Booking Service on Internet to demonstrate Distributed Transaction on CORBA

Kam Shu Kai

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Master of Science

in

Computer Science

©The Chinese University of Hong Kong
August 1999

ABSTRACT

In the earliest of 21st Century, no one will argue the fact that the world of trade will be solely dominated by Electronic Commerce, complemented with advanced distributed transaction technology. As one of this technology, the Common Object Request Broker Architecture (CORBA) is an ideal middleware to integrate different parts of a system on different hardware platforms, running in different operating systems and developed on different programming languages. In this project, distributed transaction characteristics on CORBA are demonstrated through the implementation of a commercially applicable system, the Integrated Transaction Service System (ITSS).

Results presented here show that the neutrality of Interface Definition Language (IDL) and the object-oriented design approach of CORBA together with its services can support the development of sizable multi-tiered system with the advantage of ease of performance optimization, reusability, maintainability, scalability, portability and interoperability. Moreover, several suggestions have been made on how to improve the overall system performance. Although the study does not completely utilize all the services provided by CORBA, the demonstration does sufficiently convince that it is an excellent architecture for constructing the gigantic distributed system on Internet.

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude towards my supervisor, Prof. Michael R. Lyu, for his continual supports and efforts in giving me his invaluable opinions, advices and suggestions throughout the first and second terms of this final year project and the author can gain precious experience from him.

Special thanks to Mr. Leung Kin Wai, Andrew, my partner with the same project code and all my classmates who not only guide me throughout this project but also share their wisdom with me and help me in solving problems that I encountered.

Kam Shu Kai

6th August, 1999

Table of Content

1. Introduction

1.1 Motivation

In the late 1980s, most programmers were still writing standalone multi-user computer applications. Network applications were alien. In the early 1990s, no sooner had the client/server distributed system and networking technology become commercially mature and had it been proved to be more cost-effective than the traditional centralized system, many new computer systems began to develop in this direction. However, difficulties came along because of the integration of different systems in which computer programs were written in many different languages and were run on different operating system.

Around 1991, the early members of Object Management Group (OMG) proposed a sensible step, only a few years ahead of its time: instead of building software as huge monolithic chunks and regarding network connections as unusual features, software were designed as sets of independent components or objects that could interoperate (not just cooperate) with other objects regardless of whether they were located locally or remotely from them. In this architecture, network interoperability comes naturally to every component; a big step taken in anticipation of the networked world that lay ahead.

In 1992, OMG defined the standard for an Object Request Broker (ORB), a software component that resides with or near every client and object. An ORB receives invocations from a client, and delivers these to a target object. If the client and the target object do not reside on the same machines, there are two ORBs involved: the client's ORB sends the requests over the network to the ORB of the target object, which delivers it to the object itself. Client and server codes stay simple, concentrating on core business. Network complexity is dealt with by its ORB. This is the fundamental idea behind the **Common Object Request Broker Architecture** (CORBA). It is rapidly becoming the replacement protocol for the World Wide Web. A web of 'interconnected' ORBs will form the basis of how Electronic Commerce and many other applications are conducted over the Internet.

1.2 Aims of the Project

Around the world, many researchers and numerous commercial software developing teams were focusing in this area [1]. In this project, an *Integrated Transaction Service System* (ITSS) for a coliseum will be built on CORBA architecture, through which on Internet organizations can reserve the venue to hold their functions such as music concerts. In addition, an advertising service and a booking service will be born with each of them. The date to be held, the prices for the seats and the available vacancies can all be browsed. Correspondingly, buyers can purchase tickets for such functions. Its main purpose is to demonstrate the distributed transaction characteristics on CORBA. From the OMG News Fall 1998 [2], most of the successful stories on CORBA told how its ability to leverage the legacy software systems and easily integrating them with the next-generation software. This project, on the contrary, pays emphasize on *object reusability*. The service is specially designed for further extension to provide other services, e.g. traveling agency and job advertisement, with a little additional effort.

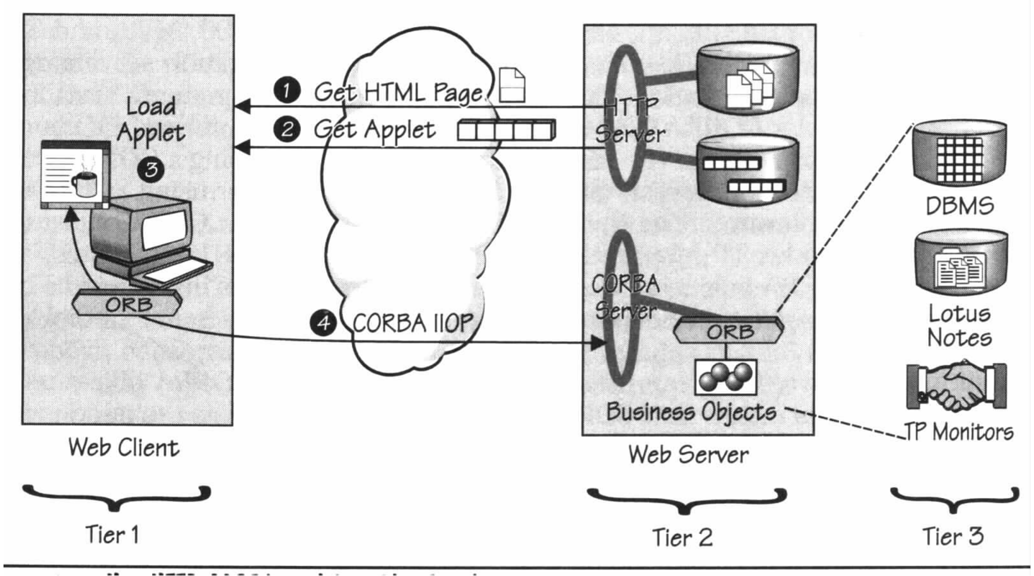


Figure 1-1 A typical 3-tier client/server application model with CORBA

Figure 1-1 above shows the design, a typical 3-tier client/server application model on the Internet, with the help of CORBA. The Web-based client interacts with its server on the Object Web as follows:

1. ***Web Browser downloads HTML page***. In our design, the page includes reference to embedded Java applets.
2. ***Web Browser retrieves Java applets from HTTP server***. The HTTP server retrieves the applet and downloads it to the browser in the form of bytecodes.
3. ***Web Browser loads applet***. The applet is first run through the Java run-time security gauntlet and then loaded into memory.
4. ***Applet invokes CORBA server objects***. The Java Applet can include IDL-generated clients stubs, which let it invoke objects on the ORB server. Alternatively, the applet can use the CORBA Dynamic Invocation Interface (DII) to generate server requests on-the-fly.

References:

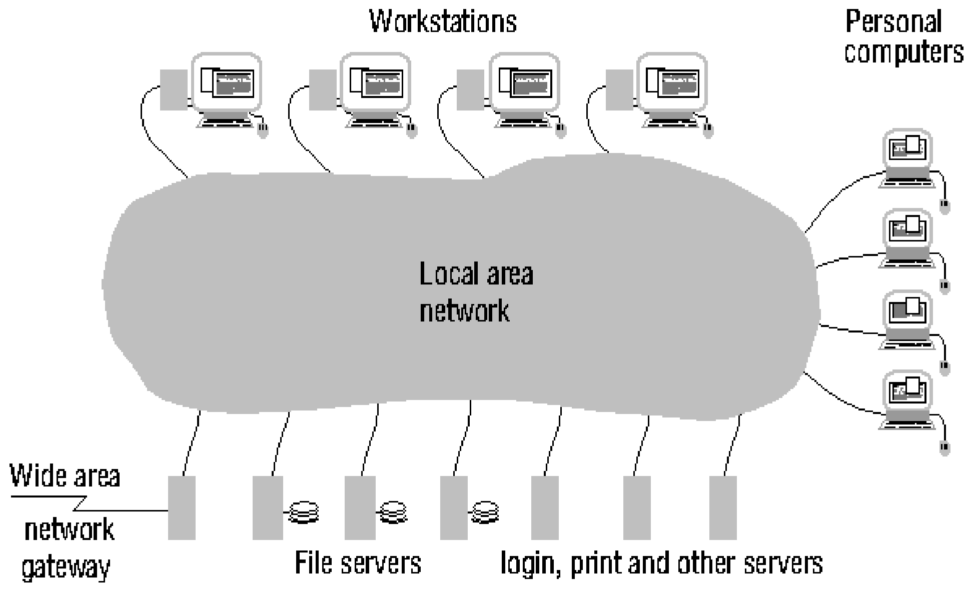
[1] CORBA Success Stories, http://www.corba.org/

[2] OMG News Fall 1998, http://www.corba.org/

2. Distributed System

2.1 Introduction

A distributed system consists of collection of autonomous computers, connected through a network and



distributed operating system software, which enables computers to coordinate their activities and to share the resources of the system - hardware, software and data, so that users perceive the system as a single, integrated computing facility [1].

Figure 2-1. A typical local area network

The development of distributed system followed the emergence of high-speed local area networks [Figure 2-1] at the beginning of the 1970s. In the early 1990s, the availability of high-performance personal computers, workstations and server computers has resulted in a major shift towards distributed systems and away centralized and multi-user computers. This trend has been speeded by the development of distributed system software, designed to support the development of distributed applications. There are many examples of commercial application of distributed system, such as the Database Management System, Automatic Teller Machine Network, and the one having numerous computers connected with the largest number of users everyday - World-Wide Web.

In a centralized system, there is a single component which possesses full control over its non-autonomous parts at all the times. If the component supports multiple users, e.g. relational database, the users share the complete component at all times. There is only a single point of control, which may be the bottleneck if the workload is heavy. Consequently, there is only a single point of failure, the most vulnerable point among all its weaknesses. The system is either running or not. If the single autonomous component fails, the whole system will not work at all.

From another aspect, a centralized system has some advantages. It runs in a single process. There is no need to take concurrency control and synchronization into account. Besides, as there are no other autonomous components, no interface is required. Thus the design is comparatively simpler than that of distributed system.

In distributed systems, there are multiple autonomous components that may be decomposed further. They possess full control over their parts at all times. Interfaces of these components must be provided for each other to use. There may be components that are not shared by all the users and resources may not be accessible. The users may use them indirectly. There are multiple points of control to avoid the bottleneck of processing but these are not totally independent. Similarly, there are multiple points of failure. If one of the machines fails, the processes running on it may be restart on other machines.

As compared with its counterpart, a distributed system runs in multiple processes. These processes are usually not executed on the same processor. Hence interprocess communication involves communication with other machines through a network. The network may have a chance of failure and it takes extra time for traveling through the network. Nevertheless, distributed systems are still more faulttolerant than a centralized one. In fact, the trade-off for its advantages comes from the design of complex communication interface.

2.2 Key Characteristics of Distributed System

Six key characteristics are primarily responsible for the usefulness of distributed system. They are resource sharing, openness, concurrency, scalability, fault tolerance and transparency. It should be emphasized that they are not automatic consequences of distribution; system must be carefully designed in order to ensure that they are achieved [1].

Resource sharing is the ability to use any hardware, software or data anywhere in the system. Resources in a distributed system, unlike the centralized one, are physically encapsulated within one of the computers and can only be accessed from others by communication. It is the resource manager to offers a communication interface enabling the resource be accessed, manipulated and updated reliability and consistently. There are mainly two kinds of model resource managers: client/server model and the object-based model. Object Management Group uses the latter one in CORBA, in which any resource is treated as an object that encapsulates the resource by means of operations that users can invoke.

Openness is concerned with extensions and improvements of distributed systems. New components have to be integrated with existing components so that the added functionality becomes accessible from the distributed system as a whole. Hence, the static and dynamic properties of services provided by components have to be published in detailed interfaces.

Concurrency arises naturally in distributed systems from the separate activities of users, the independence of resources and the location of server processes in separate computers. Components in distributed systems are executed in concurrent processes. These processes may access the same resource concurrently. Thus the server process must coordinate their actions to ensure system integrity and data integrity.

Scalability concerns the ease of the increasing the scale of the system (e.g. the number of processor) so as to accommodate more users and/or to improve the corresponding responsiveness of the system. Ideally, components should not need to be changed when the scale of a system increases.

Fault tolerance cares the reliability of the system so that in case of failure of hardware, software or network, the system continues to operate properly, without significantly degrading the performance of the system. It may be achieved by recovery (software) and redundancy (both software and hardware).

Transparency hides the complexity of the distributed systems to the users and application programmers. They can perceive it as a whole rather than a collection of cooperating components in order to reduce the difficulties in design and in operation. This characteristic is orthogonal to the others. There are many aspects of transparency, including a) access transparency, b) location transparency, c) concurrency transparency, d) replication transparency, e) failure transparency, f) migration transparency, g) performance transparency and h) scaling transparency.

