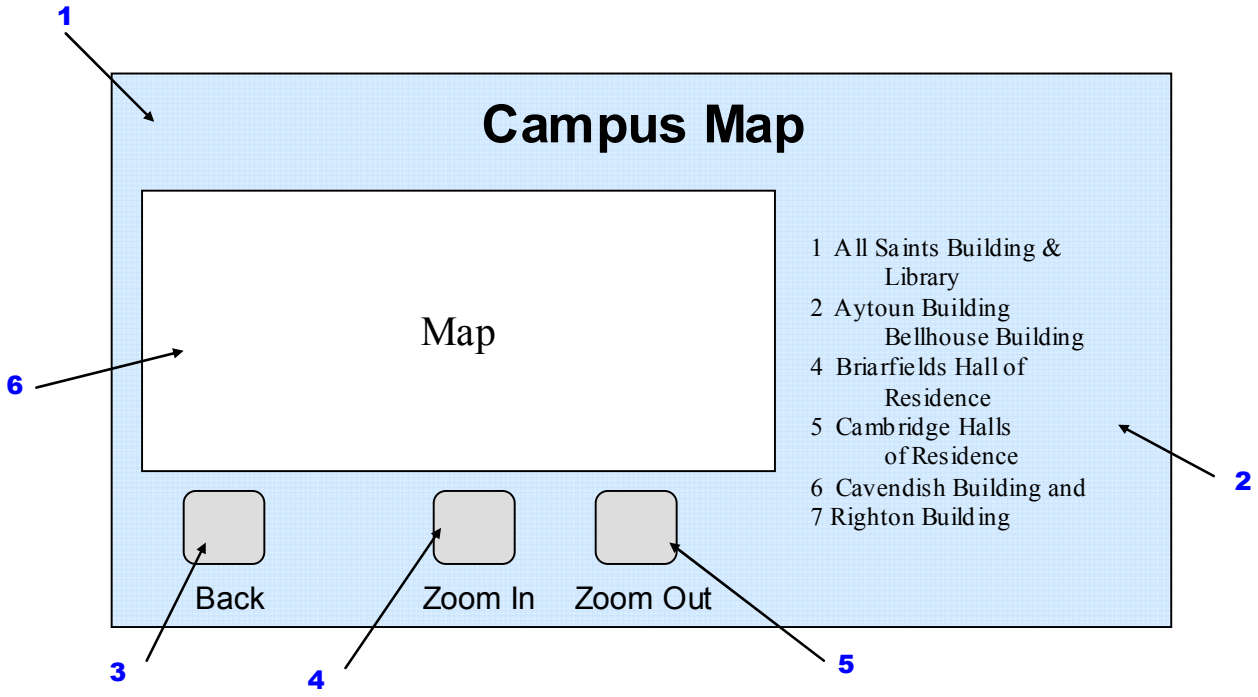
Would You Like More Time Panel



1	Title	2	Yes Button	3	No Button
Simple JLabel stating a question of is a user would like more time to complete their current task		This will include an image icon. If this button is pressed it will return to the previous panel that is was displaying		This will include an image icon. If this button is pressed it will return to the original 'introduction panel'	

N	Notes
This panel is on a timer, in which if there's no user activity, once the timer runs out will return to the 'introduction' panel.	

Campus Map Panel

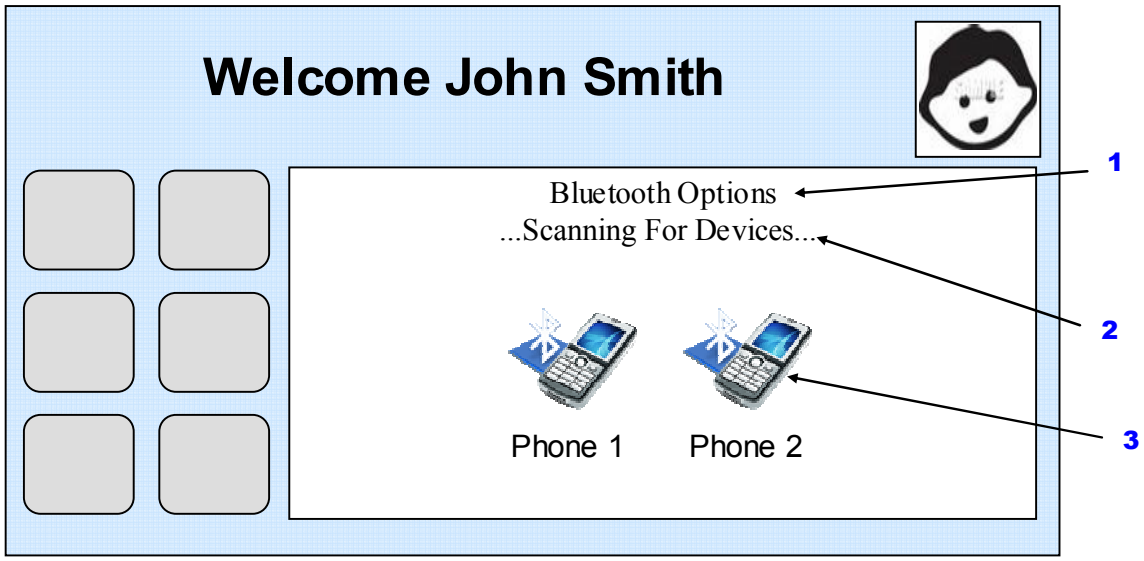


1	Title	2	Map Legend	3	Back Button
Simple JLabel Title		Simple JLabel containing the text for the legend of the map		A button (which will include an image icon) that will load the home panel (which was the previously displayed panel before this panel)	

4	Zoom In Button	5	Zoom Out Button	6	Map Legend
When this button is pressed, it will basically increase the size of the image so it appears 'zoomed in'. It should also maintain the same centre view point from it's previous size		When this button is pressed, it will basically decreases the size of the image so it appears 'zoomed out. It should also maintain the same centre view point from it's previous size		The map is an image of a map inside a JScrollPane, which will have no scroll bars and have mouse listener implementations, which will allow users to navigate around the map by pressing and dragging, much like that of the 'goodle maps' app on a touch screen phone	

N	Notes
This panel is on a timer, in which if there's no user activity, once the timer runs out will return to the 'introduction' panel.	

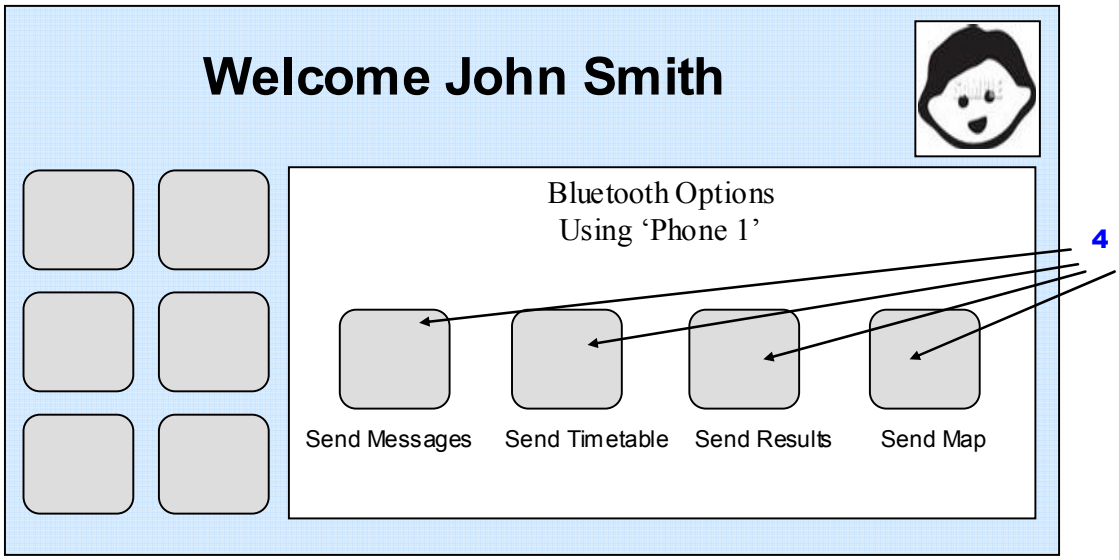
Bluetooth Pane (On Home Panel)



1	Title
Simple JLabel title	

2	Dynamic Message
This is a JLabel message for the user to inform them of events taking place and shows any warning messages if any problems occur	

3	Discovered BT Device
A button with an image icon will appear for each discovered Bluetooth device in range, along with the device's name. Once clicked, icons will other icons will appear enabling users to perform Bluetooth actions	

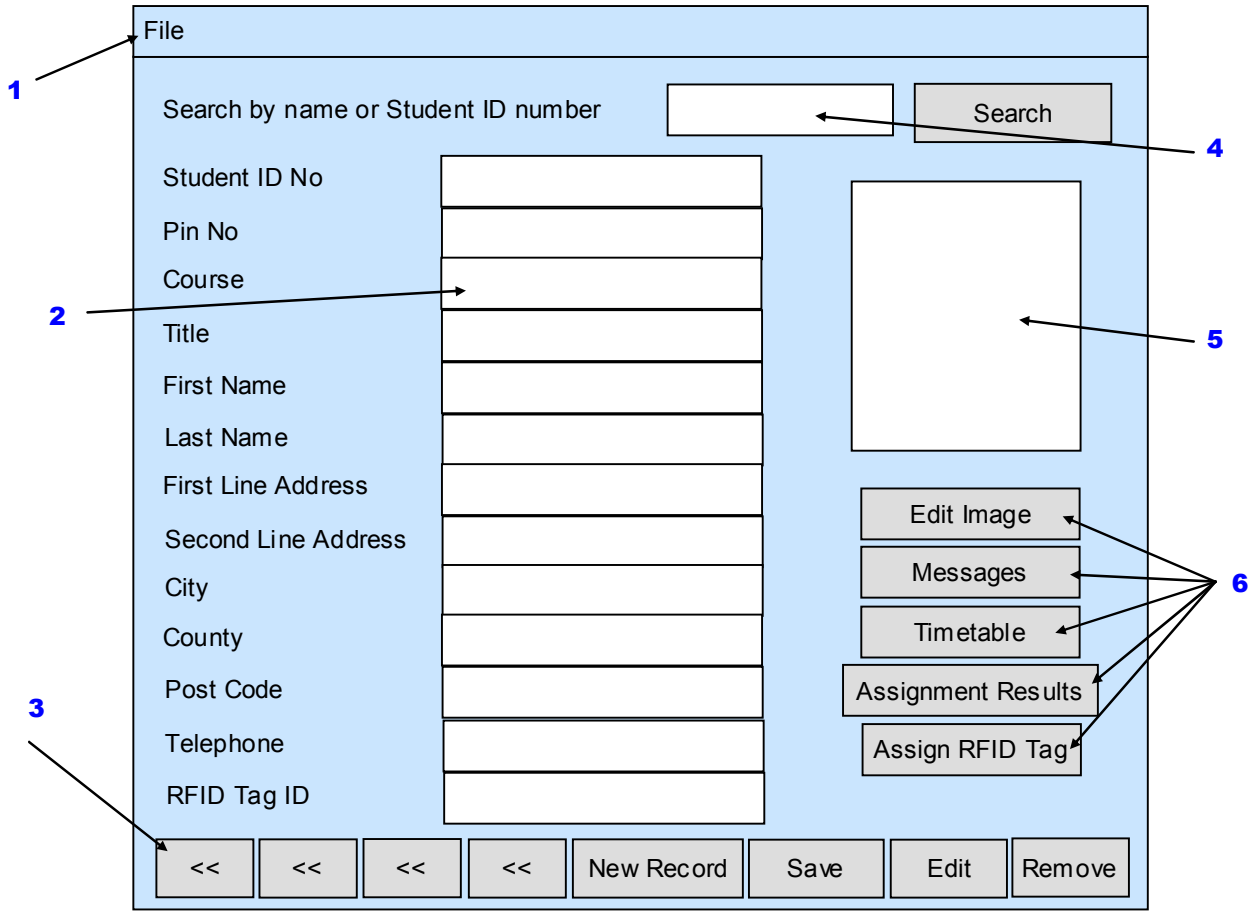


4	BT Action Buttons
When pressed, it will send a file (image or text file) to the selected Bluetooth enabled phone. Note, these buttons will include image icons	

N	Notes
When sending messages, timetable and results to a selected phone, it will send in the form of a text file, and will send the campus map as an image file.	

The Student Information System Admin Interface

Main Display GUI for the Records Manager

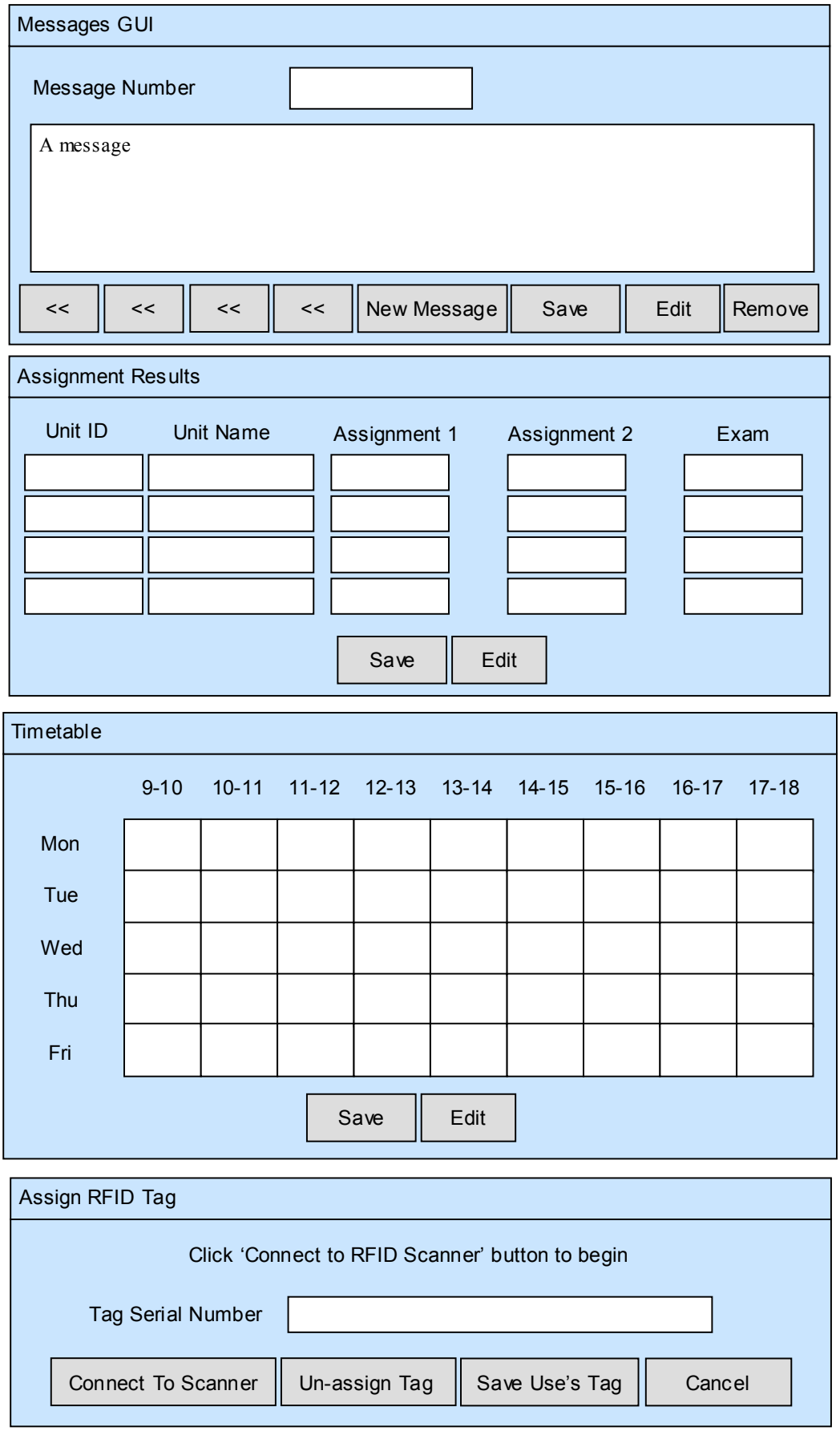


1	File Menu	2	Text Fields	3	Navigation Bar
A drop down menu which contains actions such as open store, save store, exit, load store from RMI location and save store to RMI location		JText fields, where a user would type in data about a student		Used to navigate through records, create new records, save a record, edit a record (text fields become editable) and remove a record	

4	Search Area	5	Photo Image	6	Buttons To Other GUIs
A user can quickly navigate to a record or a set of records matching the search criteria		Will display a photo image of the student (if one exists)		These are buttons to other related GUIs. The 'Edit Image' will bring up a file chooser, so a user can chose the image file from the file system	

N	Notes
The file menu items 'Open Store' and 'Save Store' will bring up a file chooser to enable a user to choose a file to save or load from the local file system (or a file system)	

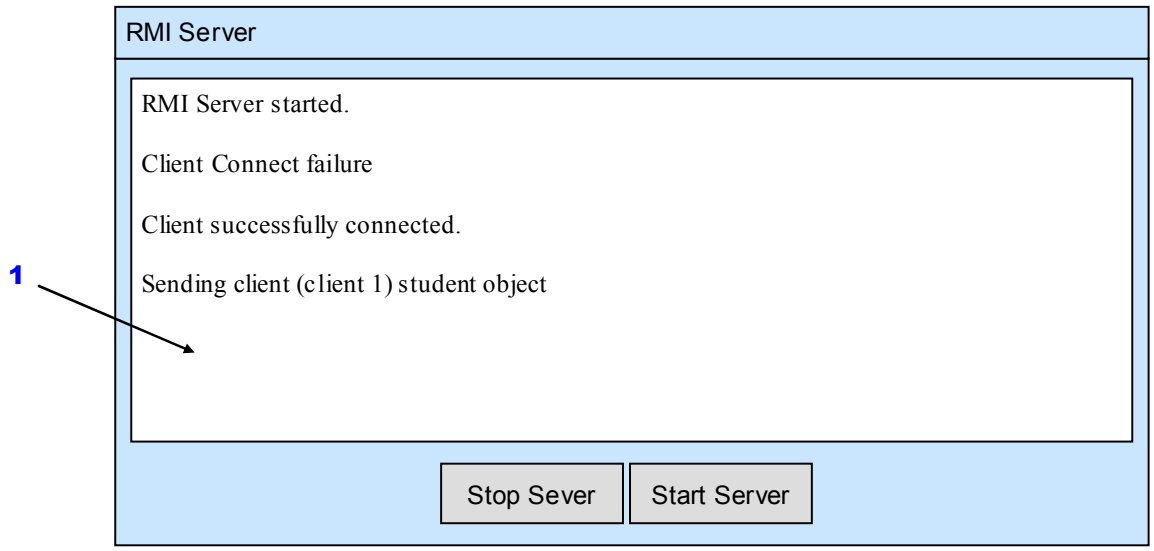
Other Record Related GUIs



The RMI Server Interface

There should also be a graphics user interface for the RMI server. Although this is not necessary, it shouldn't take long to do, and in today's world, most programs include some kind of GUI.

RMI Server GUI



1	Server Log Text Area

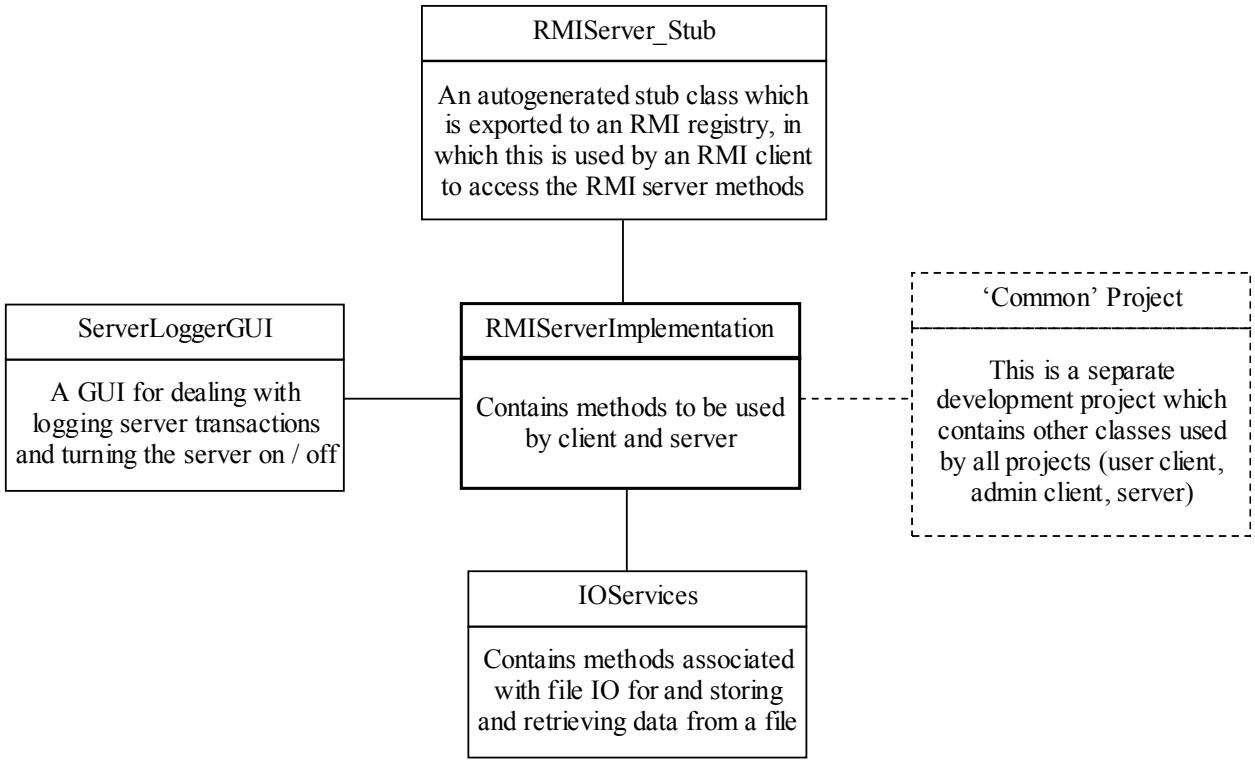
Any messages to a user would appear here. It's mainly a text area which contains a log of server events taking place. And it would state any server problems such as RMI binding issues or connection failure etc. The GUI would give the user the option to start and stop the RMI Server

Required Interface Functionality Specification

- All interfaces must provide user feedback wherever possible to allow the user to be safe in the knowledge that a task has successfully been performed
- In the event of an error, the user is to be informed immediately and must include useful information about the error in order to understand how the problem at hand can be solved
- The interfaces must incorporate extensive error handling and not just quit or crash, exceptions should be caught and handled
- The interface must provide data entry validation to encourage data coherence and reduce the human error factor. I.e. When a user is entering a telephone number into a text field, they shouldn't be able to save the details if the telephone field contains letters
- The interfaces must appear linear in structure to prevent navigation confusion
- The interface (particular the interface used by students) must be aesthetically pleasing and provide user friendliness and interaction on a number of levels, i.e. include sounds, images, colour and feedback.

General Java Class Structure Design

Server Project

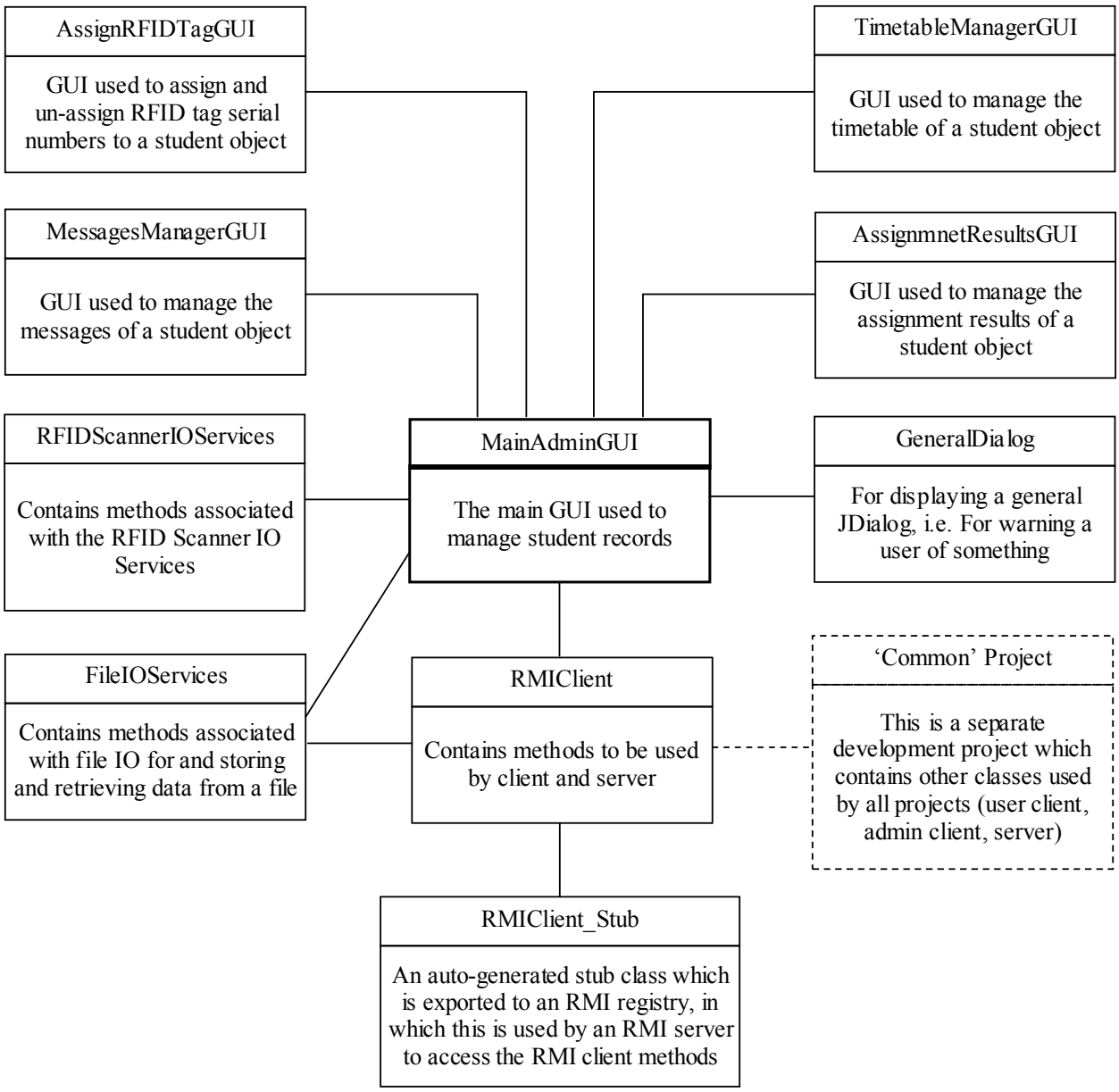


Note: The project entry class is indicated by a darker lined box

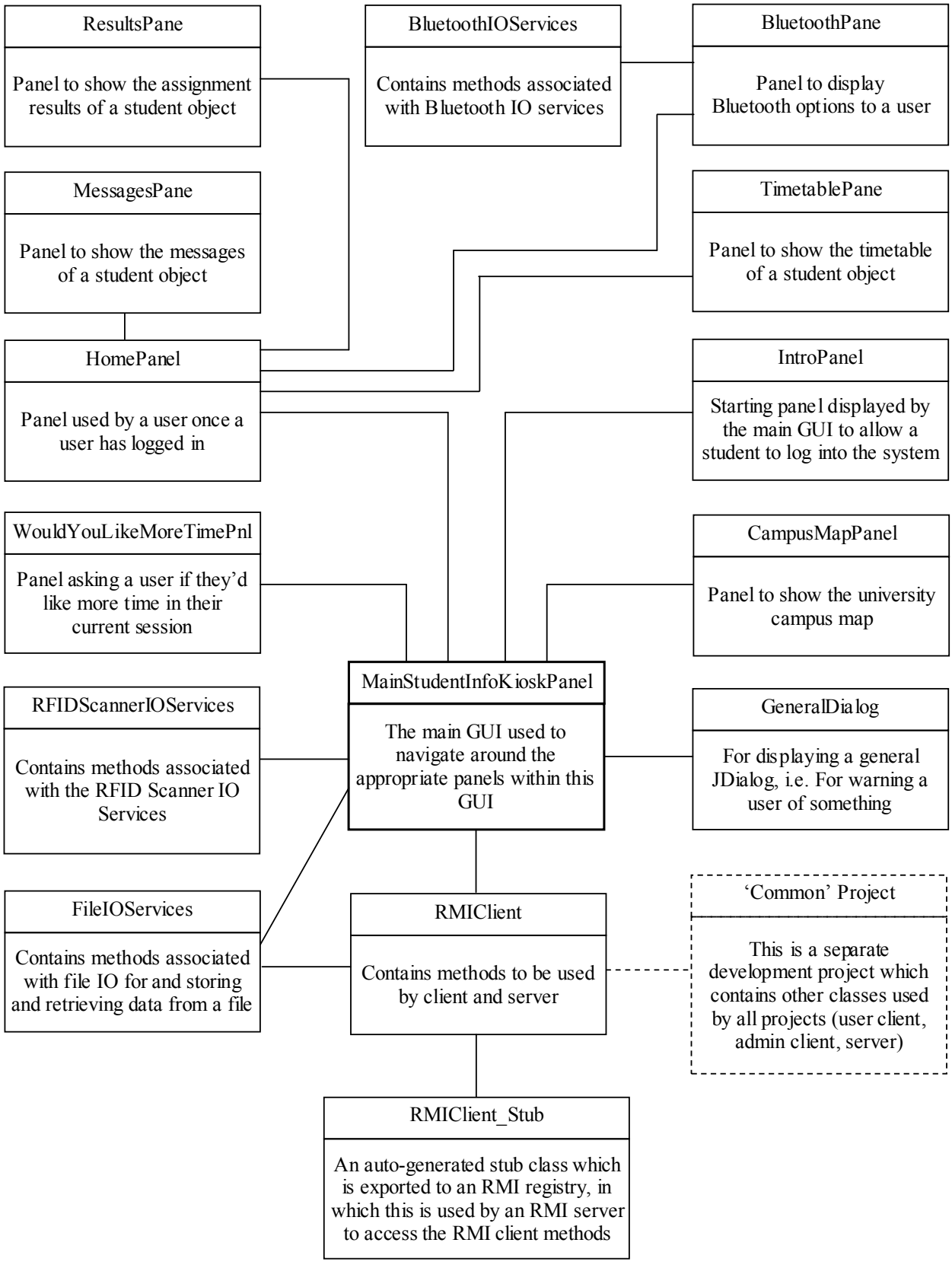
Common Project

The 'common' project contains classes used by all the other projects, which include the 'student object' class, the 'student object store' class and an 'RMI Server interface', which is required for an RMI system for method invocation across a network. A copy of these classes is in fact required by all projects, however in development, as these classes will be the same in each project, they need only be written once. When exporting a project to an executable file, say an executable jar file, the jar file will contain its own copy of these classes.

Admin Client Project



Note: The project entry class is indicated by a darker lined box



Note: The project entry class is indicated by a darker lined box

Design Patterns

The use of design patterns is a widely adopted technique to maintain standards and increase maintainability when developing programming projects. They can also speed up the development process by providing tested and proven development paradigms. When writing the code implementation of the classes, it will be beneficial to refer to a these design patterns for guidance.

The typical design patterns to follow will include:

Creational Pattern

Singleton: To ensure that a class has only once instance, in which the class includes a global point of access. The idea behind this idea is efficiency (only one object exists) but it also provides a unique way of organising objects into a tidy form

Builder: It should be possible to build some object in this system to have a different representation, while being the same object. Thus allowing better object re-usability

Lazy Initialization: Involves delaying the creation of an object until it is needed, thus increasing efficiency.

Behavioural Patterns

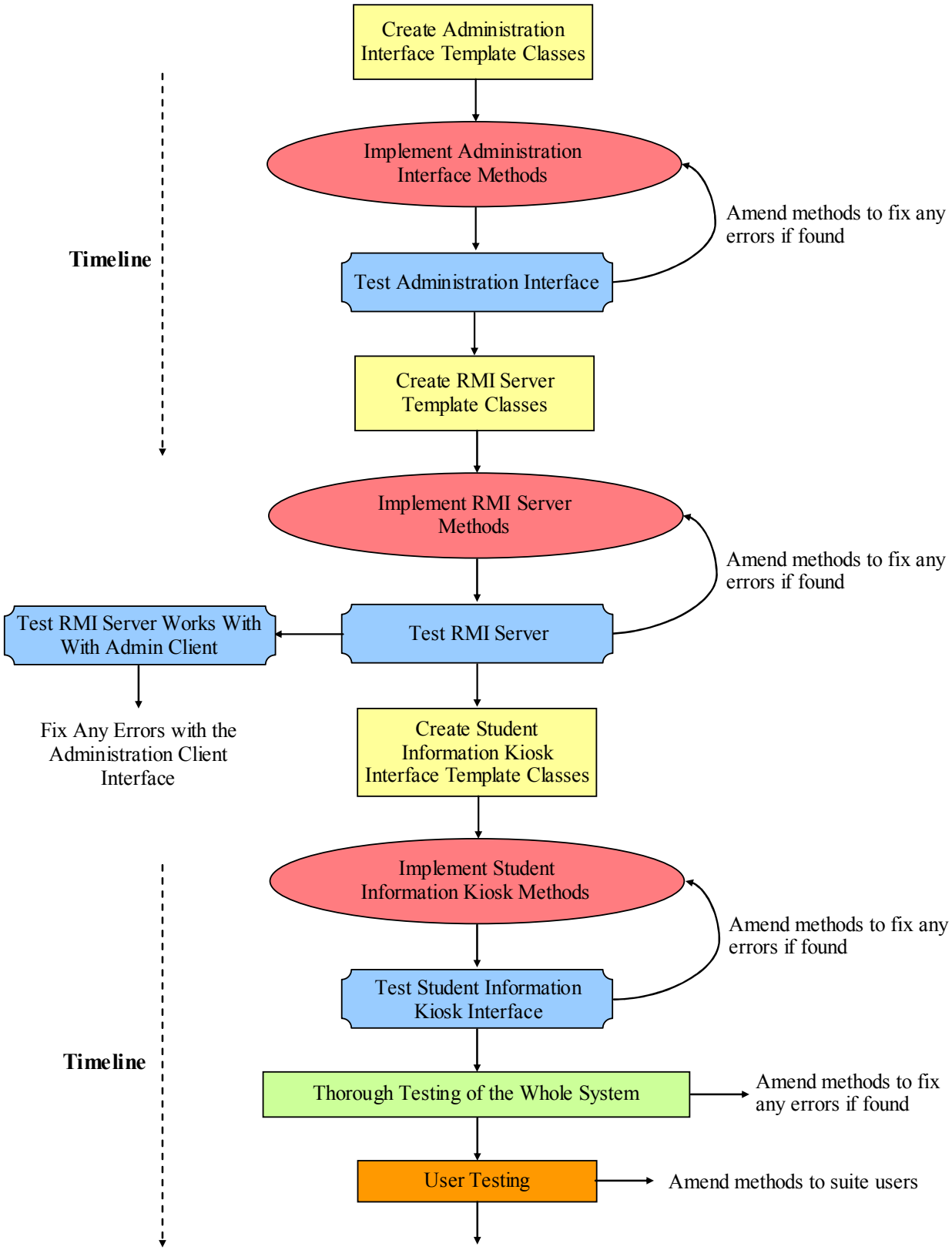
Observer: A useful pattern to use where the state of an object changes; When the state of an object changes, all of its dependencies should be notified so they can make an update automatically, rather than having to check if the state has been changed.

Template Method: To defer some steps of a method process to other 'sub methods'. This lets the sub method redefine certain steps of the method without changing its structure, thus it can be re-used in different way and thus don't have to create another method to do a 'similar' job.

(Gamma, Helm, Johnson, & Vlissides, 1994)

Implementation Draft

Implementation Plan



The diagram on the previous page shows a general plan of the stages involved in implementing the Student Information Kiosk system, along with a rough testing plan.

The first stage of the implementation process is to create template classes, which contain a list of methods without any method bodies. This will encourage that the designed structure of the system is adhered to, it also encourages forward thinking and planning, which is always a good technique, as it provides a set start and end to parts of the implementation and the implementation as a whole, rather than simply starting with a blank class and wondering where on earth to begin. It also provides a scope as to how long each part of the implementation will take, and how far away you are from finishing a part of the implementation, as the methods are visually there in front of you; they don't just contain a method body. This technique is a good way of demonstrating time management skills and it may also speed up the development process, as it will be clearer as to what needs to be done next, rather than figuring it out as you go. This technique of course is purely to provide some structure to the implementation process. It is likely that the implemented classes will differ significantly to the templates, as new ideas and better ways of coding something will be discovered along the way.

The next stage will be to actually implement each of the classes. This won't be a linear process, as programming generally isn't. They will be a lot of going back and forth in the implementation process, in which testing will be done in an 'as-you-go-technique', followed by a more thorough testing at the end of an implementation part process.

Once the whole system has been implemented, an even thorough level of testing needs to be performed to ensure the system works as it should. This will include a wider scope of testing such as platform testing, validation testing, user testing etc. All of which will be outlined in the 'testing' section of this report, and the testing results will be documented in this section.

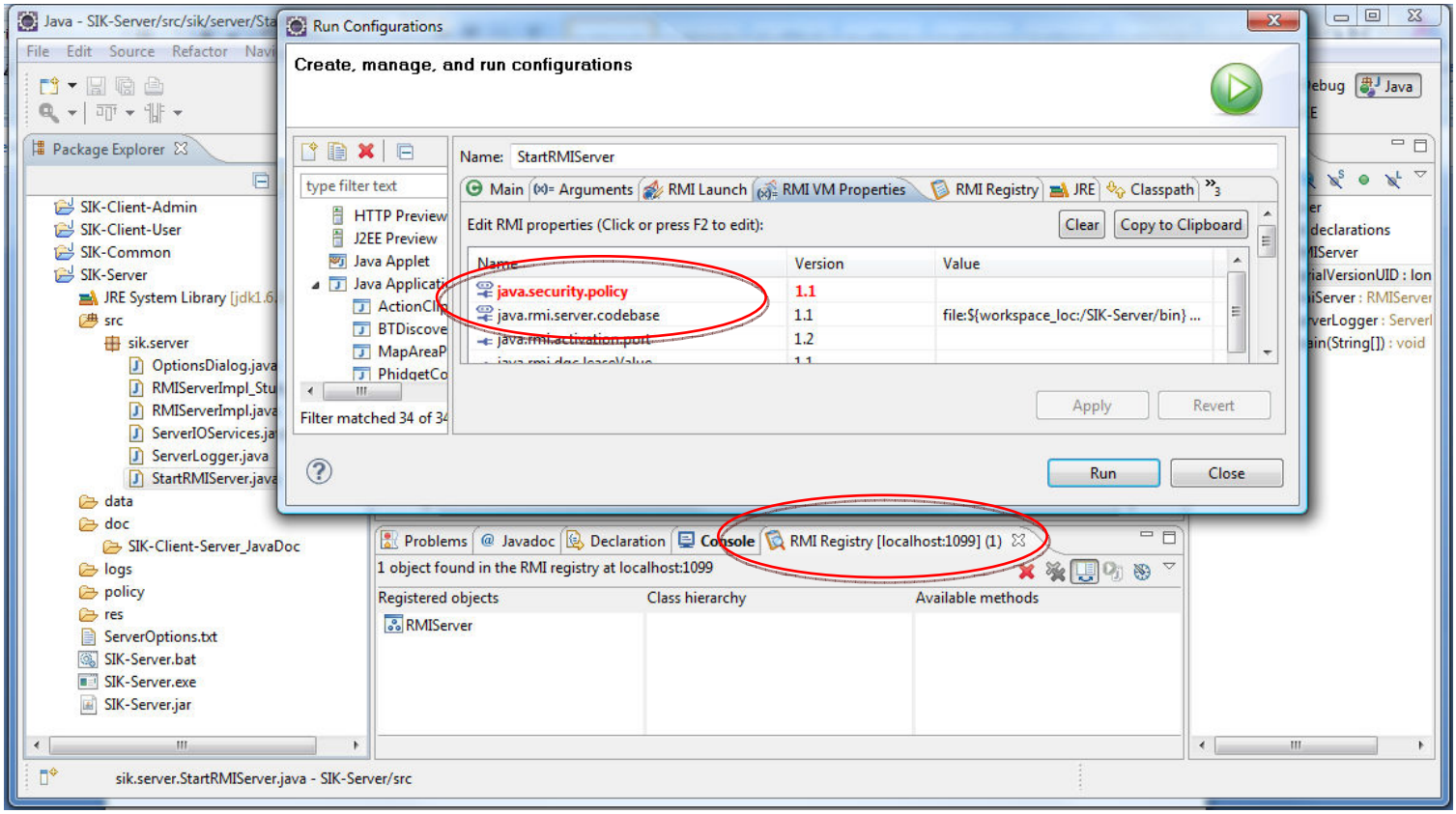
The Student Information Kiosk, as already mentioned is split into 3 separate development projects which include the 'Student Records Administration Interface', the 'Student Information Kiosk User Interface' and the 'RMI Server', with an additional 'Common' project to be used by each project. It would be a good idea to start with the 'Student Records Administration Interface', as this interface will be used to create the student objects, which are used by all three projects. In creating a graphical user interface to do this, it means that the student objects need not be coded by hand in the future.

After implementing the student records administration interface, the RMI server will be implemented and tested to see if it works with the student records administration interface (i.e. an RMI connection can be made and utilised). Once this is done, it provides a solid foundation to implementing the actual Student Information Kiosk interface, which will include similar method implementation for the technical side of things, however this project will be much larger in content, however as the foundation is there (student objects, RMI server connectivity) this shouldn't be too difficult to implement.

Once the system has been thoroughly tested and user tested, it would be good practice to go back to the implementation and make some necessary amendments or additions to fix any errors found and to comply with any user requests. However, due to time limitations this may not be fully possible. If this is the case the proposed changes will be mentioned and expanded upon in the evaluation section of this report.

Development Tools

The use of eclipse is to be used to implement the entire system. Eclipse is a multi-language software development environment comprising of an integrated development environment (IDE) and extensible plug-in system. It is written mainly in Java. Eclipse is an ideal tool to use in the implementation process of this system, as, with its extensible plug-in system, can include an RMI development plug-in which include tools to aid the development of RMI based applications such as RMI registry monitoring and an interface to set typical RMI properties such as a security policy which is required to be including when running RMI applications.



A development environment is generally used in industry today for a multitude of reasons, they provide:

- Automatic compilation of code
- Java class path handling
- On the fly compilation error notification
- On the fly package, method and variable lookup and auto-complete when typing
- Class, package and folder organisation
- Centralised development tools

All in all, it simply makes development much easier, it would be foolish not to use one. I have chosen to use 'eclipse' due to the fact that it's a free development environment, it is popular so there are always new products, plug-ins and enhancements available plus I have used it for a few years so I am familiar with it.

Interim Report

At present, the actual implementation part of the project is more or less complete, with perhaps a few minor adjustments to be made. This section has been included to document any major project design / implementation changes that have taken place and to asses my own time and project management skills, to identify where I am up to so far and to outline any major difficulties found. This section will only be brief, so as to not repeat myself in the evaluation section of this report.

Time Management

The original time plan for each section of this project went rather smoothly for the first few weeks and the time gage for each section was accurate up until the implementation section, which involved actually creating the Student Information Kiosk System. It is at this stage where the time required to implement the system was underestimated considerably, which was due to underestimating the shear amount of code required to implement the system, and running into a few technical difficulties (described later). However it is safe to say, with the implementation now complete, it is possible to still complete the report to as high a quality, as long as I put some extra effort it, however it may not be possible to complete the final stages of the report outlined in the original time plan, which involves going back to the implementation once the thorough and user testing has been complete, and making necessary amendments to reflect the results of the testing. The amendments that I would have made, however, and how I would go about implementing the changes will be discussed in the evaluation section of this report. A new, adjusted time plan is included on the next page.

Major Design / Implementation Changes

There have been no major design changes as compared to the proposal of this project; all the proposed elements of the proposal have been successfully applied, with the exception of a minor change to the implementation, which includes the user access methods to the Student Information Kiosk. Originally it was supposed to support multiple access methods to the interface, i.e. RFID scanner, Bluetooth pairing, bar code scanner etc. The system does support two ways off access, which include an RFID scanner and access by manually entering a pin, but that is all. This is due to programming language compatibility issues. It was decided earlier on that the system was to be implemented in java, and the implementation then want underway, however when it came to incorporating the multiple means of access, the hardware that I had access to only supported alternative programming languages such as C++ and C sharp.

As I had already done a lot of the implementation, it wouldn't have been feasible to start again or spend too much time attempting to create a multi-programming language based system, therefore the additional access methods were left out. However the implementation still performs the tasks to which it was intended.

There exists but a couple of other, even minor changes, such as the class layout, however these will be outlined further in the continued implementation section of this report and discussed in the evaluation.

Revised Project Plan

			June					July				August					September				
Task	**Start Date**	**End Date**	03	08	15	22	29	06	13	20	27	03	10	17	24	31	07	14	21	28	**Duration**
Introduction & ToR	03-06-10	08-06-10																			1 Week
Research	03-06-10	29-06-10																			4 Weeks
Design	29-06-10	13-07-10																			2 Weeks
Implementation (Draft)	13-07-10	31-08-10																			7 Weeks
Interim Report	31-08-10	01-09-10																			1 Day
Implementation (Finalise)	01-09-10	07-09-10																			6 Days
Software and Device Testing	07-09-10	14-09-10																			1 Week
User Testing	07-09-10	09-09-10																			2 Days
Evaluation	17-09-10	24-09-10																			1 Week
Project Review and Hand-In	24-09-09	**28-09-10**																			4 Days

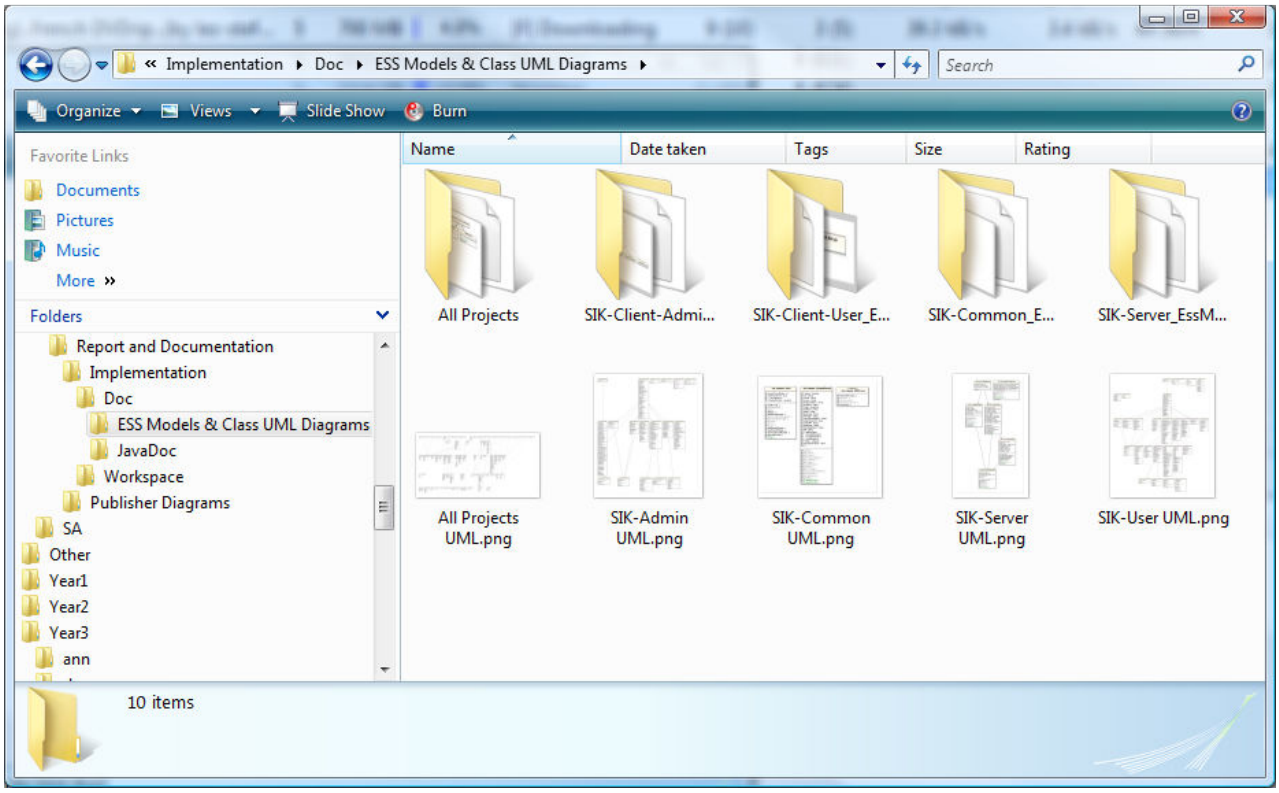
Implementation Continued

Class ESS Models, Model Details and UML Diagrams

There have been a number of class UML diagrams generated for the Student Information Kiosk System; however they're not going to be included in this report due to the fact that the models and UML diagrams can't feasibly be put on an A4 sized piece of paper. They are however included on the CD provided with this report. The ESS models and UML diagrams have been automatically generated using the 'Eldean ESS Mode Creator' software. The models provide a thorough breakdown of all the packages, classes, methods and variables within the development projects along with all their associated information, such as the 'type' of variable (integer, string, double etc), and a method's protection level (public, protected etc).

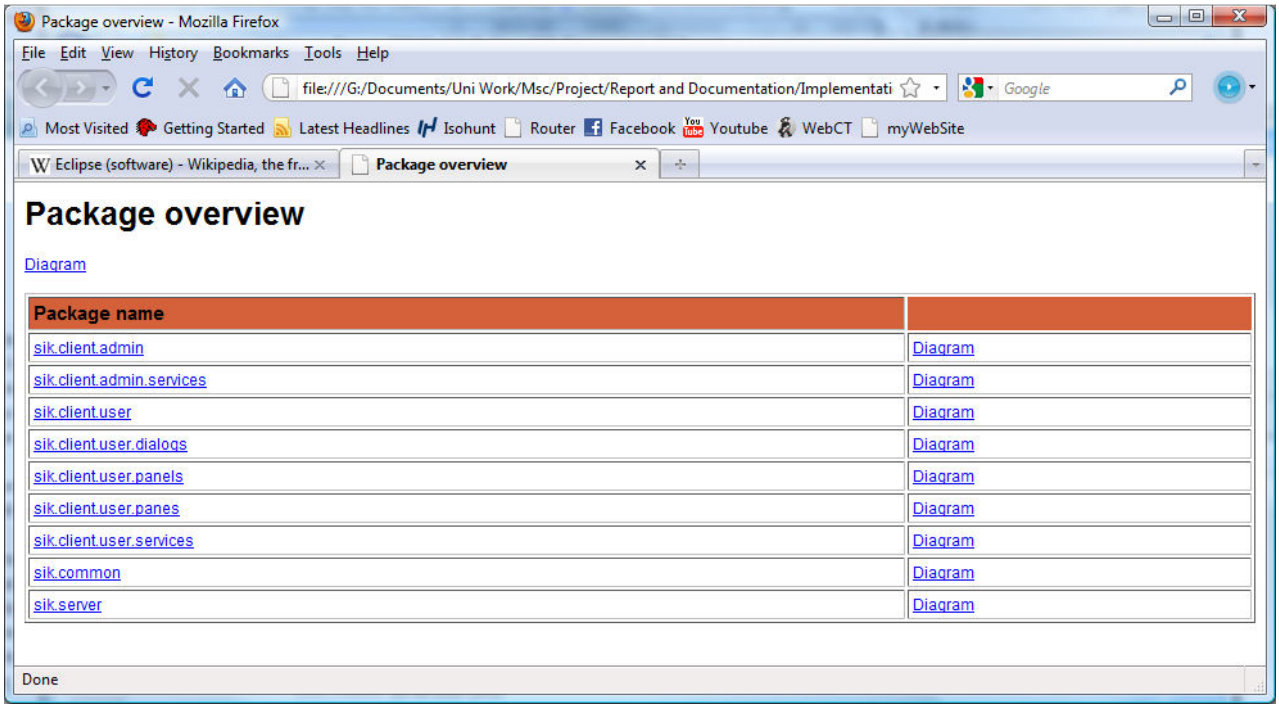
On the CD:

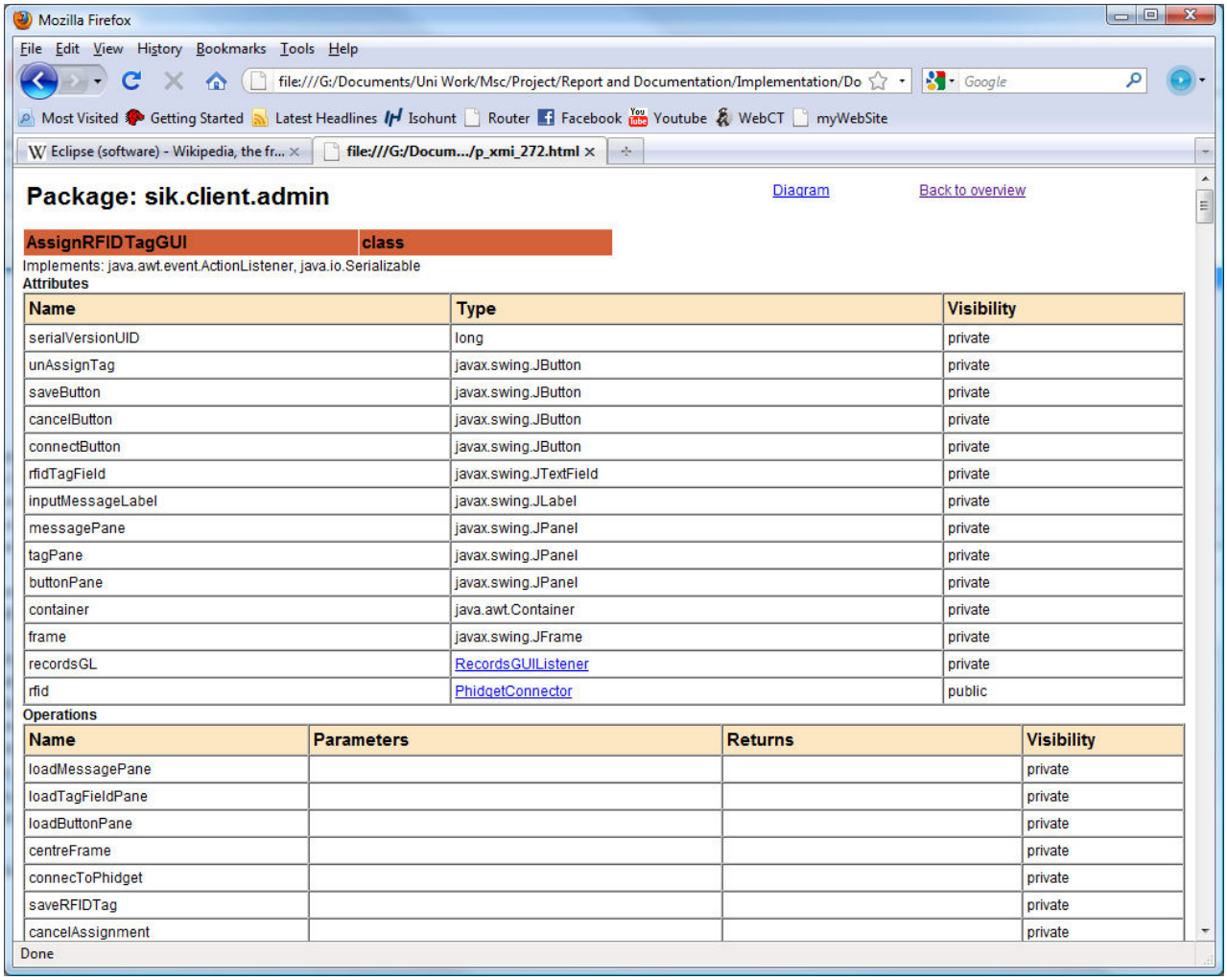
Navigate to *"Implementation" --> "Doc" -- > "ESS Models & Class UML Diagrams"* folder:

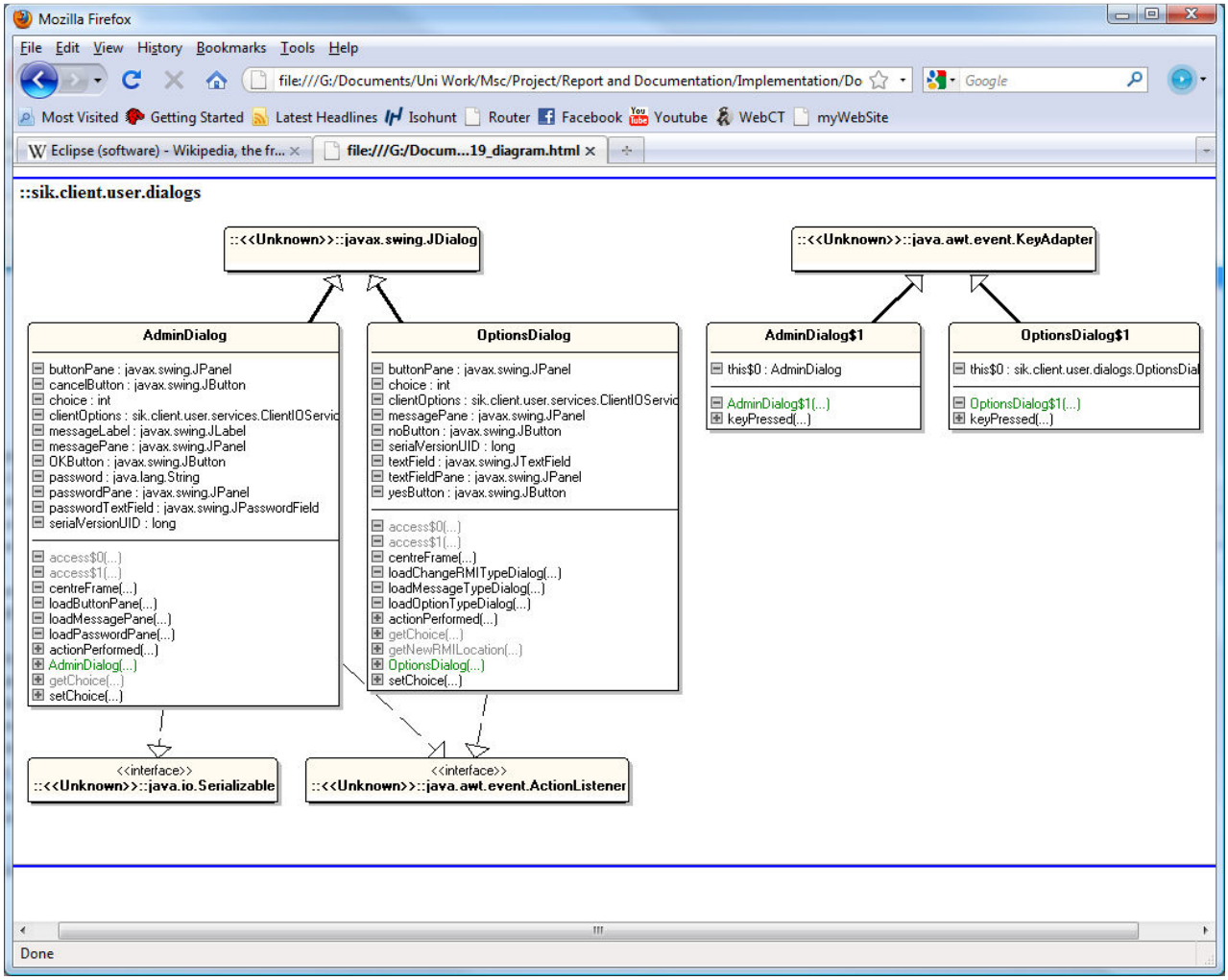


Here you will find a number of .png images which are UML diagrams of specific projects in the Student Information Kiosk system. There are also a number of folders which include ESS models and model details of specific projects/packages in the Student Information Kiosk system. Within each of these folders is a file named 'overview.html', which is the index of a HTML based overview of specific packages within the projects of the Student Information Kiosk system, which also include further 'sub' UML diagrams.

The web-page based overview: 'overview.html'







The above screenshot shows a 'standard' format in which to represent classes in UML format.

Javadoc

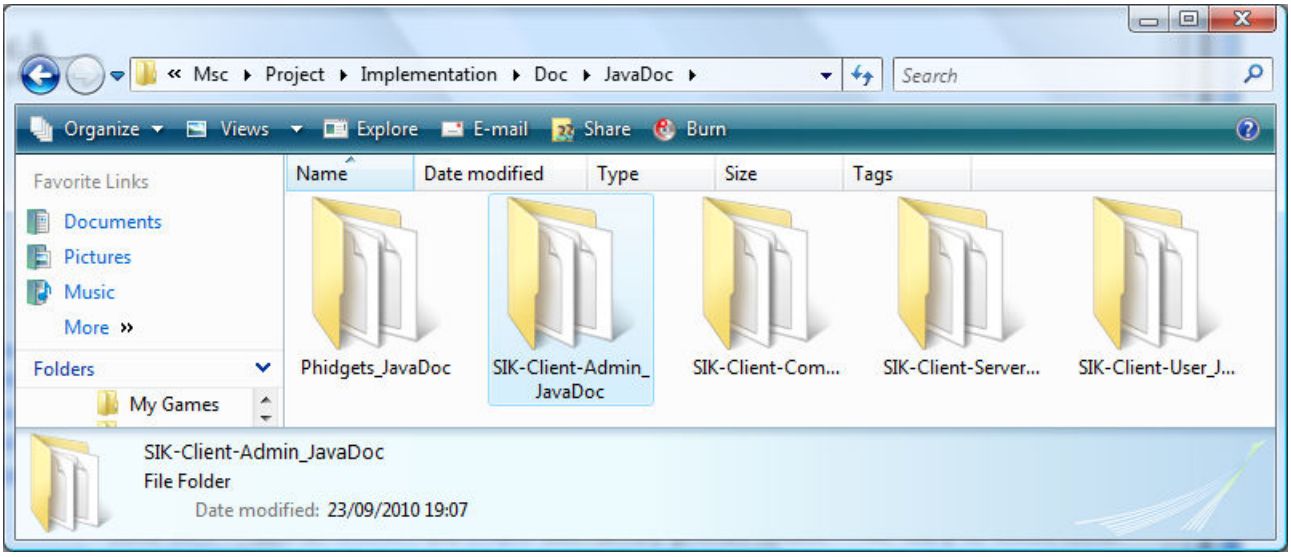
Also included on the CD are 'Javadoc' files. Javadoc is a documentation generator from Sun Microsystems for generating API documentation in HTML format from java source code. The 'doc comments' format used by Javadoc is an industry standard for documenting Java classes. Javadoc comments have been included in the Student Information System source code which describe a class's properties, the class's methods and all the associated attributes such as parameters and return values.

Javadoc comments are written in between doc comment identifiers (/** */) as follows, and specific keywords for certain fields specified after a @ sign:

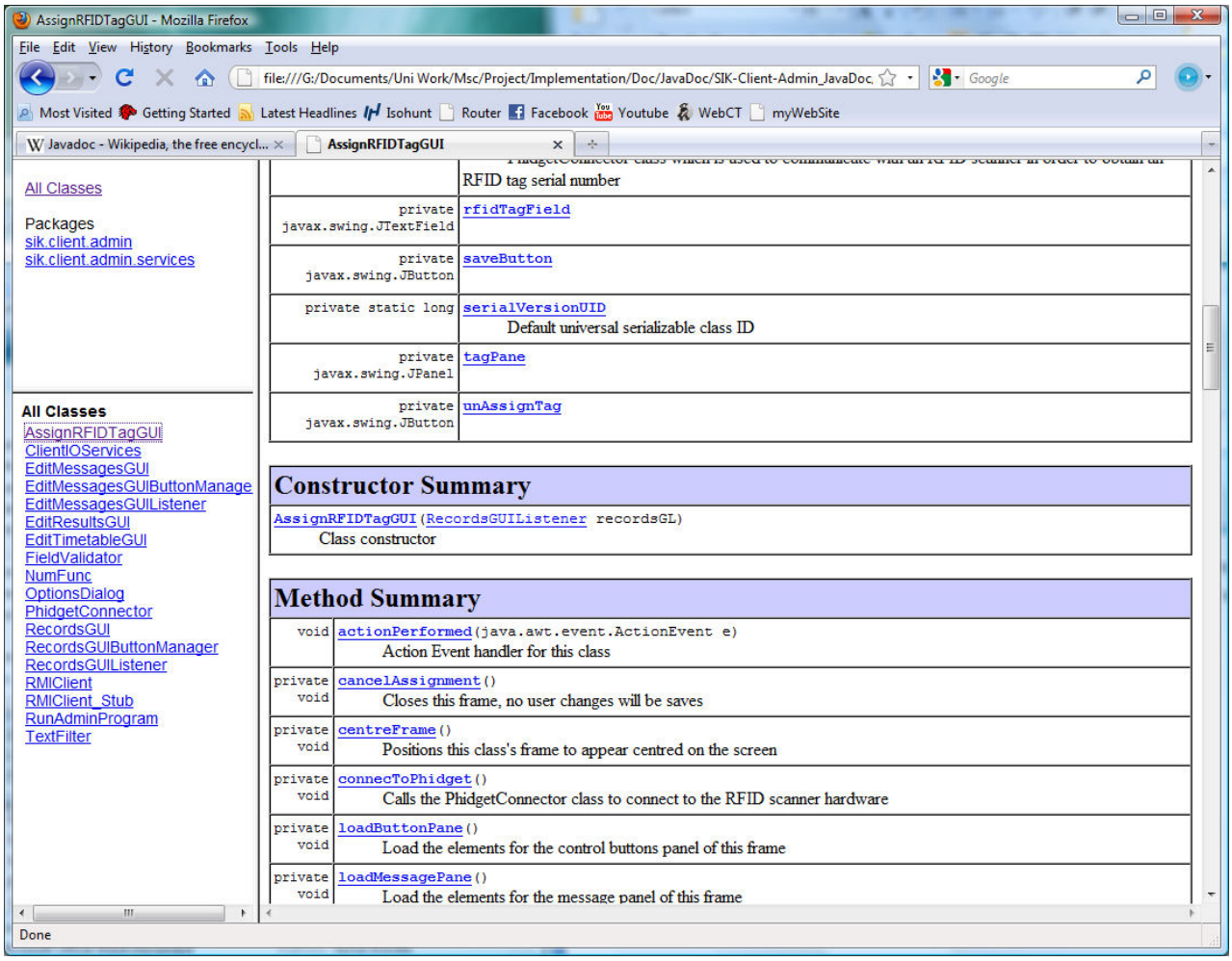
```
/**
 * Class constructor
 *
 * @param recordsGL
 *            The parent class which called this class
 * @throws PhidgetException When a connection to the phidget is encountered
 */
public AssignRFIDTagGUI(RecordsGUIListener recordsGL)
            throws PhidgetException {
```

On the CD:

Navigate to *"Implementation" --> "Doc" -- > "JavaDoc"* folder:

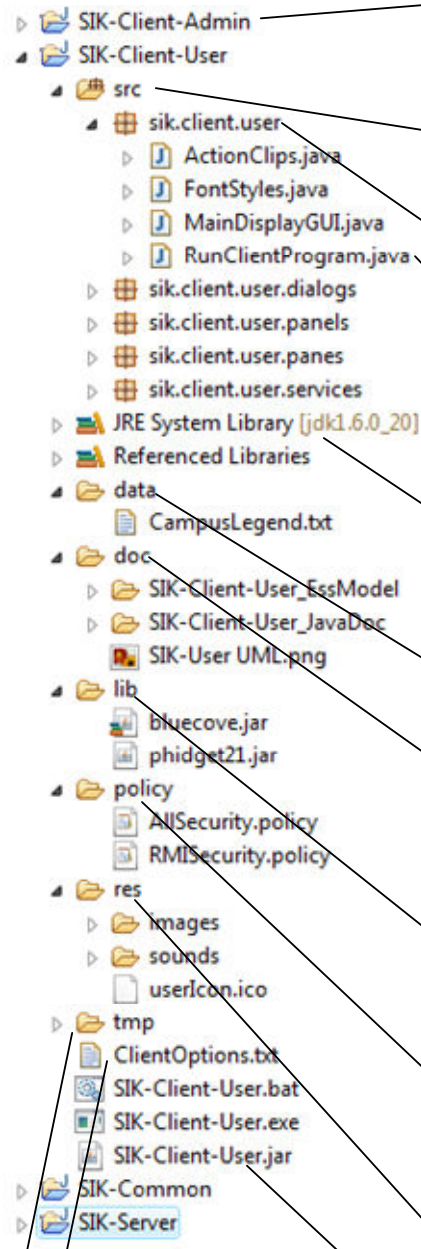


Here you will find the individual folder for each of the Student Information System projects. Within each folder is a file named 'index.html' which will present the overview of a project. They are again in HTML format.



Project Packages and Folder Layout

The projects packages and folder for Student Information Kiosk follows a typical standard:



Project
A project contains a collection of packages. Typically a project is the 'root' folder for all the folders and packages it contains and generally is not reference by any other projects unless specified

Source Folder
The source folder holds the java source files, which are often structured into 'packages'. The source folder will be translated to the 'bin' (binary) folder upon compilation, which it will store the compiled classes in a folder structure corresponding to the dots (.) in the package name.

Java Source Files
These are typically translated to java 'classes' upon compilation. These source files contain the written java code, which include methods and variables

Libraries or APIs
Libraries are a set of java classes in which developers typically use to aid project development. They are basically a large list of functions or methods that can be called and re-used. This library is from Sun's java development kit version 1.6

Data Folder
Contains and data files that will be used by the system

Documentation Folder
Contains any project development documentation that would typically be used by future developers of this project or anyone that wishes to find out any information on written classes, methods or variables and what they do

Library Folder
Contains any addition libraries that can be referenced, in which will be used in the development in this project. These libraries are generally stored in 'jar' file, which is much like a 'zip' file

Policy Folder
Contains a java security policies. These define what an end user can and can't do / have access to. A policy is not always required, but in the use of RMI application it is

Resources Folder
Contains any resources that are used in the project, such as images, text files, sounds etc

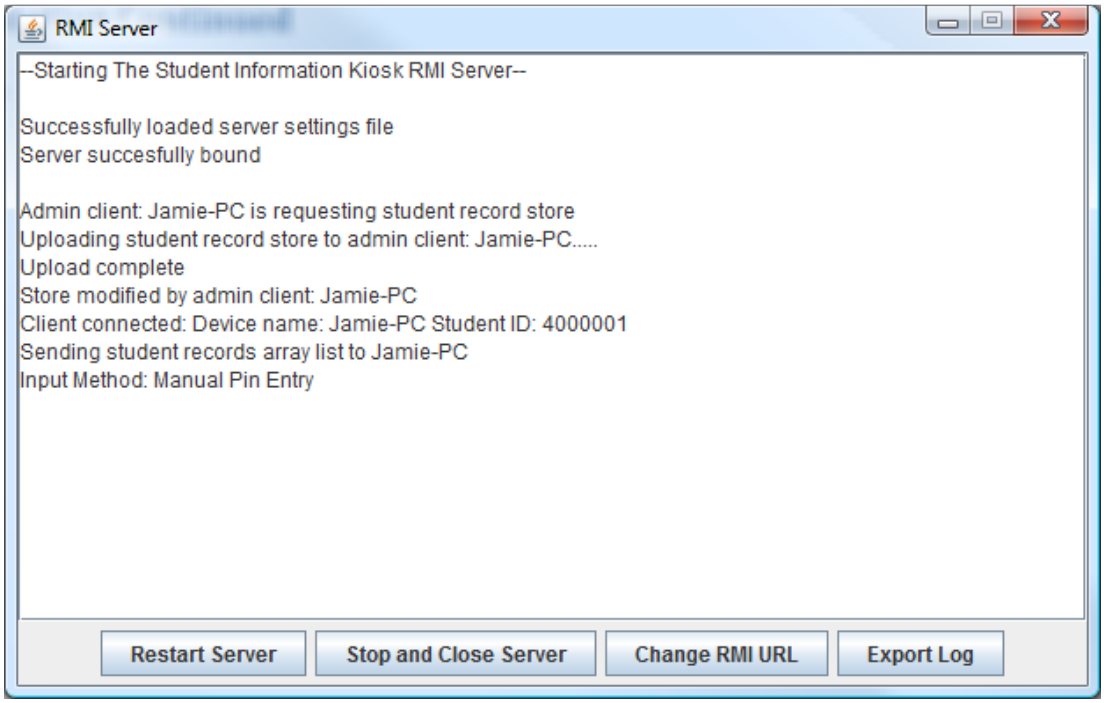
Options Text File
This is basically a 'config' text file for the system. It allows a user to amend particular settings, that the application will use at run-time. This means a that particular application preferences or settings don't have to be hard-coded into the system

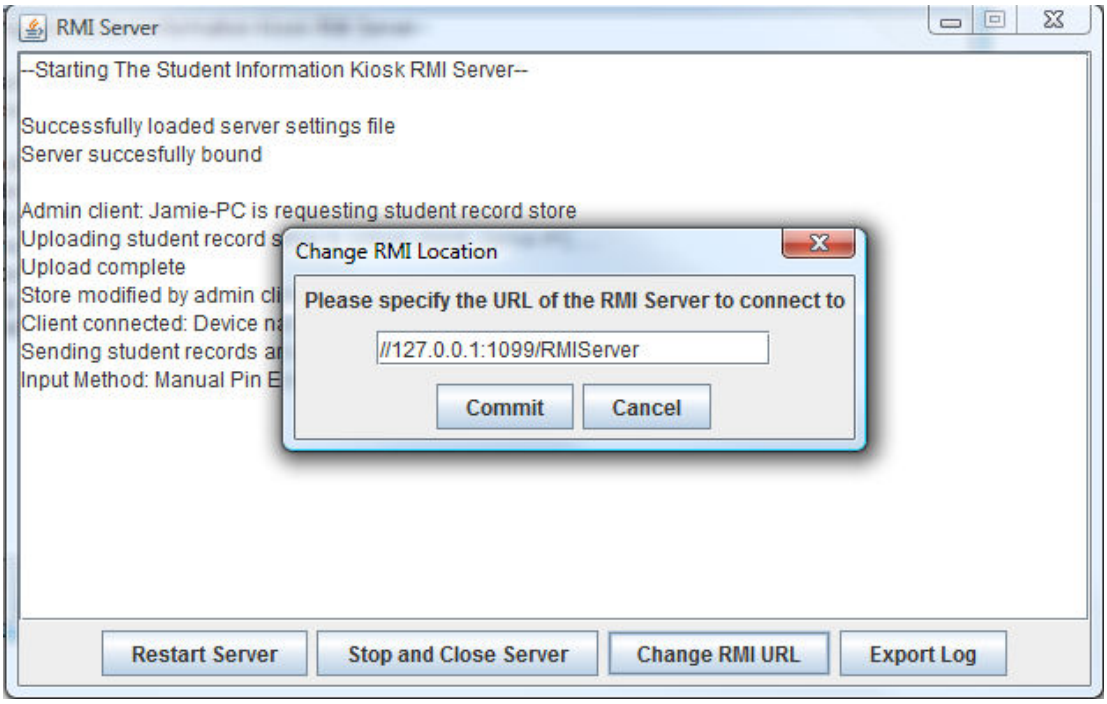
Executable or Runnable Files
These are the end result of the project. They are single executable files that contain the classes that run the actual application, which allows the user to click and run the application without having to type commands into a consol

Temporary Folder
This folder is used to store any temporary files. Temporary files are typically program generated files which can be re-generated when needed and utilised only for a short period of time

RMI Server

Screenshots





The screenshots above show the RMI server interface in action and the popup dialog on the RMI server interface that allows a user to change the RMI server location.

The Workings

The RMI server has been designed to be clean and simple with the workload being taken off the server and placed on the clients. The RMI server's main purpose is to send the clients either a student object (to be used by the Student Information Kiosk Interface) or a student object store (to be used by the Student Records Administration Interface) when a client requests, in which a client will invoke one of two methods from the server:

```
public StudentRecord getStudentRecord(String uniqueInput,
String computerName, boolean isRFIDInput) throws RemoteException {

        ...

        return StudentRecord;
}
```

The above method would be called by the Student Information Kiosk interface, in which the client must provide a number of parameters which include a value unique to an individual student, which can be a student ID number or an RFID tag serial number. The client must also supply the computer name of the calling class (this is merely for server logging) and state which unique student value it is supplying, which is a boolean value of whether it's an RFID serial value or not. The RMI server will then lookup the corresponding student object from the student objects store and send it to the client as a return object. Now the client will have the student object which will contain all the details of that student.

```
public Store getStore(String computerName) {

        ...

        return store;
}
```

The above method would be called by the Student Records Administration Interface. It works in the same manner as the method before, however as the method returns a store and not an individual record, it need not supply any other parameters.

One thing to note, at no point would the RMI server call any methods of an RMI client, in fact the RMI clients doesn't contain any methods for the RMI server to call. It's purely a 'one way method invocation scheme' and the RMI server does not have to be aware of an RMI clients existence until an RMI client calls once of its methods.

Another important method within the RMI server is:

```
public void saveStore(Store store, String computerName)
                    throws RemoteException {
        ...
}
```

This is also used by the Student Records Administration Interface to allow the RMI Server's student record to be updated remotely.

When the RMI server is first loaded, it must do a few things before it can be accessed by a remote client and to actually become an RMI Server.

The first thing it must do it define it's codebase location. This is the location where the RMI registry on the client and server will look to access an RMI Server's methods and objects. Typically this would be a remote location that can be accessed by client or server.

The next step is to define a security policy for the RMI server. Java allows a security policy to be set which defines a user's access rights to that class. This is not always required for a java application, however in order for the RMI server to work, it is required. Once this is set, a new security manager object is then created.

The next step is to actually bind an RMI server object to the 'RMI Registry' and the name of the RMI server object, which includes the location of where to find this named RMI server object. Once this is done, the RMI server has been bound to the RMI registry and can be accessed by an RMI client.

The code is outlined below:

```
if (System.getSecurityManager() == null) {
        String currentDir = System.getProperty("user.dir").replace('\\','/');
                String codebase = "file:/" + currentDir + "/bin/";
                System.setProperty("java.security.policy",
                                "policy/AllSecurity.policy");
                System.setProperty("java.rmi.server.codebase", codebase);
                System.setSecurityManager(new RMISecurityManager());
}
try {
        registry.rebind("RMIServer", this);
        Naming.rebind(bindUrl, this);
```

This 'RMI Server setup' can actually be defined outside of the RMI Server class at run-time, however this means a user must supply the codebase, security policy and security manager when running the RMI server class, however it has been coded into the RMI server class here for ease. In order for this to work, the RMI server class must also supply an RMI registry to use, or create a new one, which has been accomplished using the following code:

```
try {
        registry = LocateRegistry.createRegistry(serverOptions
                        .getRegistryPort());
} catch (RemoteException e) {

        ...
}
```

The code marked in yellow calls a method in a separate class that has been created that returns a port to use for the RMI server. As the RMI preferences have been hard coded into the RMI Server class, it doesn't give the user much of a chance to change these preferences easily. Therefore the 'ServerIOServices' class (serverOptions) has been created. This class is used to read in data from a 'config' text file, in which a user can easily change lines of text in a file to easily change the RMI server preferences or settings, in which this data will be used at run time.