Access transparency means that the operations or commands used for accessing objects are identical regardless of local or remote data access. Location transparency enables information objects to be accessed without the knowledge of their physical locations. These two transparencies usually combine as the network transparency.

Concurrency transparency enables several processes to concurrently access and update shared information without having to be aware that other processes may be accessing the information at the same time. Replication transparency enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs, such as Web pages mirroring.

Failure transparency enables the concealment of faults. Users and applications are allowed to complete their tasks despite the failure of other components. Migration transparency, an added property of location transparency, allows the movement of information objects within a system without affecting the operations of users or application programs.

Performance transparency allows the system to achieve a consistent and predictable performance level as the loads vary. Scaling transparency allows incremental growth of a system without change of its structure or application algorithms. Again the World Wide Web is the best illustration.

Designing under these six characteristics, a distributed system is capable to benefit users in lower development cost, higher system performance and better reliability over that from centralized system.

2.3 Basic Design Issues

Related to its distributed nature, design issues need to be resolved. Specifically, they are naming, communication, software structure, workload allocation and consistency maintenance.

Naming in distributed systems involves the following design considerations:

1) The choice of an appropriate name space for each type of resource. A name space may be finite or it may be potentially infinite, and it may be structured or flat. All of the resources managed by a given type of resource manger should have different names, no matter where they are located. In objected-based systems as in CORBA, all objects are uniformly named - they occupy a single naming space.
2) Resource must be resolvable to communication identifier. This is usually done by holding copies of names and their translations in a name service.

The performance and reliability of the communication techniques used for the implementation of distributed systems are critical to their performance. A design issue is to optimize the implementation of communication while retaining a high-level programming model for its use.

Openness is achieved through the design and construction of software components with well-defined interfaces. Data abstraction is an important design technique for distributed systems. Services can be viewed as the managers of objects of a given data type; the interface to a service can be viewed as a set of operations. A design issue is to structure a system so that new services can be introduced that will work fully with the existing services element. The open services brings the programming facilities of a distributed system up to the level for application programming and leave the operating system kernel services to provide the most basic of resources and services while protecting the basic hardware components from inadmissible access.

Design issue on workload allocation concerns how to deploy the processing and communication and resources in a network to optimum effect in the processing of a changing workload.

Several consistency problems arise in distributed system, such as update of data, replication of data, use of cache, failure and user interface. Their significance for design is in their impact on the performance and application. Thus the maintenance of their consistencies is important and perhaps the most difficult problem encountered in the design.

In additional to the issues above, typical user requirements must be considered in design. They are the functionality, quality of service and reconfigurability. Since distributed systems bring a richer variety of resources over the services across a network, the functionality is required to define what the system should do for users. On the other hand, quality of service defines the degree of the performance (fast response), reliability (fault tolerance) and security (privacy) while reconfigurability relates to its ability to accommodate changes on timescales., namely the short-term one in run-time condition and the medium-to-long-term one with new hardware.

Having addressed most of the design issues, CORBA provides an excellent architecture [2], together with its basic CORBAservice [3], best suited for the development of distributed system. The detail is described in depth in next chapter.

References:

[1] G. Coulouris, J. Dollimore and T. Kindberg: Distributed Systems: Concepts and

Design (2nd ed). Addison-Wesley. 1994.

[2] Common Object Request Broker Archictecture, OMG, July, 1995.

[3] Common Object Services Specification, OMG 95-3-31, 1995

3. CORBA

3.1 Background

The Object Management Group (OMG) was founded in May 1989 [1] by eight companies: 3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems and Unisys Corporation. In October 1989, OMG began independent operations as a non-profit corporation. Through the OMG's commitment to developing technically excellent, commercially viable and vendor independent specifications for the software industry, the consortium now includes over 800 members [2]. As OMG moves forward in establishing **CORBA** as the "Middleware that's Everywhere" through its worldwide standard specifications: CORBA/IIOP, Object Services, Internet Facilities and Domain Interface specifications. OMG is headquartered in Framingham, Massachusetts, USA, with international marketing partners in the UK, Germany, Japan, India and Australia.

OMG was formed to create a component-based software marketplace by hastening the introduction of standardized object software. The organization's charter includes the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development. Conformance to these specifications will make it possible to develop a heterogeneous computing environment across all major hardware platforms and operating systems. Implementations of OMG specifications can be found on over 50 operating systems across the world today. OMG's series of specifications detail the necessary standard interfaces for Distributed Object Computing. Its widely popular Internet protocol IIOP (Internet Inter-ORB Protocol) is being used as the infrastructure for technology companies like Netscape, Oracle, Sun, IBM and hundreds of others. These specifications are used worldwide to develop and deploy distributed applications for Manufacturing, Finance, Telecoms, Electronic Commerce, Realtime systems and Health Care.

OMG defines *object management* as software development that models the real world through representation of "objects." These objects are the encapsulation of the attributes, relationships and methods of software identifiable program components. A key benefit of an object-oriented system is its ability to expand in functionality by extending existing components and adding new objects to the system. Object management results in faster application development, easier maintenance, enormous scalability and reusable software.

The acceptance and use of object-oriented software is widespread and growing. Virtually every major provider and user of computer systems in the world is either using or planning to implement object-oriented tools and applications. Within the next few years, revenue from the sale of object-oriented software is projected to exceed billions of dollars in US.

3.2 Object Management Architecture

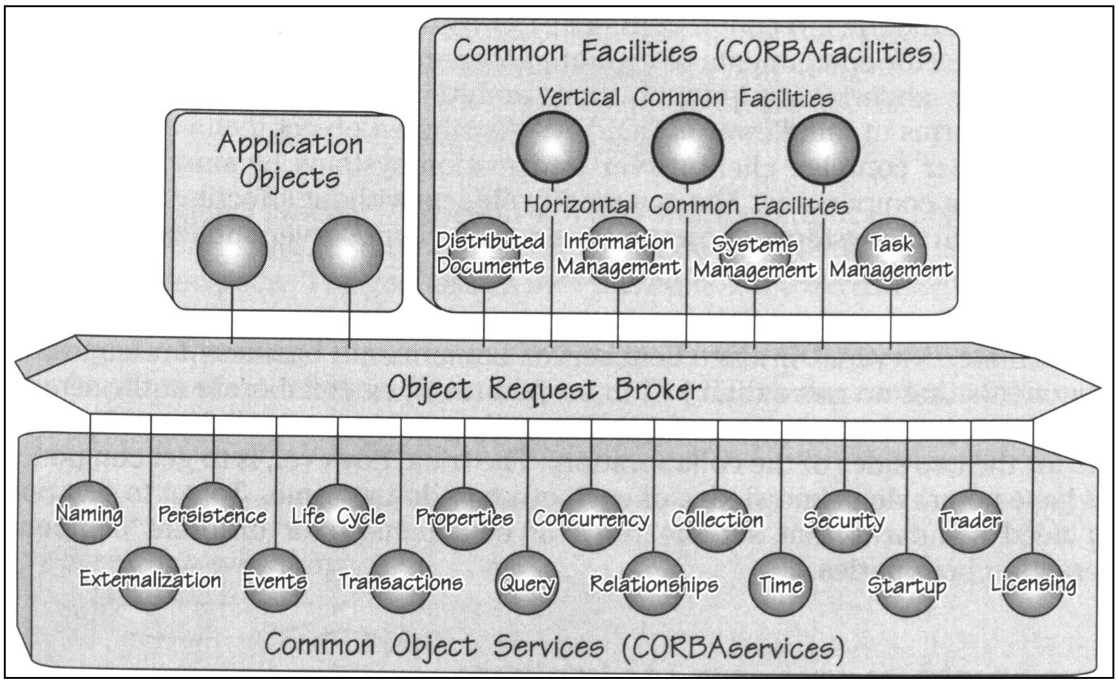


Figure 3-1 Main elements of the CORBA architecture

In the fall of 1990, the OMG first

published the Object management Architecture Guide (OMG Guide). It was revised in September 1992. The details of the Common Facilities, however, were added later in January 1995. Figure 3-1. Shows the four main elements of the architecture: 1) Object Request Broker (ORB) defines the CORBA object bus; 2) CORBAservices define the system-level object frameworks that extend the bus; 3) CORBAfacilities define horizontal and vertical application frameworks that are used directly by business objects; and 4) Application Objects are the business objects and applications - they are the ultimate consumers of the CORBA infrastructure. The following sections provide a top-level view of the four elements that make up the CORBA infrastructure.

3.2.1 Object Request Broker (ORB)

The Object request broker (ORB) is the object bus which allows objects to transparently make requests to and receive response from other objects located locally or remotely. The client is not aware of the mechanisms used to communicate with, activate, or store the server objects. The CORBA 1.1 specifications introduced in 1991 specified the Interface Definition Language (IDL) [3], language bindings and

APIs for interfacing to the ORB. CORBA 2.0 specifies interoperability across vendor ORBs.

A CORBA ORB provides a wide variety of distributed middleware services. The ORB lets objects discover each other at run time and invoke each other services. An ORB is much more sophisticated than alternative forms of client/server middleware, including the traditional Remote Procedure Calls (RPCs) [4], Message-Oriented Middleware (MOM), database store procedures, and peer-to-peer services. Benefits that every CORBA ORB provides:

Static and dynamic method invocations - A CORBA ORB allows developers to statically define method invocations at compile time, or dynamically discover them at run time. Hence, developers can get strong type checking at compile time or maximum flexibility associated with late (run-time) binding. On the contrary, most other forms of middleware only support static bindings.

High-level language bindings - A CORBA ORB allows developers to invoke methods on server objects using their own choice. CORBA separates interface from implementation and provides language-neutral data type that make it possible to call objects across language and operating system boundaries. In contrast, other types of middleware typically provide low-level, language-specific, API libraries. Also they do not separate implementation from specification. The API is tightly bound to the implementation, which makes it very sensitive to changes.

Self-describing system - CORBA provides run-time metadata for describing every server interface known to the system. Every CORBA ORB supports an Interface Repository (IR) that contains real-time information describing the functions a server provides and their parameters. The clients use the metadata to discover how to invoke services at run time. It also helps tools to generate code on-the-fly. The metadata is generated automatically either by an IDL-language precompiler or by compilers that know how to generate IDL directly form an OO language, e.g. Visigenic/Netscape's Caffeine generates IDL directly from Java bytecode. CORBA is the first and also the most mature middleware to provide this type of run-time metadata and language-independent definitions of all its services.

Local/remote transparency - An ORB can run in standalone mode on a laptop, or it can be interconnected to every other ORB in the universe using CORBA Internet Inter-ORB Protocol (IIOP) services. An ORB can broker inter-object calls within a single process, multiple processes running within the same machine, or multiple processes running across networks and operating systems. This is completely transparent to objects

Built-in security and transactions - The ORB includes context information in its messages to handle security and transactions across machine and ORB boundaries.

Polymorphic messaging -In contrast to other forms of middleware, an ORB does not simply invoke a remote function. It invokes a function on a target object, which means that the same function call will have different effects, depending on the object receives it.

Coexistence with existing systems - CORBA's separation of an object's definition from its implementation is prefect for encapsulating existing application. Using CORBA IDL, a developer can make his own existing code look like an object on the ORB, even if it is implementation in stored procedures. This enables CORBA an evolutionary solution.

3.2.1.1 Anatomy of a CORBA 2.0 ORB

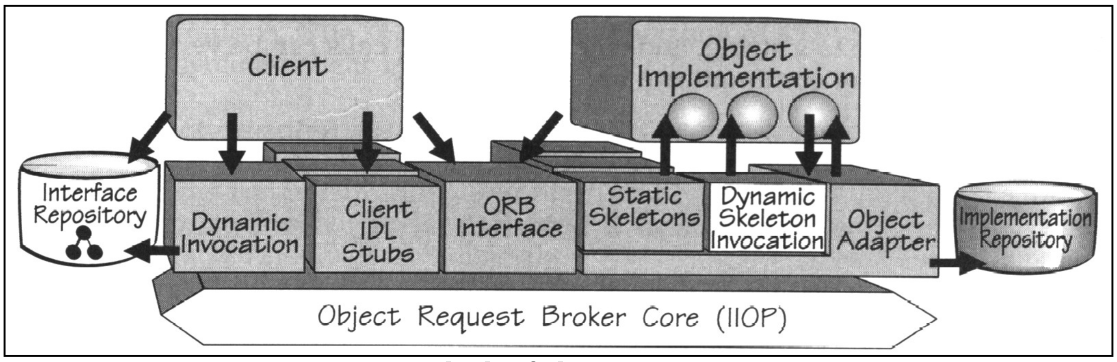


Figure 3-2 Block of elements in CORBA 2.0

Figure 3-2 shows the client and server sides of a CORBA ORB. The light areas are new to CORBA 2.0. The client does not have to be aware of where the object is loaded, its programming language, its operating system, or any other system aspects that are not part of an object's interface.

On the client side:
The *client IDL stubs* provide the static interfaces to the object services. These precompiled stubs define how clients invoke corresponding services on the servers. From a client's perspective, the stub acts like a local call. It is a proxy for a remote server object. The stub perform marshaling so that the operations and the parameters are encoded and decoded into flattened message formats to send to the server.

The *Dynamic Invocation Interface (DII)* allows the discovery of methods to be invoked at run time. CORBA defines standard APIs for looking up the metadata that defines the server interface, generating the parameters, issuing the remote call and getting back the results.

The *Interface Repository APIs* let developers obtain and modify the descriptions of all the registered component interfaces, the methods they support, and the parameters they required. CORBA calls these description *method signatures*. The Interface Repository is a run-time distributed database that contains machine-readable versions of the IDL-defined interfaces. The APIs allow components to dynamically accessed, tore, and update metadata information. This pervasive use of metadata allows every components that lives on the ORB to have self-describing interface.

The *ORB Interface* consists of a few APIs to local services that may be of interest to an application. For example, CORBA provides APIs to convert an object reference to a string, and vice versa. These calls can be very useful if the object reference store and communication is need.

On the server side
The *Server IDL Skeletons* provide static interfaces to each service exported by the server. These skeletons, like the stubs on the client, are created using an IDL compiler.

The *Dynamic Skeleton Interface* (DSI), introduced in CORBA 2.0, provides a run-time binding mechanisms for servers that need to handle incoming method calls for components that do not have IDL-based complied skeletons. The Dynamic Skeleton looks at parameter values in an incoming message to figure out the target object and method. In contrast, normal compiled skeletons are defined for a particular object class and expect a method implementation for each IDL-defined method. Dynamic Skeletons are very useful for implementing generic bridges between ORBs. They can also be used by interpreters and scripting languages to dynamically generate object implementation. The DSI is the server equivalent of a DII. It can receive either static or dynamic client invocations.

The *Object Adapter* sits on top of the ORB's core communication services and accepts request for service on behalf of the server's objects. It provides the rum-time environment for instantiating server objects, passing requests to them, and assigning them object references. The Object Adapter also registers the classes it supports and their run-time instances with the Implementation Repository (below). CORBA specifies that each ORB must support a standard adapter called the Basic Object Adapter (BOA). Servers may support more than one object adapter.

The *Interface Repository* provides a run-time repository of information about the classes a server supports, the objects that are instantiated, and their IDs. It also serves as a common place to store additional information associated with the implementation of ORBs, including trace information, audit trails, security and other administrative data.

The *ORB Interface* consists of a few APIs to local services that are identical to those provided on the client side.

3.2.2 CORBAservices

CORBAservices are collections of system-level services packaged with IDL-specified interfaces. They are used to augment and complement the functionality of the ORB. OMG has published standards for sixteen object services:

The *Life Cycle Service* defines operations for creating, copying, moving and deleting components on the bus.

The *Persistence Service* provides a single interface for storing components persistently on a variety of storage servers, including Object Databases (ODBMS), Relational Databases (RDBMSs), and simple files.

The *Naming Service* components on the bus to locate other components by name; it also supports federated naming contexts. The service also allows objects of be bound to existing network directories or naming contexts, including ISO’s X.500, OSF’s DCE, Sun’s NIS+ etc.

The *Event Service* allows components on the bus to dynamically register or unregister their interest in specific events. The service defines a well-known object called an event channel that collects and distributes events among components that know nothing of each other.

The *Concurrency Control Service* provides a lock manager that can obtain locks on behalf of either transactions or threads.

The *Transaction Service* provides two-phase commit coordination among recoverable components using either flat or nested transactions.

The *Relationship Service* provides a way to create dynamic associations between components that know nothing or each other. It also provides mechanisms for traversing the links that group these components. You can use the service to enforce referential integrity constraints, track containment relationships, and for any type of linkage among components.

The *Externalization Service* provides a standard way for getting data into and out of a component using a stream-like mechanism.

The *Query Service* provides query operations for objects. It is a superset of SQL. It is base on the upcoming SQL3 specification and the Object Database Management Group's (ODMG) Object Query Language (OQL).

The *Licensing Service* provides operations for metering the use of components to ensure fair compensation for use. The service supports any model of usage control at any point in a component's life cycle. It supports charging per session, per node, per instance creation and per site.

The *Properties Service* provides operations that let developers associate named values (or properties) with any components. Using this service, the properties can be associated with a component's state, e.g. a title or date.

The *Time Service* provides interface for synchronizing time in a distributed object environment. It also provides operations for defining and managing time-triggered events.

The *Security Service* provides a complete framework for distributed object security. It supports authentication, access control lists, confidentiality and non-repudiation. It also manages the delegation of credentials between objects.

The *Trader Service* provides a "Yellow Pages" for objects; it allows objects to publicize their services and bid for jobs.

The *Collection Service* provides CORBA interfaces to generically create and manipulate the most common collections.

The *Startup Service* enables requests to automatically start up when an ORB is invoked.

All these services enrich a distributed component's behavior and provide the robust environment in which it can safely live and play.

3.2.3 CORBAfacilities

CORBAfacilities are collections of IDL-defined frameworks that provide services of direct use to application objects. The two categories of common facilities, horizontal and vertical, define rules of engagement that business components need to effectively collaborate. In October 1994, the OMG issued the Common Facilities Request for Proposal 1 (RFP1) to obtain technology submissions for compound documents. In March 1996, OMG adopted OpenDoc as its compound document technology (Distributed Document Components, DDCF). DDCF specifies presentation services for components and a document interchange standard based on OpenDoc's Bento.

The Common Facilities that are currently under construction include mobile agents, data interchange, business object frameworks and internationalization. Like the highway system, Common Facilities are an unending project. The work will continue until CORBA defines IDL interfaces for every distributed service today, as well as ones that are yet to be invented.

3.2.4 CORBA Business Objects

Business objects provide a natural way for describing application-independent concepts such as customer, employee, account, payment and patient. They encourage a view of software that transcends tools, applications, databases, and other system concepts. The ultimate promise of object technology and components is to provide these medium-grained components that behave more like the real world does. A business object, by definition, independent of any single application, is an application-level component that can be used in unpredictable combinations. It represents a recognizable everyday life entity. In contrast, system-level objects represent entities that make sense only to information systems and programmers.

In the CORBA model, a business object consists of three kinds of objects:

a) *Business objects* encapsulate the storage, metadata, concurrency and business rule associated with an active entity. They also define how the object reacts to changes in the views.

b) *Business process objects* encapsulate the business logic at the enterprise level. In traditional Model/View/Controller (MVC) systems, the controller is in charge of the process. In the CORBA model, short-lived process functions are handled by the business object. Long-lived processes that involve other business objects are handled by the business process object. The process object typically acts as the glues that unites the other objects. For example, it defines how object reacts to a change in the environment.

c) *Presentation Objects* represent the object visually to the user. Each business object can have multiple presentations for multiple purposes. The presentations communicate directly with the business object to display data on the screen. The OMG also recognizes that there are non-visual interfaces to business objects.

Typically, a business object may have different presentation objects spread across multiple clients. The business object and the process object may reside in one or more servers. The beauty of a CORBA-based architecture is that all the constituent objects have IDL-defined interfaces and can run on ORBs [Figure3-3]. It does not matter if the constituent objects run on the same machine or on different machines. As far as clients are concerned, they are dealing with a single business object component, even though it may be factored into objects running different machines. A well-designed business object builds on the CORBA services. For example, the concurrency and transaction services can be used to maintain the integrity of the business object's state.

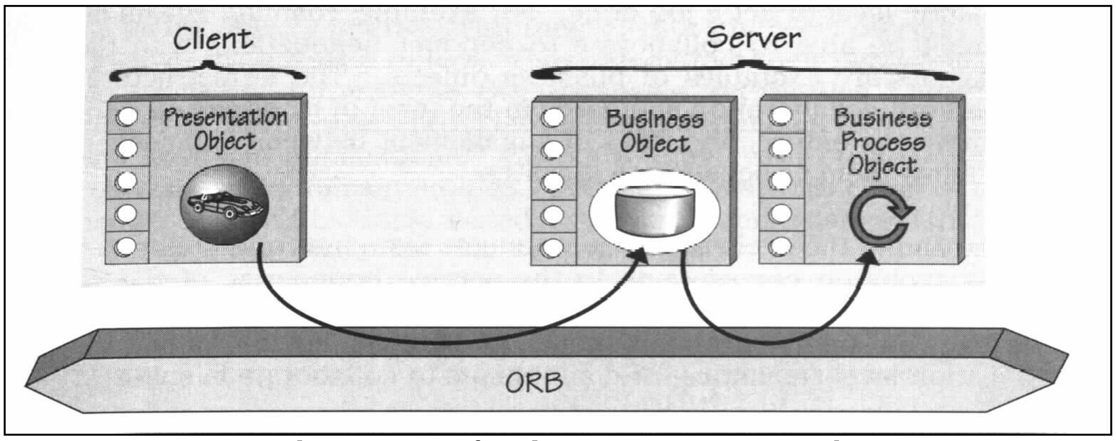


Figure 3-3. The anatomy of a Client/Server Business Object

3.3 Conclusion

Undoubtedly CORBA is the most suitable architecture for distributed systems to be built on. It is more complete and mature than other similar middlewares. Distributed CORBA objects modeled as business objects are an excellent fit for 3-tier client/server architecture. They provide scalable and flexible solutions for intergalactic client/server environments and for the Internet and Intranets. More importantly, business objects are evolutionary. The existing applications are preserved. They can be encapsulated and developer can incrementally add new intelligence, one component at a time.

References:

[1] Background of OMG http://www.omg.org/omg/background.html

[2] Member of OMG http://www.omg.org/cgi-bin/membersearch.pl

[3] IDL defined by OMG http://www.omg.org/library/idlindx.html

[4] RPC http://www-lc.llnl.gov:8080/library/all/SR-2089_9.0/

4. System Description

4.1 Introduction

In a distributed system, there may be multiple clients simultaneously communicating with the same server or one client may communicate with multiple servers, or even multiple clients simultaneously communicating with multiple servers. On Internet, however, the first case occurs more frequently. To demonstrate transaction on CORBA, a booking service is provided through the Internet which enables a party to purchase a service, advertises it and then sells it to other parties. As an example named before, an organization (first party) can book the venue of a coliseum to hold concert and then sell tickets to the public (second party). Specially designed in object-oriented approach, the booking service can be easily extended for *Integrated Transaction Service System* (ITSS). Types of venue may include cinema, swimming pool, tennis court, gymnasium etc. Other kinds of advertisement can be posted and transactions can be made through the system. The overall system design is defined in detail below.

4.2 Terminology

Before the full description of the system, some terms which will be extensively used later in this chapter are defined first.

System - refer to the software product for the Internet transaction – ITSS

Service - a general term which includes both service and goods provided by the system and other sellers

User - refer to those who are using the system. A user can have two roles, namely seller (service producer) and buyer (service consumer). A user may be a buyer and a seller at different situation

User Identity - simply denoted as UserId, together with a password is given to the user who have registered

Registered - refer to a person or a party who has already successfully applied for the use of the system. One must register before he/she can use the system.

Transaction - refer to the “buy and sell” of a service between two parties in which payment is transferred from one’s bank account to the other’s

Organization - refer to the seller of a service who have rented a venue for an activities. These activities usually imply a music concert, talk show or a match.

Super User – refer to the one who has special rights other than the ordinary users have.

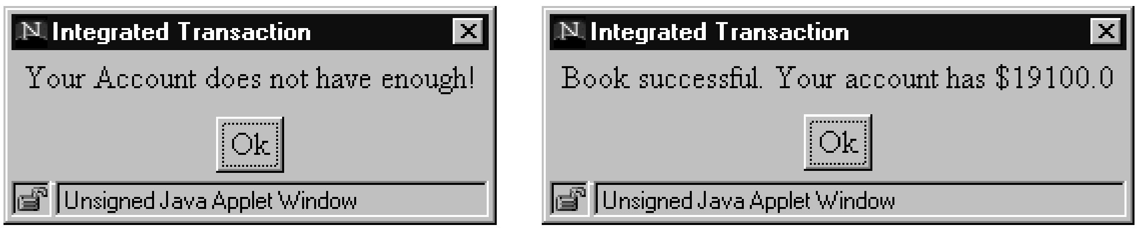
4.3 System Requirement

The system provides an Integrated Transaction Service on Internet. Users may be organizations or individual ticket-buyers. They have to apply in person (for security purpose) for a user account before they can use the service. An account together with the corresponding password will then be given. In the homepage of the system, a friendly graphical user interface is provided for the user to interact with. A user has to type in his UserId and password every time before his/her entry to the system. After the authentication, the user will be invited to a screen for browsing information as well as purchasing service.