This ‘config’ file is named ‘ServerOptions.txt’ and is located within the root folder in each project. Its contents look like the following:

```
WARNING -- change the variables but do NOT change anything else
-
RMI Address:
//127.0.0.1:1099/RMIServer
-
Registry Port:
1099
-
Store File Name:
data/StudentRecords.dat
-
```

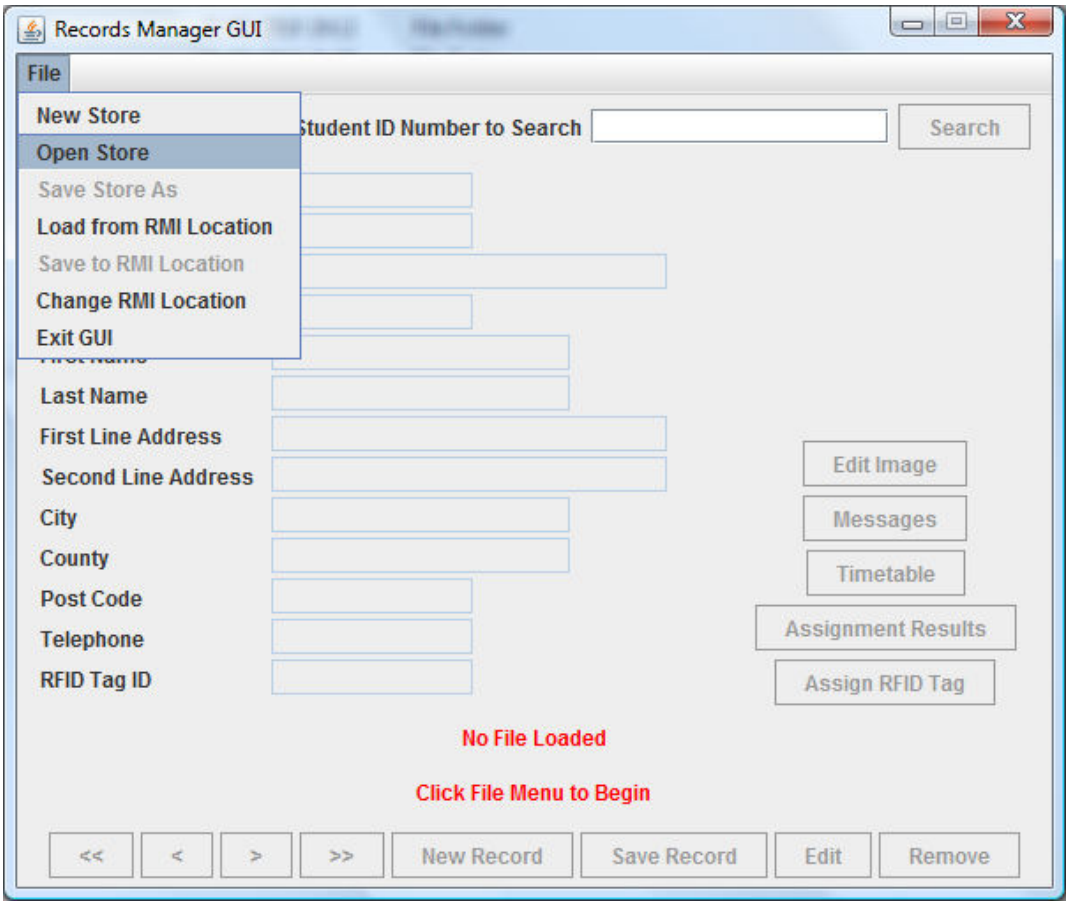
RMI technology, as it turns out is rather sensitive when it comes to defining and using preferences and can often result in a lot of error throwing when trying to get it to work properly, and these errors tend to be very un-informing as they provide little in terms of clues to what caused an exception to be thrown. The clues that are found tend to be the roughly the same for each kind of error, therefore as much information gathering techniques have been created (and the same for the RMI clients) in order to figure out what is causing an exception to be thrown, which involve many try catch statements, which then can be used to inform the user or programmer so they can make the necessary changes required for the RMI server to work. I.e:

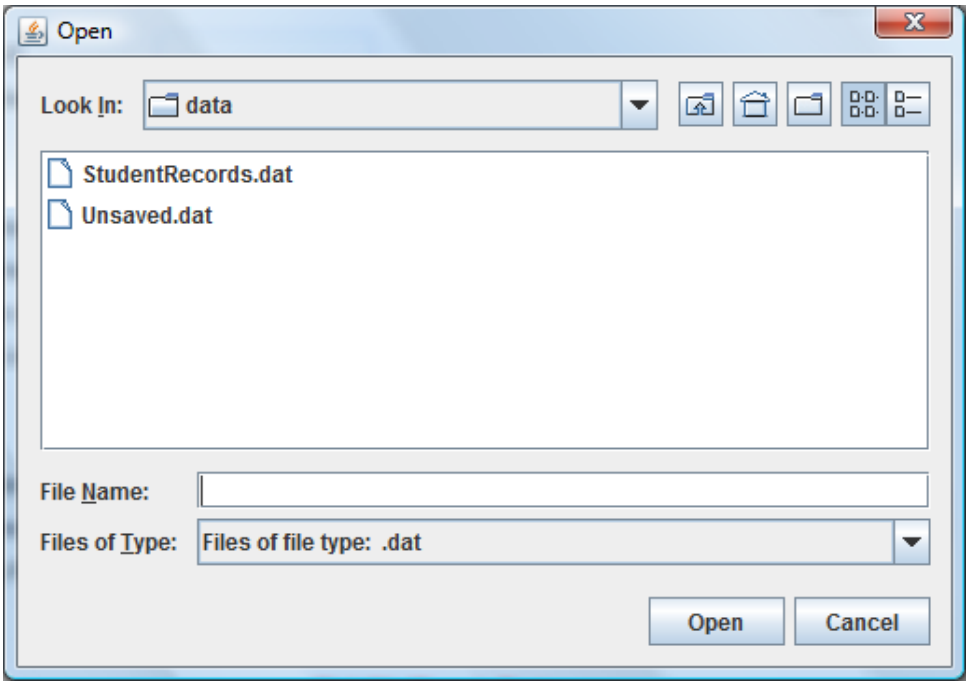
```
private String getRegistryDetails() {
	String registryDetails = "";
	registryDetails += "\nChecking registry details...\n";
	registryDetails += "Currently using host: " + bindUrl + "\n";
	registryDetails += "Checking host naming binding...\n";
	try {
		registryDetails += "..." + Naming.lookup(bindUrl).toString() + "\n";
	} catch (MalformedURLException e) {
		registryDetails += "Error: Wrong url syntax\n";
		e.printStackTrace();
		return registryDetails;
	} catch (RemoteException e) {
		registryDetails += "Error: Some kind of remote connection failure\n";
		e.printStackTrace();
		return registryDetails;
	} catch (NotBoundException e) {
		registryDetails += "Error: The RMI Server stub has not bound
                                successfully\n";
		e.printStackTrace();
		return registryDetails;
	}
		.
		.
		.
}
```

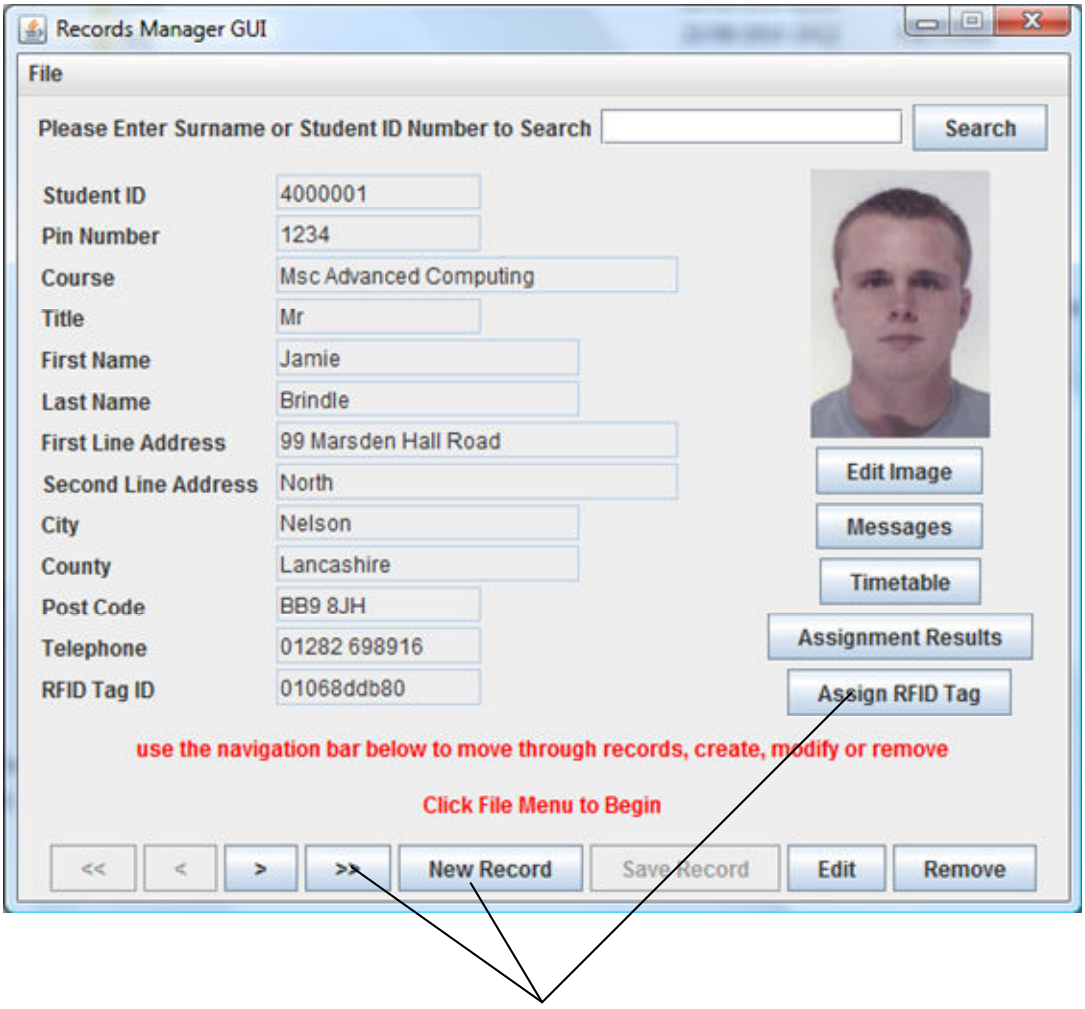
The information that is gathered is would then be displayed on the Server Logger (the user interface for the RMI server) in a more readable and understandable form

Student Records Administration Interface

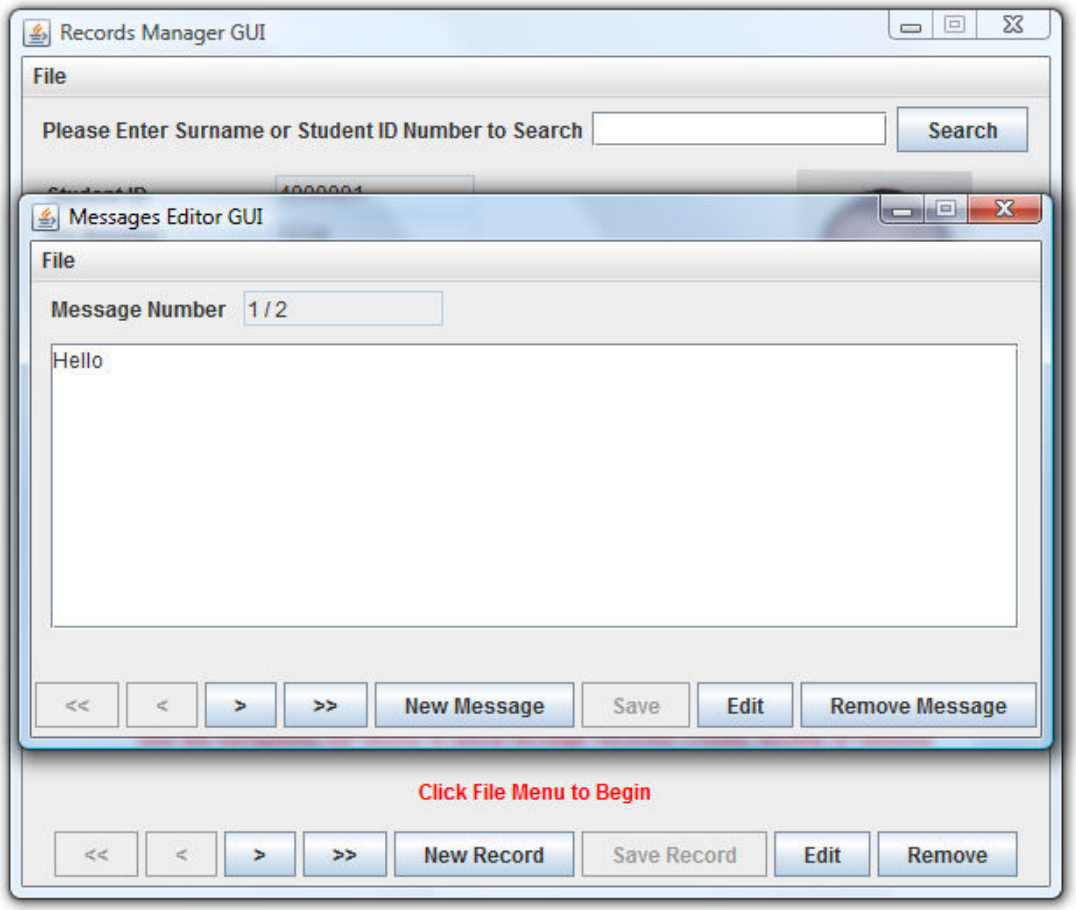
Screenshots

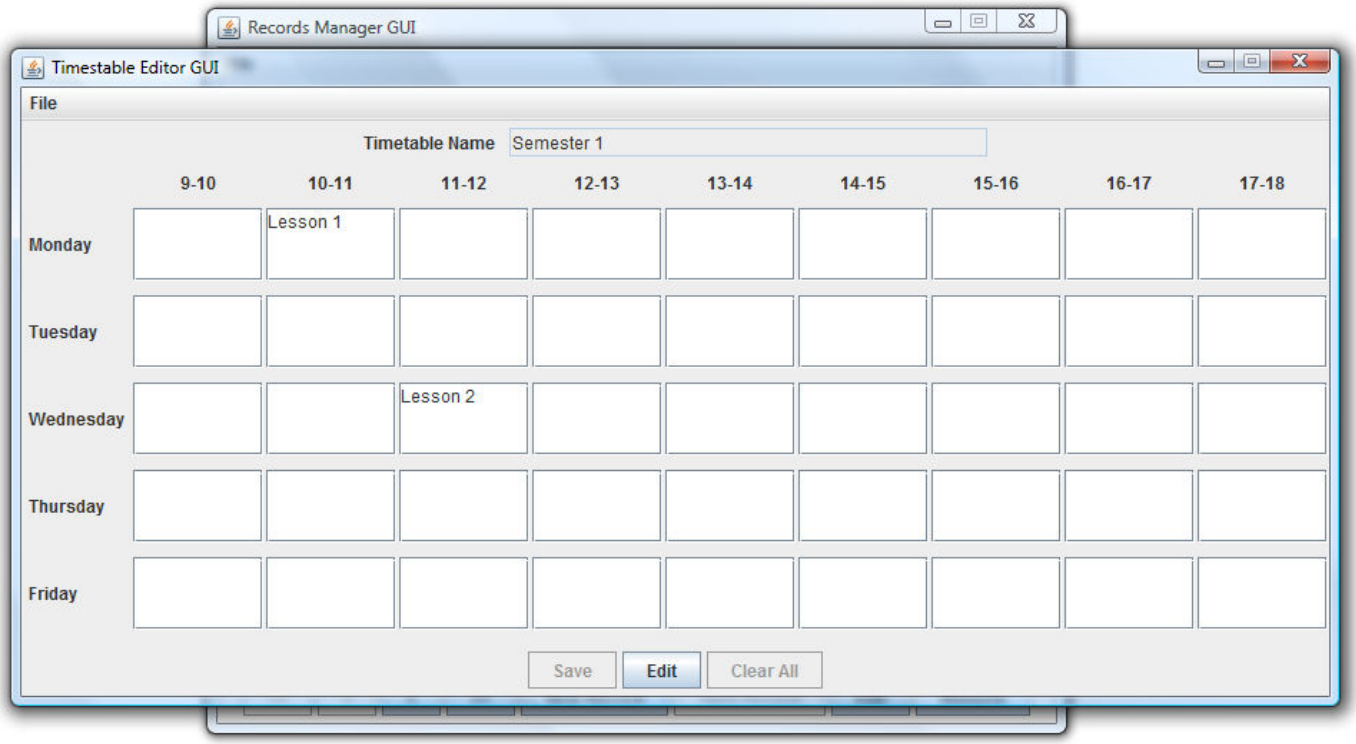


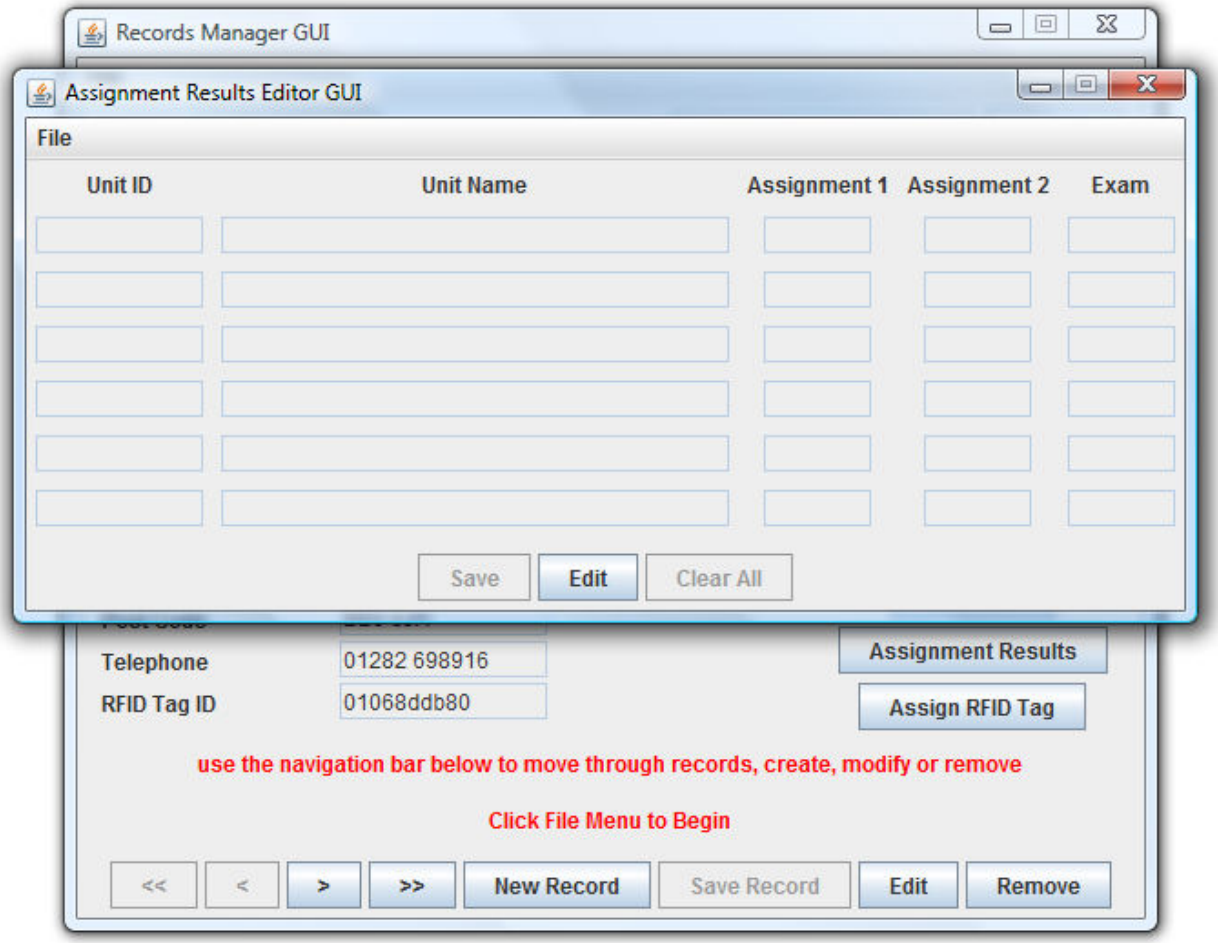


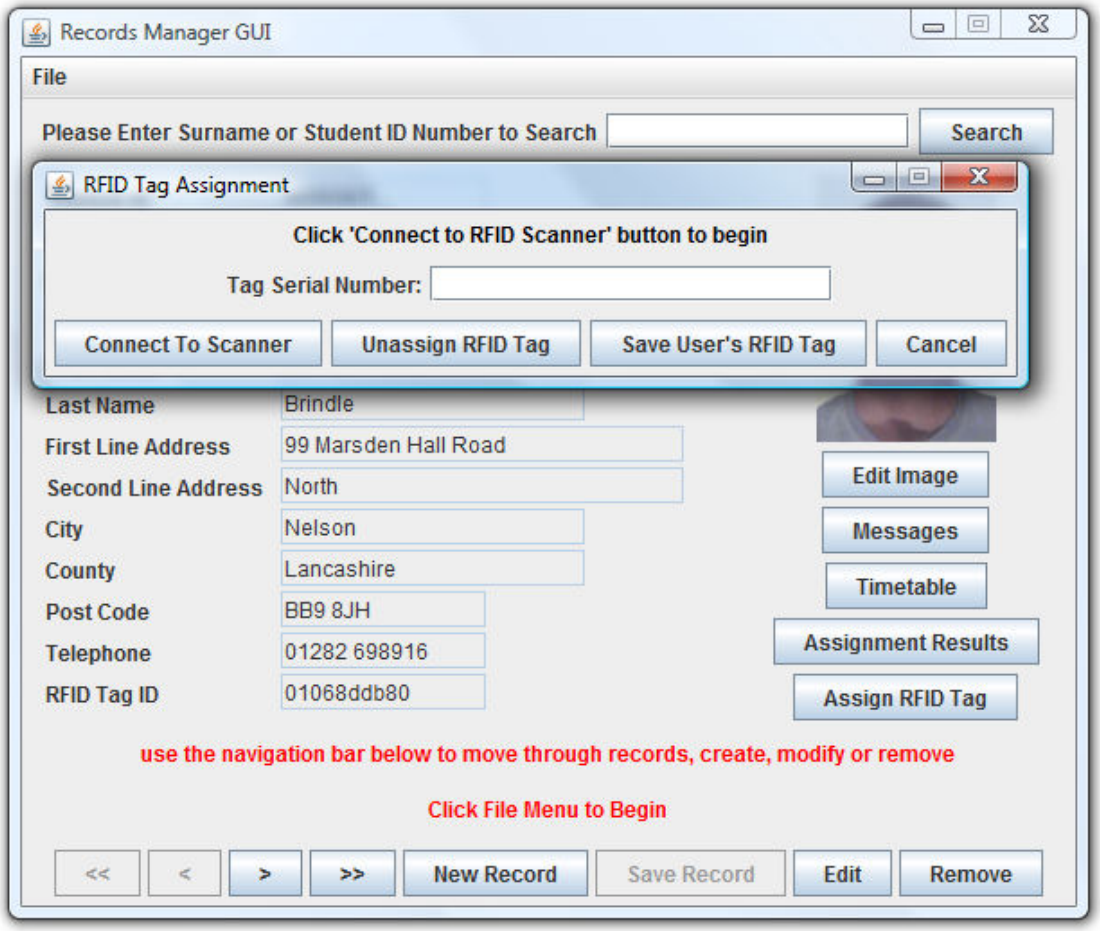


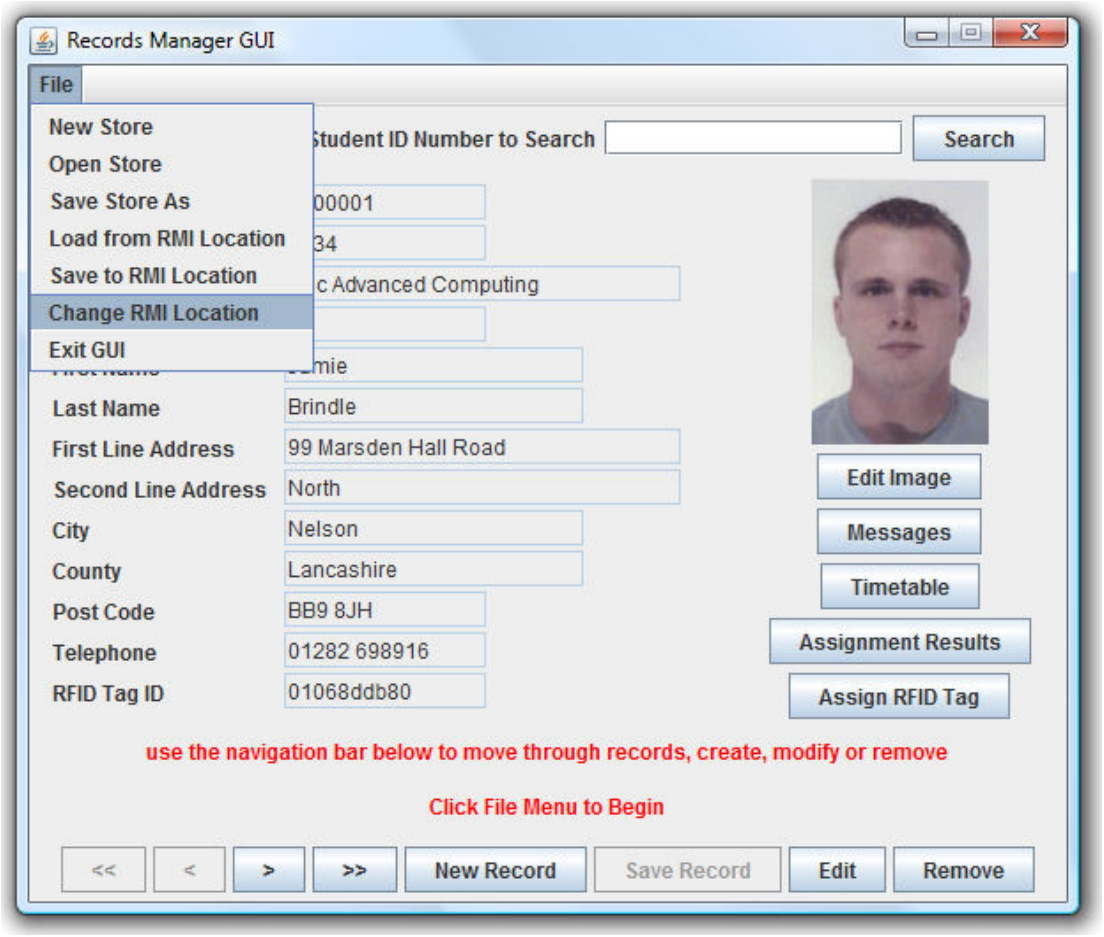
Certain buttons become enabled/disabled
depending On what task a user is performing

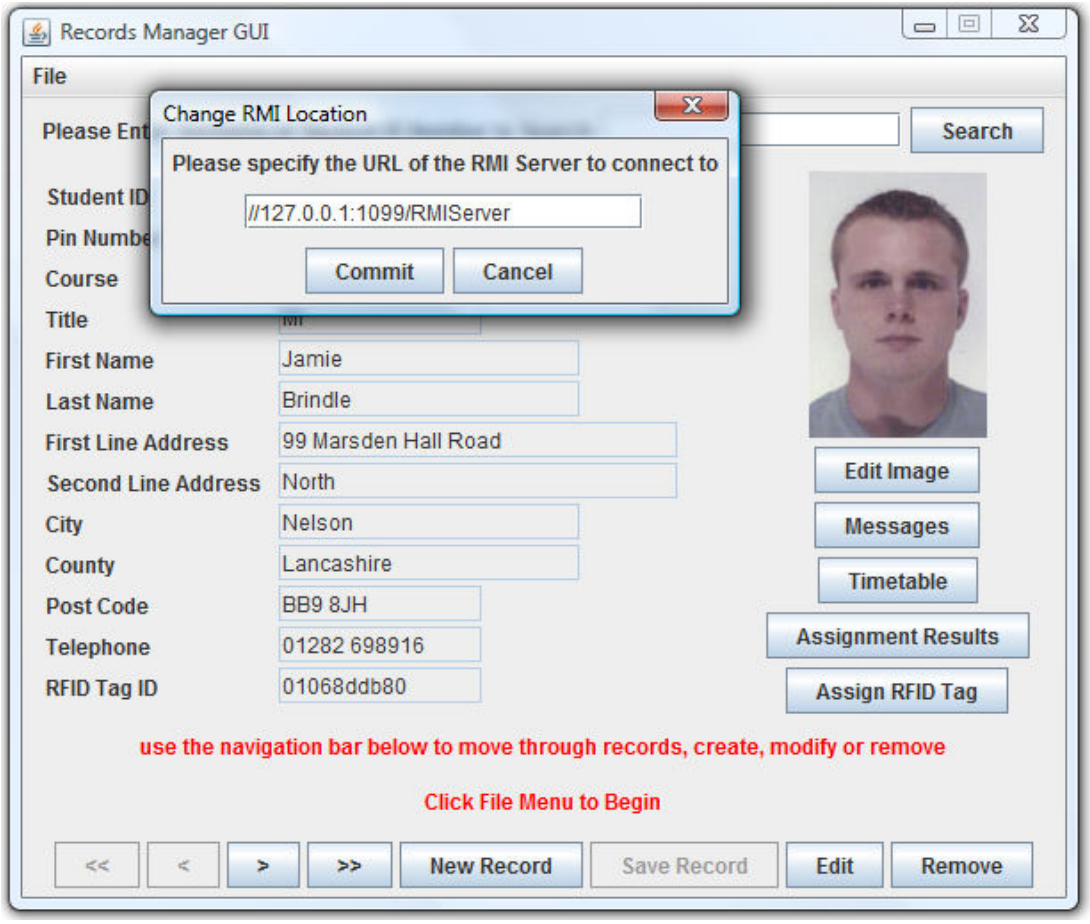

Records Manager GUI
File
Please Enter Surname or Student ID Number to Search
Search
Messages Editor GUI
File
Message Number
1 / 2
Hello
<<
<
>
>>
New Message
Save
Edit
Remove Message
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove


Records Manager GUI
Timestable Editor GUI
File
Timetable Name
Semester 1
9-10
10-11
11-12
12-13
13-14
14-15
15-16
16-17
17-18
Monday
Lesson 1
Tuesday
Wednesday
Lesson 2
Thursday
Friday
Save
Edit
Clear All


Records Manager GUI
Assignment Results Editor GUI
File
Unit ID
Unit Name
Assignment 1
Assignment 2
Exam
Save
Edit
Clear All
Telephone
01282 698916
RFID Tag ID
01068ddb80
Assignment Results
Assign RFID Tag
use the navigation bar below to move through records, create, modify or remove
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove

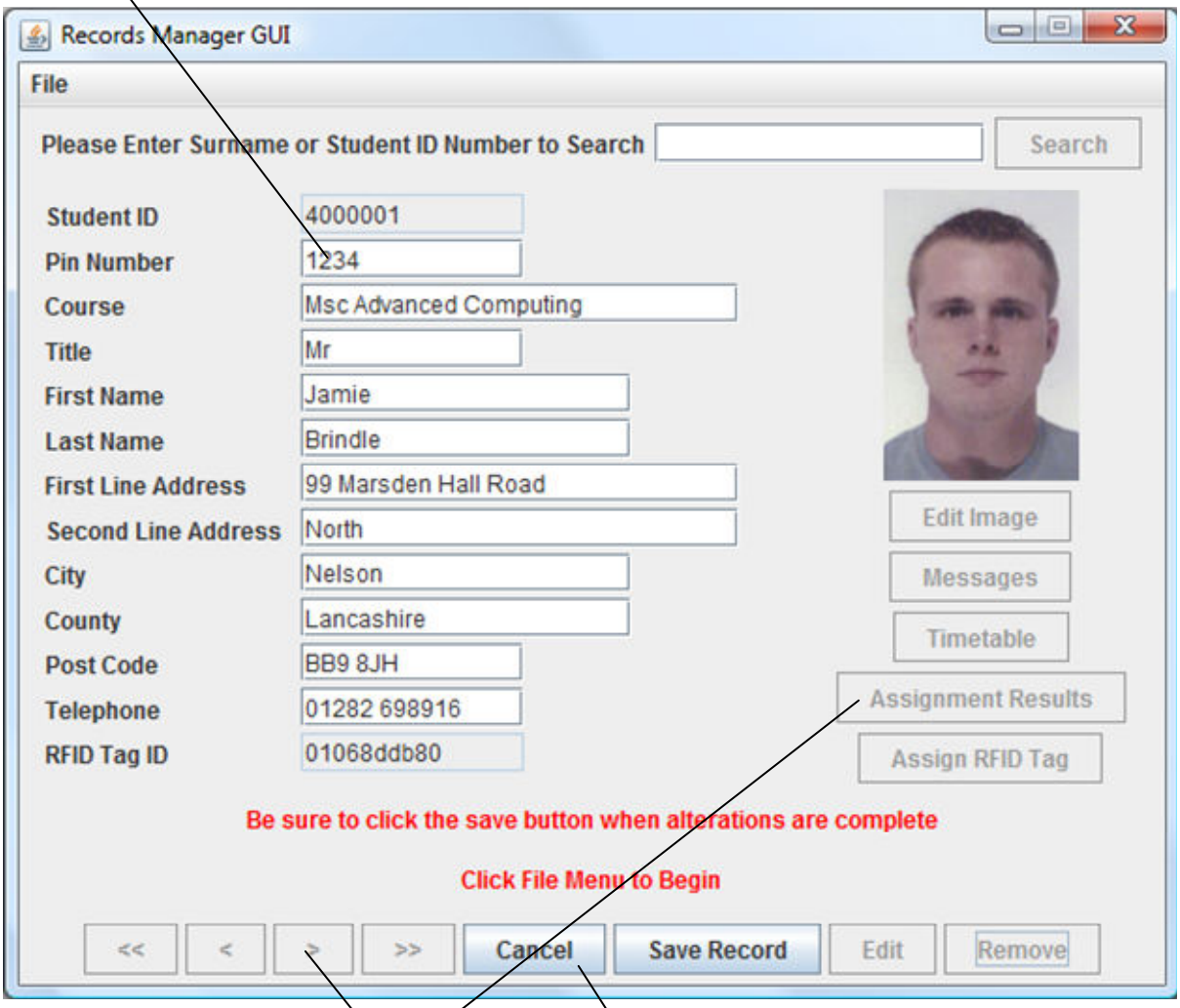

Records Manager GUI
File
Please Enter Surname or Student ID Number to Search
Search
RFID Tag Assignment
Click 'Connect to RFID Scanner' button to begin
Tag Serial Number:
Connect To Scanner
Unassign RFID Tag
Save User's RFID Tag
Cancel
Last Name
Brindle
First Line Address
99 Marsden Hall Road
Second Line Address
North
City
Nelson
County
Lancashire
Post Code
BB9 8JH
Telephone
01282 698916
RFID Tag ID
01068ddb80
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
use the navigation bar below to move through records, create, modify or remove
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove


Records Manager GUI
File
New Store
Open Store
Save Store As
Load from RMI Location
Save to RMI Location
Change RMI Location
Exit GUI
Student ID Number to Search
Search
00001
34
c Advanced Computing
Last Name
Brindle
First Line Address
99 Marsden Hall Road
Second Line Address
North
City
Nelson
County
Lancashire
Post Code
BB9 8JH
Telephone
01282 698916
RFID Tag ID
01068ddb80
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
use the navigation bar below to move through records, create, modify or remove
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove


Records Manager GUI
File
Change RMI Location
Please specify the URL of the RMI Server to connect to
//127.0.0.1:1099/RMIServer
Commit
Cancel
Search
Student ID
Pin Numbe
Course
Title
First Name
Jamie
Last Name
Brindle
First Line Address
99 Marsden Hall Road
Second Line Address
North
City
Nelson
County
Lancashire
Post Code
BB9 8JH
Telephone
01282 698916
RFID Tag ID
01068ddb80
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
use the navigation bar below to move through records, create, modify or remove
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove

Editing and Creating New Student record Objects:

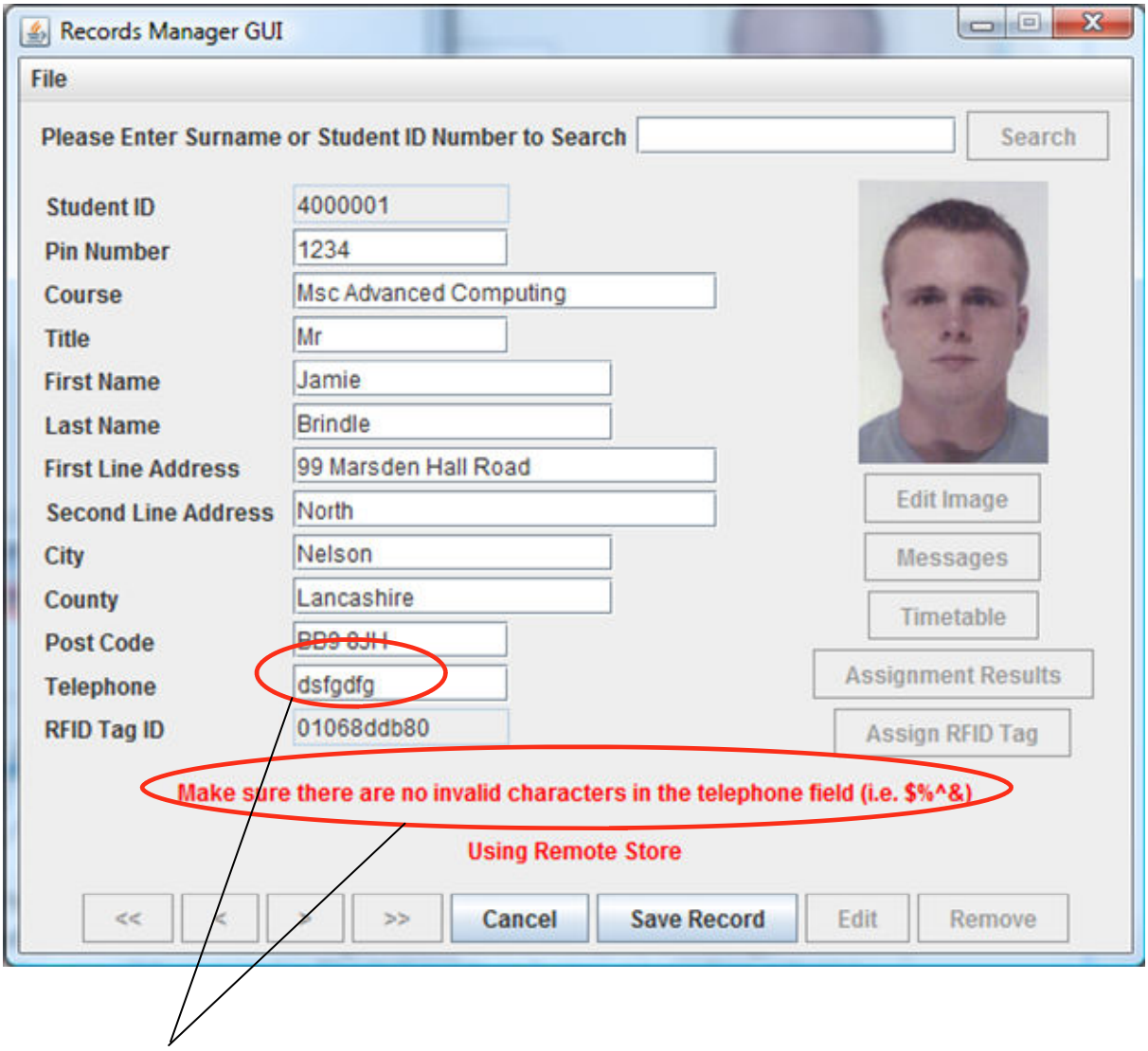
Text fields become editable



Particular buttons that you won't use when performing this task become disabled

'New Record' button becomes the 'cancel' button

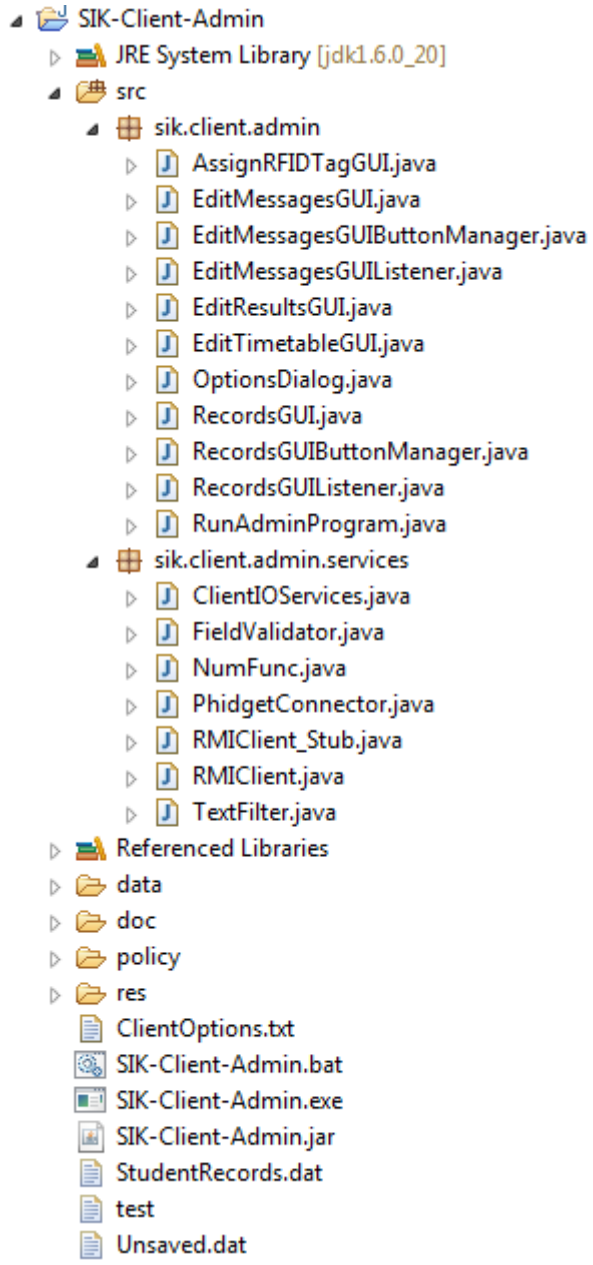
User input Validation:



Each text field is validated so when attempting to save a record that clearly contains errors it will prevent a user from saving a record and will display a message informing the user of their mistake. This helps reduce human error when entering this information.

The Workings

The Classes



The Student Records Manager Interface is divided into two separate packages. The *sik.client.admin* package and the *sik.client.admin.services* package. The sik.client.admin package contains the GUI class which are used to display the Student Records Manager Interface and to interact with the user.

As these files can get quite large to containing a large amount of elements and objects, some of the work done by each class is passed onto another class. For example the 'RecordsGUI' would contain all the template elements of the GUI and it passes on the user interaction handling to the 'RecordsGUIListener' class, which basically contains action event handlers. This class is also quite large, and it too passes some of the work to another class, the 'RecordsGUIButtonManager' which deals with managing the attributes of the GUI's button when performing certain tasks.

The 'RecordsGUI' is the main GUI for the interface. The other GUI's (AssignRFIDTagGUI, EditMessageGUI, EditResultsGUI, EditTimetableGUI) are 'sub GUIs' which are navigated to from the RecordsGUI. These GUI's are referenced to the RecordsGUI.

The 'sik.client.admin.services' package contains classes which act as services for the GUI classes which are less tightly coupled with one another. The 'ClientIOServices' class deals with importing and exporting data to and from a text file. This class is very similar to the 'ServerIOServices' class in the RMI Server project. The text file is basically a 'config' file, which a user may amend to specify preferences of the Student Records Manager interface, such as the RMI server location. This allows particular preferences to be defined at run time rather than them being hard coded into a class.

The 'FieldValidator' class is used by, but not referenced to the RecordsGUI class which reads in a String and returns its validity apparent to the context of the string. For example a method 'validatedTelephoneNumber(String string)' which check the characters in the string and return a boolean value 'true' or 'false' if the characters in the String conform to that of a telephone number. This class is most often used to validate what a user has entered into the RecordsGUI's text field are correct in order to reduce human error when entering the details for a student record.

The ‘FieldValidator’ works closely with another class. The ‘NumFun’ class. NumFun stands for ‘number functions’. That is it provides a number of functions to work with number. This class is quite often used by other classes in the Student Records Manager interface. The functions it contains are ‘static’ methods which often convert the parameter fed into the method into a different type and return the original parameter, except in a different type. For example:

```
public static long stringToLong(String aString) {
	return Long.parseLong(aString);
}

public static String numberToString(double aDouble) {
	return Double.toString(aDouble);
}
```

The fact that these methods are ‘static’ means they can be called without having to create an instance of the NumFun object. Once would typically write ‘NumFun.numberToString(“aString”);’ in order to utilise that method.

These types of methods are useful when working with elements in the GUIs such as text fields, in which the text values they contain are always in string format, and some values stored in the student record object are integers. So often it is necessary that conversions take place. Additionally when performing a calculation of numbers, if they are in string format they needed to be converted first.

The NumFun also contains other methods which return the validity of an input according to its context. For example, the below method would validate if a string input contains only numbers:

```
public static boolean containsOnlyNumbers(String aString) {
	if ((aString == null) || (aString.length() == 0)) {
		return false;
	}

	for (int i = 0; i < aString.length(); i++) {

		if (!Character.isDigit(aString.charAt(i))) {
			return false;
		}
	}

	return true;
}
```

The ‘TextFilter’ class is merely a simple class to work with Swing’s standard ‘JFileChooser’, which is used to view the files on a file system. The TextFilter class simply tells the file chooser to only show files of a particular type. I.e. .jpg

The ‘PhidgetConnector’ class provides services for the RFID scanner hardware, which basically contains a list of particular listeners and listener event handlers for the RFID scanner. The RFID scanner is basically defined as an object in which certain listeners are assigned to it, then event handlers can manage what to do once that RFID scanner object changes its state. For example, the below method defines what to do once the RFID scanner retrieves an RFID tag signal, in which it calls a method to set a text field to the tag serial number it has gained:

```
public void tagGained(TagGainEvent e) {
	tagID.setText(e.getValue());
}
```

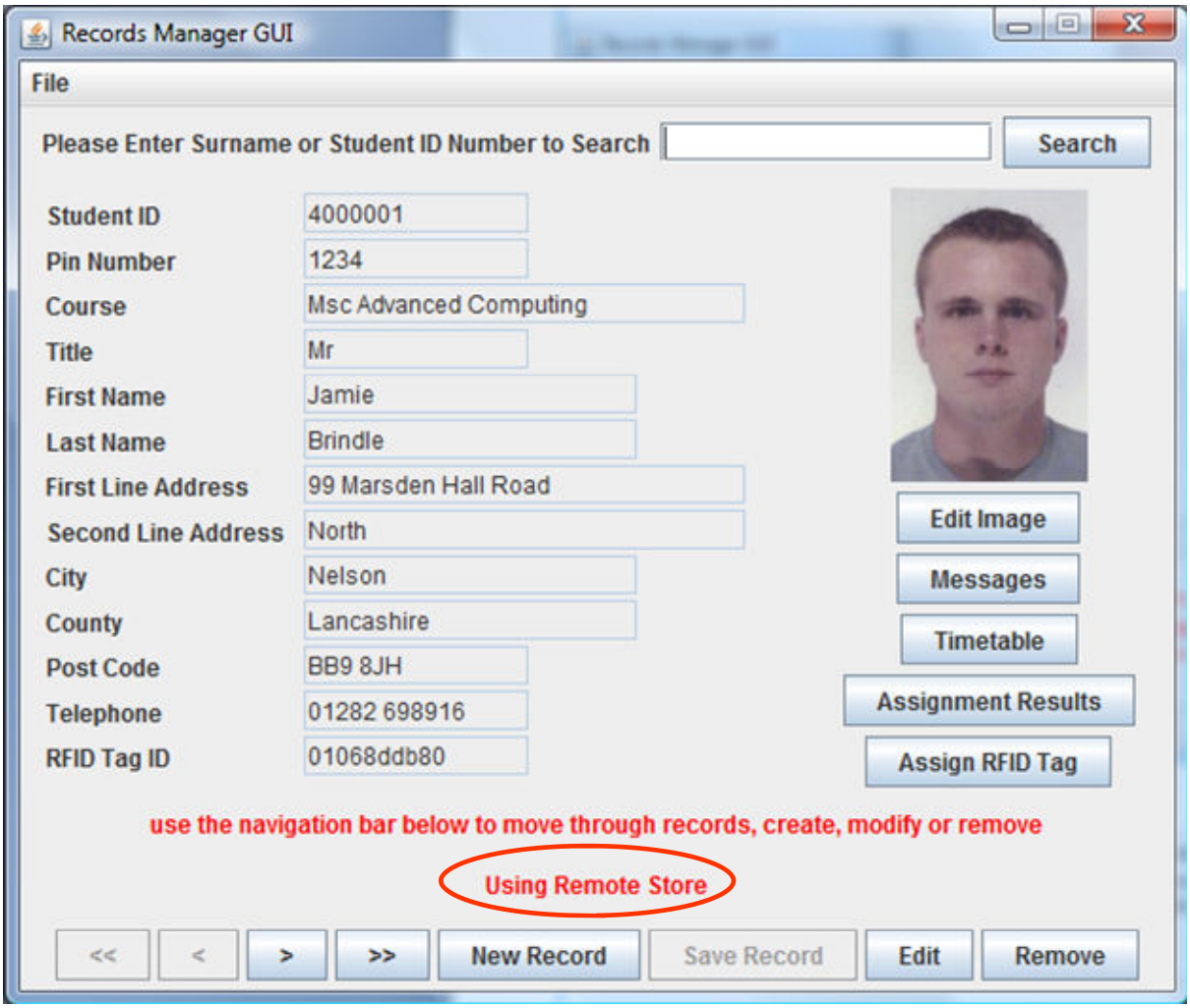
The ‘RMIClient’ class deals with invoking methods on the RMI server and is responsible for storing a retrieved student object store. In order for the RMI client to connect to the RMI server and utilise its methods, it must make a ‘lookup’ to a remote RMI registry and look for the RMI server object, in which is then cast (converted) to an RMI Server interface which is local:

```
rmiServer = (RMIServer) Naming.lookup(bindName);
```

The RMI client, like the RMI server must also provide a security policy and security manager:

```
if (System.getSecurityManager() == null) {
        System.setProperty("java.security.policy", "policy/AllSecurity.policy");
        System.setSecurityManager(new RMISecurityManager());
}
```

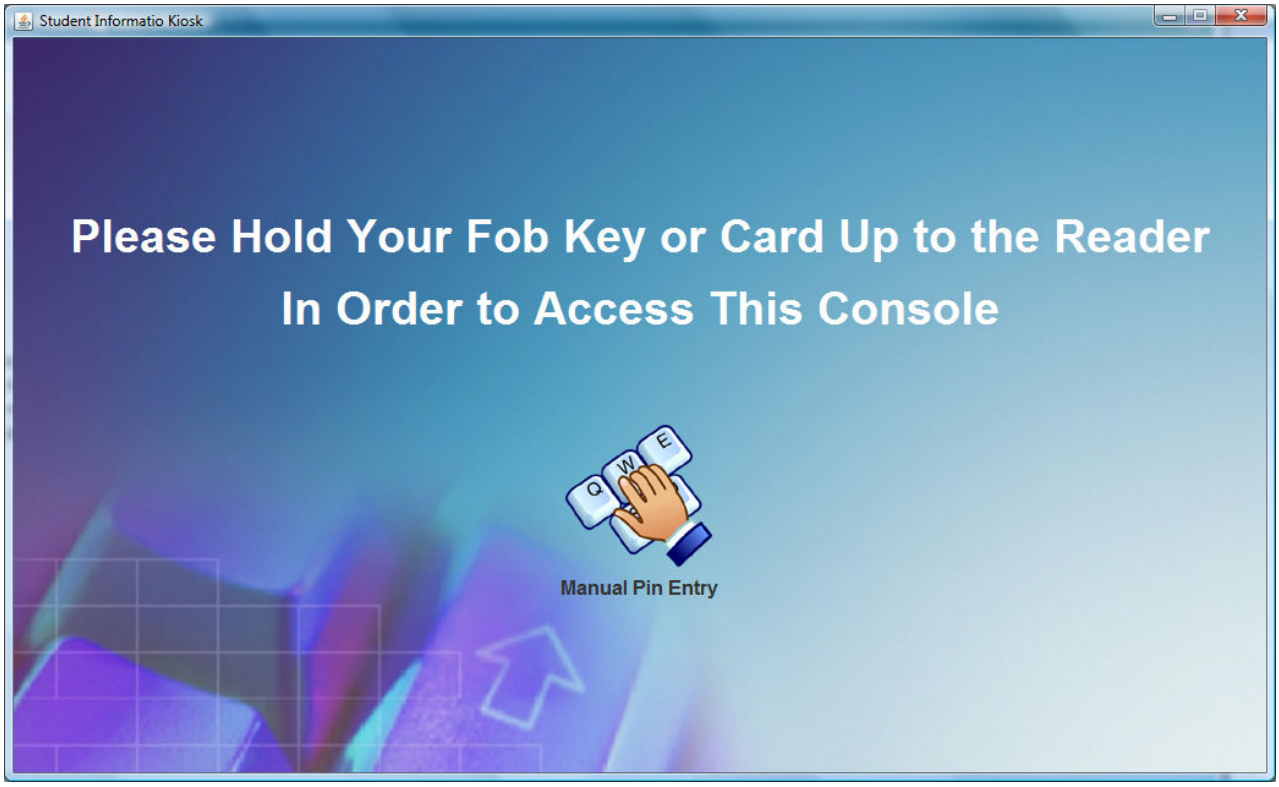
The Student Records Manager Interface does rely and making a connection to the RMI server in order for it to work, as previously discussed, it can either work in standalone mode or RMI mode. When in standalone mode, the Student Records Manager Interface can work with a local student objects store file. It only uses the RMI client to load and export a student objects store from and to a remote location. The Student Records Managers Interface will remind a user if they are dealing with a remote or local store as a message at the bottom of the interface:



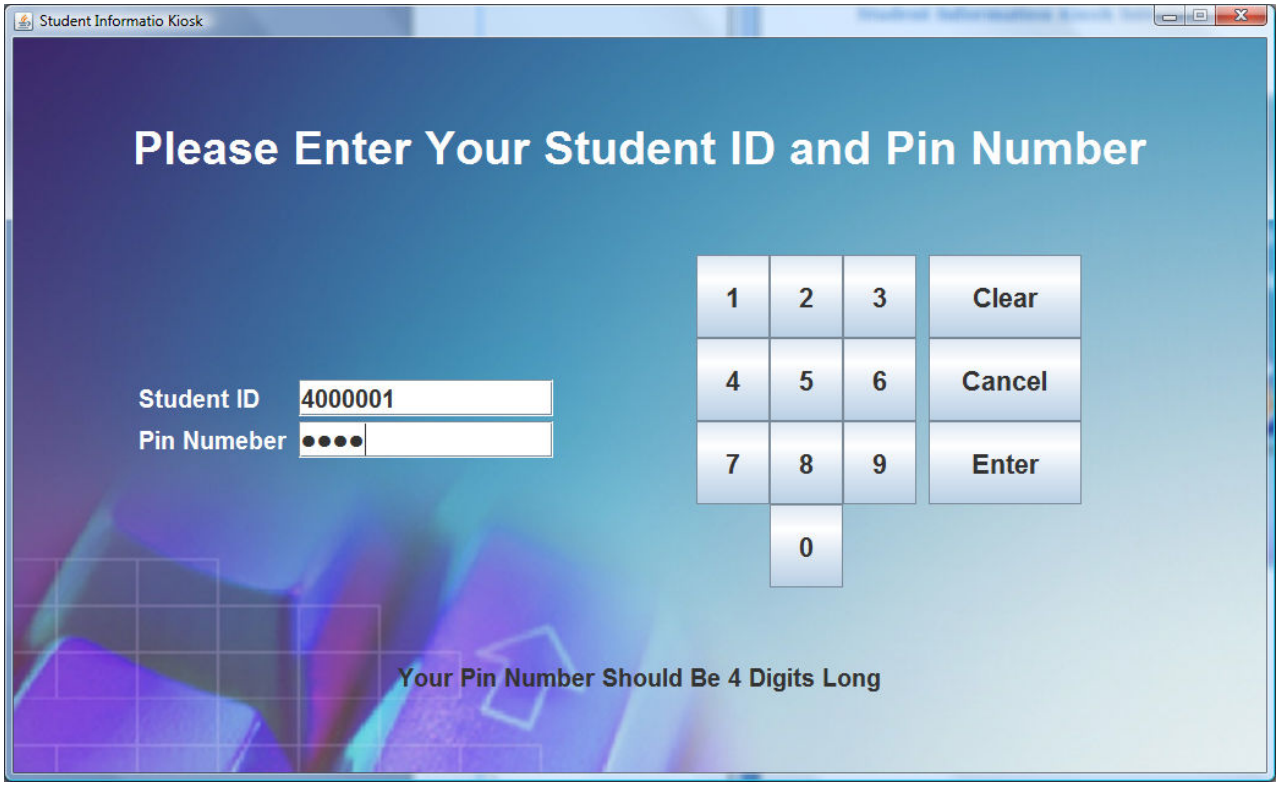
Student Information Kiosk Interface

Screenshots

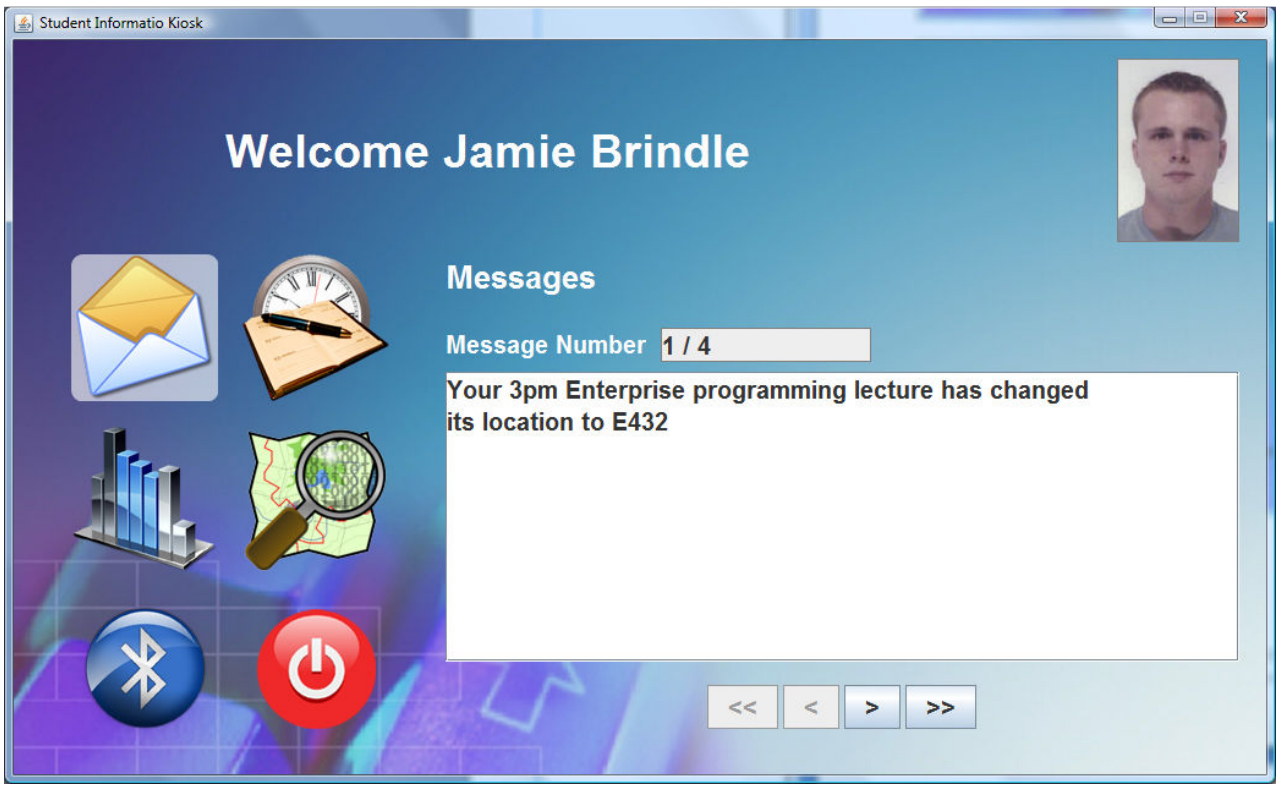
The Introduction Panel:



The Manual Pin Entry Panel:

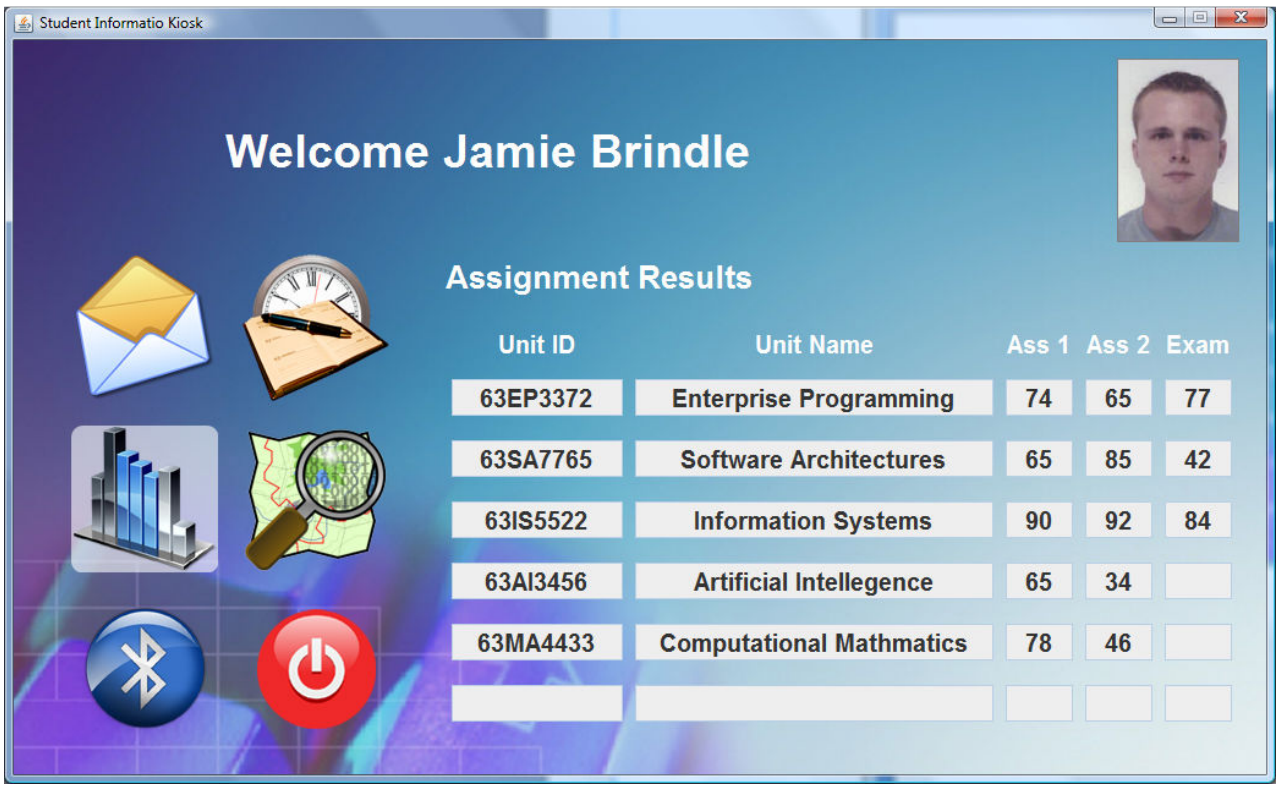


The Home Panel: Displaying the messages pane:

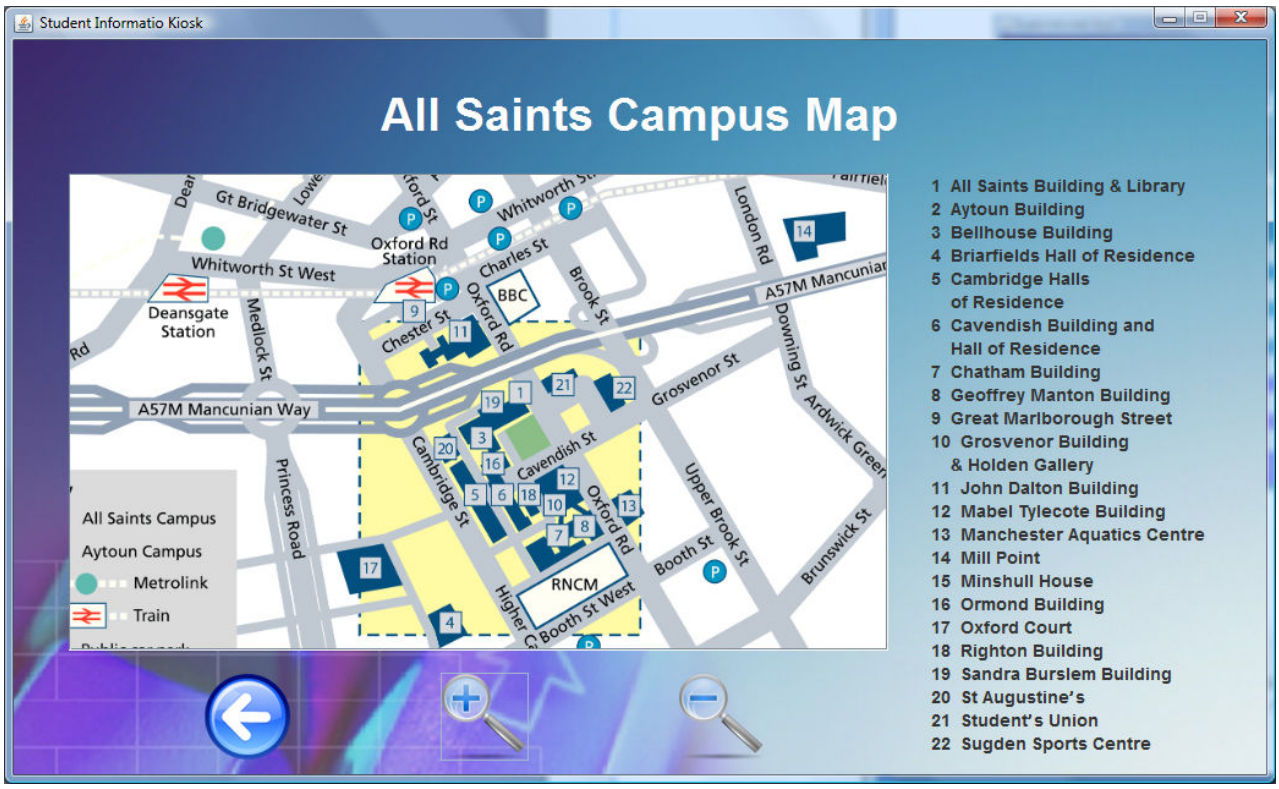


The Home Panel: Displaying the timetable pane:

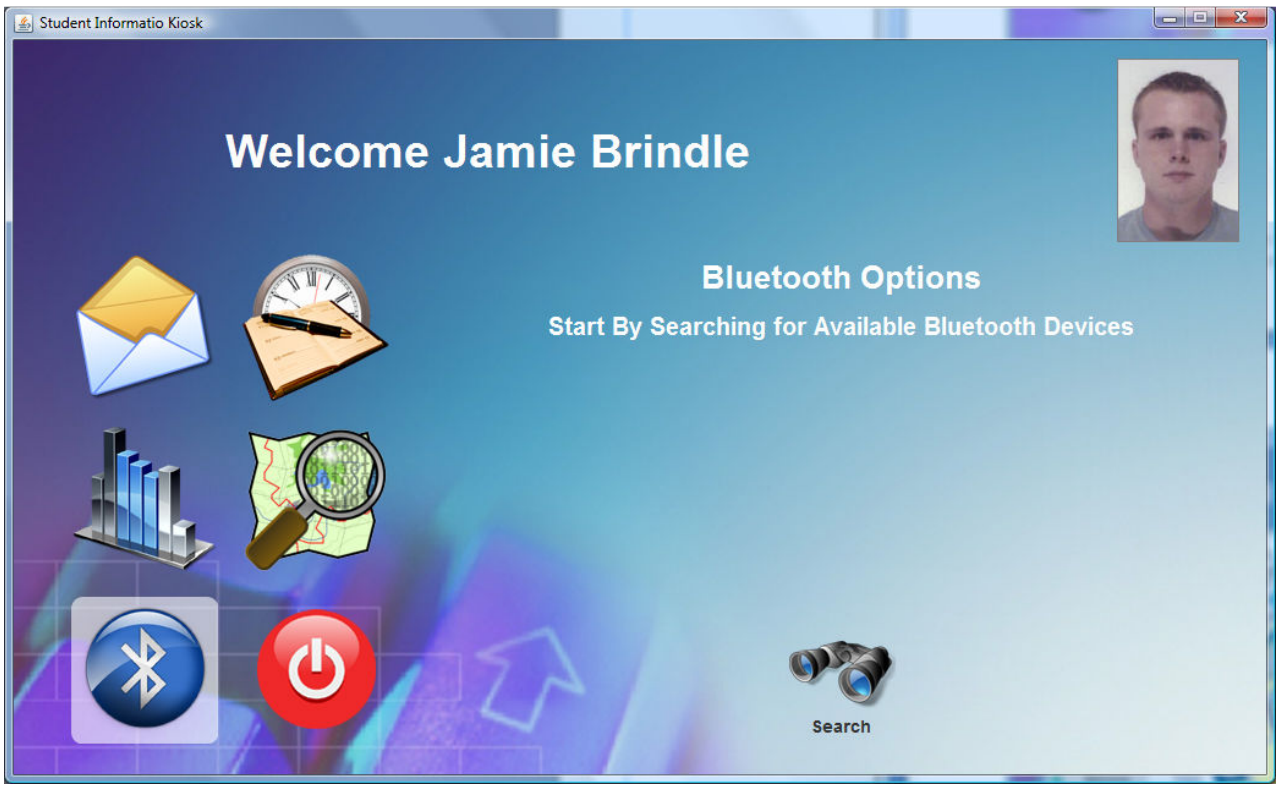
The Home Panel: Displaying the assignment results pane:



The Campus Map Panel:



The Home Panel: Displaying the Bluetooth options pane (no devices discovered)

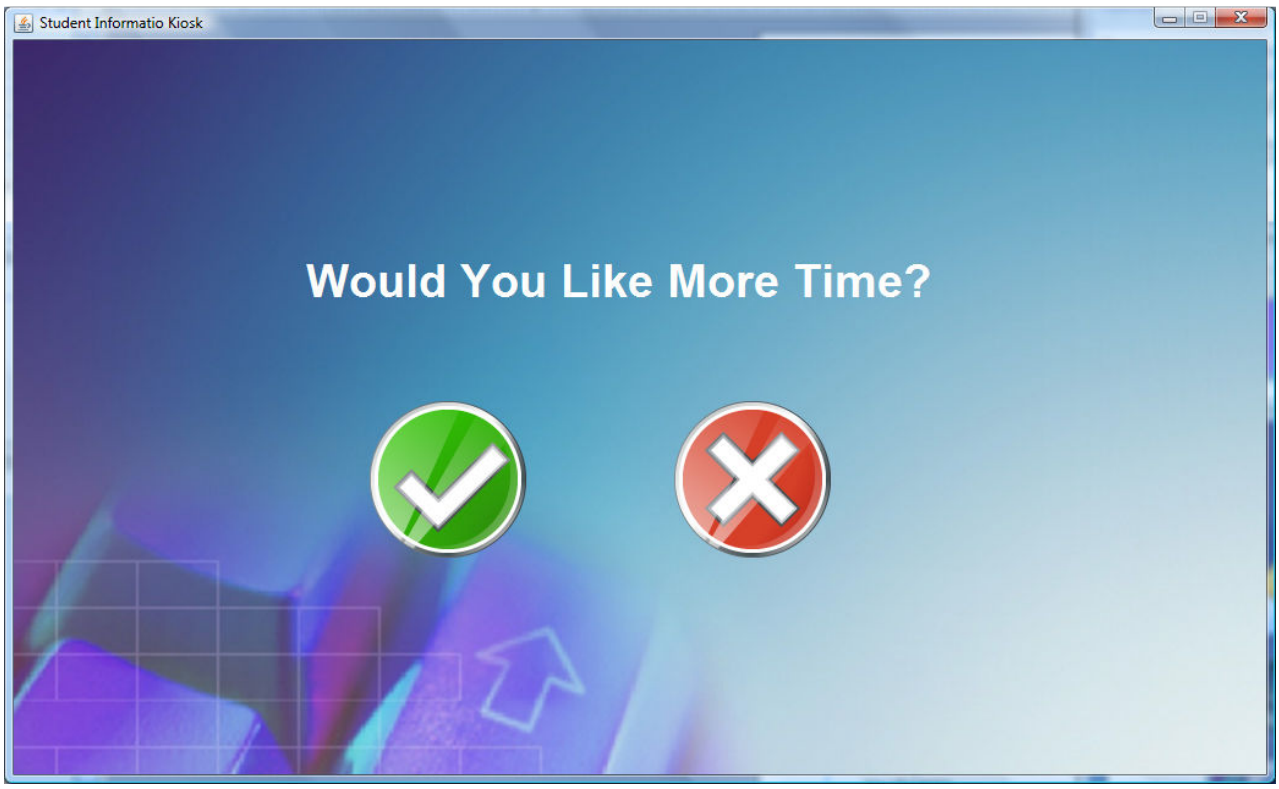


Home Panel: Displaying the blue options pane (device discovered)

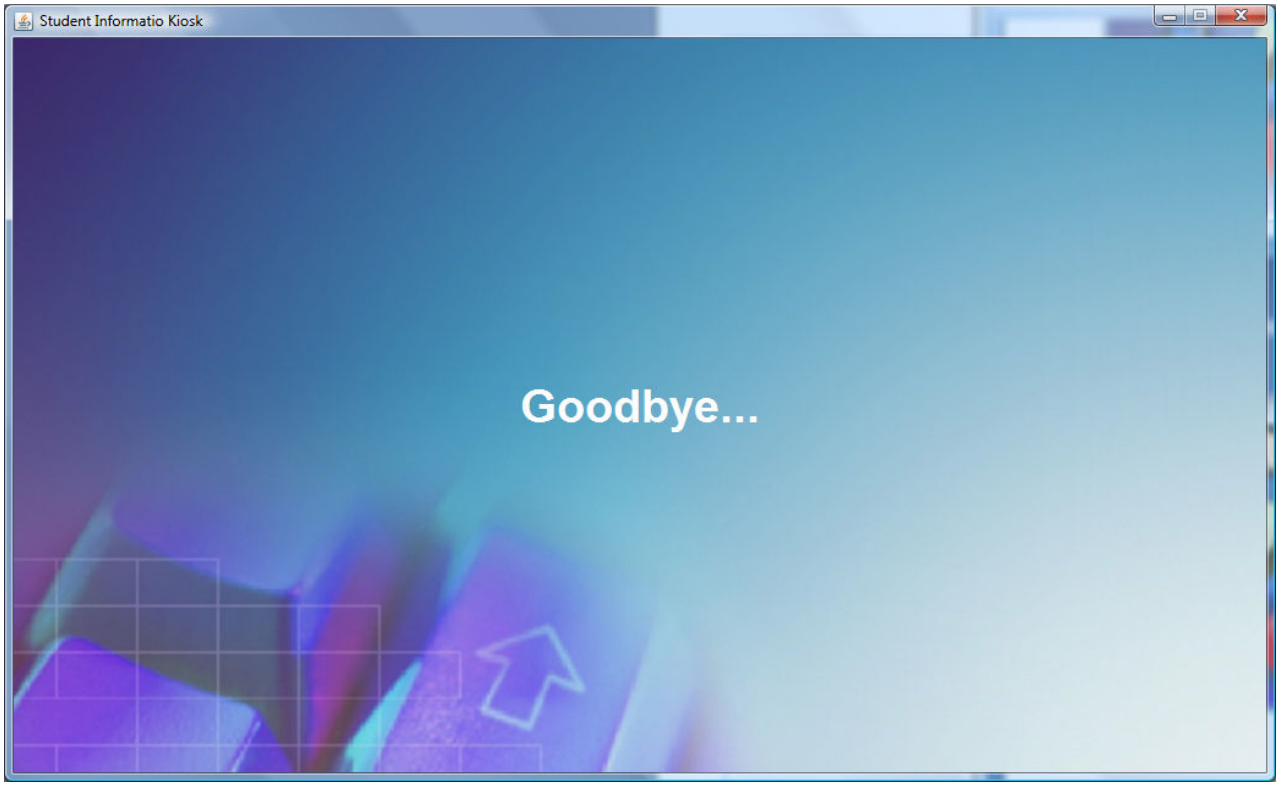
Home Panel: Displaying the Bluetooth options pane (when device to use is selected)

Home Panel: Displaying the Bluetooth options pane (Sending a file)

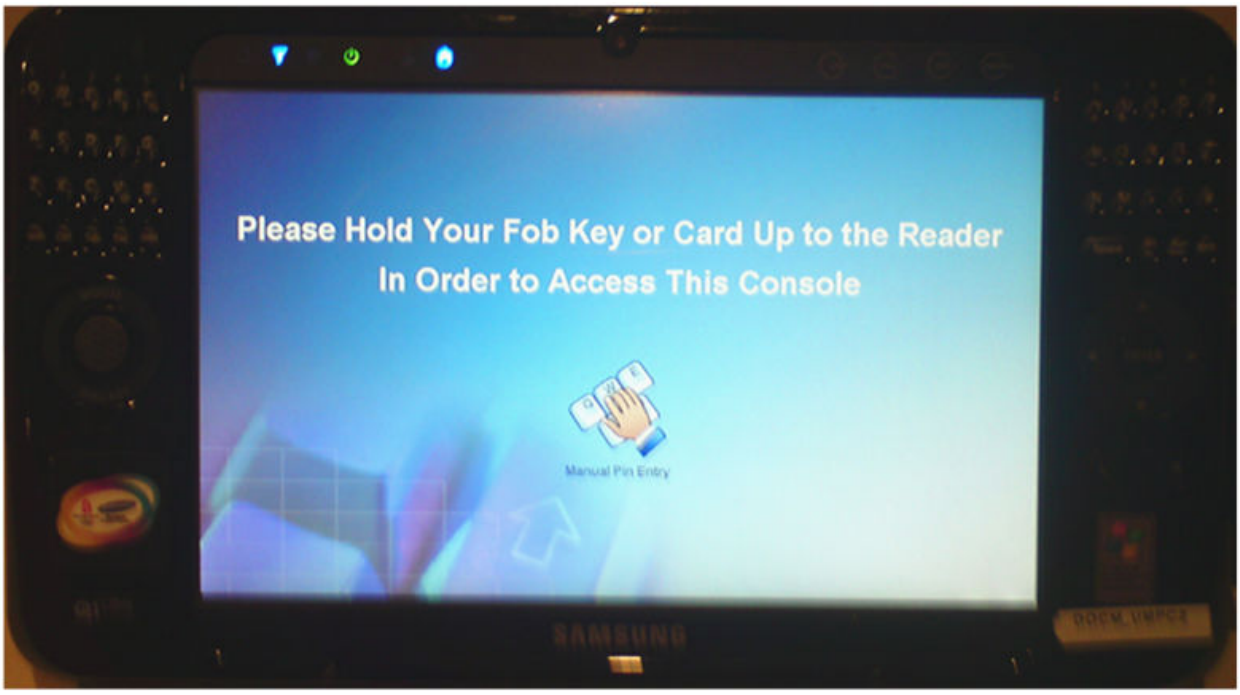
Timeout Panel: When there's no user activity for a set amount of time



Goodbye Panel: When the user wishes to end a session

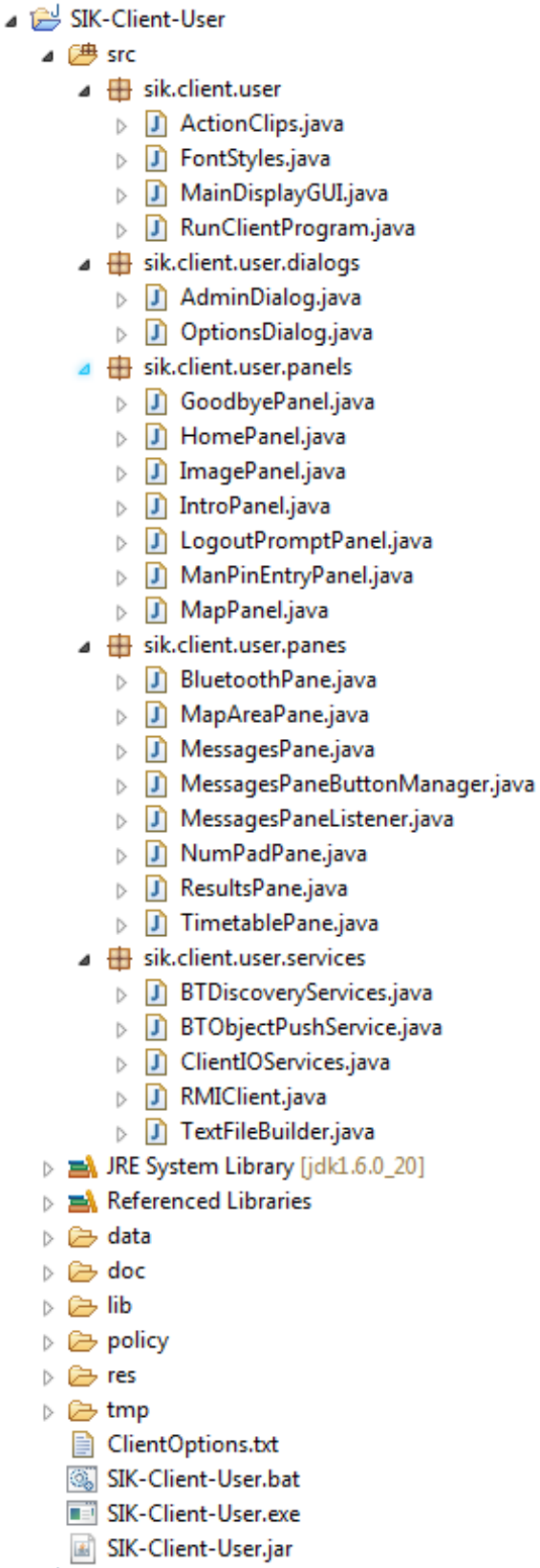


Student Information Kiosk on the Samsung Q1 Tablet



The Workings

The Classes



The Student Information Kiosk Interface is split 5 packages, which group together classes of a similar nature.

sik.client.user Package

This package contains classes which are used throughout the Student Information kiosk project, with the exception of the sik.client.user.services package. It contains the main display GUI (MainDisplayGUI) which is the main container of all the corresponding panes, panels and dialogs. It is responsible for loading in and out (displaying and hiding) of the panels, in which a user would navigate through.