At the highest level, the system basically provides many kinds of places such as coliseum, stadium, sport centers for different activities. On one hand, an organization can book the vacant venues for its activities. The venue fee will be deducted from the users’ bank account. Beside, the organization is invited to fill an advertisement and the system can automatically build a service to sell tickets. On the other hand, users (spectators) can browse the information about the programs and buy tickets they like, such as music concerts, football matches or tennis competitions, without any wastage of time in long queues. Consequently, the cost of the fees will be transferred from the ticket-buyers’ bank accounts to the organizations’ bank accounts. Organizations can query for what activities they hold and similarly buyers can query for the tickets they have purchased. Also they can browse their personal information and change their password, user name, contact telephone number, mailing address and Email address.

Organizations are allowed to book venues even if they have enough money in bank. Similarly, user must have enough money in bank before they can purchase the tickets. Error message boxes [Figure 4-1] and notification boxes will be shown appropriately for invalid data entry or important notifications. Moreover, the system should support multiple accesses and allow multiple servers running on different machines for fault tolerance as well as performance optimization. Below is a checklist for the system functions.

a. access of multiple users
b. only registered user can access the system
c. system still works even in case of any failure of some machine so that users should not be aware of
d. users can browse their personal information and change their password and some others non-sensitive information
e. users can purchase services easily by clicks on items they want
f. users can query for what they have brought
g. users can see the total cost of the service they have chosen in a transaction and their bank balance simultaneously
h. users are asked to confirm/reconsider before any transaction they make
i. users are notified for any error they make
j. user can see the balance after any transaction
k. users are notified if a new service is added on real time
l. only the role of 'super user' can browse all the transactions and make cancellations



Your Account does not have enough!

Book successful. Your account has $19100.0

Figure 4-1 Some Error message boxes/notification boxes

4.4 Design Details

The system consists of ten screens. They are the a) Opening Screen, b) the Main Screen, c) the Venue Booking Screen, d) the Seat Reservation Screen, e) the Service Transaction Screen, f) the Reservation Transaction Screen, g) the List Service Screen, h) List Reservation Screen, i) the Information Update Screen and j) Change Password Screen. Their functions are described below.

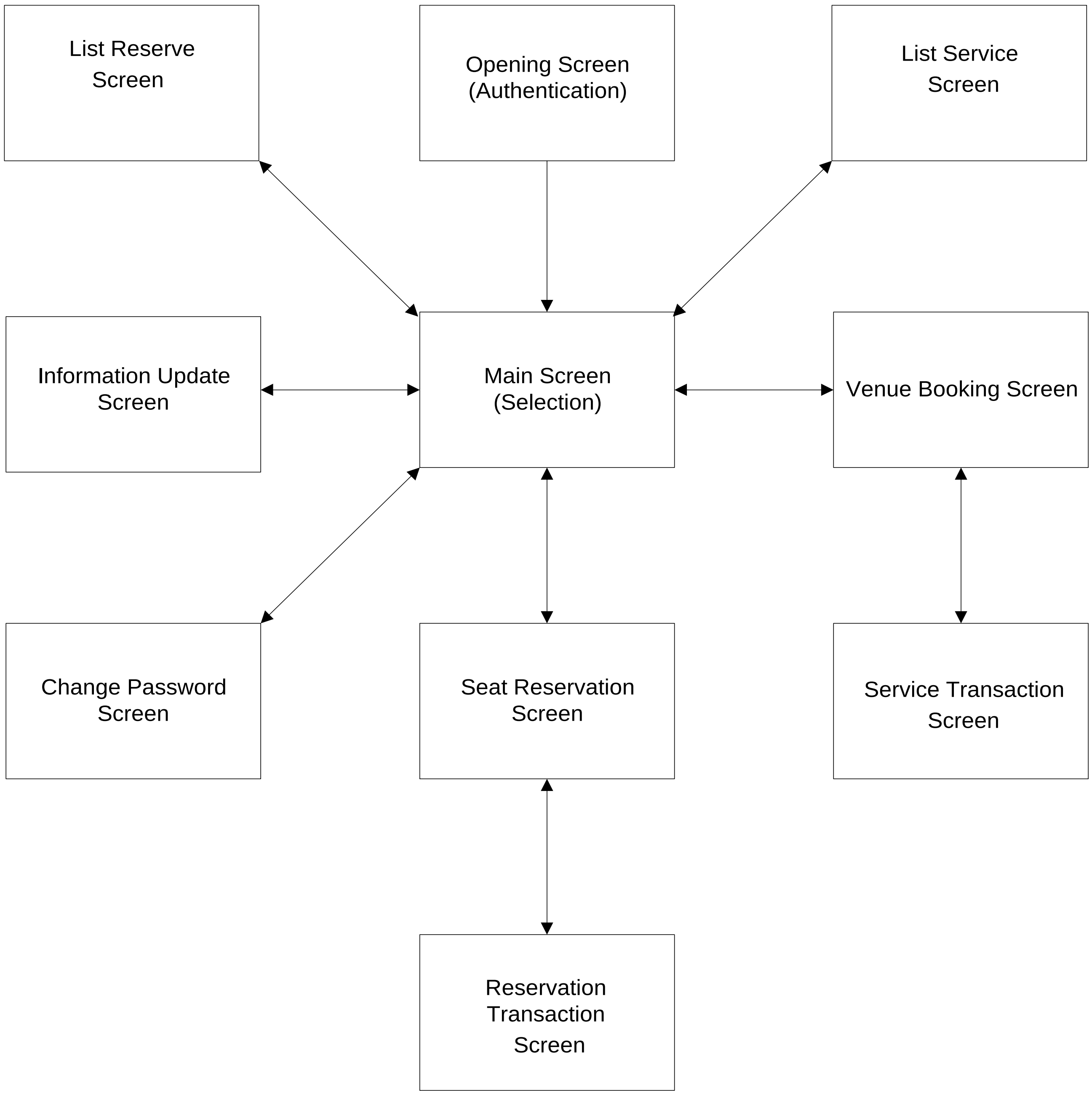


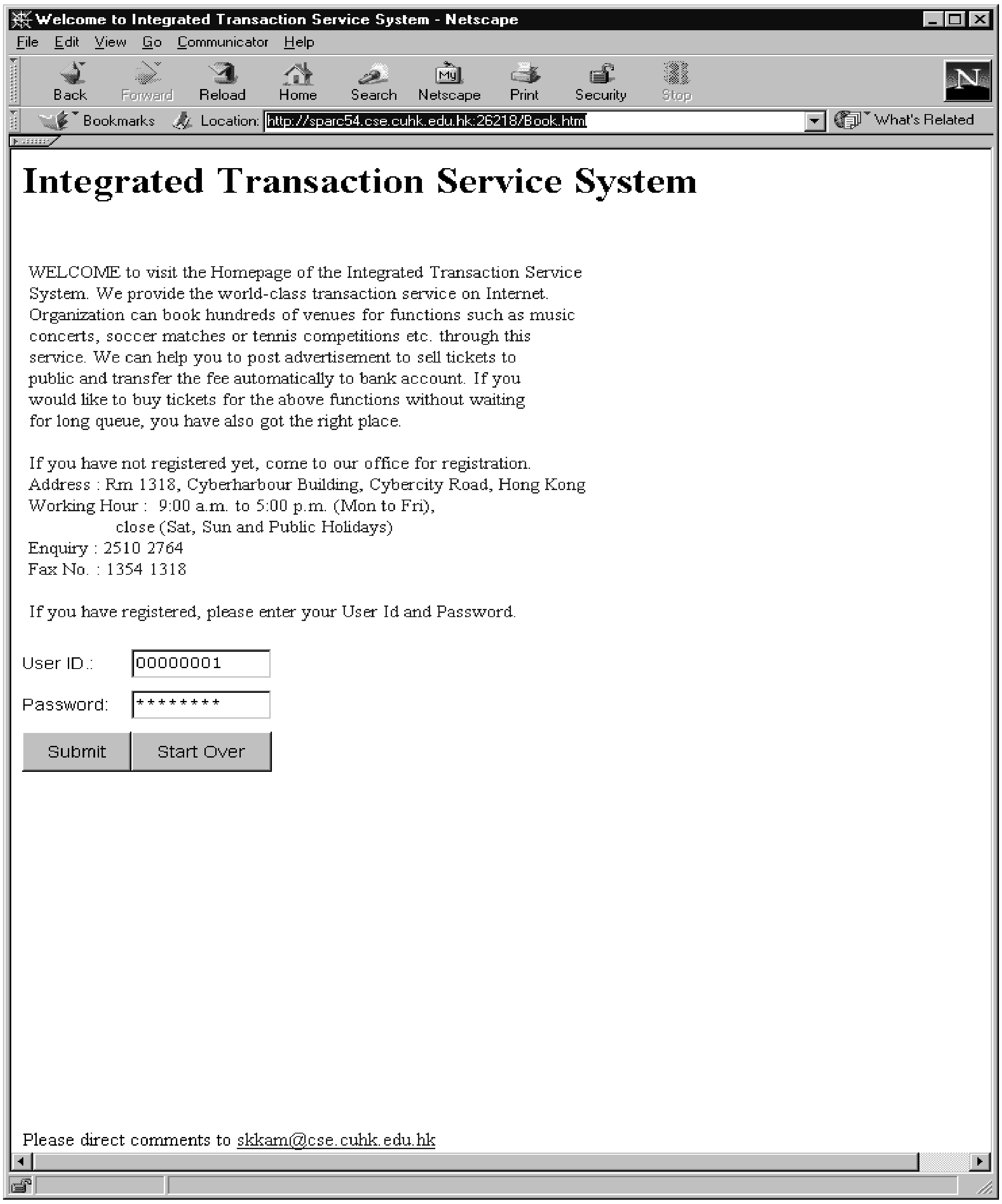
Figure 4-2 Screen for ITSS

4.4.1 Workflow

a) Opening Screen

   i. An introduction of the service and a brief description on how to use the service is shown [Figure 4-3].

   ii. User has to type in his/her UserId and the corresponding password (which will be returned in echo character '*') in order to proceed.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

WELCOME to visit the Homepage of the Integrated Transaction Service System. We provide the world-class transaction service on Internet. Organization can book hundreds of venues for functions such as music concerts, soccer matches or tennis competitions etc. through this service. We can help you to post advertisement to sell tickets to public and transfer the fee automatically to bank account. If you would like to buy tickets for the above functions without waiting for long queue, you have also got the right place.

If you have not registered yet, come to our office for registration.
Address : Rm 1318, Cyberharbour Building, Cybercity Road, Hong Kong
Working Hour : 9:00 a.m. to 5:00 p.m. (Mon to Fri),
close (Sat, Sun and Public Holidays)
Enquiry : 2510 2764
Fax No. : 1354 1318

If you have registered, please enter your User Id and Password.

User ID.: 00000001

Password: ********

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-3 Opening Screen for ITSS

b) Main Screen

i. The screen consists of a Venue List, a Program List and the following six buttons [Figure 4-4]:
   a) ‘Book Venue’ button,
   b) ‘Reserve Seat’ button,
   c) ‘List Program’ button,
   d) ‘List Reserve’ button,
   e) ‘Update Info.’ Button,
   f) ‘Change Pass.’ Button.

ii. The Venue List consists of all the venues that are available. Each row represents a venue item, including its name and its location. User can select a venue and click on the ‘Book Venue’ button to go to the Venue Booking screen to book a venue and add a new service (program).

iii. The Program list contains all the activities that are organized in the venue selected in the Venue List and will be held in dates that are not earlier than current date. Each row represents a program, showing its title and the date it held. User can select a program and click on the ‘Reserve Seat’ button to go to the Seat Reservation screen for that service where he/she is interested in to reserve the seats.