This package also contains the 'ActionClips' class, which is contains a list of static methods for playing audio files, which is part of the interaction process of the GUI. For example, when a user clicks a button, the button event handler will call a method in the ActionClips class to play a particular sound file, in which informs the user that button has been pressed and provides a modern appeal to the GUI.

The 'FontStyles' class is similar to the ActionClips class in the fact that it contains a list of static font variable styles to be used by the GUI. As there are a number of font styles used within the GUI, it's beneficial to keep them organised and put into one place and this provides a central access point to these fonts. This also allows a font style to be easily changed if desired, without having to search for a single line of code in a large class. This technique has been used numerous times in the projects. This makes the project easily maintainable and it becomes easier for future development.

sik.client.user.dialogs Package

Simply contains dialog classes. This dialogs are used as pop-ups for the GUI which can show a user a message or a warning, or can be used as a password input pop-up. In fact this dialogs have been designed to provide multiple functionality and are re-used quite often in this project and the Records Management Interface project.

The OptionDialog class method would typically be called using the following call:

```
JDialog dialog = new OptionsDialog(new JFrame(),
        "Title", "Message, "Type");
```

This creates a new OptionsDialog JDialog object with a particular title, display message and type of OptionsDialog. The type could be a message, warning, an option (yes or no) or a password field. In being able to re-define the OptionsDialog, it means that it can be re-used throughout the projects.

sik.client.user.panels Package
This package contains a number of JPanel object classes, which are contained within the MainDisplayGUI. Each panel contains a number of elements (objects) such as JTextFields, JLabels, JTextAreas, ImageIcons, JButtons and other JPanels. Each panel acts as a separate 'page' for the GUI, (like web pages), in which will be displayed to the user depending on their navigation through GUI. Having a number of separate panel classes allows them, and all their contained objects to be loaded in and out of view and to be called easily by various other classes.

The 'GoodbyePanel' class simply display a 'goodbye' message to a user at the end of their user session, the 'LogoutPromptPanel' class displays a message to the user, asking if they'd like more time in their current session, which would be called when there has been no user activity for a set amount of time. The 'ImagePanel' class is used by the MainDisplayGUI class to create a panel with a background image.

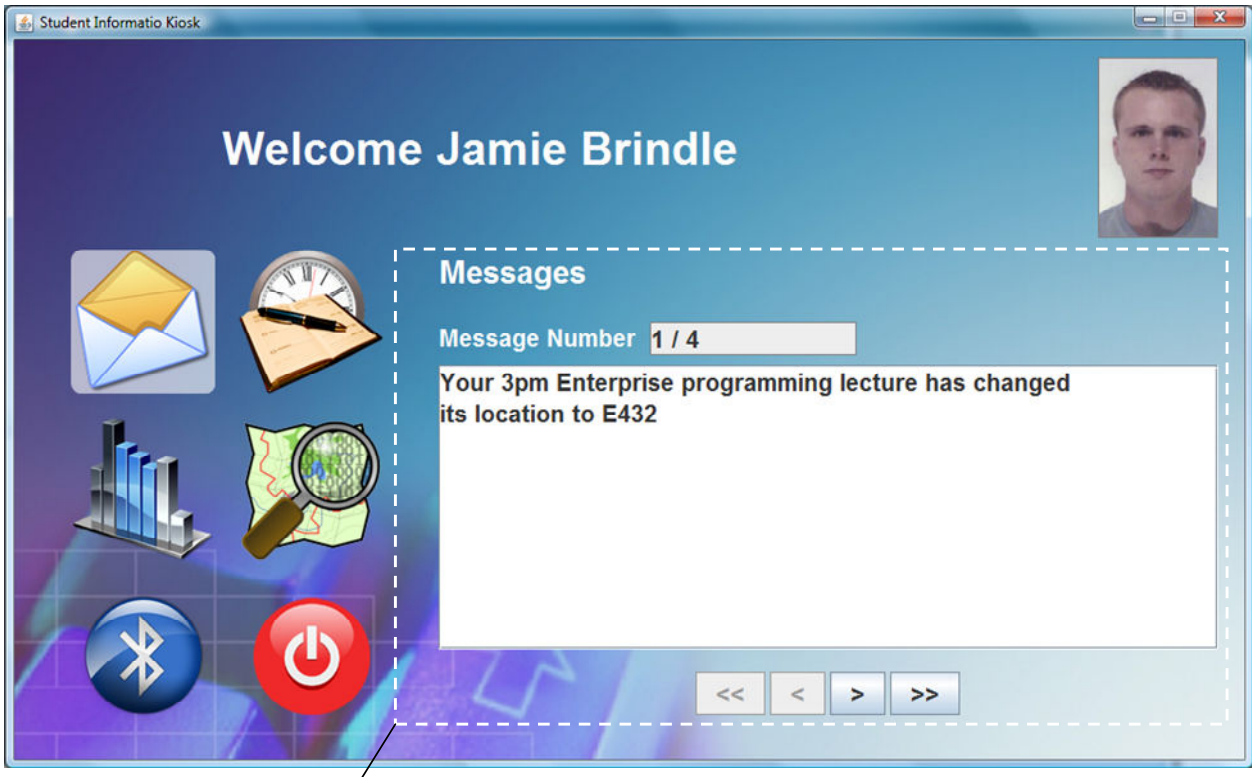
The 'MapAreaPanel' class displays a map and legend for the university campus map. The actual map itself is contained in a separate panel the 'MapAreaPane', which is located in the sik.client.user.pane package. This map image is displayed in a JScrollPane without the scroll bars and has been assigned mouse listeners in which aim to partly reproduce a map interaction style similar to that of the google maps app on a touch screen phone (as this system is displayed on a touch screen tablet), which enables a user to point to the map and drag the map around to navigate. This particular panel will be expended upon later in this report. See 'The Campus Map' implementation for more details.

The 'IntroPanel' class is an important class as it is the first and last panel to be displayed in the MainDisplayGUI. It also makes use of the Phidgets API so that the RFID scanner can be used as a method of logging into the Student Information Kiosk using an RFID tag. This class also makes a call to the 'RMIClient' class, which in turn communicates with the RMI server in order to log a user into the system and display student record details.

The 'ManPinEntryPanel' class also uses the RMIClient class to log a user into the system manually using their unique student ID number and pin code. This is an alternate means of logging into the system.

The 'HomePanel' class is also an important class in this project. It's the panel that is displayed to a user once a user has successfully logged into the system. This panel is similar to the MainDisplayGUI as it acts as a container for multiple other panels, which are named 'panes' (sik.client.user.panes package) in order to define a structure to the system. The HomePanel however, is still contained in the MainDisplayGUI. This panel is where a user would view their messages, timetable or results, view the campus map image and send items to their view via Bluetooth.

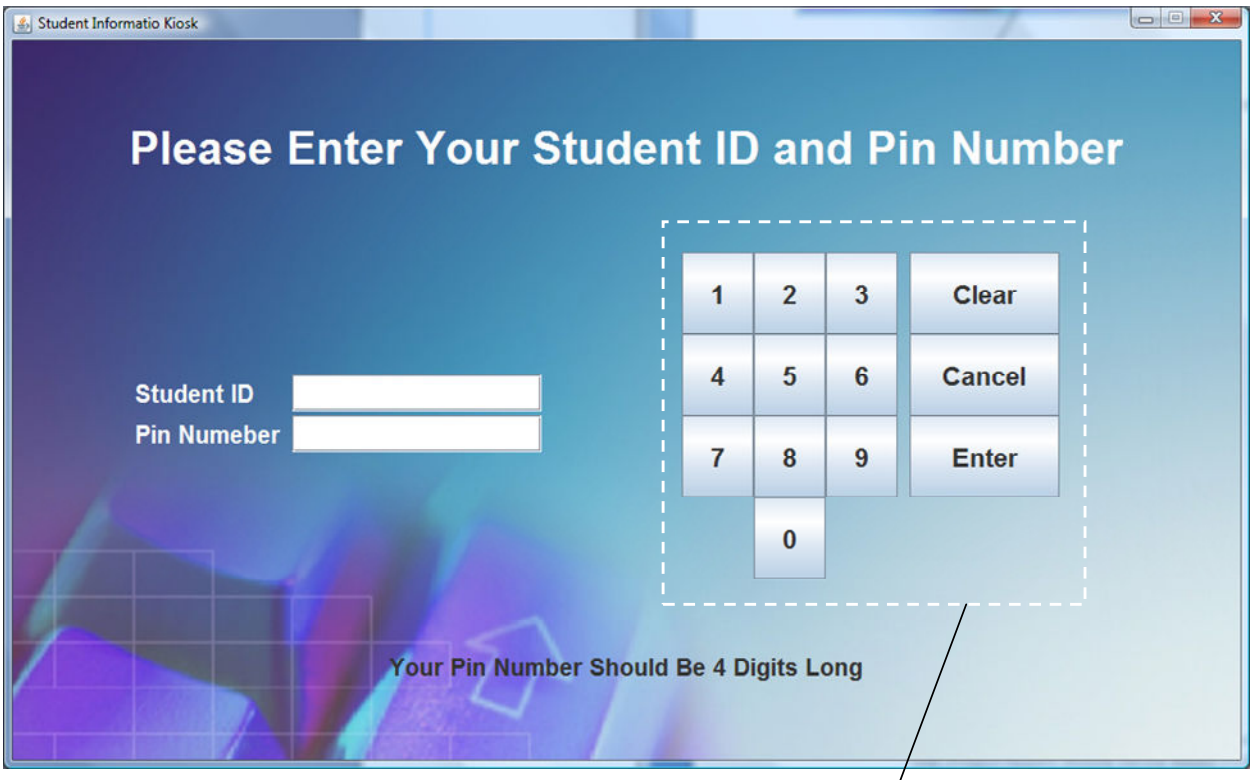
sik.client.user.panes Package
This package contains a number of panels, which are named 'panes'. Each pane is a different 'sub-page' displayed within the HomePanel, depending on which area / subject / task the user is performing, i.e. viewing messages, viewing the timetable, viewing the assignment results or performing Bluetooth actions:



The Pane

There are two pane classes within this package however, which are not contained within the HomePanel. These include the 'NumPadPane' and the 'MapAreaPane'.

NumPadPane:



The Pane

MapAreaPane:

The Pane

They are named 'panes' in order to provide structure to the system, as these panels play an important role and are contained within another panel and in which performs tasks more complex in nature than a typical JPanel containing only visual type objects.

There are other classes in this package that aren't panes, such as the 'MessagesPanelListener'. As a class can get quite large in content, such as the MessagesPane, it is often beneficial to pass some of its workload to another class. The MessagesPanelListestener is an example of this, in which it purely contains the action event handlers for a number of objects in the MessagesPane. The same applies for the 'MessagesPaneButtonManager' which simply deals with altering the properties of particular JButton objects of the MessagesPane.

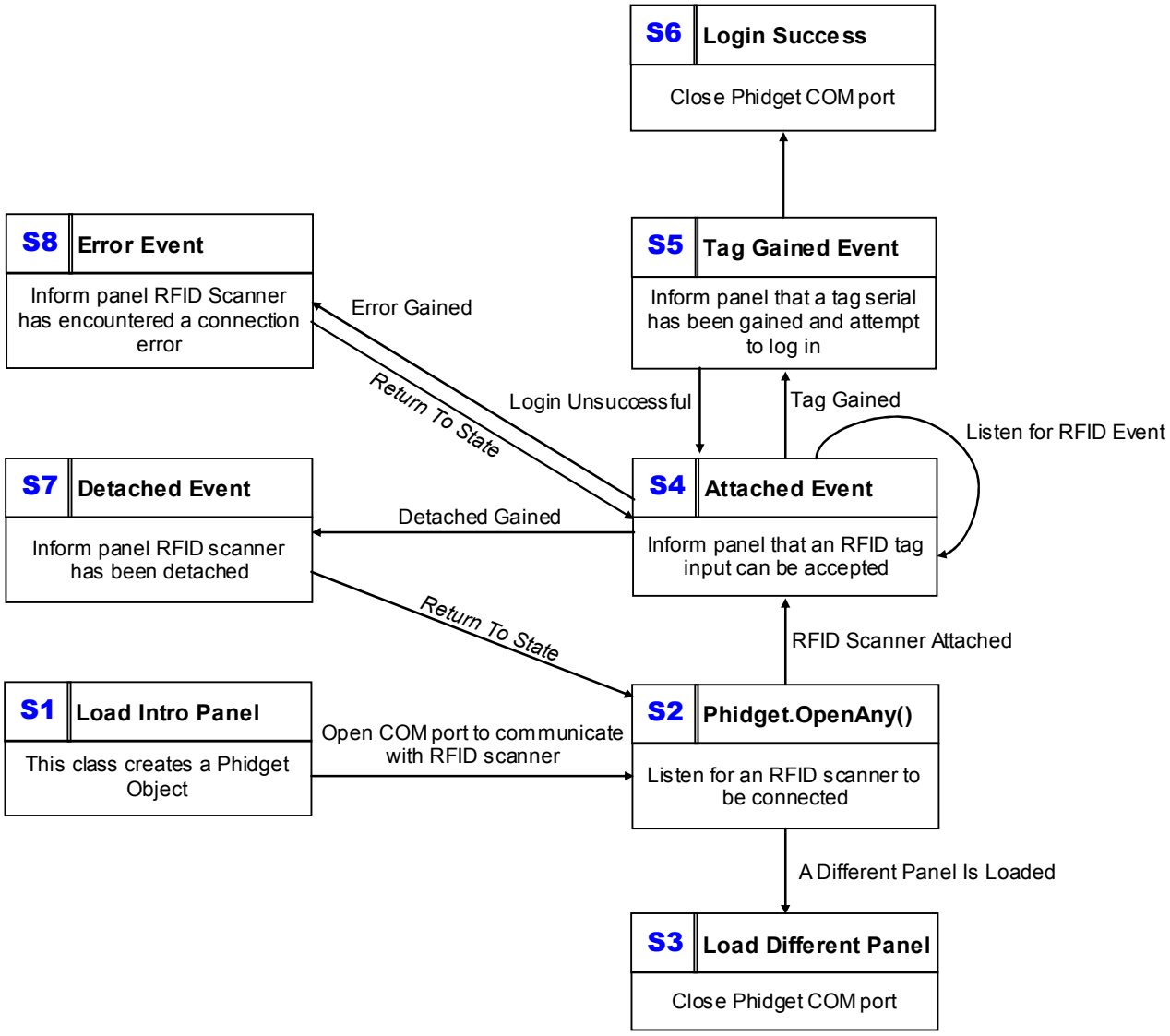
The 'BluetoothPane' class is quite a complex class. It is used to display 'Bluetooth options' to the user, which a user can discover Bluetooth enabled devices in range and send a file to a selected device. These files include text fields such as the student messages, results or timetable or an image file of the campus map. This BlutoothPane makes use of the 'BTDiscoveryServices' class and the 'BTObjectPushServices' class within the 'sik.client.user.services' package, and the 'TextFileBuilder' class in the same package in order to do this. More on Bluetooth options side of things will be discussed later on in this report. See 'Using Bluetooth Technology'.

sik.client.user.services Package

So far in this project the packages and corresponding classes have been directly apparent to the graphical user interface used by the user. This package contains classes that are used by the user interface. These include 'BTDisoveryServices', 'BTObjectPushedServices' and 'TextFileBuilder' which are used to send files to a Bluetooth enabled device (discussed later). A ClientIOServices class to read data from a config file, which are preference settings for the system, which are stored in a text file so they can be easily changed and used at run-time (mentioned a few times already in the workings of previous packages) and the 'RMIClient' (also discussed earlier) which is used by the system to invoke the methods of an RMIServer, and which is responsible for the storing and handling the student record object.

RFID Scanner (Phidgets) State Transition

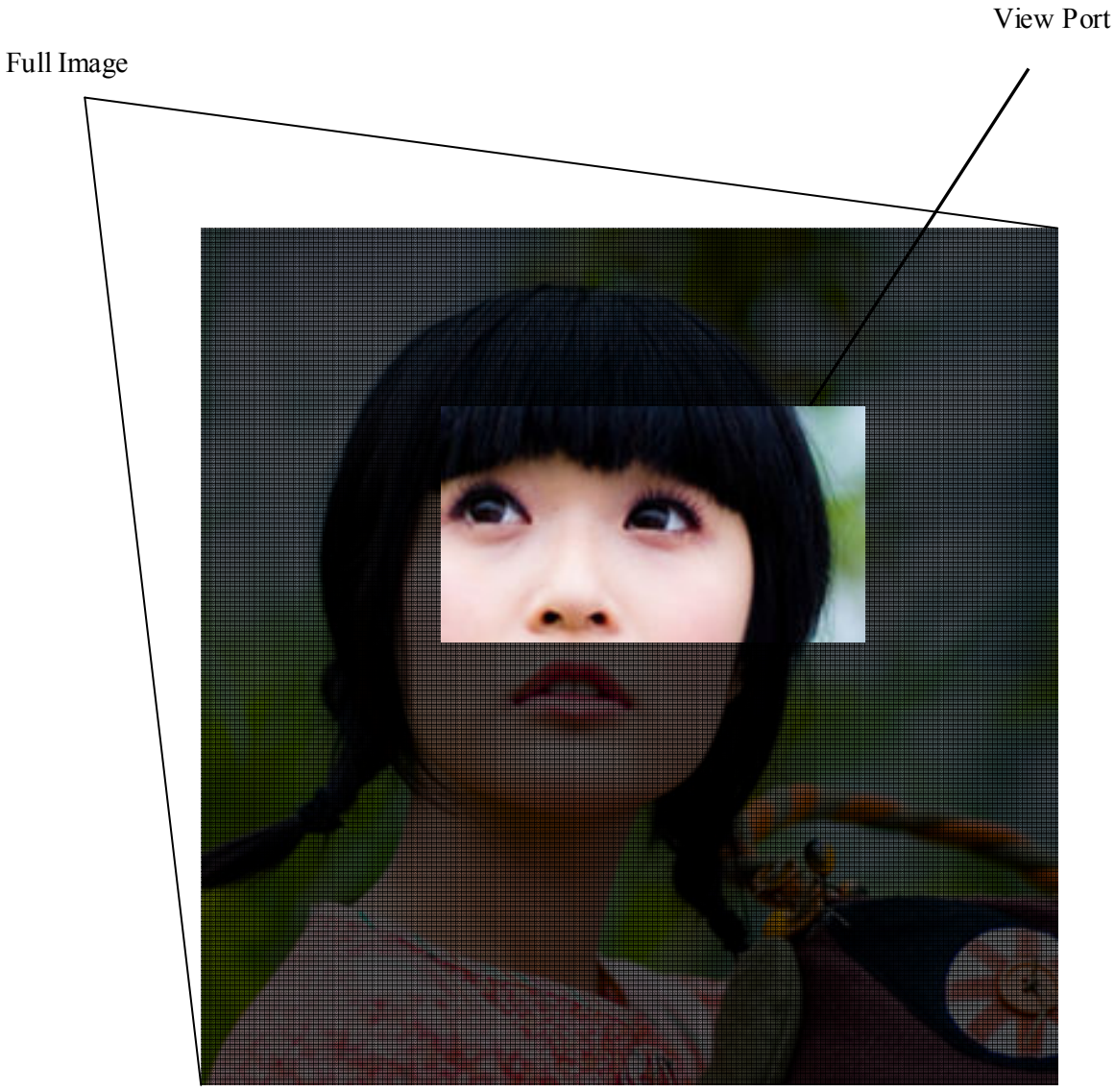
The student information kiosk utilises the Phidgets API which allows communication with an RFID scanner, and it set up to follow a pre-defined state transition:



The Campus Map

The campus map section of the projects refers to the ‘MapAreaPanel and ‘MapAreaPane’ classes. They are used to show an interactive map of the university campus to the user. Although when looking at the screen shot it appears simple and effortless, quite a lot of work went into it.

The map image itself is contained within a JLabel which is in turn is contained within a JScrollPane, which allows ‘sections’ of the full image to viewed at a time, and gives the user the ability to navigate around the image, viewing different sections (scrolling). The view section which is visible to the user is called the ‘view port’:



This is how a user is able to navigate around the campus map image. In addition to this, the scroll pane has been assigned mouse movement listeners, in which the mouse movement event handlers have been carefully coded to allow the view port to be altered by simply clicking and dragging.

```
private class mapImageMouseListener extends MouseAdapter {
	private BoundedRangeModel horizontalModel = mapScrollPane
			.getHorizontalScrollBar().getModel();
	private BoundedRangeModel verticalModel = mapScrollPane
			.getVerticalScrollBar().getModel();
	private int x0, y0;
	private Point p0;

	public void mousePressed(MouseEvent e) {
		homePanel.mainGUI.MapAreaPanel.durationTimer.restart();
		x0 = horizontalModel.getValue();
		y0 = verticalModel.getValue();
		p0 = e.getLocationOnScreen();
	}

	public void mouseDragged(MouseEvent e) {
		homePanel.mainGUI.MapAreaPanel.durationTimer.restart();
		Point p1 = e.getLocationOnScreen();
		int x1 = p0.x - p1.x + x0;
		int y1 = p0.y - p1.y + y0;

		horizontalModel.setValue(x1);
		verticalModel.setValue(y1);
	}
}
```

The map can also be ‘zoomed in’ and ‘zoomed out’, which in actual fact simply makes the image bigger or smaller for it to appear zoomed in or out. This however caused 3 different problems. Firstly, the impact on the image quality that java has when resizing an image, which results in a poor quality resulting image.

Secondly, the performance overheads required when dealing with large images. The map image has to initially be quite large to have the quality content there when zooming in, rather than just scaling an image up from a small image, which looks very poor in quality. As this image is quite large, and java converts all images it uses to raw format (bmp – very large in byte size) at run time by the java virtual machine, this can (and has) caused memory issues, such has a big impact on the performance of the Student Information System.

The last problem that was encountered, is when resizing the image (zooming in or out), the view port either re-sets to its default location, which is at the top left or appears to view a random point on the image, which is not desirable when zooming in or out of a map. Instead it would be more beneficial and less annoying if the view port gets set so that it displays the centre of what was previously displayed.

Image Quality Vs Rendering Performance

It was possible in the end to find a technique to resize an image and maintain quality; in fact swing provides a method for doing this which is:

```
Img = imageRef.getScaledInstance(<new width>, <new height>, Image.SCALE_SMOOTH)
```

This worked fine, however it had a big effect on performance. The Image.SCALE_SMOOTH parameter is an image scaling hint, this particular hint would use 'Bilinear' filtering, which, without going into too much depth about image filtering and interpolation, is one of the better techniques, resulting in a smoother after image.

The other rending hint is Image.SCALE_FAST, which uses 'Trilinear' filtering or something worse and results in an image quality that is just unacceptable, however the performance was ok.

After much reading around a new technique was discovered, which is the use of buffered images, in which buffering by itself is designed for optimisation, but using buffered images also allows more control over the image due to the availability of methods in its API, in which the following method was created to resize an image:

```
Object renderingHint = RenderingHints.VALUE_INTERPOLATION_BILINEAR;

public BufferedImage getScaledInstance(BufferedImage img, int targetWidth,
            int targetHeight, Object hint) {
      int type = (img.getTransparency() == Transparency.OPAQUE) ?
                                          BufferedImage.TYPE_INT_RGB
                  : BufferedImage.TYPE_INT_ARGB;
      BufferedImage returnImage = img;
      int width, height;

      do {
            if (width > targetWidth)) {
                  width /= 2;
                  if (width < targetWidth) {
                        width = targetWidth;
                  }
            }

            if (height > targetHeight)) {
                  height /= 2;
                  if (height < targetHeight) {
                        height = targetHeight;
                  }
            }

            BufferedImage tempImage = new BufferedImage(width, height, type);
            Graphics2D g2 = tempImage.createGraphics();
            g2.setRenderingHint(RenderingHints.KEY_INTERPOLATION, hint);
            g2.drawImage(returnImage, 0, 0, width, height, null);
            g2.dispose();

            returnImage = tempImage;
            tempImage.flush(); // keep memory usage down
      } while ((width != targetWidth) || (height != targetHeight));

      return returnImage;
}
```

This technique proved to be very successful. It still uses bilinear filtering to maintain image quality with the addition of a multi-step technique. This provides a simple yet very fast means to rescale an image an keep the quality, as with each consecutive image scale step, the changes to the image pixels are only minor, so it doesn't take long to calculate the new pixels and this also keeps the quality as the new pixels are calculated on a newly created, slightly scaled image, so it doesn't have to create many new pixels, and the pixel data is present, rather than having to try and calculate many in-between pixels, with only the pixel data from the original image.

Although this technique improved image rendering dramatically, there still was a noticed delay when loading the MapAreaPane, which occurred as the original image (large image) was loaded when the MapAreaPane was loaded, and was not kept in memory when switching to a different panel, which produced a delay each and every time the MapAreaPane was loaded. To counteract this, the buffered image is loaded at run-time, when the Student Information Kiosk interface is first started, and as kept in memory for when it is needed. The results are pleasing.

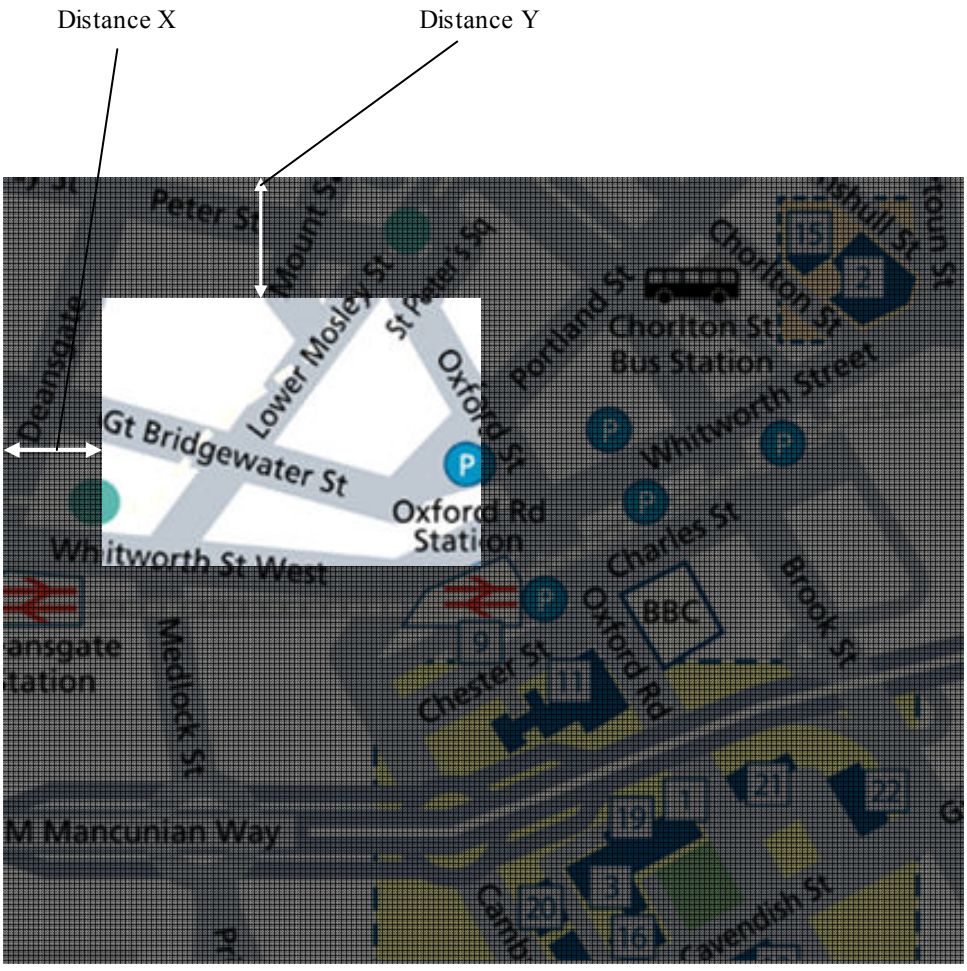
Solving the View Port Problem

As discussed earlier, the problem with the view port, is that when scaling an image up or down in a JScrollPane, the view port of the scroll pane resets to its default location, which is at the top left hand size of the of its content. Sometime the view port would set itself somewhere randomly on the content. It was bazaar.

The solution however was quite simple, all that was needed it to set the scroll bars (even though the scroll bars aren't set to visible, they can still be set) so that what was originally displayed in the view port is what will be in the centre of the view port after the image is resized. This has actually produced quite complex code, however the idea is simple.

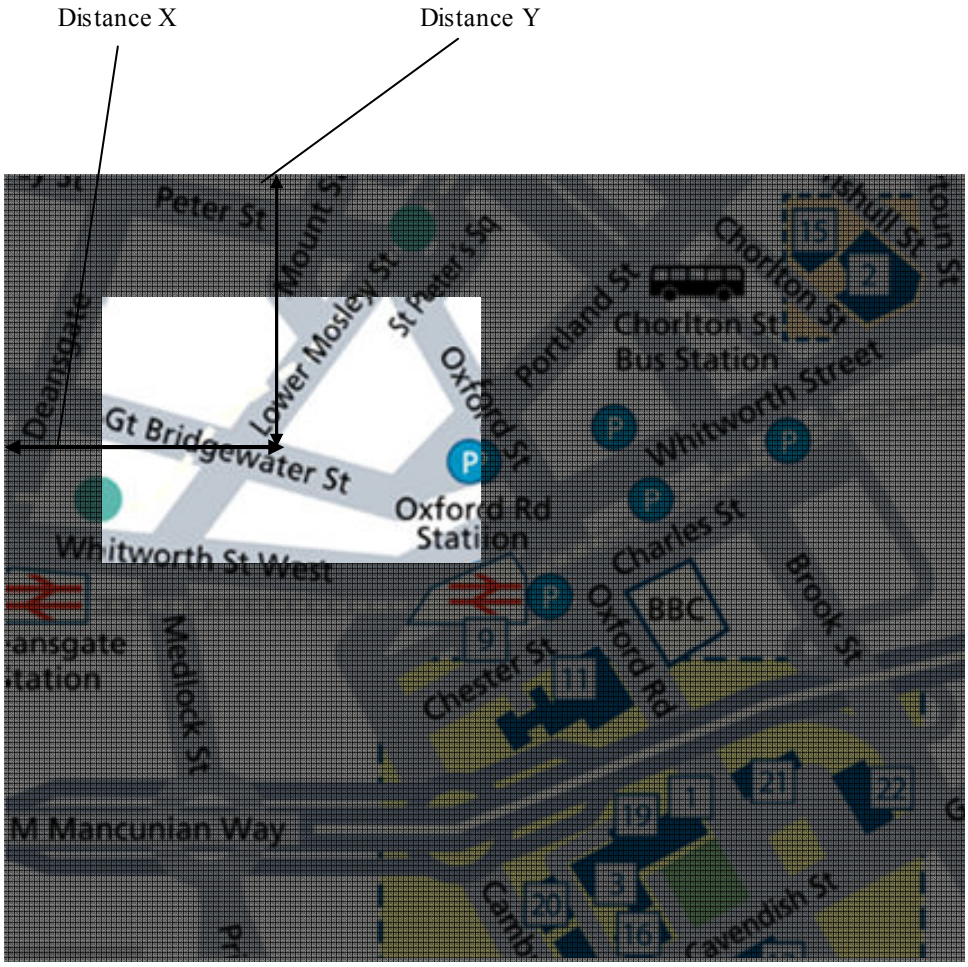
Using the size of the image, and how far away the left hand side of the view port is away from the left edge of the image, we can work out this distance as a percentage of the image. The percentage is also calculated for the distance of how far away the top of the view port is from the top of the image.

i.e.



Once this percentage has been calculated, the same percentage view port distances need to be applied (set) to the new image, and this works fine, however when the view port is at the fare left or the far top, the distance that the view port is away from the edge is now 0, in which the percentage distance is worked out to be 0, which obviously, when you scale the image up the new view port will still be at the far left or far top, which won't keep what was previously displayed in the centre of the view port in view.

What was then discovered is that this distance needs to be worked out from the centre of the view port, not the left or top:



This performed well, as it did not calculate any '0' values, however it did mean a lot more calculation needed to be performed, in which the calculation is attempted to be described as simply as possible on the next page.