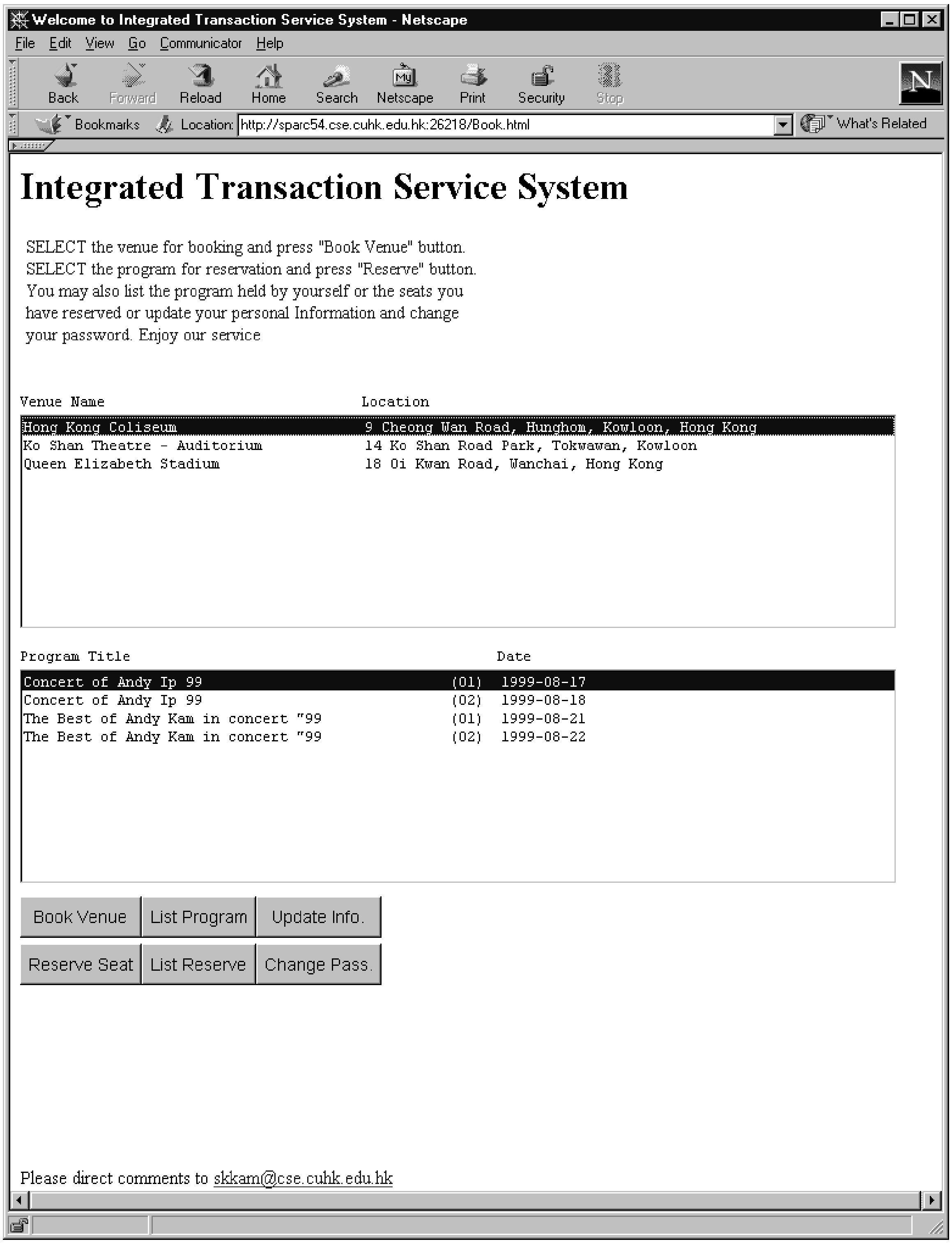
iv. User can click on ‘List Program’ button to view all the program/service held by him/her. The super user can view all.

v. User can click on ‘List Reserve’ button to view all the seats reserved by him/her. Again, the super user can view all.

vi. User can click on the ‘Change Info.’ button to go the Information Update Screen.

vii. User can click on the ‘Change Pass.’ button to go the Change Password Screen.

viii. The Program List will be re-populated automatically if a new program/service is successfully added.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

SELECT the venue for booking and press "Book Venue" button.
SELECT the program for reservation and press "Reserve" button.
You may also list the program held by yourself or the seats you have reserved or update your personal Information and change your password. Enjoy our service

```
Venue Name                                Location
Hong Kong Coliseum                        9 Cheong Wan Road, Hunghom, Kowloon, Hong Kong
Ko Shan Theatre - Auditorium              14 Ko Shan Road Park, Tokwawan, Kowloon
Queen Elizabeth Stadium                   18 Oi Kwan Road, Wanchai, Hong Kong
```

```
Program Title                                            Date
Concert of Andy Ip 99                         (01)  1999-08-17
Concert of Andy Ip 99                         (02)  1999-08-18
The Best of Andy Kam in concert "99           (01)  1999-08-21
The Best of Andy Kam in concert "99           (02)  1999-08-22
```

Please direct comments to skkam@cse.cuhk.edu.hk

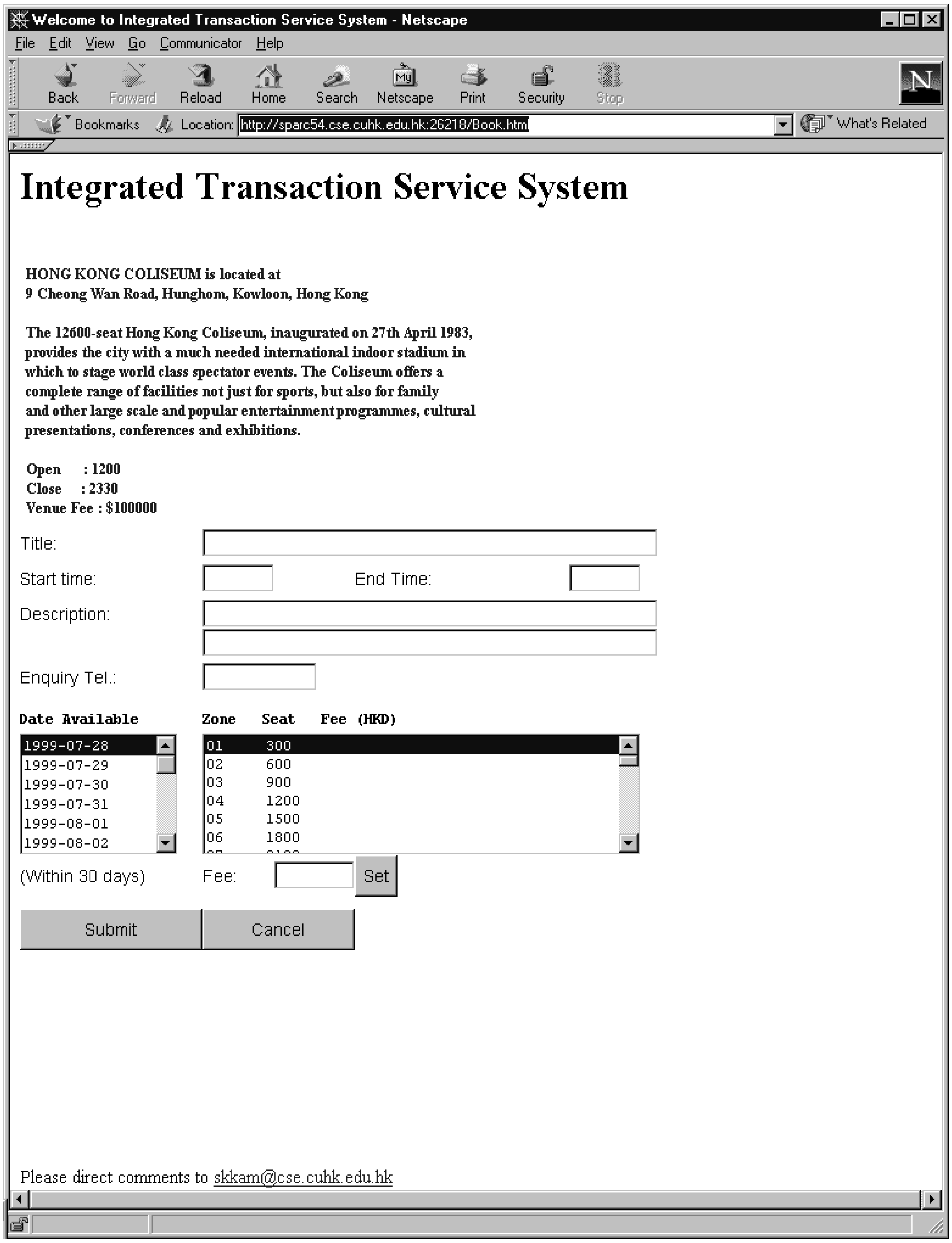
Figure 4-4 Main Screen for ITSS

c) Venue Booking Screen

   i. The screen [Figure 4-5] consists of the followings. 1) The details of the venue, including its name and location, a brief description of what kinds of activities that can be hold, opening and closing hours and the venue fee. 2) The Date Available List shows the available period within 30 days. 3) A Zone List for setting the fee for each zone. 4) A group of text boxes for the organizer to fill in the details of the program, including the Title, the start time and the end time, description and the enquiry telephone number. 4) A 'Submit' button and a 'Cancel' button.
   ii. User must fill in all the boxes correctly and select at least one date and at least set the fee for the first zone. (Refer to user guide for details)
   iii. User has to click the 'Submit' button if he/she has finished. All the fields' validations are checked at this moment. Error message will be displayed appropriately to notify the user if any validation fails. If all the validation and correct, it will go to the Service Transaction Screen for confirmation.
   iv. User can click the 'Cancel' button to go back to the Main Screen and ignore the changes.

d) Service Transaction Screen

   i. The screen on Figure 4-6 exactly displays what the user has entered in the Venue Booking Screen. The Date List show only those being chosen. Also the total fee for the dates is shown.
   ii. User can click the 'Confirm' button to make transaction. It will then be back to the main screen no matter if it is success or not. Notification will be shown to inform the result. The bank balance will be displayed for successful transaction.
   iii. User can click the 'Reconsider' button for reconsideration. It will go back to the venue Booking Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

HONG KONG COLISEUM is located at
9 Cheong Wan Road, Hunghom, Kowloon, Hong Kong

The 12600-seat Hong Kong Coliseum, inaugurated on 27th April 1983, provides the city with a much needed international indoor stadium in which to stage world class spectator events. The Coliseum offers a complete range of facilities not just for sports, but also for family and other large scale and popular entertainment programmes, cultural presentations, conferences and exhibitions.

Open : 1200
Close : 2330
Venue Fee : $100000

Title:
Start time: End Time:
Description:
Enquiry Tel.:

Date Available
1999-07-28
1999-07-29
1999-07-30
1999-07-31
1999-08-01
1999-08-02

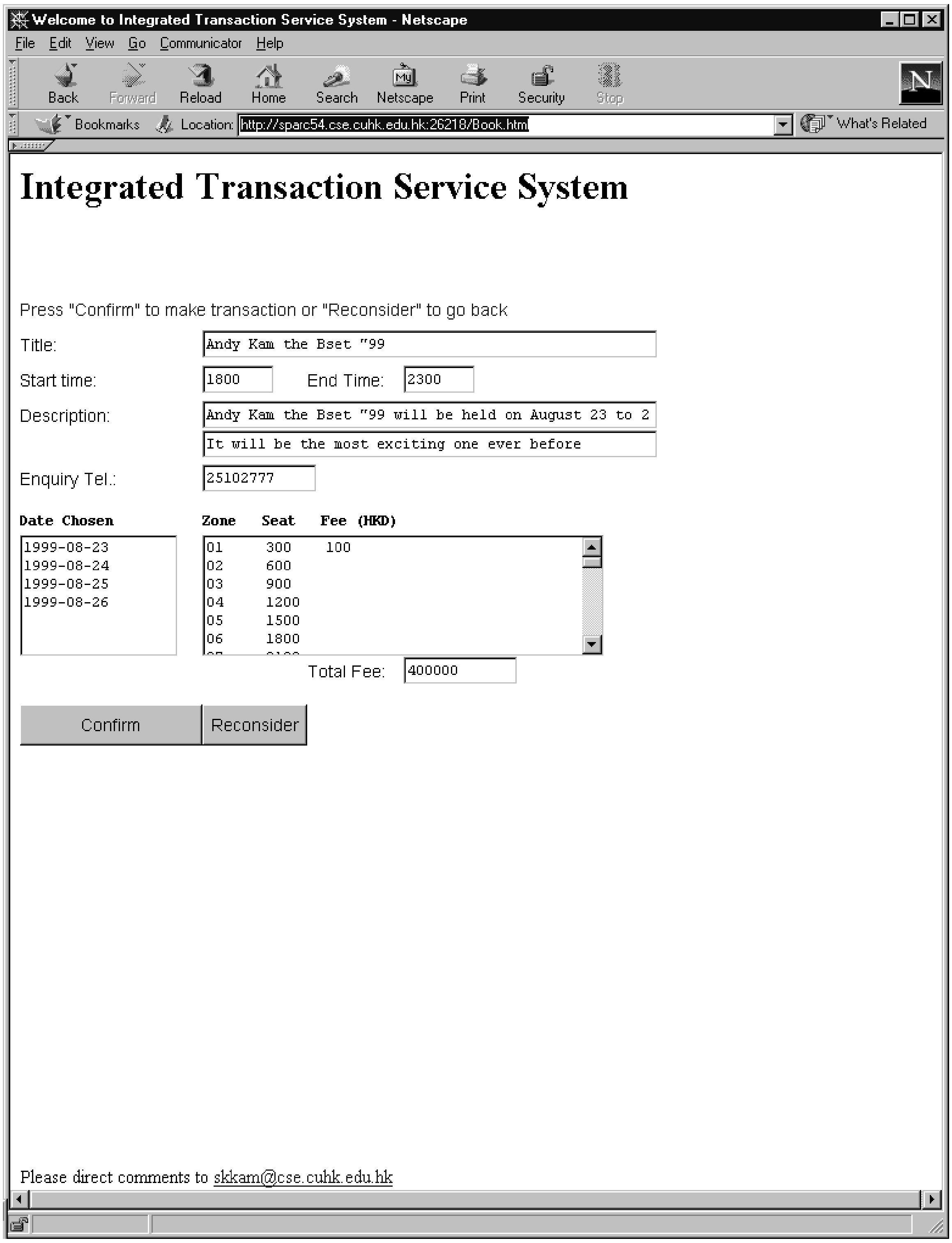
Zone	Seat	Fee (HKD)
01	300	
02	600	
03	900	
04	1200	
05	1500	
06	1800	