Get Old View Port Distances

- Distance from centre of viewport to left of image = viewport x co-ordinate of left edge of viewport on the image + half the viewport width
- Distance from centre of viewport to top of image = viewport y co-ordinate of top edge of viewport on the image + half the viewport height
- Percentage of this horizontal distance = distance from centre of viewport to left of image / image width * 100
- Percentage of this vertical distance = distance from the centre of viewport to top of image / image height * 100

Set New View Port Distances

- New distance of centre of view port to left of image = image width / 100 * old percentage of this distance
- New distance of centre of view port to top of image = image height / 100 * old percentage of this distance

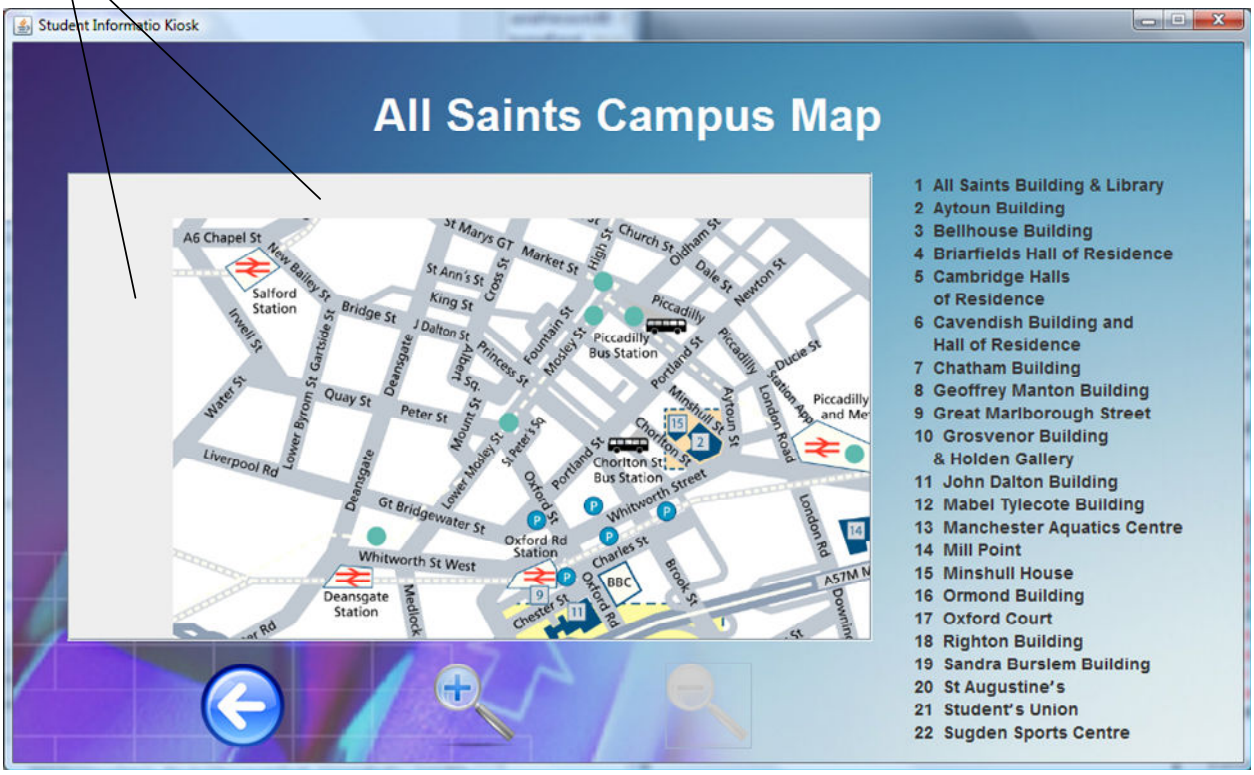
***** *When setting the values of the view port scroll bars, you can only set the top left values, so more calculation need to be done*

- New horizontal scroll bar value of viewport = new distance of centre of view port to left of image – viewport width / 2
- New vertical scroll bar value of viewport = new distance of centre of view port to top of image – viewport height / 2

Another factor affecting the viewport, is that when the viewport is set to top left of the image zoom in, when you zoom back out using the calculation above to keep what was previously displayed in the centre of the viewport can't possibly be kept in the centre of the new view port without exceeding the edge of the image, therefore an exception to the rule needs to be introduced, whereby if the image is currently zoomed and the viewport is positioned to the left or top of the image, then don't worry too much about the previous displays centre point, which can simply be done as follows, which simply sets the viewport to be at the far left or far top of the image:

```
if (newScrollX < 0) {
            newScrollX = 0;
      }
      if (newScrollY < 0) {
            newScrollY = 0;
      }
```

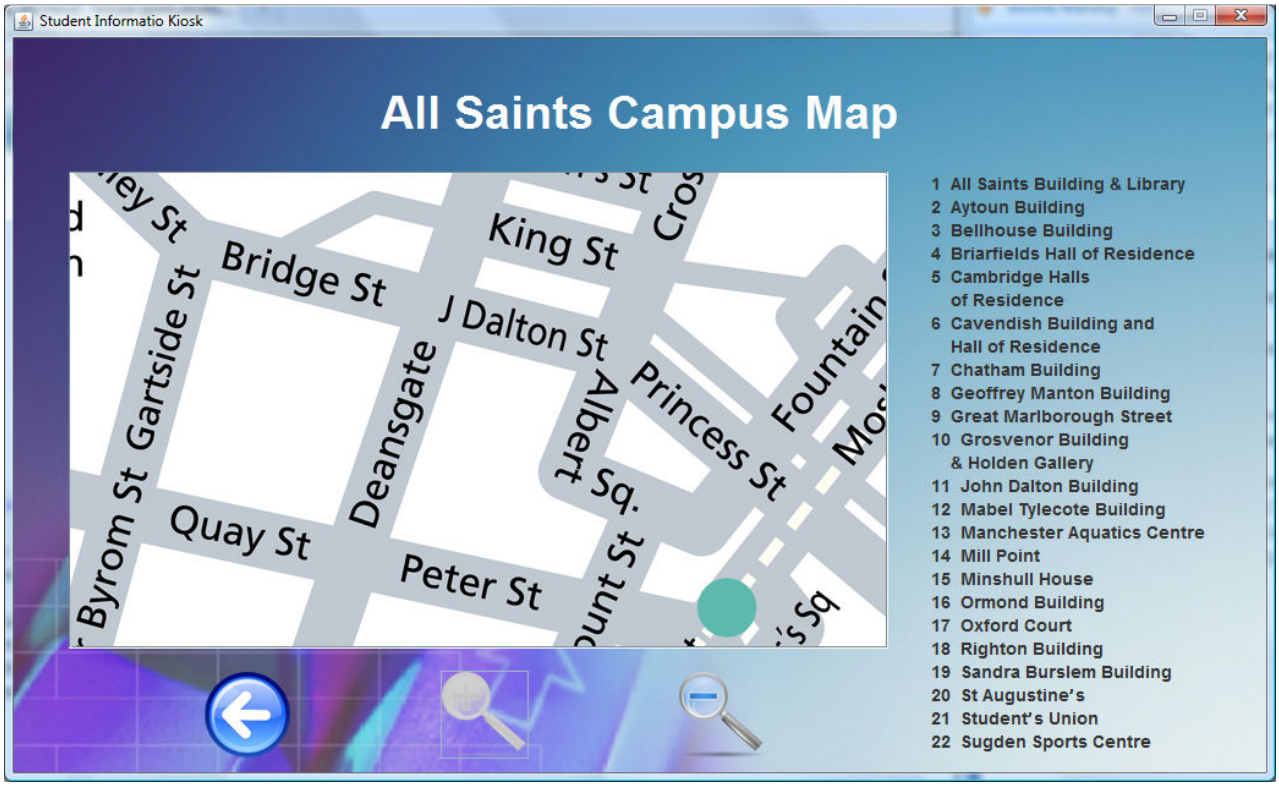
Without the exception to the rule:



With the exception to the rule:

Image Quality Results: Note: it would be better to view the actual interface in action.

Maximum Zoom In:



Maximum Zoom Out:

Using Bluetooth Technology

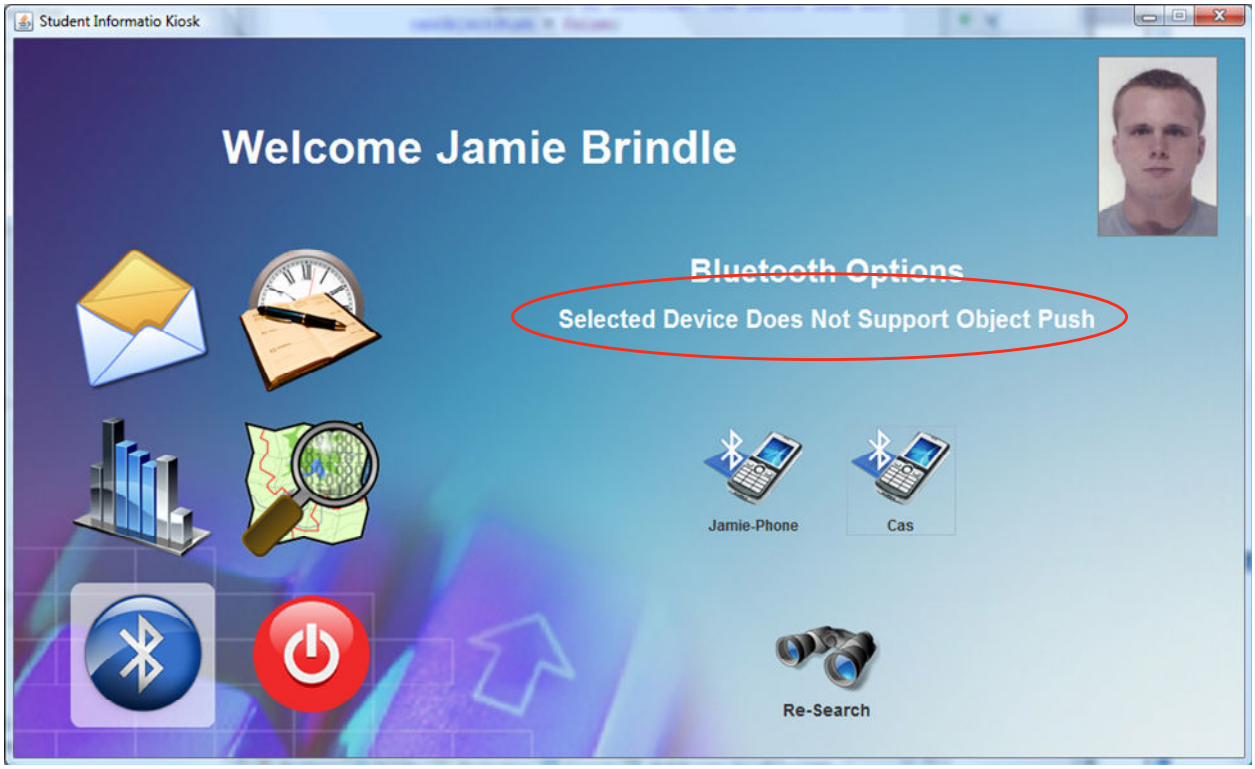
The Student Information kiosk includes two classes which are used in order to send a file to a Bluetooth enabled device. These include 'BTDiscoveryServices' and 'BTObjectPushService' which are located in the 'sik.client.user.user.services' package in the Student Information Kiosk project. A file is sent via Bluetooth's object push technology, which doesn't require any pairing of the Bluetooth host and Bluetooth client. Pairing is used for security purposes, where a client and host agree on a set 'shared key' however in using object push it doesn't reduce the security, as a client has to accept the file manually when the host wishes to transfer a file.

Before a file can be sent, a device must first be discovered, which is what the BTDiscoveryServices class is used for. It will scan for any Bluetooth devices in range and the ones it find get stored in an array of Bluetooth devices, along with their physical Bluetooth address. In order to prevent locking up the Student Information Kiosk interface while this discovery takes place, this class's discovery method is executed in a separate thread of execution.

Once the devices are discovered and their physical Bluetooth addresses are stored, it is possible to attempt to send the Bluetooth device a file, however, not all Bluetooth devices support the 'object push' Bluetooth technology (although most do nowadays). So the BTDiscoveryServices class also contains a method to query the Bluetooth devices and fetch back the Bluetooth device's available services. Once it is known that the device supports object push, then the Student Information Kiosk can proceed with the sending of a file, which is done using a method in the BTObjectPushServices class, which again, to prevent halting the system, is called in a separate thread of execution and the address of the Bluetooth device is already stored in the BTDiscoveryServices class when it performed its device discovery.

Both classes provide the BluetoothPane (within the Student Information Kiosk Interface), with discovered, success or failure messages, and details of any errors if they occur.

When a Bluetooth device is discovered, it immediately informs the Bluetooth pane, which adds a button to the pane, which when pressed, will discover the services of that selected device. If the device doesn't support object push, the user will be informed:



If the device does support object push however, the Bluetooth Panel will display a number of options of which file to send to the selected Bluetooth device:

When a user proceeds to send the device a file, the user is then prevented from attempting to send another file until the current file sending process is complete or cancelled, otherwise this would produce another thread of execution of file sending, which would throw an exception and lock the Student Information Kiosk, as the Bluetooth hardware doesn't support multiple file sends at once:

Once the process is complete however, the buttons are re-enabled to allow sending of more files, and the Bluetooth panel displays a 'send successful' message:

User Feedback and Error Reporting

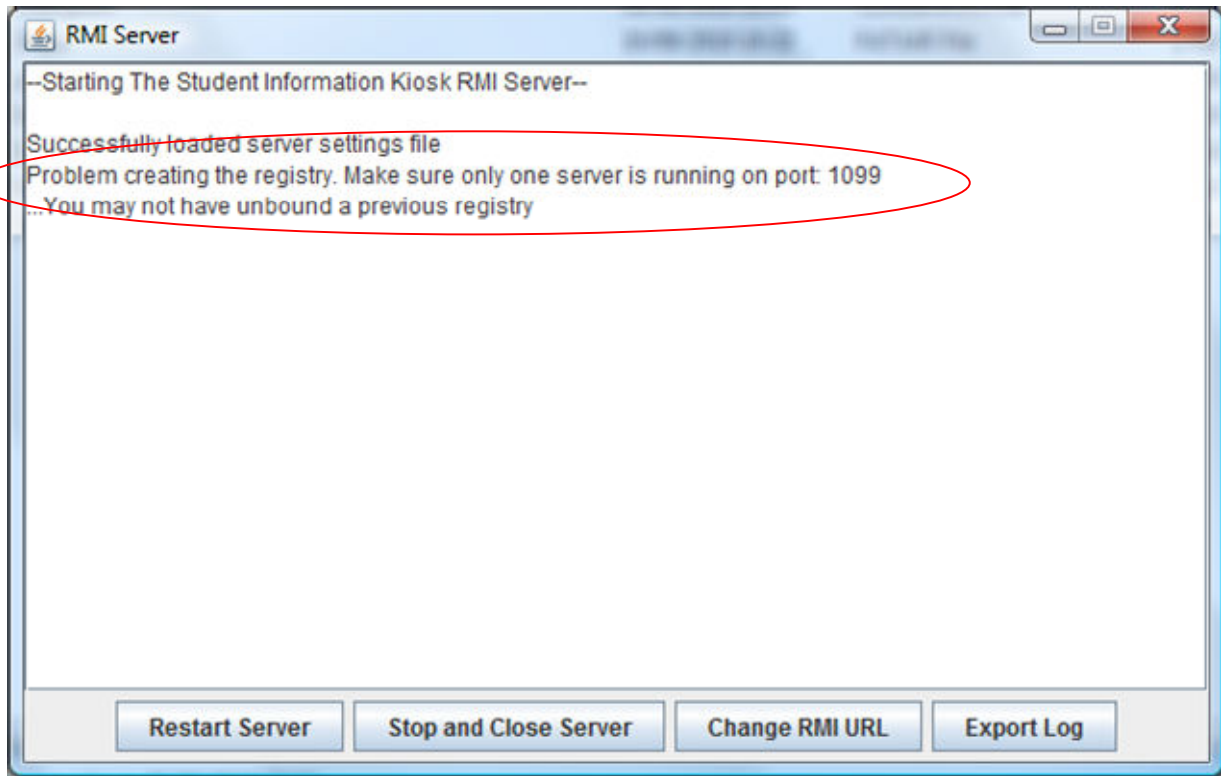
A considerable amount of time and effort has been spent making sure that both the Student Information Kiosk Interface and the Student Records Management Interface supply the user with feedback when performing particular tasks, which generally involves displaying useful messages to the user, such as in the screenshot on the previous page, where the Bluetooth options panel displays a 'successfully sent' message. A great deal of effort has been spent making sure these message or correct and where errors occur, both interfaces have the ability to provide useful information as to what the problem is or why a particular task didn't execute successfully.

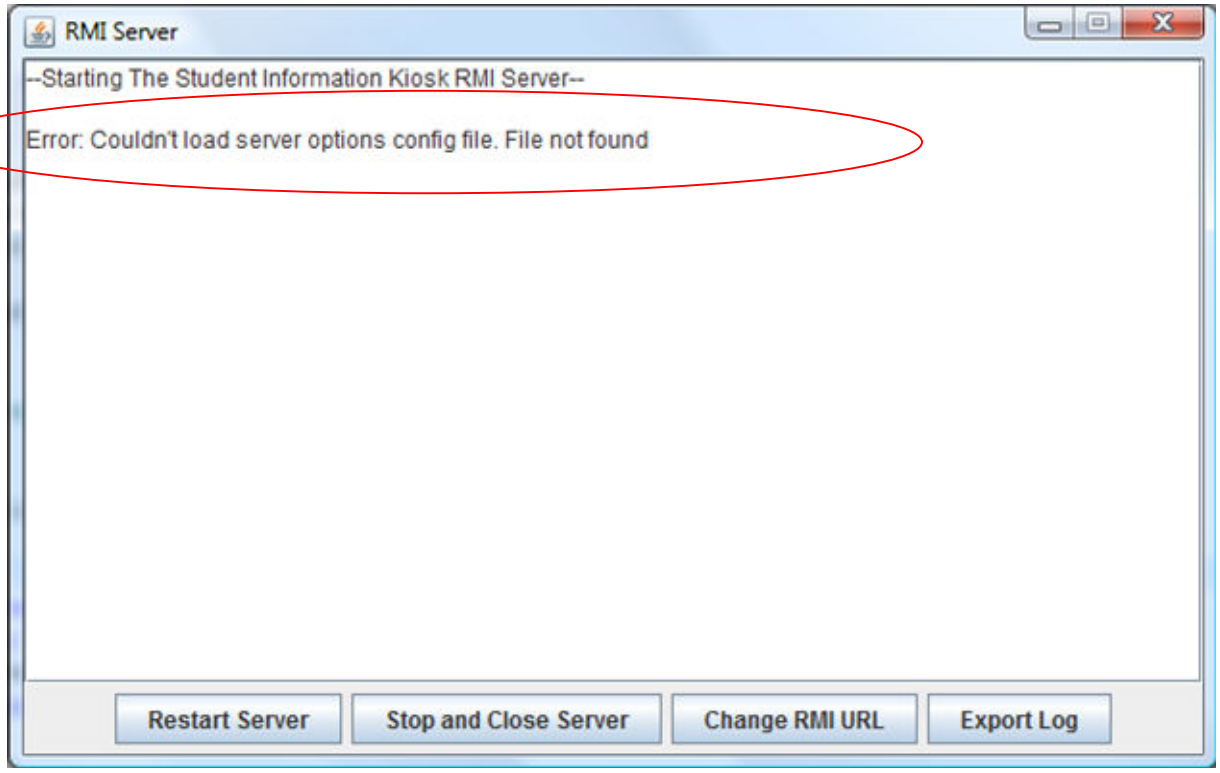
Other types of feedback include audio action clips being played when a user clicks a button or performs a particular task and the visual effects on the buttons change when it is pressed, so the user knows a particular button has been selected:

It is possible that there may however be one particularly important type of feedback left out which you may see in the screen shot above, which is the fact that the main category buttons to the left of the panel don't include any text describing what the button is or does. This is by design. Originally the buttons did included text, but they looked visual ugly and so an idea has been taken from the design of most modern mobile phones that display buttons as icons, which you will find that the main menu button's don't contain text, instead the button text when selected is displayed at the top of the menu and the same has been applied to this interface, where the title is displayed in the pane to the right (i.e. Bluetooth options in the above screenshot). Also, there aren't many buttons to choose from, and I believe is quite obvious to where the buttons take you. If it's not obvious it won't take long to find out.

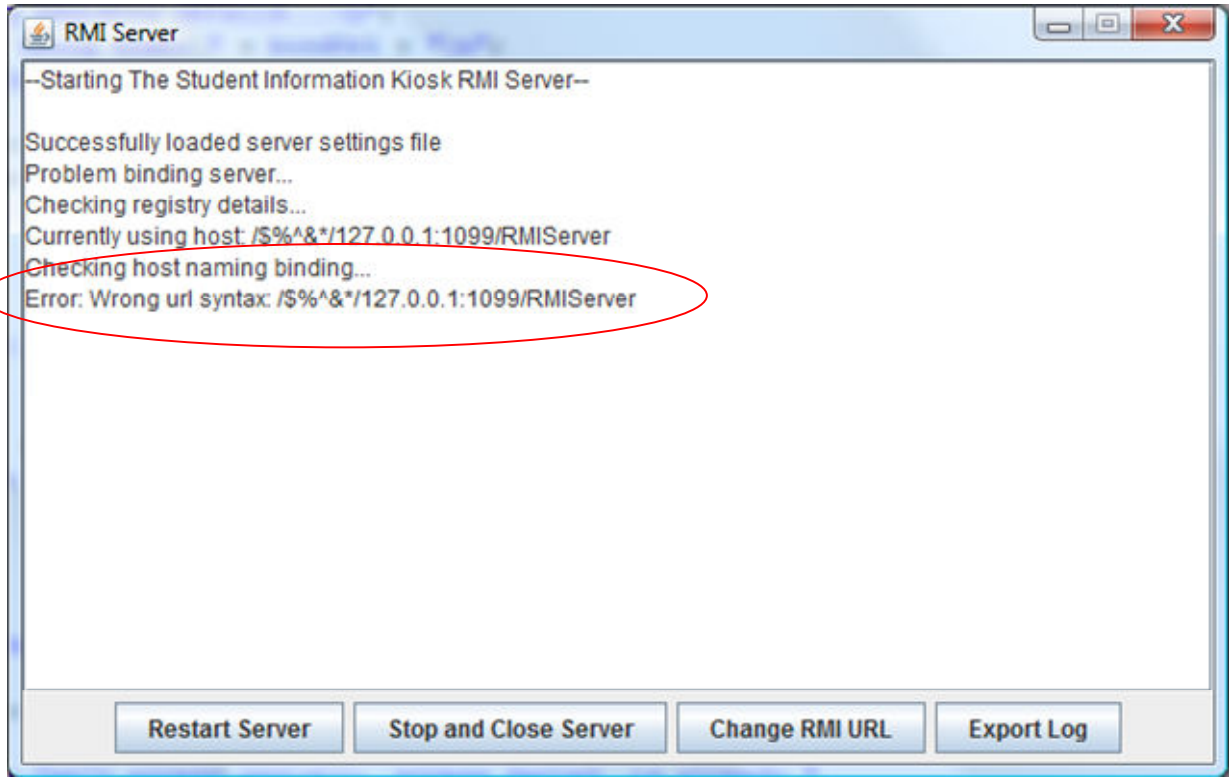
Examples of Error messages

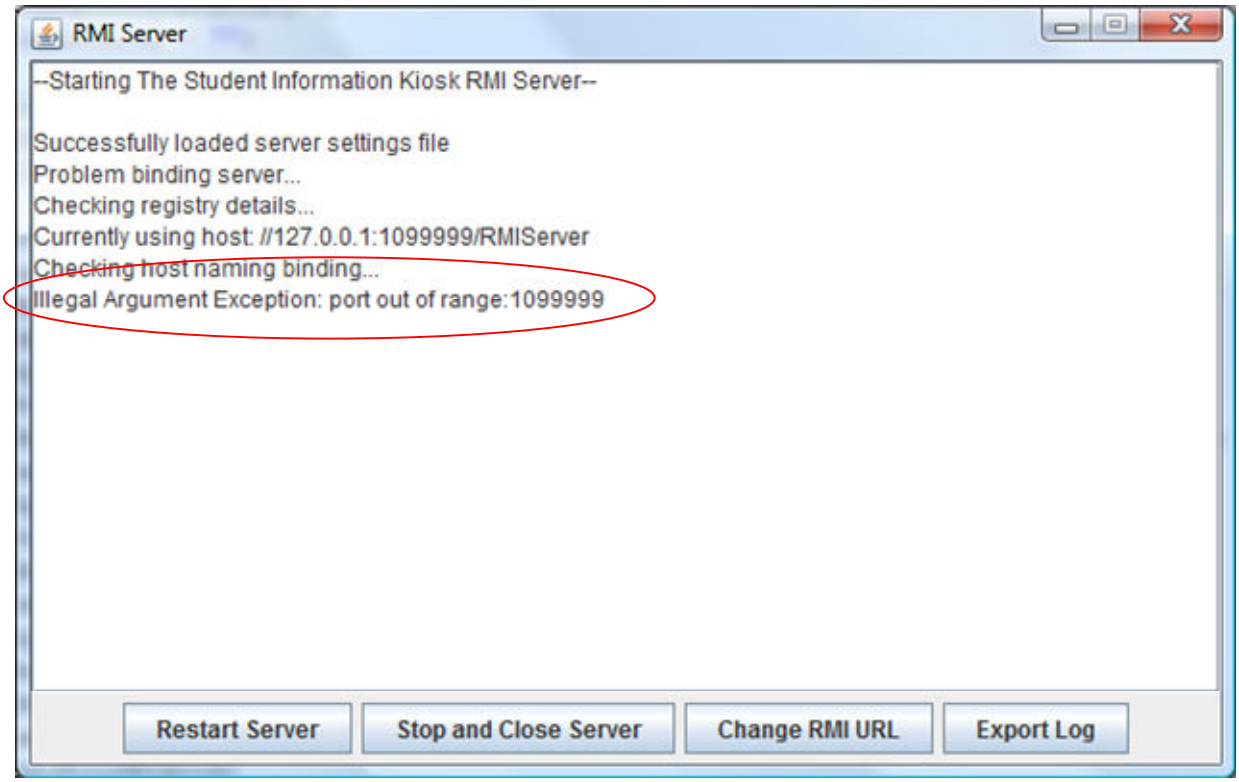
RMI Server





When there is an error binding the server to the RMI registry, it could be due to a number of reasons, in which a 'test' method has been created to determine the source of the problems. This test method would test for malformed URL exceptions (wrong bind URL syntax), remote exceptions, server not bound exceptions (usually wrong URL address but also if something is already bound in the desired bind location), illegal argument exception, which could be for a variety of reason, such as trying to use a binding port that is out of range.





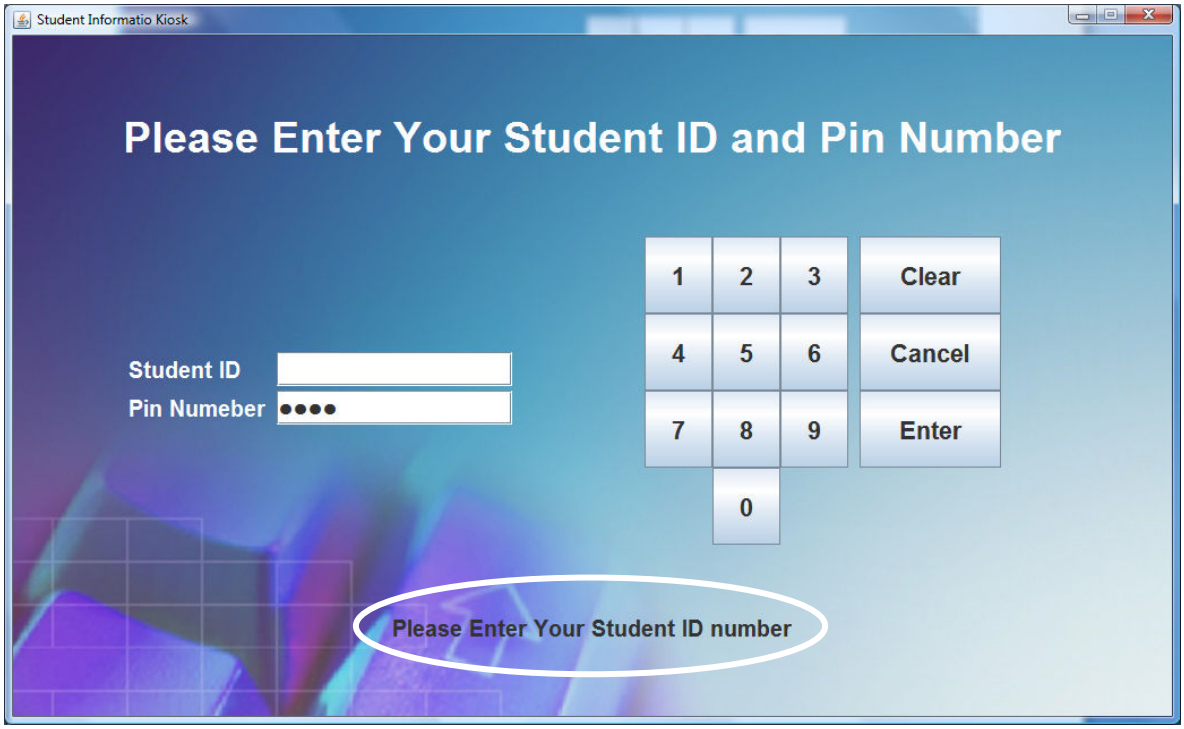
In order to generate the correct message to be displayed, an extensive series of 'try-catch' statements have been used throughout of the projects. It wouldn't be feasible to include of all the areas in the projects where these try catch statements have been used as they are simply too many of them, however an example of these are included here:

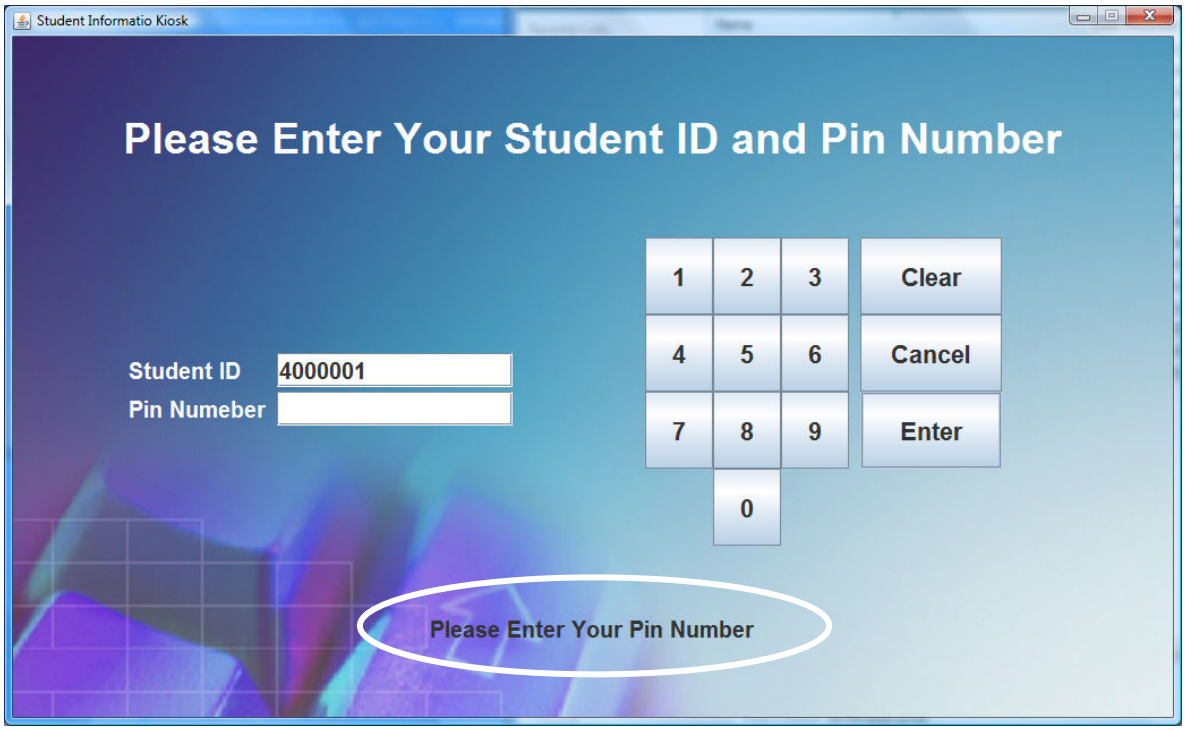
The getRegistryDetails() method in the RMIServerImpl class, which creates generates a message to determine the success or failure of an RMI binding, and includes the details of any problems found:

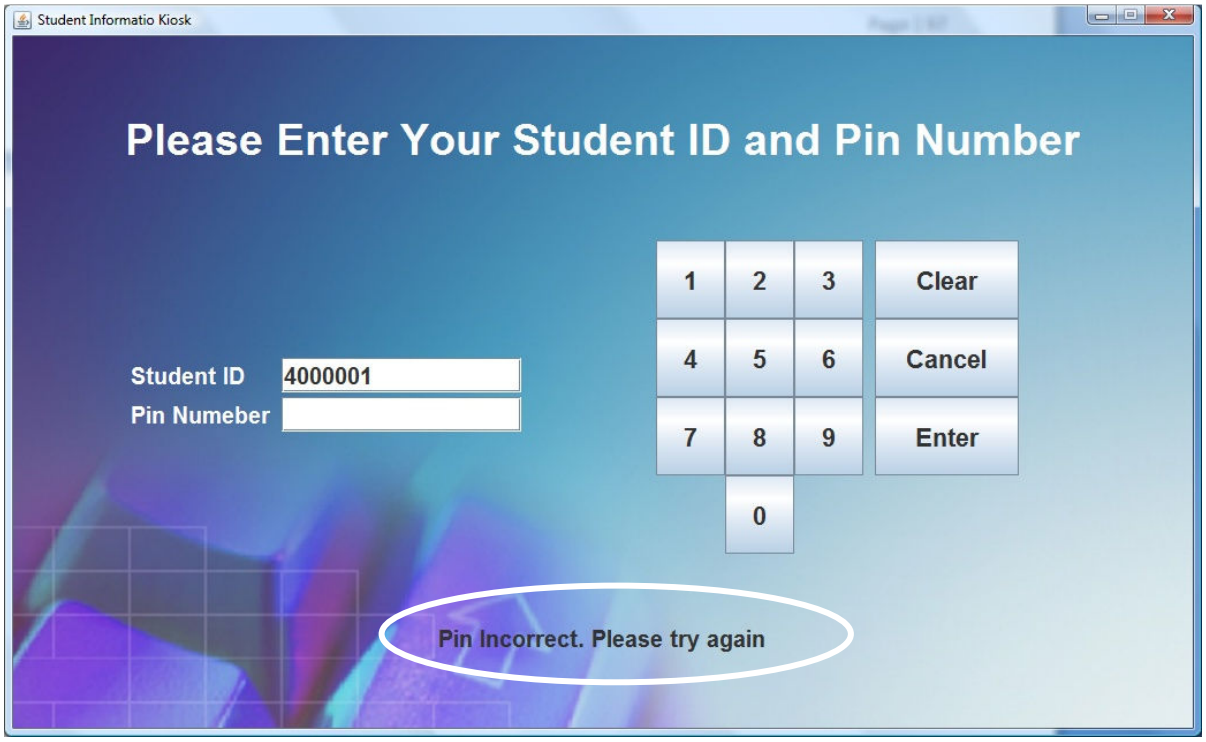
```
private String getRegistryDetails() {
        String registryDetails = "";
        registryDetails += "\nChecking registry details...\n";
        registryDetails += "Currently using host: " + bindUrl + "\n";
        registryDetails += "Checking host naming binding...\n";
        try {
                registryDetails += "..." + Naming.lookup(bindUrl).toString() + "\n";
        } catch (IllegalArgumentException e) {
                registryDetails += "Illegal Argument Exception: " + e.getMessage();
                e.printStackTrace();

                return registryDetails;
        } catch (MalformedURLException e) {
                registryDetails += "Error: Wrong url syntax: " + erverOptions.getRMIAddress() +
                                                                "\n";
                e.printStackTrace();

                return registryDetails;
        } catch (RemoteException e) {
                registryDetails += "Error: Some kind of remote connection failure\n";
                e.printStackTrace();
                return registryDetails;
        } catch (NotBoundException e) {
                registryDetails += "Error: The RMI Server stub has not bound successfully\n";
                e.printStackTrace();

                return registryDetails;
        }
                registryDetails += "Checking host registry binding...\n";
        try {
                registryDetails += "..." + registry.lookup("RMIServer") + "\n";
        } catch (AccessException e) {
                registryDetails += "Error Can't access registry, access denied, is already "
                                + "in use, not found or not allowed\n";
                e.printStackTrace();

                return registryDetails;
        } catch (RemoteException e) {
                registryDetails += "Error: Some kind of remote connection failure\n";
                e.printStackTrace();
        return registryDetails;
        } catch (NotBoundException e) {
                registryDetails += "Error: Error: The RMI Server stub has not bound successfully\n";
                e.printStackTrace();

                return registryDetails;
        }
                registryDetails += "It appears the registry has been set up successfully";

                return registryDetails;

}
```

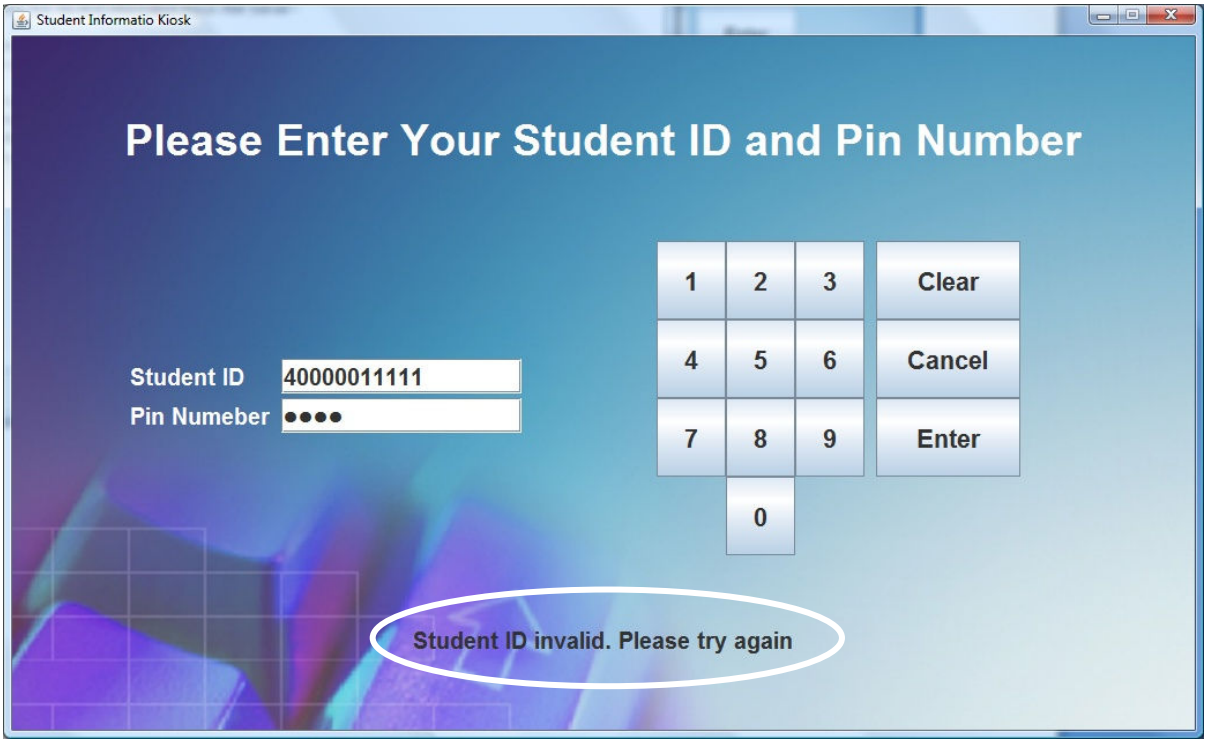
Student Information Kiosk Interface

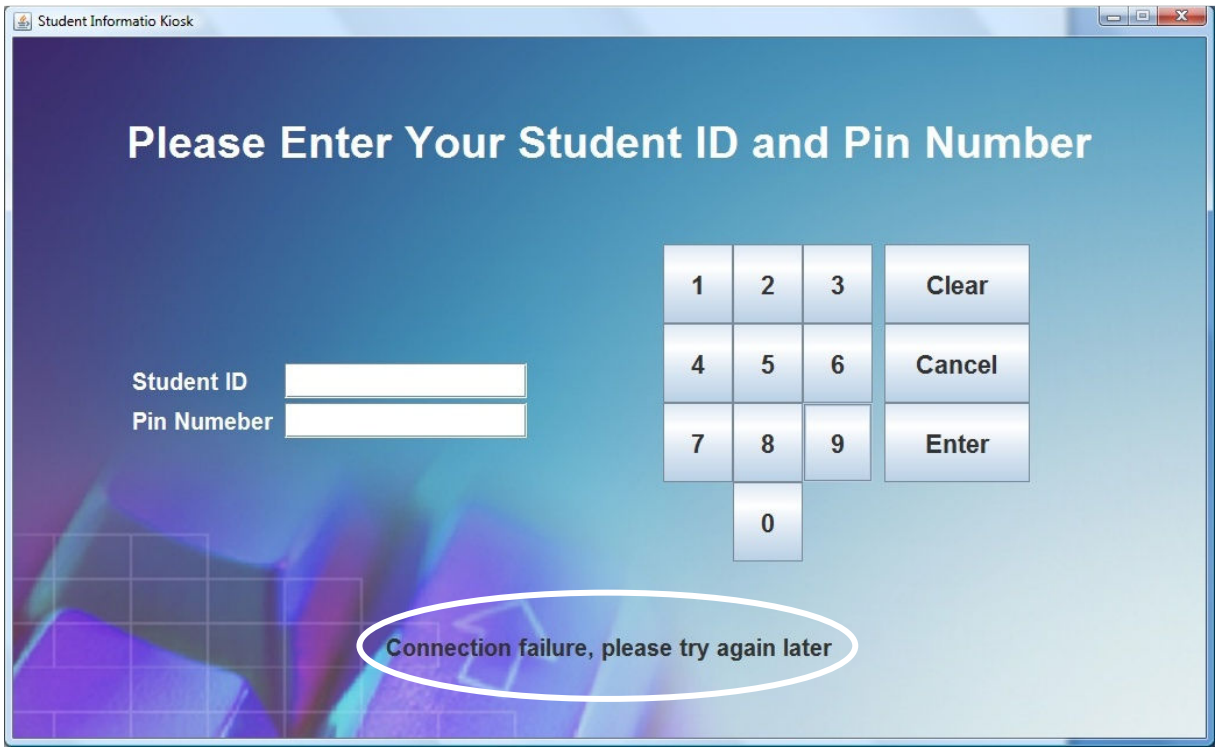
Another good example of using a series of 'try-catch' statements to determine if a problem is found is within the 'ManPinEntryPanel' class in the 'sik.client.user.panels' package where a user enters a unique student ID and pin number in order to log into the system. The method that includes these try-catch statement also includes numerous 'if else' statements to determine if the user has entered the correct credentials in order to log in, which tests if various text fields haven't been left empty (in which case the system won't try to connect to the RMI server, as there's no point, the result would be that the user hasn't provided enough information) and which credential the user has provided incorrectly:



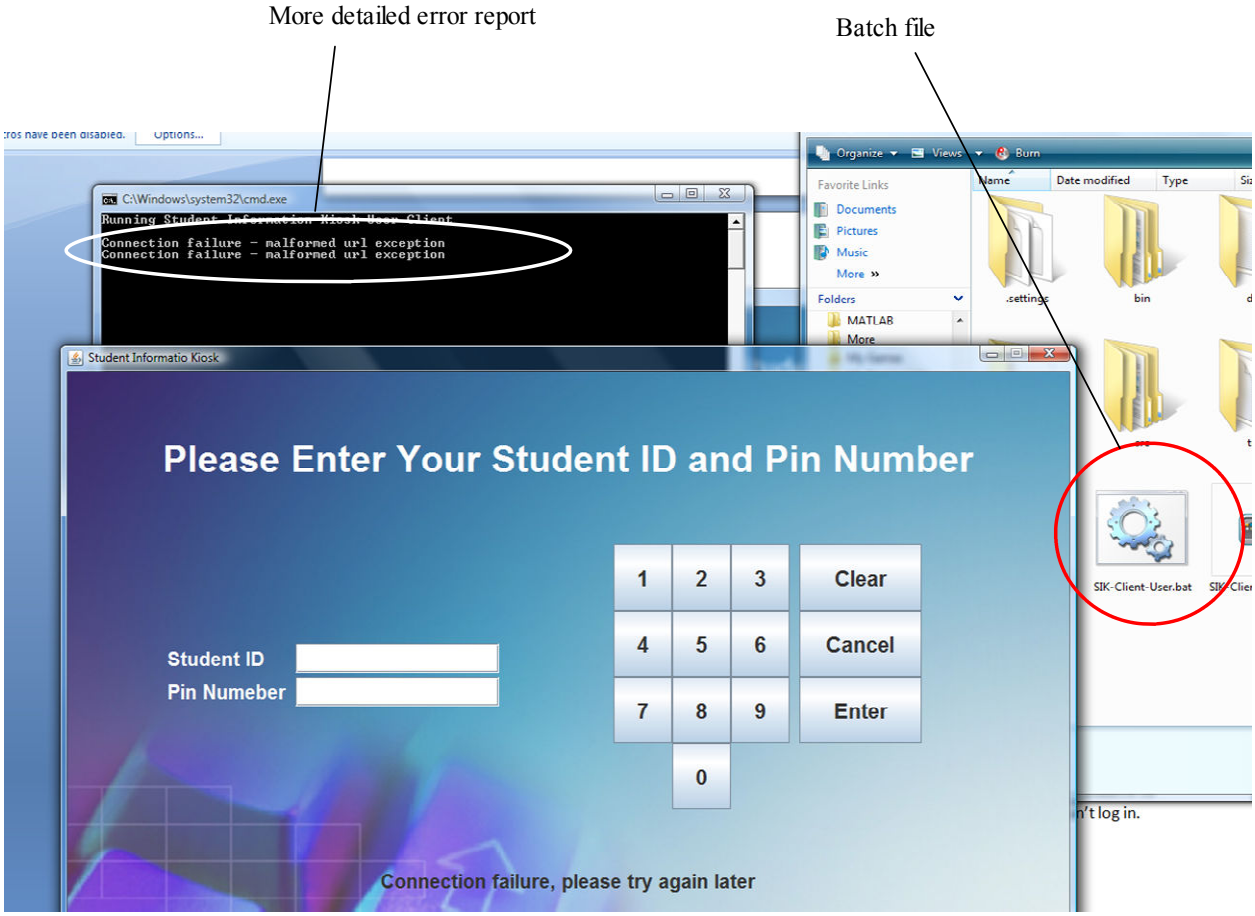



Student Informatio Kiosk
Please Enter Your Student ID and Pin Number
Student ID
4000001
Pin Numeber
1
2
3
Clear
4
5
6
Cancel
7
8
9
Enter
0
Pin Incorrect. Please try again


Student Informatio Kiosk
Please Enter Your Student ID and Pin Number
Student ID
40000011111
Pin Numeber
1
2
3
Clear
4
5
6
Cancel
7
8
9
Enter
0
Student ID invalid. Please try again

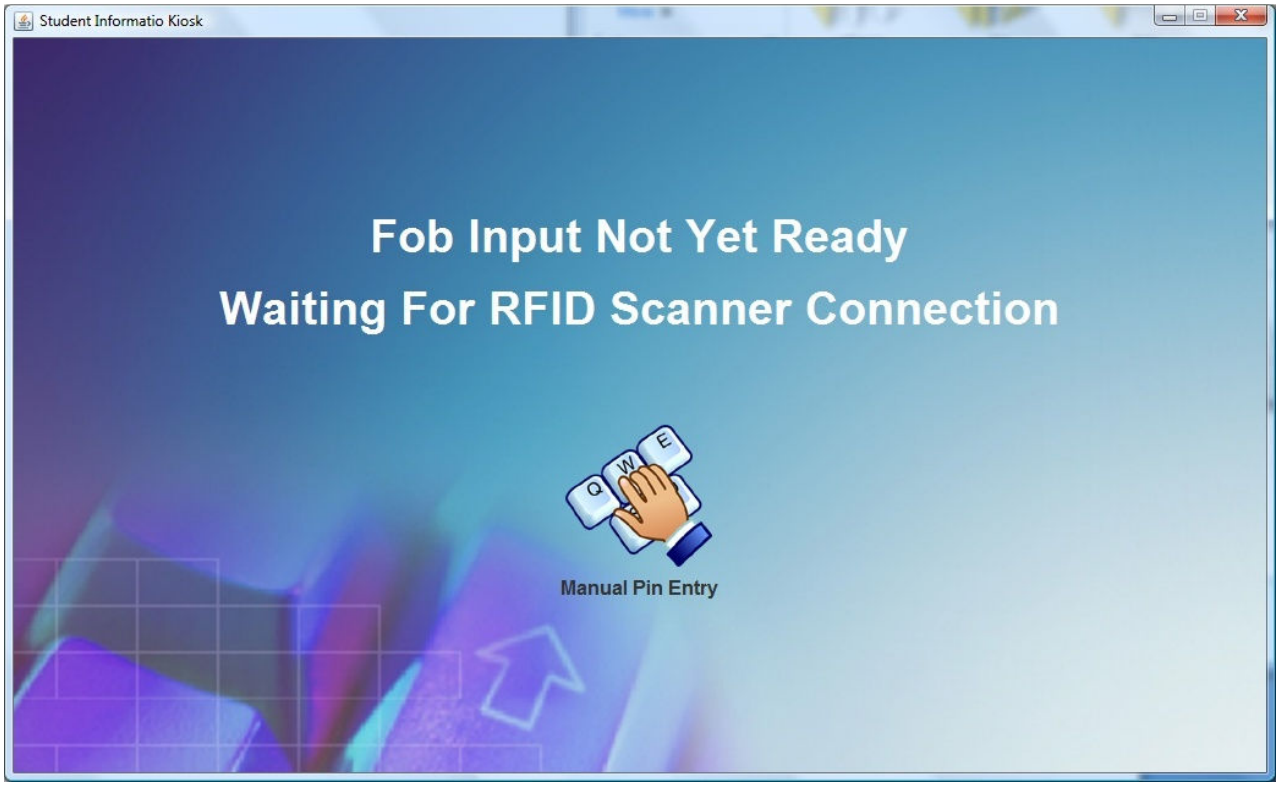


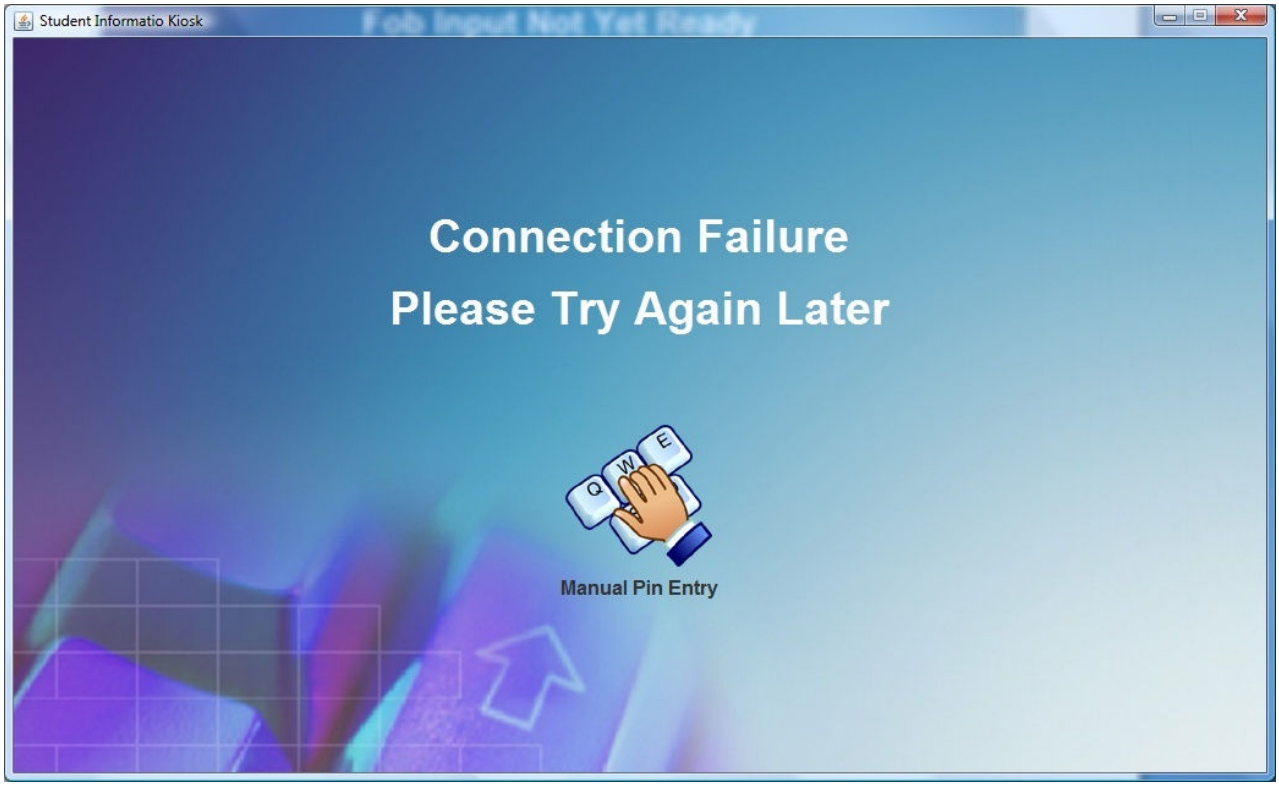
It is not necessary to display to the user why there is a connection failure, as it is not down the a student to fix the problem, all they need to know is that a connection failure exists and that's why they can't log in. However, to determine the cause of the problem, there has been a batch file created, which basically executes the runnable jar of the Student Information Kiosk from a consol, which will show more information as to why an error occurred:



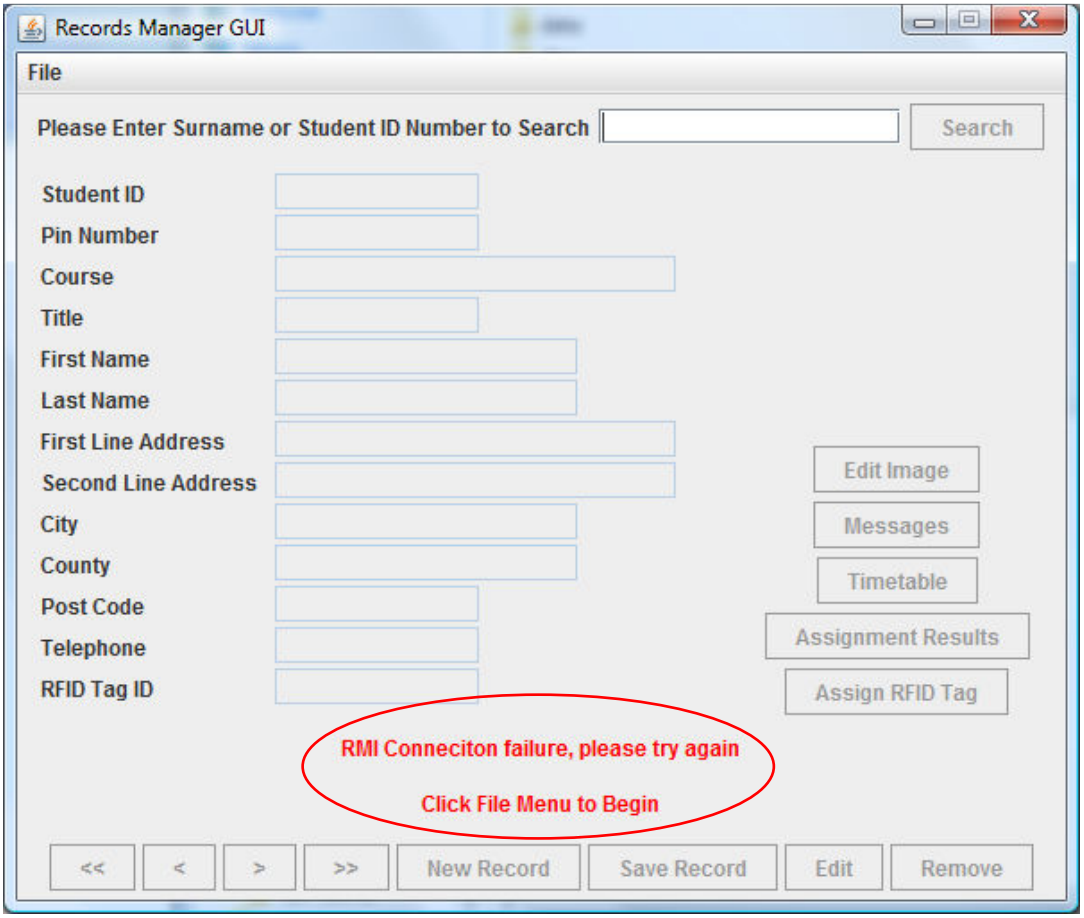
Other Message Examples

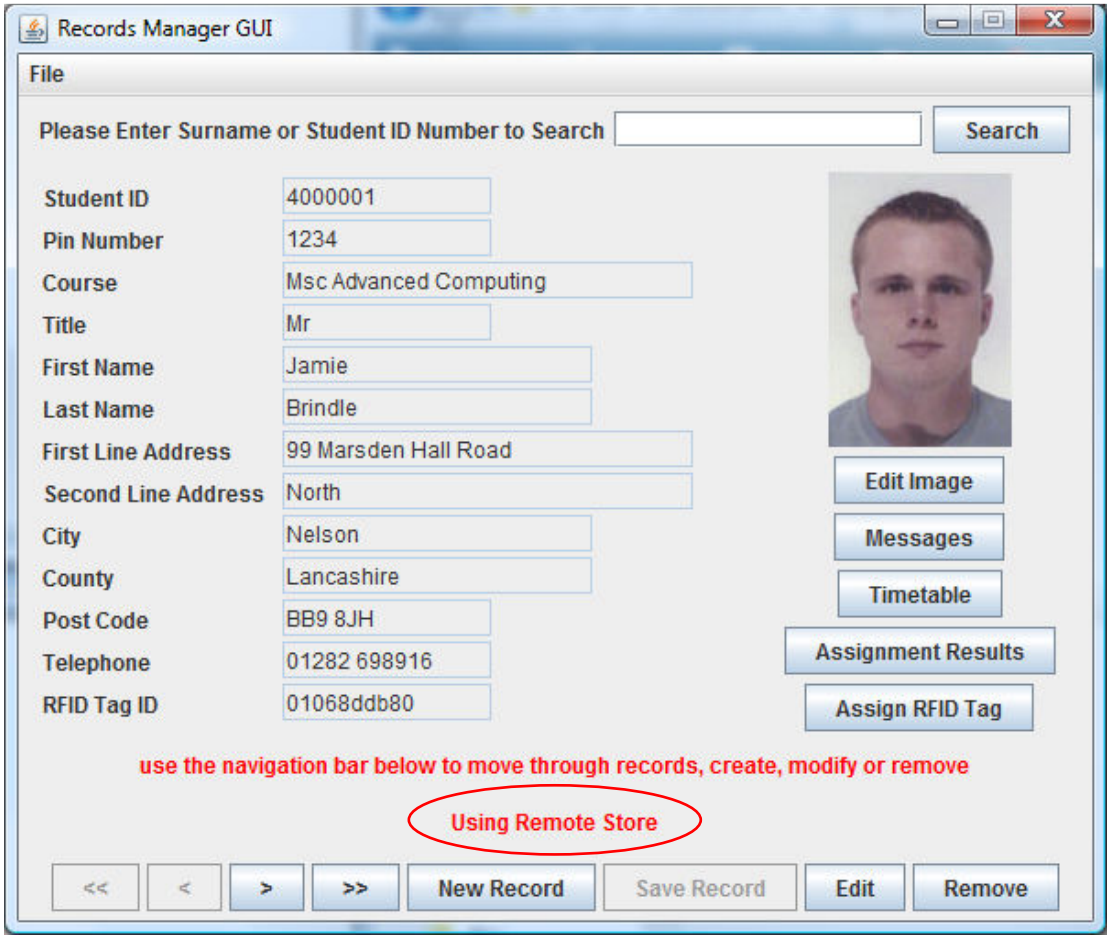
The Introduction Panel:

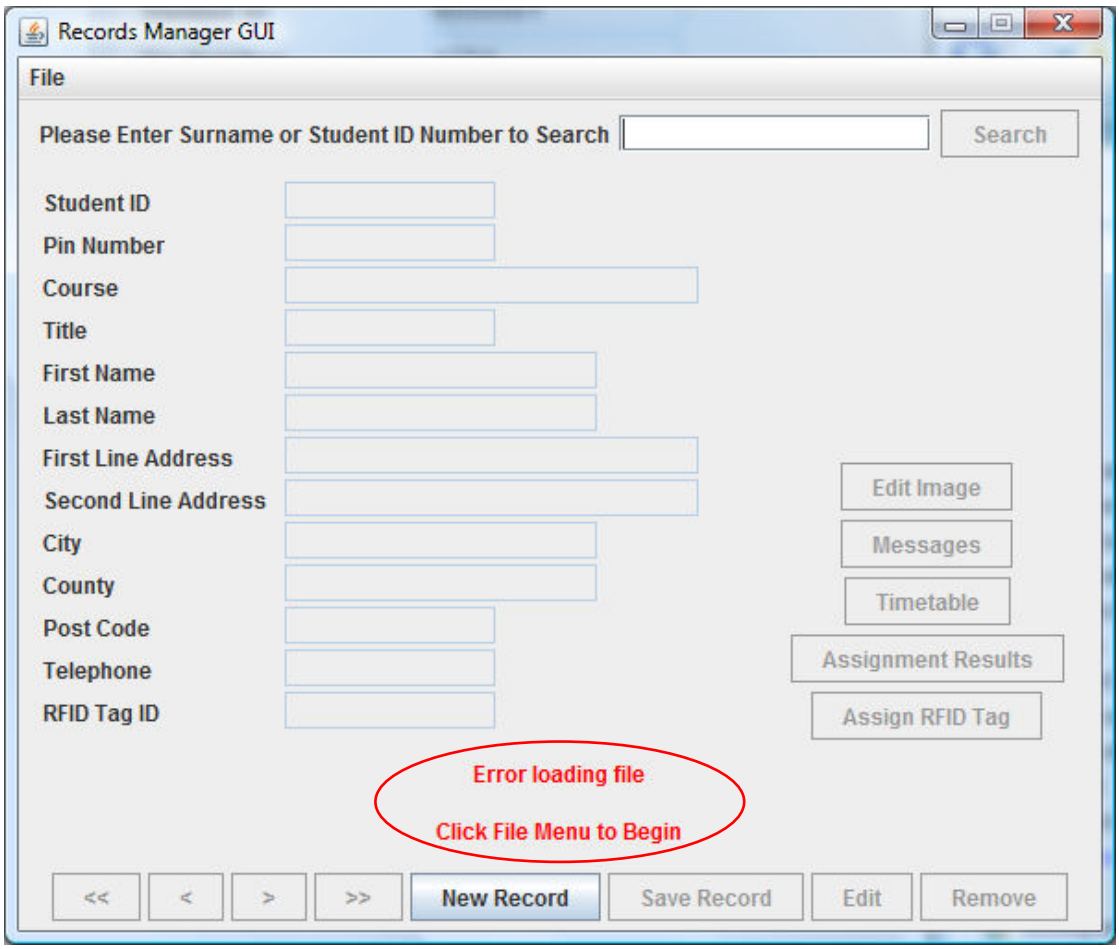





Student Informatio Kiosk
RFID Tag Unassigned
Please Contact A Technician To Re-Assign Your Tag
Manual Pin Entry


Records Manager GUI
File
Please Enter Surname or Student ID Number to Search
Search
Student ID
Pin Number
Course
Title
First Name
Last Name
First Line Address
Second Line Address
City
County
Post Code
Telephone
RFID Tag ID
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
RMI Conneciton failure, please try again
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove


Records Manager GUI
File
Please Enter Surname or Student ID Number to Search
Search
Student ID
4000001
Pin Number
1234
Course
Msc Advanced Computing
Title
Mr
First Name
Jamie
Last Name
Brindle
First Line Address
99 Marsden Hall Road
Second Line Address
North
City
Nelson
County
Lancashire
Post Code
BB9 8JH
Telephone
01282 698916
RFID Tag ID
01068ddb80
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
use the navigation bar below to move through records, create, modify or remove
Using Remote Store
<<
<
>
>>
New Record
Save Record
Edit
Remove


Records Manager GUI
File
Please Enter Surname or Student ID Number to Search
Search
Student ID
Pin Number
Course
Title
First Name
Last Name
First Line Address
Second Line Address
City
County
Post Code
Telephone
RFID Tag ID
Edit Image
Messages
Timetable
Assignment Results
Assign RFID Tag
Error loading file
Click File Menu to Begin
<<
<
>
>>
New Record
Save Record
Edit
Remove

The Use of Design Patterns and Other Programming Techniques

Design Patterns

Creational Pattern

Singleton: To ensure that a class has only once instance, in which the class includes a global point of access. The idea behind this idea is efficiency (only one object exists) but it also provides a unique way of organising objects into a tidy form.

The singleton design pattern idea has been used within this project. It may not follow the pattern exactly, however the idea is the same. An example this is the 'RMIClient' class, found within the sik.client.admin.services and sik.client.user.services packages in the Student Records Manager Interface and Student Information Kiosk Interface projects.

Only one instance of the RMIClient is every created. An instance of a student object or student object store also exists within the RMIClient, which, as only once instance of the RMI client is created, then only one instance of the student object or student object store is created. In using the idea of this design pattern for this one class has resulting in a beneficial knock-on effect elsewhere in the project. For an unreferenced class to access the RMIClient and its methods/objects, it would simply access its parent classes until it gets to the class which created the RMIClient instance:

```
if (homePanel.mainGUI.rmiClient.studentRecord.getMessages() == null) {
        messageNumberTextField.setText("0 / 0");
} else {
        messagesArrayList = homePanel.mainGUI.rmiClient.studentRecord
                            .getMessages();
```

The example above is found in the loadMessagesArea() method from the MessagesPane class, found within the sik.client.user.panes package in the Student Information Kiosk Interface project.

Builder: It should be possible to build some objects in this system to have a different representation, while being the same object, thus allowing better object re-usability.

The idea of this pattern should generally always be used in a large enough project as it basically defines the point of object oriented programming, which is to be able to redefine and re-use objects. An example of this is the use of the OptionDialog (extends JDialog) which is used in all of the projects in the system.

The OptionDialog class method would typically be called using the following call:

```
JDialog dialog = new OptionsDialog(new JFrame(),
        "Title", "Message, "Type");
```

This creates a new OptionsDialog JDialog object with a particular title, display message and type of OptionsDialog. The type could be a message, warning, an option (yes or no), or a password field. In being able to re-define the OptionsDialog, it means that it can be re-used throughout the projects.

Lazy Initialization: Involves delaying the creation of an object until it is needed, thus increasing efficiency.

This technique has been used in the Student Information Kiosk Interface project to try and keep the memory consumption of the interface to a minimum. Referring to the MainDisplayGUI class in the sik.client.user package, there are a number of methods to load particular panels onto the MainDisplayGUI e.g

```
public void loadManualPinEntryPanel() {
	manPinEntryPanel = new ManPinEntryPanel(this);
	mainDisplayPanel.add(manPinEntryPanel, mainDisplayLayout);
	manPinEntryPanel.studentIDTextField.requestFocusInWindow();
}

public void loadGoodbyePanel() {
	goodbyePanel = new GoodbyePanel(this);
	mainDisplayPanel.add(goodbyePanel, mainDisplayLayout);

}
```

These methods are accessed multiple times, in which they create new panel objects. The classes that call these classes would remove and dispose themselves or the panel before calling the above methods to ensure that only few panels remain in memory, rather than simple setting a panel '.visible=true' or '.visible=false'.

There are cases when this is not desired however, such as in the case of the campus map, when the map image is to remain in memory, and when the 'LogoutPromptPanel' is displayed, in which the previous panel simply gets hidden, as the chances are that the previous panel will be displayed again immediately after the LogoutPromptPanel, and it is desired that the current task that a user is performing is returned to once the LogoutPromptPanel is removed, which couldn't easily be the case if the previous panel were to be remove from the MainDisplayGUI.

Behavioural Patterns

Observer: A useful pattern to use where the state of an object changes; When the state of an object changes, all of its dependencies should be notified so they can make an update automatically, rather than having to check if the state has been changed.

An obvious example composing of the observer pattern is the use of event listeners and event handles, which are widely used within this project.

Template Method: To defer some steps of a method process to other 'sub methods'. This lets the sub method redefine certain steps of the method without changing its structure, thus it can be re-used in different way and thus don't have to create another method to do a 'similar' job.

The use of this pattern defines part of the underlying point of object oriented programming and has been widely used thought this project.

(Gamma, Helm, Johnson, & Vlissides, 1994)

Other Programming Techniques Used

Enumerators

Enumerators are an often forgotten about in application development. They provide a useful means to define a set of values in an easier to remember way and in which encapsulate these values. It also provides a means to retrieve these values in a number of different ways. A good example of where an enumerator has been used in this project is in the 'MapAreaPane' in the sik.client.user.panes package in the Student Information Kiosk Interface:

```
public static enum ZoomType {
	ZOOM_MINUS_4 (670, 606),
	ZOOM_MINUS_3 (811, 733),
	ZOOM_MINUS_2 (952, 861),
	ZOOM_MINUS_1 (1093, 988),
	ZOOM_ZERO (1235, 1117),
	ZOOM_PLUS_1 (1376, 1244),
	ZOOM_PLUS_2 (1518, 1372),
	ZOOM_PLUS_3 (1659, 1500),
	ZOOM_PLUS_4 (1800, 1627);

	private Dimension dimension;
	private int width, height;

	private ZoomType(int width, int height) {
		this.dimension = new Dimension(width, height);
		this.width = width;
		this.height = height;
	}

	public Dimension getDimension() {
		return dimension;
	}

	public int getWidth() {
		return width;
	}
```

This enumerate is used to get and set the image size of the map, which can be retrieved as a 'Dimension' (object which contains both x and y values), in which the parameter to set the image would be used as follows:

```
ZoomType.ZOOM_MINUS_2.getDimension();
```

Or the enumerator can return the individual x and y dimensions as follows:

```
ZoomType.ZOOM_MINUS_2.getWidth();
ZoomType.ZOOM_MINUS_2.getHeight();
```

Abstraction and Encapsulation

Abstraction and encapsulation are very important factors in part of the general foundation of object oriented programming and have been widely used throughout this project from the use of getters and setters to the use of abstract objects and method overriding.

Refactoring

Refactoring is the re-structuring, re-organisation of, re-writing and general tidying up of code once it has been written, to not only make code more readable and more organised, but to also implement the previously mentioned design patterns and programming techniques which results in a higher quality, more maintainable, more re-usable and more efficient code that conform to good practice programming. Refactoring is widely adopted as it is easier to do once the code is there in front of you, and also, coding takes time. It is often necessary, particular in industry to try and complete a class or method as quickly as possible and then go back and to code and make alterations at a later stage in development. A considerable amount of time has been spent refactoring in this project in order to get the code to a level of industry standard.

The Use of Convention

A convention is a set of agreed, stipulated or generally accepted standards and is widely used in practically all levels of programming. The main reasons for this are that:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

The Code Conventions for the Java Programming Language that Sun follow and recommend that others follow covers filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions and programming practices.

(*Web-page / Document:* Code Conventions for the Java Programming Language – *By Sun Microsystems Inc.* - oracle.com)

There is a document provided on the CD handed in with this project called 'Code Conventions.pdf' which is found in the 'Misc' folder. This document reflects the Java language coding standards presented in the Java Language Specification , from Sun Microsystems, Inc. Major contributions are from Peter King, Patrick Naughton, Mike DeMoney, Jonni Kanerva, Kathy Walrath, and Scott Hommel. The conventions used within this project match the convention specification outlined within this document.

Testing

Testing Plan

Testing will be done in a number of stages which include:

- **User Interface Testing** – Testing that each part of the interface is working correctly – i.e. buttons and basic commands function correctly and how the system can hope will errors and network failures.
- **Data Testing** – basic and complex data testing, imputing a variety of levels of data to see how the system responds and copes with the variety of data including valid and invalid data
- **Platform and Network Testing** – Transferring the system to other machines and platforms and used over a variety of network structures to test the independence and transferability of the system.
- **Usability Testing** – Getting a number of volunteers to use and test the interface to see what they thought of the system – a specific questionnaire will be created for this section of the testing.

User Interface Testing

A considerable amount of testing has already been performed on the system as the 'test as you go' method has been used, which is generally an unavoidable thing to do when developing an application such as this. Proof that the system works is shown in the screen shots of the system in action which are included earlier on in this report.

Data Testing

As the first bullet point in the testing plan (user interface testing) has already been performed while developing the system, the first 'testing stage' will technically be the data testing. Data testing can only be performed on the Student Records Management Interface as this is the only project that requires a user to enter any information. The other projects (Student Information Kiosk Interface and RMI Server) does not require any data input other than the press of a button.

The data testing results for the Records Management Interface follows.

Key:
== : equivalent
! : not
> : more than
< : less than
=> : equal to or more than
<= : less than or equal to
Null : empty
&& : and
|| : or

Editing a Record

('What Occurred' refers to what happened when the save button is pressed)

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
Pin Number	== 4 number characters only	< 4 number characters && ! null	Warning message displayed, database not updated	Successful
		> 4 number characters	Warning message displayed, database not updated	Successful
		null	Warning message displayed, database not updated	Successful
		4 letter characters	Warning message displayed, database not updated	Successful
		3 number characters and 1 letter character	Warning message displayed, database not updated	Successful
		4 number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Course	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
Title	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow null values
		Invalid characters: !"£$%^&*(){][ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
First Name	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){][ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
Second Name	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){][ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
First Line Address	! null && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Second Line Address	No invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
City	No invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful: should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
County	! null && no invalid characters (i.e. !"£$%^&*) && no number characters	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful: should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
Post Code	! null && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Telephone	! null && no invalid characters (i.e. !"£$%^&*) && no letter characters	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Warning message displayed, database not updated	Successful
		Overall result:	Successful	

Creating a New Record

('What Occurred' refers to what happened when the save button is pressed)

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
Pin Number	== 4 number characters only	< 4 number characters && ! null	Warning message displayed, database not updated	Successful
		> 4 number characters	Warning message displayed, database not updated	Successful
		null	Warning message displayed, database not updated	Successful
		4 letter characters	Warning message displayed, database not updated	Successful
		3 number characters and 1 letter character	Warning message displayed, database not updated	Successful
		4 number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Course	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
Title	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow null values
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
First Name	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
Second Name	! null && no number characters && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful, should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
First Line Address	! null && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Second Line Address	No invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
City	No invalid characters (i.e. !"£$%^&*)	null	Save appears successful, no warning messages appear, the database was updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful: should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
County	! null && no invalid characters (i.e. !"£$%^&*) && no number characters	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Unsuccessful: should not allow number characters
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Unsuccessful: Changes need to be made	
Post Code	! null && no invalid characters (i.e. !"£$%^&*)	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Overall result:	Successful	
Telephone	! null && no invalid characters (i.e. !"£$%^&*) && no letter characters	null	Warning message displayed, database not updated	Successful
		Invalid characters: !"£$%^&*(){}[ ]:;'@#~<>,./?|\-`¬ *(Attempted individually)*	Warning message displayed, database not updated	Successful
		Number characters	Save appears successful, no warning messages appear, the database was updated	Successful
		Letter characters	Warning message displayed, database not updated	Successful
		Overall result:	Successful	

Searching for a Record

Field	Desired Allowed Input	Attempted Input	What Occurred	Validation Success
Search	Anything If only numbers entered, will only search on account numbers If any other character entered, will search on last name If null is entered, will reset the search and display all records in present store	4000001	Successfully loaded a single account: 4000001	Successful
		400	Loaded all accounts, as all accounts contain 400 in account number field	Successful
		40000000000	No account loaded as there were no matches	Successful
		null	Cleared search, loaded all accounts within present store	Successful
		S	Loaded all accounts who's second name contain an 's' which = 2 (4000002 and 4000003)	Successful
		B	Loaded one account who's second name contained a B which = 1 (4000001)	Successful
		X	No account loaded as there were no matches	Successful
		Brindle	Loaded one account who's second name contained a Brindle which = 1 (4000001)	Successful
		£$%^& entered individually	No account loaded as there were no matches	Successful
		Overall result:	Successful	

Adding and Editing Messages

Any characters are allowed here.
The result: Successful

Editing Timetable

Any Characters are allowed here.
The result: Successful

Assignment Results

Any Characters are allowed here.
The result: Successful

Assign RFID Tag

There is are no validation check on this tag serial number field, this is due to the fact that a user shouldn't enter an RFID tag number, the tag serial field should be populated automatically via the RFID scanner tag event gained event. Changes need to be made here.

The result: Un-successful

Data Testing Results and Changes to be Made

1. The desired input for the 'Title' field should not allow null values
2. The desired input for the 'Title' field should not allow number characters
3. The desired input for the 'First Name' field should not allow number characters
4. The desired input for the 'Second name' field should not allow number characters
5. The desired input for the 'City' field should not allow number characters
6. The desired input for the 'County' field should not allow number characters

Re-Testing After Alterations to the Implementation Have Been Made

Field	Previous Problem	Attempted Input	What Occurred	Validation Success
Title	The desired input for the 'Title' field should not allow null values	null	Warning message displayed, database not updated	Successful
Title	The desired input for the 'Title' field should not allow number characters	Mr7	Warning message displayed, database not updated	Successful
First Name	The desired input for the 'First Name' field should not allow number characters	Ja3mie	Warning message displayed, database not updated	Successful
Second Name	The desired input for the 'Second Name' field should not allow number characters	Bri24ndle	Warning message displayed, database not updated	Successful
City	The desired input for the 'City field should not allow number characters	Ne254lson	Warning message displayed, database not updated	Successful
County	The desired input for the 'County field should not allow number characters	Lanc235ashire	Warning message displayed, database not updated	Successful
	Overall result:	Changes made to the implementation were successful		

Platform and Network Testing

As Java uses its own virtual machine, known as 'Java Virtual Machine' then it should be able to run on any platform that has the java runtime environment set up and installed. Java was originally designed to be platform independent and so transferring the Student Information Kiosk Interface and running it on different platforms shouldn't prove to be a problem. Also, most machines and platforms are able to use the TCP/IP protocol that RMI / sockets use, so in theory the Student Information Kiosk will be able to run over most modern network structures, provided that the installed JRE (Java runtime environment) includes an RMI API. Never the less the system is to be tested on a Linux and Windows operating system and test whether the RMI Server and Student Information Kiosk Interface can communicate with each other when both the client and server are run under the same platform and if they can communicate with each other when the client and server is run under different platforms.

Results

Client (admin & user)	RMI Server	Can Run	Can Communicate	End Result
Windows	Windows	Yes	No	Unsuccessful
Linux	Linux	Yes	No	Unsuccessful
Windows	Linux	Yes	No	Unsuccessful
Linux	Windows	Yes	No	Unsuccessful

It appears we've run into a major problem. The RMI client of the Student Information Kiosk and Student Records Management Interfaces won't find and communicate with the RMI server on another machine.

For some reason, the RMI server gets successfully bound to the RMI registry, but the RMI client doesn't seem to be able to locate the remote registry, which is odd as this has briefly been tested before and has turned out to be a success. Numerous attempts have been made along with numerous changes to the RMI client and server code and binding preferences; however it appears something is being missed out. The RMI client and server only seem to be able to communicate when they are running on the same machine. I suspect this has something to do with the java codebase property which defines where the RMI server and client locate remote objects. It may also be a routing/firewall issue. This will be looked into.

One success however, is that the all project interfaces will run on different machines with different operating systems.

Usability Testing

Testing the usability of the Student Information Kiosk Interface will be done by people other than me who are willing to participate in simply using the system and then filling out a questionnaire which has been written by myself. This kind of testing will hopefully bring up some usability issues that I have not noticed myself. I believe this to be the best way as the designer and creator of any software will no doubt know exactly how to use the software they have just created, and so certain issues may go un-noticed.

A sample questionnaire is found on the next 2 page:

The questionnaires that have been filled in are included in the appendix of this report.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☐ No ☐

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☐

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☐

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☐

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☐

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☐

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☐ Not at All ☐

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☐

Comment:……

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☐

If Yes, which area?:……

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☐

If Yes, which areas?:……

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Questionnaire Results

There were 10 people in total who filled out the questionnaire. The filled out questionnaires can be found in the appendix.

Question No	The Question	Result
1	Have you ever used graphical user interface like the Student Information Kiosk before?	4 people responded 'no' and 6 people responded 'yes' to this question.
2	What did you think of the overall response time of the information kiosk?	5 people responded 'neutral' 2 people responded 'good' and 3 people responded 'very good'.
3	How did you find learning to use the information kiosk?	9 people responded 'very easy' and 1 person responded 'easy' to this question.
4	How did you find the structure and layout of the information kiosk?	9 people responded 'very simple' and 1 person responded 'simple' to this question.
5	Did you find that the information kiosk got straight to the point?	10 people responded 'yes' to this question.
6	Did you find the amount of information that was displayed on the screen helpful enough?	9 people responded 'yes' and 1 person responded 'just enough' to this question.
7	How much did you feel the information kiosk differs from other graphical user interfaces?	7 people responded 'not a lot' and 3 people responded 'not at all' to this question.
8	What did you think of the overall usability of the information kiosk?	7 people responded 'very good' and 3 people responded 'good' to this question.
9	Were there any areas of the information kiosk that you did not like?	8 people responded 'no' and 2 people responded 'yes' to this question.
10	Were there any areas of the information kiosk that you though were useless or not required?	10 people responded 'no' to this question.

Evaluation

Overall the project has been a success. It has been a worthwhile, educational and generally enjoyable experience, apart from the odd stressful occasions experienced when discovering a number of issues while implementing the system. Another slightly stressful experience is also occurring at this present moment while racking my brain trying to think of how to actually start writing the evaluation. I good place to start would probably be to outline the issues discovered, how the issues have or haven't been resolved and what would be done differently given the knowledge of hindsight. This would also provide an advantageous steppingstone for describing any future development that maybe possible for the Student Information Kiosk system and to outline a number of usage scenarios.

Design Flaws / Implementation Issues

A number of issues have already been identified in the implementation section of this report in which the section also describes how the issue has been overcome. These include the campus map image rending quality versus performance trade off, the issue discovered of keeping the centre of the viewport in view when zooming in and out of the image, the issue of the Bluetooth options panel halting due to Bluetooth host and device communication failure and the use of batch files to allow a technician to be able to retrieve information on any errors that occur that wouldn't normally displayed to the user.

It is actually beneficial to run into problems particularly in a university project because it then serves the purpose of doing the project in the first place, which is to learn. The issues outlined in the implantations are also beneficial as they then enabled me to demonstrate problem solving skills, intuitiveness and some level of programming aptitude when describing why the issues are occurring and how they are resolved.

Some issues however haven't been described in the implantation section of the report, which are to be outlined next.

The Use of RMI

If it was required to describe the experience obtained from the learning of and usage of RMI within this project in two words, it would have to be 'never again'. This is due to the fact a vast majority of the problems discovered while implementing the system were due to the use of, or possibly in this instance the incorrect use of Java's RMI API.

The reason for using RMI as a web service protocol however is still sound. It does provide a unique way to invoke the methods of a server as if they were local and provides the ability to send java objects, such as the student object and student object store in this case across a network with very little intervention with the addition of extra security you wouldn't easily be able to obtain from the use of standard sockets for example. However the problems encountered from using RMI has forced me to re-evaluate if RMI is in fact required to be used at all in this system.

Firstly, is security really essential? Currently the RMI server sends the client a student object which contains all the information of a student, which include their messages (university related), timetable, assignment results and personal details such as their address and telephone number. The information kiosk however only uses particular information from the student object, such as the timetable, messages and results. It does not use the personal information of the student, so it's pointless sending it. The only reason to send it is because it's easier to transfer the whole object, rather than retrieving certain details individually and sending either a new object or the required details separately, in which case there would be no personal information transferred and so security isn't really required. Plus using this method would mean less data is required to be transferred across the network, thus reducing network traffic.

Secondly, is the higher level of functionality provided by the use of RMI effectively utilised? The answer to this no, it is not. RMI provides the facade of local method invocation, in which in a large enough system, where many different methods calls are made this would prove to be very useful, however in this system the RMI server contains very few methods. Its purpose is to merely send one of two objects to the clients as a return value from one of two methods which contain the student information. That is all. In which case it would have been just as effective to create a synchronizable object (object converted into byte stream) and send it via the use of sockets, using java's standard sockets API. This would produce the same effect and produce far less problems as the whole process is simpler and basic and requires no 'codebases', 'security policies' and 'security managers' properties to be properly configured, which it where a lot of the problems have been encountered and it also doesn't require RMI interfaces to have to be created or registry RMI objects to be bound to the RMI registry etc and it is easier to bypass or configure a firewall to allow RMI communication across a network, which is a common issue in the use of RMI.

It is therefore concluded that the use of sockets would have been a feasible and adequate means of client/server communication in this system. The only real benefit foreseen is that it demonstrates a higher aptitude of programming, as RMI is more complex, and it supports easier possible future development in the future as the use as RMI provides more functionality and it is now there in the foundation of the system ready to be expended upon.

The Lack of the Use of an SQL Database

It has already been mentioned in the research analysis, pre-design section of this report that an SQL database would be an appropriate means of storing the details of students, particularly in a university where there are essentially tens of thousands of students as. SQL databases provide a fast, disk-space efficient, multi-access means of managing large amounts of data. The reason it hasn't been used is because the goal of this project is focused on user interface design and mobile technology, not on database design, and therefore it didn't seem feasible to waste too much time implementing another level to the system that isn't required, and that could potentially incur more time costs due to discovering issues with the additional 3rd party software usage and the fact that this system has been developed as a prototype.

It would have however shown additional skills in the use of a proper database platform and I really wish I had used t. I don't believe it would have taken up too much extra time and if this system were to ever be real life tested with the current amount of students in a university it would simply not work. This is due to the fact that all student objects are stored in a single 'store' object, which is in fact an array list. This

means that every time the server starts up it must load the whole store object before it can retrieve a single student object from it, which would either fail or take up a considerable amount of time and resources as this store would be very large if there were as many student object stored in it as there are students in a university. ImageIcons are even stored in the student object. If the student details were store in a database, all the RMI server needs to do is consult with the database and retrieve only the required information, not everything.

There is a simple solution to problem however that doesn't require too many changes to the existing coding of the system however, as the system has been designed to be easily maintainable and easy for future development due the quality of the code, all that would be required in an additional class, named say 'StudenObjectConstructor'. This class would include SQL database connectivity methods, which would retrieve the required student information from the database and construct a student record object equal to the current student record which is currently retrieved from the student object store. Therefore the RMI server and the rest of the system could utilise the student object in the same way as it already does.

Conclusion Drawn From the User Testing and Questionnaire

Question 1: Have you ever used graphical user interface like the Student Information Kiosk before?

4 people responded 'no' and 6 people responded 'yes' to this question. The question was set out to determine if people thought the user interface overly dissimilar to other graphical user interfaces. It was surprising to discover that just under half of the people thought it was very different as the GUI is icon based, however looking back to the wording of the question, it's probably too vague to get a quality result, not to mention 2 people made the comments: 'what is it like?' and 'what kind of interface is it?', which leads me to believe that the question caused some confusion.

Question 2: What did you think of the overall response time of the information kiosk?

5 people responded 'neutral' 2 people responded 'good' and 3 people responded 'very good'. These results were expected, as occasionally the response time of the interface was a little 'laggy', however once the interface had properly loaded it was quite fast. This is due to using the Samsung Q1 tablet computer that the users used in testing the interface being quite slow in terms of hardware performance with the addition of the software currently installed on the Samsung Q1 using up too much memory. The Student Information Kiosk Interface isn't exactly an overly large or complex application, it should run relatively smoothly.

Question 3: How did you find learning to use the information kiosk?

9 people responded 'very easy' and 1 person responded 'easy' to this question. This was expected as the user interface doesn't provide an overwhelming amount of functionality. It is a 'point and click' type of interface and the interface provides an adequate amount of instructional messages. Overall the result to this question is pleasing.

***Question 4*:** How did you find the structure and layout of the information kiosk?

9 people responded 'very simple' and 1 person responded 'simple' to this question. The person who responded 'simple' to this question also responded 'easy' to the previous question, which would show a correlation between the two questions; If the structure and layout of the user interface is simple, then it musts be easy to use. This result was expected as the user interface was designed to be as simple as possible.

Question 5: Did you find that the information kiosk got straight to the point?

All 10 people responded 'yes' to this question and is what was expected. Again, the user interface is design to be just 'point and click' and then an action is performed.

Question 6: Did you find the amount of information that was displayed on the screen helpful enough?

9 people responded 'yes' and 1 person responded 'just enough' to this question. The person who responded 'just enough' also commented on the fact that the buttons on the home panel don't have any text describing what the button is or does, which they don't. Overall the response to this question is pleasing. The decision to have the buttons not contain text has been discussed early on in this project.

Question 7: How much did you feel the information kiosks differs from other graphical user interfaces?

7 people responded 'not a lot' and 3 people responded 'not at all' to this question. This result is rather confusion as this question is similar to question 1, in which most people said they had never used an interface like this one before. The results to this question are therefore inconclusive and as already mentioned in question 1; it's a rather vague question. One person even commented 'what do I compare it to?'

Question 8: What did you think of the overall usability of the information kiosk?

7 people responded 'very good' and 3 people responded 'good' to this question. What was expected that most people were to respond 'very good' to this question, however it was also expect that more people would make that response especially when compared to the results of question 3 and 4 where most people responded that the user interface is simple, easy to use and easy to learn, which would directly have an effect on the usability of the question, however, when reading some of the comments to this question, people wrote 'yes! I could use it' which could suggest that people didn't fully understand the question. It might have been more beneficial to replace the word 'usability' in this question to something more definitive.

Question 9: Were there any areas of the information kiosk that you did not like?

8 people responded 'no' and 2 people responded 'yes' to this question. This result was expected, there is always something on a user interface that a user will dislike. It is very difficult to please everyone. One person commented on the fact that the buttons on the text field did not contain text again (same person who mentioned this for question 6). One person who responded 'yes' to this question mentioned that the when on the Bluetooth options panel it 'froze' the interface. This may be a technical issue that needs to be looked into. The other person who responded 'yes' to this question commented on the fact that when on the manual pin entry panel, after entering the student ID and pin number, if the pin number is incorrect that interface will clear both of the text fields rather than just the pin number text field, which means the

user must re-enter the student ID pin number, leading to wasted time and a further increase in the probably of making another mistake when entering the student ID.

This particular question made the user testing and questionnaire section worth doing, as this particular design flaw would have gone un-noticed, and yes, it is annoying to have to enter the student ID number again for no reason, particularly when the user interface clearly knows that it was the wrong pin number entered into the text field, not the wrong student ID number as it displays an appropriate message saying so. However within a university with a large number of users, it may be possible to enter someone else's ID number by mistake. So this design could actually prove helpful to the user, as this means if they have entered the wrong ID number, then they don't have to clear the text field and also and if they have entered the wrong student ID number, they may not recognise that they have done so at first glance. Therefore there are two ways of looking at this.

The person who also made the above comment also mentioned in the additional comments that the size of the text on the timetable panel was too small. Which is true, as it was necessary in order to fit a student's timetable on the panel, however this person suggested that I could make the panel be scrollable in order to allow the text to be bigger. This is a valid point and is something which was not thought of in the design or implementation stage of the Student Information Kiosk.

Question 10: Were there any areas of the information kiosk that you though were useless or not required?

10 people responded 'no' to this question, which is what was expected, as the specification for the user interface specified that it should not include any useless information purely for function quantity sake.

Overall

Overall the user testing and questionnaire was worth doing. It helped highlight some key issues of the Student Information Kiosk, in which the kiosk could now use a few tweaks to improve its user friendliness and quality. However as time is now very limited this won't be possible.

As well as the user testing and the questionnaire highlighting a number of issues with the user interface, it can also be used to measure the success of the user interface in terms of how the implementation of the system meets its design requirement; in which I am pleased to say it has turned out to be a success; with the results reflecting what the system was designed to be and thus what was expected.

The Use of Java Swing Applets

I believe that the choice made to use java's swing API to produce an applet was the right choice as compared to the other options that were mentioned in the research analysis, pre-design section of this report. The other option included a browser based service such as the use of servlets or JSP/ASP. The benefits of using this would be that a Java runtime environment doesn't need to be installed on a client machine, the client merely needs to use a standard web browser, and in using this it could run over the firewall friendly HTTP protocol on port 80 and use a mark-up language to implement the interface, which would require far less code than the current method. However I believe the use of swing is generally tidier and produces a faster interface response due to not having to load separate pages for each update when the client performs a single task. Also java is popular enough to justify installing a JRE, plus it is platform independent.

The other alternative is similar to the use of applets which is the use of a C++ or C# windows app, which means using a different programming language altogether, however seen as though the use of RMI was decided to be used earlier on in the development of this project, java had to be used as RMI is available only for java. Also I have more experience programming in java and a windows application is not platform independent.

Time and Project Management

A time and project evaluation has already been discussed in the interim report section of this report. It evaluated my time and project management skills up until completing the implementation of the Student Information Kiosk System, which described that the estimated times to complete each section of the project, went well up until the implementation of the system, which was underestimated considerably. This section evaluates my time and project management skills for the period after the implementation of the system.

I am pleased to say that it was only the implementation process that was underestimated. The rest of the sections of the report were completed on time and went as planned. The only issue is that it meant a little extra effort was needed to get the rest of the sections described in the project plan to be completed on time; however it also meant that one important part of the implementation had to be missed out.

It was originally planned that after completing the initial implementation of the system that it were to be thoroughly tested and user tested and then after which I was supposed to go back to the implementation and make some changes. However due to unforeseen time limitations this was not fully possible, however I did manage to make minor alterations to the system after the data testing. The changes that would have been made have already been discussed previously in this evaluation.

There is a learning benefit here however, which is that I now know what is involved in developing a system to this scale and how much time to account for in future development

Future Development

As well as the additions or alterations mentioned, such as the use of an SQL database and the use of sockets as opposed to RMI, and the graphical user interface alterations drawn from the user testing, there are a number of other areas where future development may require some attention.

Firstly, the Student Information Kiosk Interface may want to include more functionality. Currently the functionality it provides is to allow a student to log into the system using an RFID tag without having to enter a password, it allows the user to view messages, timetable, assignment results, view a campus map and send the above to a Bluetooth enabled device. There are a multitude of additional functions that might want to be included in the system, such as an additional means of logging into the system, such as via the use of finger print recognition as well as using a RFID scanner, or a student using their Bluetooth enabled device to pair with the system in order to log in as the system, which could be easily possible as the system already provides Bluetooth connectivity. It was mentioned in the introduction of this report that it was a possibility that this idea would be implemented.

Another use using Bluetooth might be for the system to advertise messages to users who have paired with the system in the past such as timetable reminders, where the kiosk might send a message to a device such as 'it's 11am you should be in lecture hall C0.1 for Enterprise Programming' or even send the device the assignment results as and when the result come through, so a user doesn't even need to access the student information kiosk to get the results.. Of course an additional 'preference' panel would need to be included on the information kiosk that allows a user to control if they wish to retrieve Bluetooth messages or not.

Another function could be that the Student Information Kiosk is 'location aware', therefore when viewing the campus map it would show current location. This could easily be done by marking a point on the map and telling each client to which point on the map it belongs, which could be done as a setting in part of the ClientOptions.txt configuration file, in which it includes a location key, which would be different for each client. As the system is now location away it also could incorporate some of the previously mentioned additional functions such as the Bluetooth message sent to a device stating 'It's 11am you should be in lecture hall C0.1 for Enterprise Programming and you're late and in the wrong building'.

Other functions that could be easily developed onto the existing system include an assignment results calculator and graph generator. It would add more of a graphical appeal to the interface and also provide more meaningful data about the results obtained. The calculator might be able to be used so it can calculate if a student is obtaining a 1st 2.1 or 2.2 etc class degree or how many marks they need on the next exam to gain this result.

The additional functions and extra content that could be added to the system are endless. It wouldn't be too difficult to implement additional functions without having to re-write the existing code. For example, the interface currently uses different 'panes' for tasks or what information it is displaying. Each pane is responsible for its own tasks, so therefore all that is required is to add corresponding additional 'panes' to the interface, which are viewed within a the current 'view pane' which is contained within the home panel. It's rather like a web page within a web page type of implementation; therefore all that is needed to be done is to add an additional 'web page' to view.

There is one problem with this however, that involve the current home panel buttons which are used to navigate around the information kiosk. Currently these buttons are hard coded in the system and due to size area limitations on the home panel, only 6 of these buttons can exist. Therefore if we wish to add additional panes then we need a new way of displaying these buttons. One way of implementing this might be to contain the buttons on a JSrollPane and be able to scroll up and down which would allow more buttons. This is common technique used in practically all operating systems.

Besides including more functionality, another area that might need some attention is the window size of the interface. Currently the interface has a set window size of 1024 x 600 pixels. This is the full resolution of the Samsung Q1 Tablet PC that the information kiosk has been designed to be run on. This means that the system can only be properly used on a display of that size, which doesn't generate much of a target audience for the interface. What would be expected it to be able to be use the interface on various machines. Therefore it might be beneficial to incorporate a means to have a 'size' option, where an administrator could set the window size of the interface, or better, the interface could resize itself according to the current screen size.

This addition might involve quite a bit of code, as pretty much every panel and object within the interface relies on the current 1024 X 600 window size for placeholder positions and so the interface may look a distorted when trying to alter the window size, however I don't believe it would be too tricky to make the adjustments. One reason for this is that all the font styles (which would need altering when changing the window size to maintain the text's proportion within that window) are stored within a single class, as static objects for easy access. Therefore all that is required is additional font style classes in which different fonts get loaded depending on the window size.

This is to name but a few of the ideas for future development of the interface, the list can go on.

References

Barcode Reader. (2010, June 04). Retrieved June 2010, 09, from Wikipedia: http://en.wikipedia.org/wiki/Barcode_reader

Bluetooth. (2010, June 08). Retrieved June 09, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Bluetooth

Bluetooth Basic. (2010, June 9). (Bluetooth) Retrieved June 09, 2010, from Bluetooth.com: http://www.bluetooth.com/English/Technology/Pages/Basics.aspx

Comer, D. (1996). *Internetworking With TCP/IP Client-Server Programming and Applications* (Vol. 3). Pentice Hall.

Downing, T. B. (1998). *Java RMI.* Whiley Publishing.

Fingerprint Recognition. (2010, June 05). Retrieved June 09, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Fingerprint_recognition

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* (Vol. 1). Addison-Wesley Professuibak.

Lui, S. (2009, June 15). *Bluetooth Technology*. Retrieved June 09, 2010, from Programing Tutorials: http://progtutorials.tripod.com/Bluetooth_Technology.htm

Magnetic Stripe Card. (2009, December 12). Retrieved June 08, 2010, from Wikipedia.org: http://en.wikipedia.org/wiki/Magnetic_stripe_card

McCacken, N. (2000, July 05). *Java RMI.* Retrieved June 09, 2010, from TutorialPDF: http://aspen.ucs.indiana.edu/webtech/foils2/javarmi00.pdf

Microsystems, S. (1999, April 20). *Code Conventions for the Java Programming Language*. (Sun Microsystems) Retrieved August 25, 2010, from Oracle.com: http://www.oracle.com/technetwork/java/codeconv-138413.html

MiniMag Duo. (2009, January 15). Retrieved June 08, 2010, from Kestronics: http://www.kestronics.co.uk/minimagduo-page.htm

pcProx Proximity Card Reader - USB. (2010, June 07). (Grid Connect Ltd) Retrieved June 09, 2010, from gridconnect.com: http://www.gridconnect.com/pcproxusb.html

Satpute, A. (2008, August 09). *RMI - Remote Method Invocation.* Retrieved June 09, 2010, from Career Ride: http://www.careerride.com/RMI-Defined.aspx

Shneiderman, B. (2005). *Designing the User Interface* (Vol. 4). Pearson Education, Inc.

Sorfa, P. (2002, January 10). Using and Writing Servlets. *With the power of the Java API set, servlets can provide complex, dynamic web pages and form processing.*

Tanenbaum, A. S. (2002). *Computer Networks* (Vol. 4). Pearson Education, Inc. Publishing as Prentice Hall.

Tarnay, K. (1991). *Protocol Specification and Testing.* Plenum Press.

Use of Barcode Scanners. (2010, March 8). Retrieved June 09, 2010, from National Barcode: http://www.nationalbarcode.com/info/use-of-barcode-scanners.html

Verifi P4000 Fingerprint Device. (2010, June 08). (ZVETCO Biometrics) Retrieved June 09, 2010, from zvetcobiometrics.com: http://www.zvetcobiometrics.com/Business/Products/P4000/overview.jsp?gclid=CNqdwZqik6ICFVKX2AodF35RcA

What is RFID. (2010, April 19). Retrieved June 08, 2010, from RFID Reader: http://www.rfidreader.com/

Zeiger, S. (1999, November 04). *Servlet Essentials.* Retrieved June 09, 2010, from Novacode.com: http://www.novocode.com/doc/servlet-essentials/

Literature Used to Assisting Learning of Java Programming, Sockets and RMI

Schildt, H. (2005). *Java: The Complete Reference J2SE 5 Edition.* McGraw-Hill.

Ince, D. F. (1997). *Programming the Internet with Java.* Addison Wesley Longman Limited.

Resources Used in the Development of this Project

Development Tools

Eclipse Helios Development Enviornment (IDE) with Genady's RMI plug-in

Documentation

Eldean ESS Model Generator (class model and uml creation)

Sun Javadoc Generator (Javadoc creation)

Microsoft Office Word 2007 (report write-up)

Graphics Tools

Microsoft Office Publisher 2007 (diagram creation for the write-up)

Adobe Photoshop 7.0 (image creation and modifcation for the write-up and the Student Information Kiosk Interface)

Copyright Free Images

Icon Archive - http://www.iconarchive.com

Very Icon - http://www.veryicon.com

Copyright Free Sounds

Stonewashed.net - http://www.stonewashed.net

Appendix

Completed Questionnaires

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☑ No ☐

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☑ Good ☐ Very Good ☐

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☐ Not at All ☑

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment: Yes. I could use it.

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Well done!

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes [✓] No [ ]

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad [ ] Bad [ ] Neutral [✓] Good [ ] Very Good [ ]

Comment: Perhaps a faster computer would be better

3. How did you find learning to use the information kiosk?

Very Difficult [ ] Difficult [ ] Neutral [ ] Easy [ ] Very Easy [✓]

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex [ ] Complex [ ] Neutral [ ] Simple [ ] Very Simple [✓]

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No [ ] Not Enough [ ] Neutral [ ] Just Enough [ ] Yes [✓]

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No [ ] Not Enough [ ] Neutral [ ] Just Enough [ ] Yes [✓]

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot [ ] Not a Lot [✓] Not at All [ ]

Comment: They all have words and buttons

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Liked it, nice sound effects

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☐ No ☑

Comment: I can't really think of anything similar

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☑ Good ☐ Very Good ☐

Comment:

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☑ Yes ☐

Comment: I didn't know what the buttons did

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☑ Not at All ☐

Comment:

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☑ Very Good ☐

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:..... Apart from the button .....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☒ No ☐

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☒

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☒

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☒

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☒

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☒

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☐ Not at All ☒

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☒

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☒

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☒

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes [✓] No [ ]

Comment: i guess, what kind of an interface is it?

2. What did you think of the overall response time of the information kiosk?

Very Bad [ ] Bad [ ] Neutral [ ] Good [✓] Very Good [ ]

Comment: a little slow on bringing up the home panel

3. How did you find learning to use the information kiosk?

Very Difficult [ ] Difficult [ ] Neutral [ ] Easy [ ] Very Easy [✓]

Comment: it's a simple interface

4. How did you find the structure and layout of the information kiosk?

Very Complex [ ] Complex [ ] Neutral [ ] Simple [ ] Very Simple [✓]

Comment: although at first glance, you don't know what the button do as they have no words

5. Did you find that the information kiosk got straight to the point?

No [ ] Not Enough [ ] Neutral [ ] Just Enough [ ] Yes [✓]

Comment: just press and it does

6. Did you find the amount of information that was displayed on the screen helpful enough?

No [ ] Not Enough [ ] Neutral [ ] Just Enough [ ] Yes [✓]

Comment: very informing

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot [ ] Not a Lot [✓] Not at All [ ]

Comment: i guess... like what? anything to compare it to?

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☑ No ☐

If Yes, which area?: when you enter a wrong pin number, the place where the ID number goes disappears so you have to enter it again

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

overall a very nice looking program, you've got good design skills,

There was one area I thought looked a bit odd, which was the timetable page, as the text was a lot smaller than in the rest of the program. Perhaps you could make the text bigger ~~so you~~ and you could scroll up and down. The text was a little small to read, but other than that, it's pretty good

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☑ No ☐

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☑ Good ☐ Very Good ☐

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☑ Not at All ☐

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☑ Very Good ☐

Comment: I could use it?

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Good work!

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☐ No ☑

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☑ Not at All ☐

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☑ No ☐

If Yes, which area?: froze on bluetooth options screen

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☐ No ☑

Comment: what is it like?

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☐ Not at All ☑

Comment:

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☑ No ☐

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☑ Good ☐ Very Good ☐

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☑ Very Easy ☐

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☑ Very Simple ☐

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☑ Not at All ☐

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☑ Very Good ☐

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

Student Information Kiosk Interface Questionnaire

Please tick only one box of how well you thought of a particular aspect of the Student Information Kiosk Interface unless indicated. Any other comments you wish to add will be of great help.

1. Have you ever used graphical user interface like the Student Information Kiosk before?

Yes ☐ No ☑

Comment:.....

2. What did you think of the overall response time of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☑ Very Good ☐

Comment:.....

3. How did you find learning to use the information kiosk?

Very Difficult ☐ Difficult ☐ Neutral ☐ Easy ☐ Very Easy ☑

Comment:.....

4. How did you find the structure and layout of the information kiosk?

Very Complex ☐ Complex ☐ Neutral ☐ Simple ☐ Very Simple ☑

Comment:.....

5. Did you find that the information kiosk got straight to the point?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

6. Did you find the amount of information that was displayed on the screen helpful enough?

No ☐ Not Enough ☐ Neutral ☐ Just Enough ☐ Yes ☑

Comment:.....

7. How much did you feel the information kiosks differs from other graphical user interfaces?

A Lot ☐ Not a Lot ☑ Not at All ☐

Comment:.....

8. What did you think of the overall usability of the information kiosk?

Very Bad ☐ Bad ☐ Neutral ☐ Good ☐ Very Good ☑

Comment:.....

9. Were there any areas of the information kiosk that you did not like?

Yes ☐ No ☑

If Yes, which area?:.....

10. Were there any areas of the information kiosk that you though were useless or not required?

Yes ☐ No ☑

If Yes, which areas?:.....

Additional Comments:
(Please feel free to add any additional comments about the Student Information Kiosk):

Thank you very much for taking the time to test the Student Information Kiosk Interface and for filling out this questionnaire.

User Guide

Before Running the Student Information Kiosk System

As the Student Information Kiosk system is designed to work over a network it is required that the client know the location (preferably the IP address, or simply a computer name if running over a local network) of the server and the port number that the server is running on. Currently the server is set to listen on port 1099 which should be a free port on most networks and is the default RMI port, in which this system uses the RMI protocol for client/server communication.

The clients (on the Student Information Kiosk Interface GUI and Student Records Administration GUI) are set up to connect to the server on that port and is currently set to locate the server on IP address: 127.0.0.1 which is a local loop back address which can be used if running the client and server on the same machine. If you wish to use the client and server over a network then you need to change the server IP address in the **ClientOptions.txt** file which is located in the 'SIK-Client-User' and 'SIK-Client-Admin' folder in the 'Implementation/Workspace' folder on the CD. This may be the IP address of the server or alternately the name of the computer running the RMI Server if running over a local network. If you wish to change the port number then you need to change these in both the **ClientOptions.txt** file and also the **ServerOptions.txt** file. The serverOptions.txt file is located in the 'SIK-Server folder inside the 'Implementation/Workspace' folder on the CD.

The address should look something like this:

```
//127.0.0.1:1099/RMIServer
```

'RMIServer' is the name given to the RMI Server object which is bound the RMI registry, which can be changed but it needs to be the same in both the **ServerOptions.txt** and **ClientOptions.txt** files.

Installing the 'Phidgets' Library for the use of the RFID Scanner

Before you can run the Student Information Kiosk you will need to install the Phidgets (RFID Scanner API) C++ library files to the working machine, regardless of whether the RFID scanner is to be used or not.

The installation files are found in the folder in the CD:

Implementation\Phidgets

The file that needs to be run is:

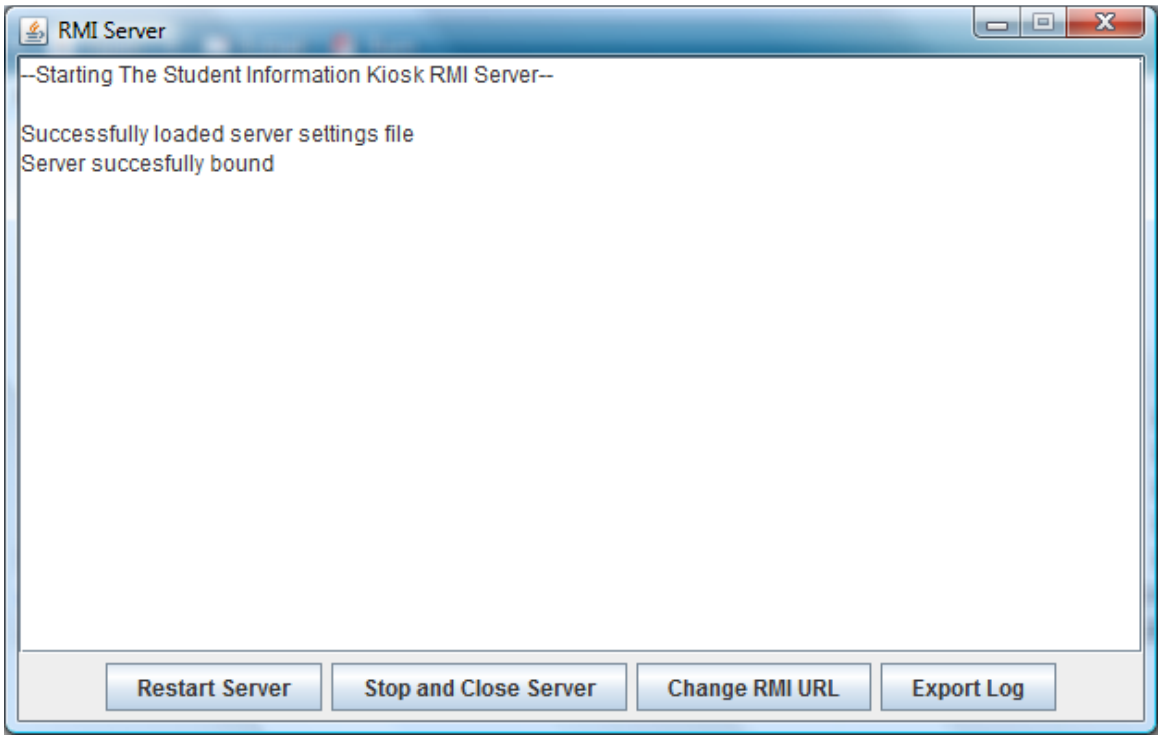
- Implementation\Phidgets \Phidget-x86_2.1.7.20100803 (if using 32-bit Windows)
- Implementation\Phidgets \Phidget-x64_2.1.7.20100803 (if using 64-bit Windows
- Phidgets\Phidgets-LinuxSource\configure (if using Linux)

Running the RMI Server (Needs to be done first before the client can connect):

Simply double click one of the following files:
- Implementation\Workspace\SIK-Server\SIK-Server.jar
- Implementation\Workspace\SIK-Server\SIK-Server.exe
- Implementation\Workspace\SIK-Server\SIK-Server.bat (will also show a consol window)

The .bat file shows a console windows as if the program was run from a consol window, in which can be used to obtain additional message of what the program is doing and will display any error messages if there are any which contain more details as compared to the error messages displayed in the GUI.

When running the RMI Server it should load up a 'server logger GUI' which can be used to control the server, export a server log and view server messages (i.e. what task the server is currently performing):



Please note, only one RMI server can be run at a time, otherwise the server cannot be successfully bound to the RMI registry, which is required in order for the RMI server object to be found by the RMI client. Additional RMI servers maybe run, but the name of the RMI server object and port number would need to be changed in the **ServerOptions.txt** file.

Running the Student Information Kiosk GUI

Simply double click on of the following files:
- Implementation\Workspace\SIK-Client-User\SIK-Client-User.jar
- Implementation\Workspace\SIK-Client-User\SIK-Client-User.exe
- Implementation\Workspace\SIK-Client-User\SIK-Client-User.bat (will also show a consol window)

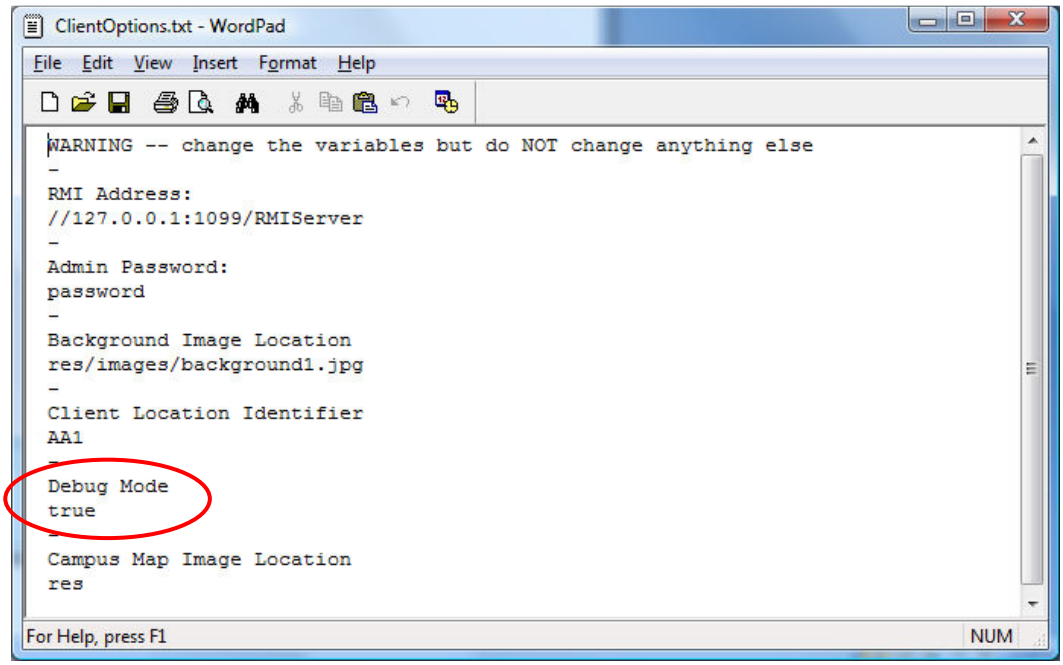
The .bat file shows a console windows as if the program was run from a consol window, in which can be used to obtain additional message of what the program is doing and will display any error messages if there are any which contain more details as compared to the error messages displayed in the GUI

For testing purposes you may use the student ID number: **4000001** (5 zeros) and pin number: **1234** when logging into the interface using the manual pin entry method.

Running the Student Information Kiosk GIU on the Samsung Q1

One thing to be aware of if attempting to the run the Student Information Kiosk GUI on the Samsung Q1 is 'window decoration'. Window decoration basically means that the application is contained within a window frame, which can then be used to move the application around the desktop, minimize and maximize the window and close the application.

The window decoration can be turned on and off for the Student Information Kiosk via the **ClientOptions.txt** file by changing the value below 'debug mode' to either true or false. When debug mode is set to true, the kiosk will include window decoration. When set to false it won't.



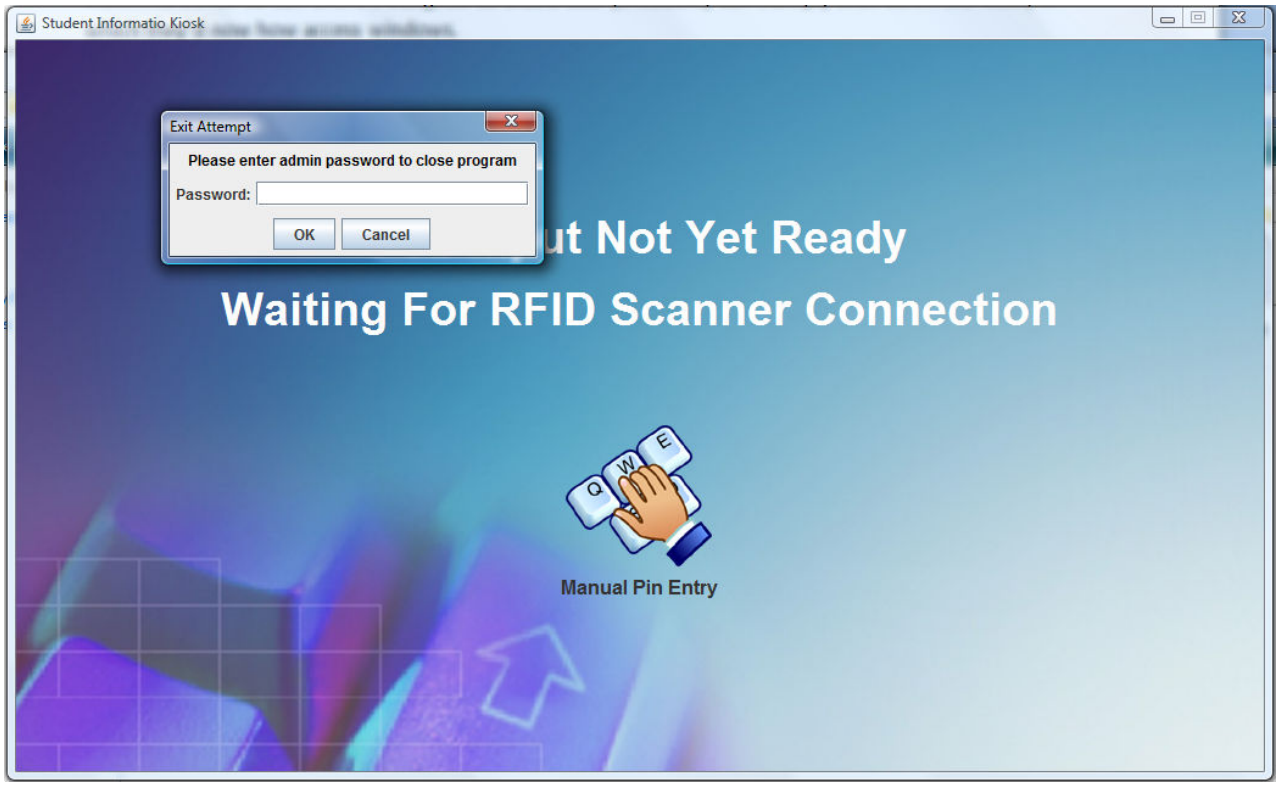
Why turn window decoration on and off? Because when running the kiosk on the Samsung Q1 you ideally want it to be full screen and it's designed so that a user (a student) can't simply shut down the kiosk, in which they'd now how access to the operating system.

Turning the Student Information Kiosk 'Off' when it is in Full Screen Mode on the Samsung Q1

Of course, when you turn the window decoration off you can no longer close the interface; which is why a hot key has been assigned to the GIU.

If you press **'q'** key on the keyboard it will display a dialog requesting a password to quit, again this is for security purposes so a user can't close down the kiosk.

The password is currently set to '**password**', which, when entered press the ok button and the interface will close:



This password can also be changed in the **ClientOptions.txt** file

Also, if you press the '**r**' key on the keyboard the same dialog will appear, however once the password has been entered another dialog will appear that will allow you to change the RMI location to the RMI server with having to alter the clientOptions.txt file.

Running the Student Records Administration GUI

Simply double click on of the following files:
- Implementation\Workspace\SIK-Client-Admin\SIK-Client-Admin.jar
- Implementation\Workspace\SIK-Client-Admin\SIK-Client-Admin.exe
- Implementation\Workspace\SIK-Client-Admin\SIK-Client-Admin.bat (will also show a consol window)

The .bat file shows a console windows as if the program was run from a consol window, in which can be used to obtain additional message of what the program is doing and will display any error messages if there are any which contain more details as compared to the error messages displayed in the GUI

Importing Projects into Eclipse

1. Open up eclipse
2. Go to file -> import
3. Within the 'General' folder, select 'Existing Project into Workspace'
4. Click 'Browse'
5. Within the 'Implementation/Workspace select which project to import and click 'import'

You can also use the 'switch workspace' option from the file menu in eclipse to point to 'Implementation/Workspace' which would present all the projects in the workspace.

Distributing the Projects

To distribute a project to a elsewhere such as to a different machine, simple go to the 'Implementation/Workspace' folder on the CD then right click and 'copy' the desired project and paste it to your desired location. Each folder in the workspace folder is an individual project that isn't referenced to any other project; therefore you can deploy just the server in one location and the information kiosk in another without having the copy the entire project**.** It is very important however if you wish to import the project into eclipse from the new location to also copy the **'SIK-Common'** folder, which is used in development and in which all other projects are referenced to it.