(Within 30 days) Fee: Set

Submit Cancel

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-5 Venue Booking Screen


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

Press "Confirm" to make transaction or "Reconsider" to go back

Title: Andy Kam the Bset "99

Start time: 1800 End Time: 2300

Description: Andy Kam the Bset "99 will be held on August 23 to 2
It will be the most exciting one ever before

Enquiry Tel.: 25102777

Date Chosen
1999-08-23
1999-08-24
1999-08-25
1999-08-26

Zone	Seat	Fee (HKD)
01	300	100
02	600	
03	900	
04	1200	
05	1500	
06	1800	

Total Fee: 400000

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-6 Service Transaction Screen

e) Seat Reservation Screen

i. The screen on Figure 4-7 shows the details of the service (program) chosen, including the followings:

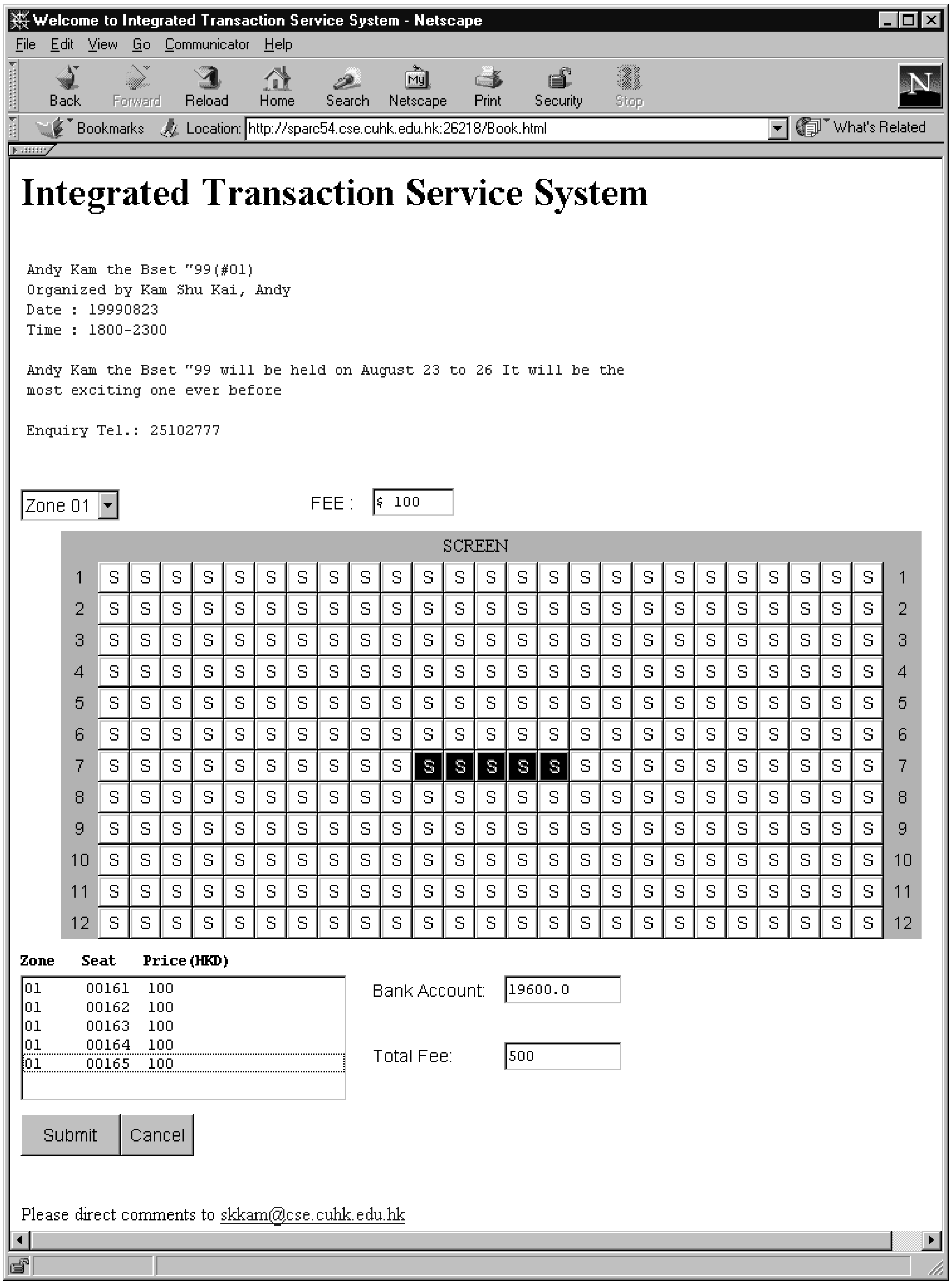
   1) title,
   2) organizer,
   3) date,
   4) start and end time
   5) enquiry telephone for details
   6) a zone choice and its corresponding fee
   7) a seating plan consist of the seats of the zone chosen
   8) a selection list to show the zone, seat number and the price
   9) bank account of the user
   10) total fee of the seats chosen

ii. User can click the zone choose any zone available. The corresponding fee and the seating plan will be shown. The seat that has already been reserved will be disabled for click and displayed in reverse color. User can reserve a seat by a click on it. It will be added to the selection list. Also the total fee will be accumulated. To cancel a selected seat, user can double click on that item in the selection list. The total fee will be adjusted accordingly.

iii. A maximum of 20 seats can be chosen at a time. Also if the bank account does not have enough, no more seats can be reserved.

iv. User has to click the ‘Submit’ button if he/she has finished. At least one seat must be reserved.

v. User can click the ‘Cancel’ button to go back to the Main Screen neglect the changes.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

Andy Kam the Bset "99(#01)
Organized by Kam Shu Kai, Andy
Date : 19990823
Time : 1800-2300

Andy Kam the Bset "99 will be held on August 23 to 26 It will be the
most exciting one ever before

Enquiry Tel.: 25102777

Zone 01 FEE : $ 100

SCREEN

1	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	1
2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	2
3	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	3
4	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	4
5	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	5
6	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	6
7	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	7
8	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	8
9	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	9
10	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	10
11	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	11
12	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	12

Zone	Seat	Price (HKD)
01	00161	100
01	00162	100
01	00163	100
01	00164	100
01	00165	100

Bank Account: 19600.0

Total Fee: 500

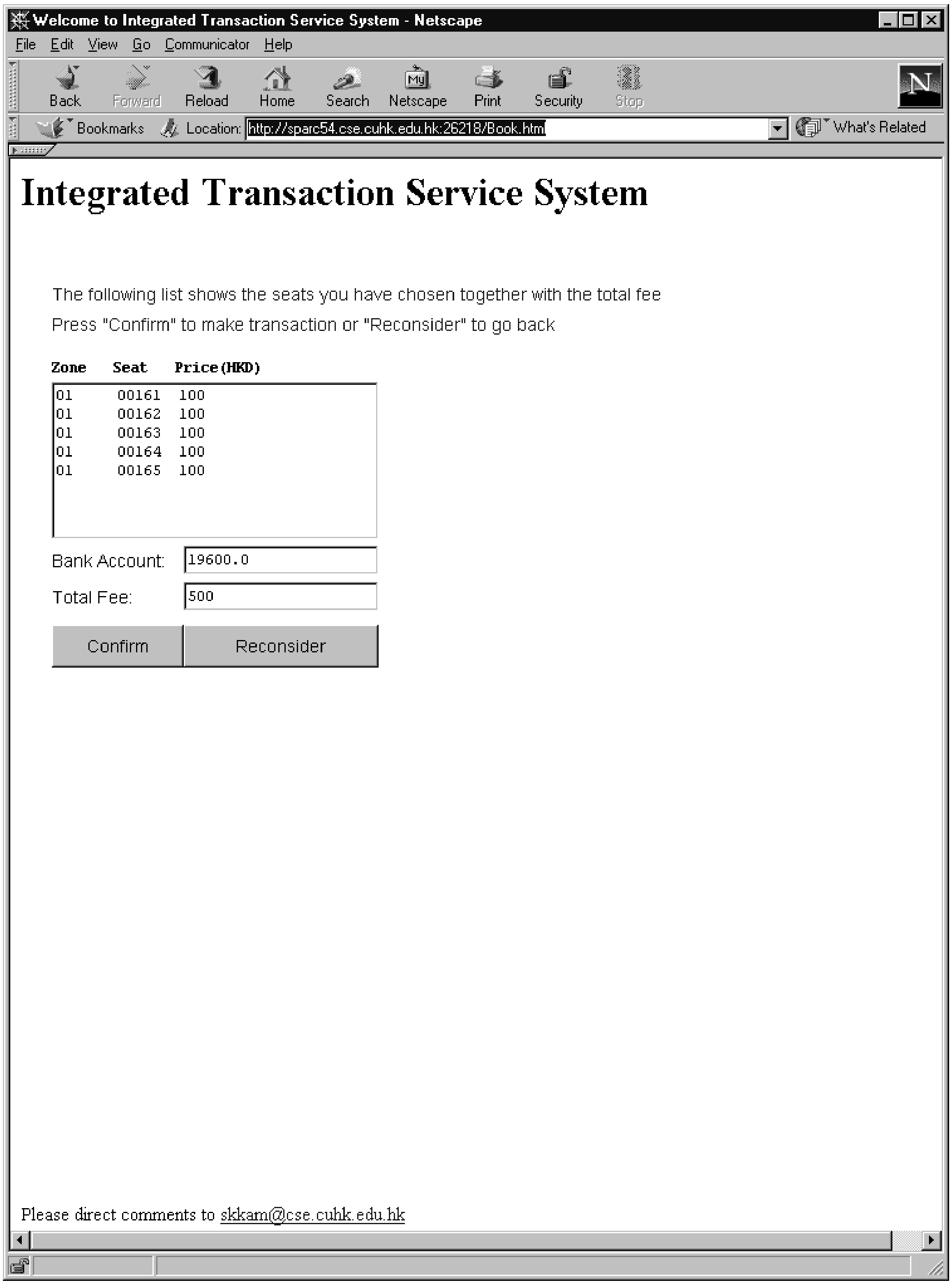
Submit Cancel

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-7 Seat Reservation Screen

f) Reservation Transaction Screen

i. The screen on Figure 4-8 shows the selection list, the bank account and the total fee as the previous one.
ii. User can click ‘Confirm’ button to make transaction. It will then be back to the main screen no matter if it is success or not. Notification will be shown to inform the result. The bank balance will be displayed for successful transaction.
iii. User can click the ‘Reconsider’ button for reconsideration. It will go back to the Seat Reservation Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

The following list shows the seats you have chosen together with the total fee

Press "Confirm" to make transaction or "Reconsider" to go back

Zone	Seat	Price(HKD)
01	00161	100
01	00162	100
01	00163	100
01	00164	100
01	00165	100

Bank Account: 19600.0

Total Fee: 500

Confirm Reconsider

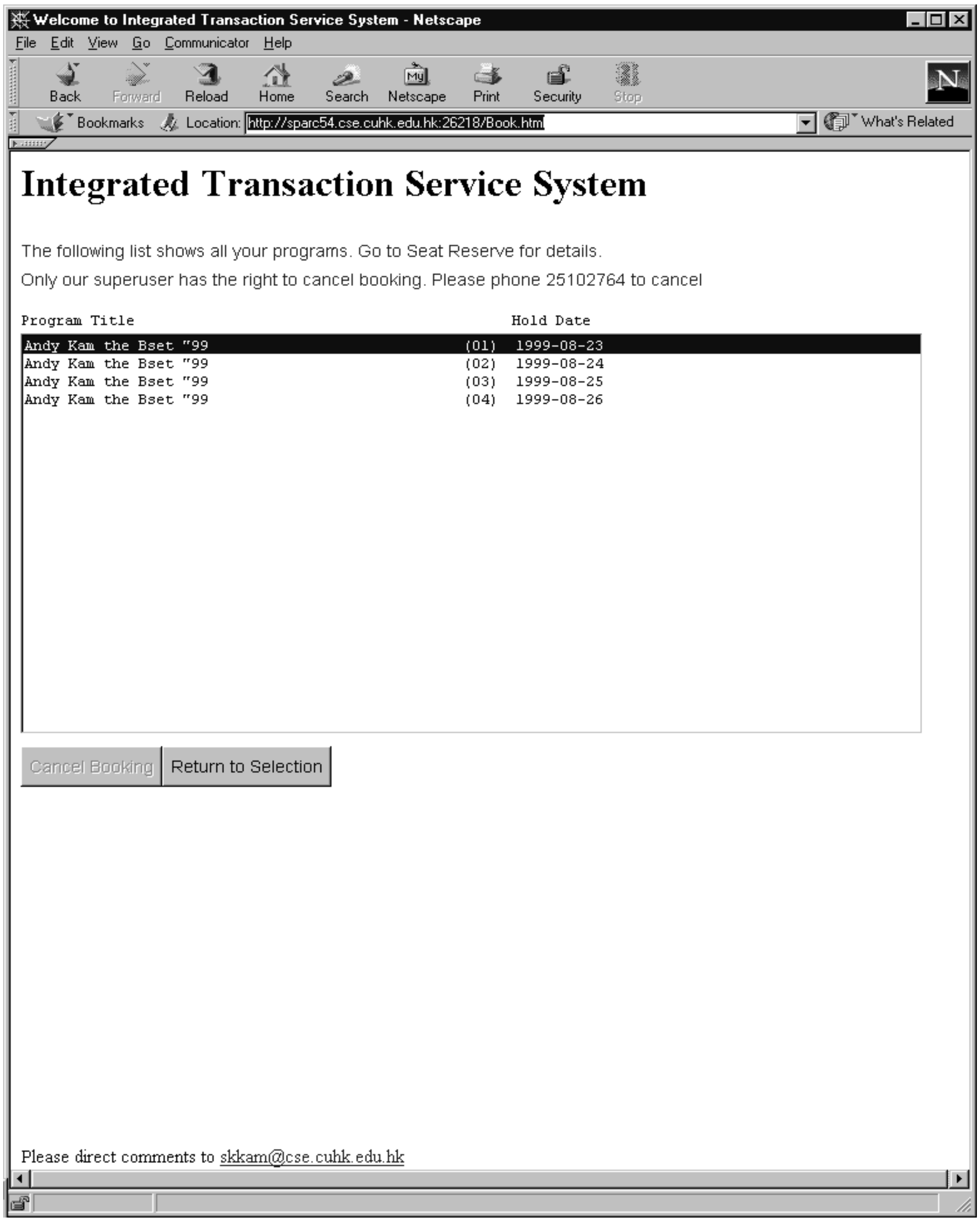
Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-8 Reservation Transaction Screen

g) List Service Screen

i. The screen on Figure 4-9 lists the program/service held by the user. The program title and the hold date are shown. Only the super user can see all the programs and make cancellation.

ii. User can click the 'Return to Selection' button to return to the main screen


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related
Cancel Booking
Return to Selection

Integrated Transaction Service System

The following list shows all your programs. Go to Seat Reserve for details.
Only our superuser has the right to cancel booking. Please phone 25102764 to cancel

```
Program Title                                         Hold Date
Andy Kam the Bset "99                            (01)  1999-08-23
Andy Kam the Bset "99                            (02)  1999-08-24
Andy Kam the Bset "99                            (03)  1999-08-25
Andy Kam the Bset "99                            (04)  1999-08-26
```

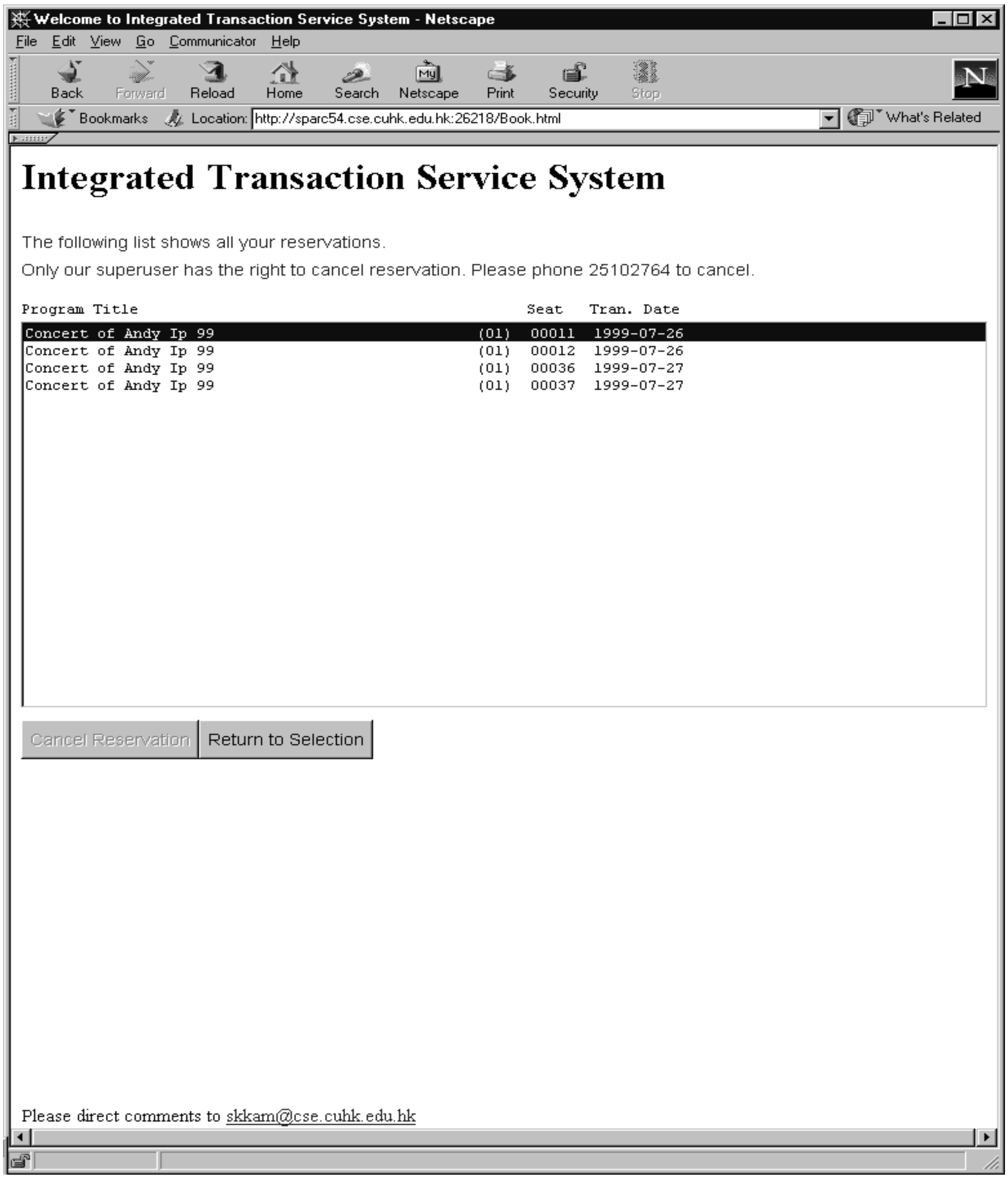
Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-9 List Service Screen

h) List Reservation Screen

i. The screen on Figure 4-10 lists all the seats reserved by the user. It shows the program title, the seat number and the transaction date. Again only the super user can see all the reservations and make cancellation.

ii. User can click the 'Return to Selection' button to return to the main screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related
Cancel Reservation
Return to Selection

Integrated Transaction Service System

The following list shows all your reservations.
Only our superuser has the right to cancel reservation. Please phone 25102764 to cancel.

```
Program Title                                       Seat   Tran. Date
Concert of Andy Ip 99                          (01)  00011  1999-07-26
Concert of Andy Ip 99                          (01)  00012  1999-07-26
Concert of Andy Ip 99                          (01)  00036  1999-07-27
Concert of Andy Ip 99                          (01)  00037  1999-07-27
```

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-10 List Reservation Screen

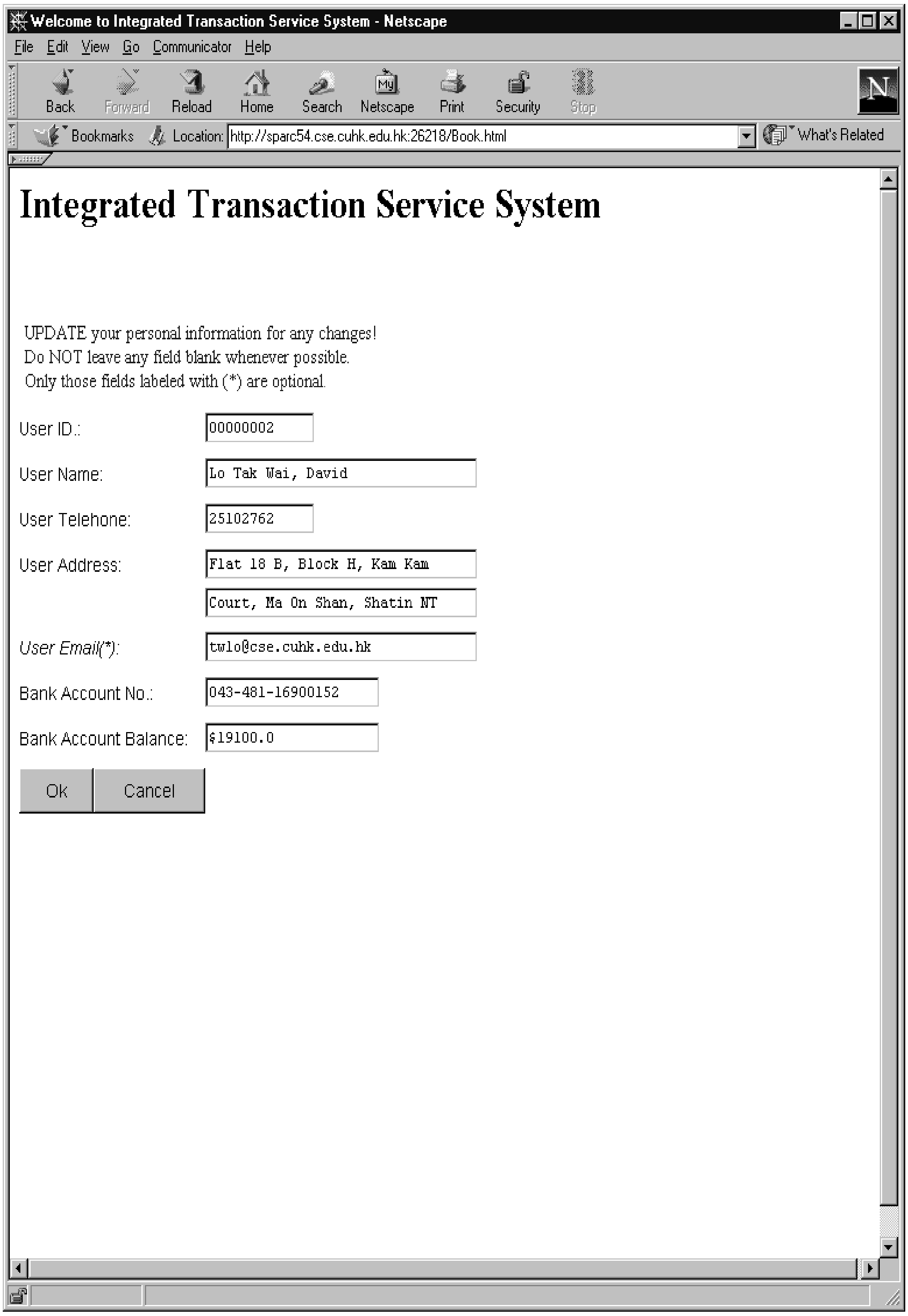
i) Information Update Screen

i. The screen on Figure 4-11 shows the user’s information: User Identity, User Name, User Telephone, User Address, User Email Address, User Bank Account Number and the Bank Balance.

ii. Only the User Name, User Telephone, User Address and User Email Address are allowed to be changed and they are all compulsory except the User Email Address.

iii. User can click ‘Ok’ button to confirm. The editable fields are checked. Error message will be displayed for any invalid entry. If success, the new information will be updated and it will be back to the Main screen.

iv. User can click the ‘Cancel’ button to ignore the changes. It will be back to the Main screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

UPDATE your personal information for any changes!
Do NOT leave any field blank whenever possible.
Only those fields labeled with (*) are optional.

User ID.:	00000002
User Name:	Lo Tak Wai, David
User Telehone:	25102762
User Address:	Flat 18 B, Block H, Kam Kam
	Court, Ma On Shan, Shatin NT
User Email(*):	twlo@cse.cuhk.edu.hk
Bank Account No.:	043-481-16900152
Bank Account Balance:	$19100.0

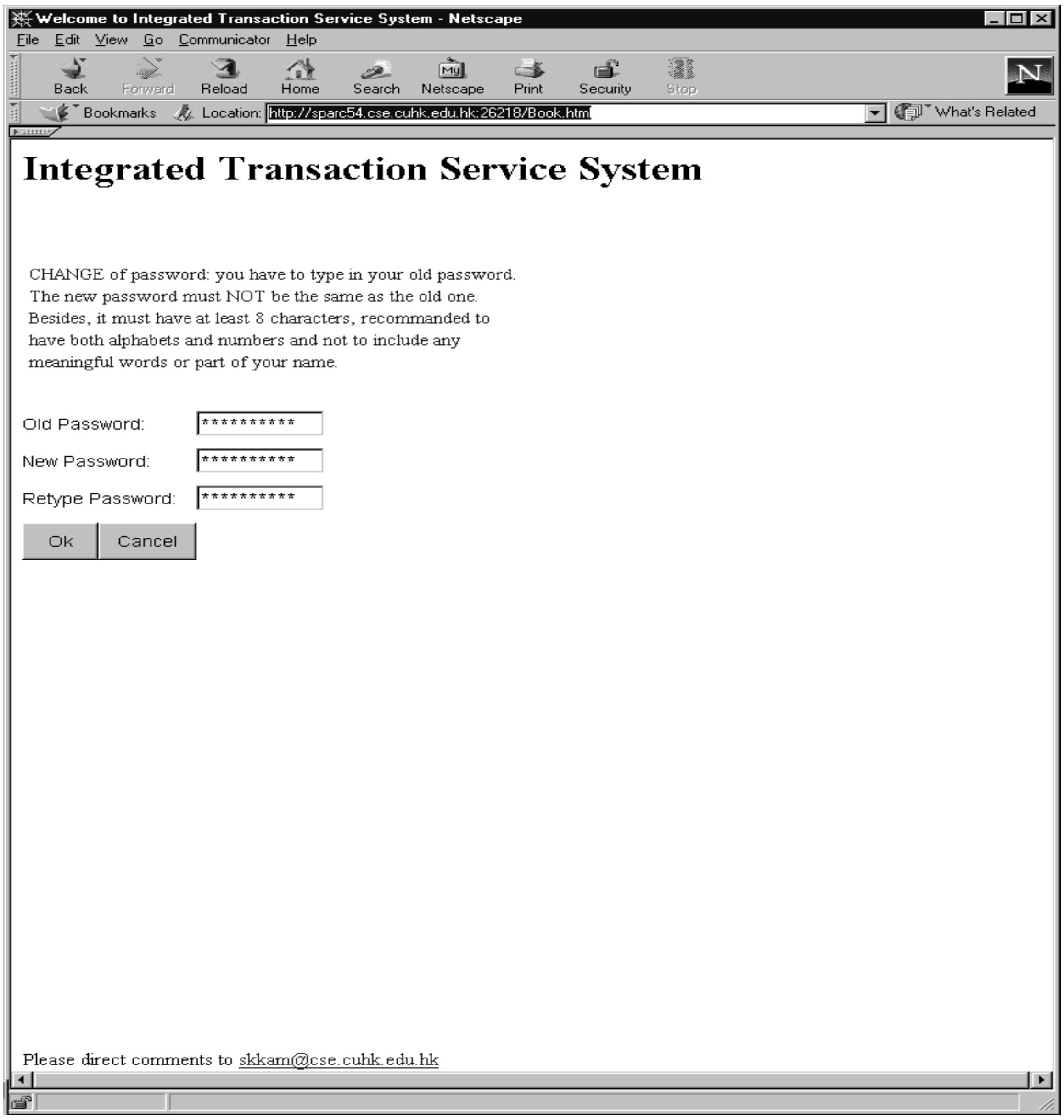
Figure 4-11 Information Update Screen

j) Password Change Screen

i. The screen on Figure 4-12 shows a brief description about changing the password. User is required to type in the old password, the new password and retype it again for security.

ii. User can click ‘Ok’ button to confirm. Error message will be displayed for invalid entry. If valid, the new password will be in effect instantly and it will be back to the Main screen.

iii. User can click the ‘Cancel’ button to ignore the change. It will be back to the Main screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.htm
What's Related

Integrated Transaction Service System

CHANGE of password: you have to type in your old password. The new password must NOT be the same as the old one. Besides, it must have at least 8 characters, recommanded to have both alphabets and numbers and not to include any meaningful words or part of your name.

Old Password:	***********
New Password:	***********
Retype Password:	***********

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 4-12 Change Password Screen

The overall system is summarized on Figure 4-2. Access to the web site will directly go to the Opening screen. Valid input of the UserId and password will lead to the Main screen, which is the responsible for selection of all functions. Venue booking or seat reservation will then comes to the Service Transaction Screens or the Reservation Transaction Screen where user can confirm to make transaction or cancel to ignore. The overall work flow is completed.

4.4.2 Database Design

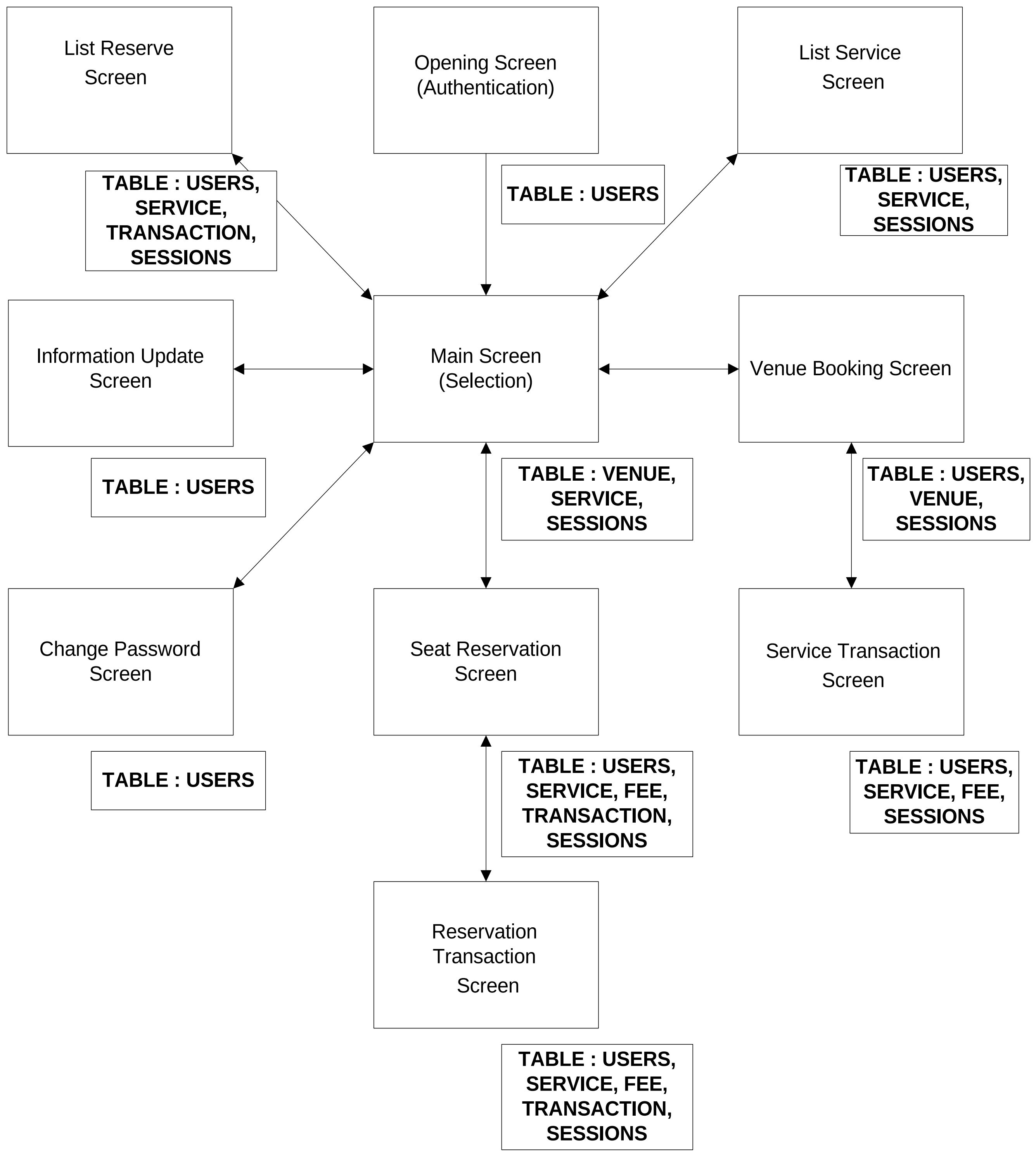

List Reserve Screen
Opening Screen (Authentication)
List Service Screen
TABLE : USERS, SERVICE, TRANSACTION, SESSIONS
TABLE : USERS
TABLE : USERS, SERVICE, SESSIONS
Information Update Screen
Main Screen (Selection)
Venue Booking Screen
TABLE : USERS
TABLE : VENUE, SERVICE, SESSIONS
TABLE : USERS, VENUE, SESSIONS
Change Password Screen
Seat Reservation Screen
Service Transaction Screen
TABLE : USERS
TABLE : USERS, SERVICE, FEE, TRANSACTION, SESSIONS
TABLE : USERS, SERVICE, FEE, SESSIONS
Reservation Transaction Screen
TABLE : USERS, SERVICE, FEE, TRANSACTION, SESSIONS

Figure 4-13 Screens and tables for ITSS

(The box on the lower right corner besides each screen shows the related tables)

The database consists of seven tables to store the information for the ITSS. Figure 4-13 shows the screens with the related tables on its bottom right. Field Names in **Bold Type** suffixed with **(P)** are primary keys. Field Description indicated with ‘display only’ implies that they are not allowed to be altered through the Internet Service for security reasons. Unique Indexes for each table are formed implicitly by the Primary Keys and other indexes added for performance improvement are shown later in the implementation part.

(i) Table Name: VENUE (Venue Information table)

Field Name	**Field Definition**	**Field Description**
VenueRefNo (P)	CHAR(4)	Venue Reference Number
Open	CHAR(4)	Venue Open Hour
Close	CHAR(4)	Venue Close Hour
Venue	CHAR(40)	Venue Name
Location	CHAR(50)	Venue Location
Description	VARCHAR2(400)	Venue Description
VenueFee	NUMBER(6,0)	Venue Fee for a Unit of time
Seat	NUMBER(5,0)	Number of seats
Zone	CHAR(2)	Number of zones
Zrow	NUMBER(2,0)	Number of seats in a row
Zcolunm	NUMBER(2,0)	Number of seats in a column
Unit	CHAR(1)	‘D’ day, ‘HD’ for half day or ‘H’ hour

(ii) Table Name: ZONE (Zone table)

Field Name	**Field Definition**	**Field Description**
VenueRefNo (P)	CHAR(4)	Venue Reference Number
Zone (P)	CHAR(2)	Zone Number
SeatMarker	NUMBER(5,0)	To mark the largest seat no. within the zone

(iii) Table Name: USERS (User Personal Information table)

Field Name	Field Definition	Field Description
UserId (P)	CHAR(8)	User Identity; display only
UserPass	VARCHAR2(10)	User Password
UserName	VARCHAR2(30)	User Name; compulsory
UserTel	CHAR(10)	Contact telephone number; compulsory
UserAddress	VARCHAR2(60)	Mailing address; compulsory
UserEMail	VARCHAR2(30)	EMail address; optional
UserBankAccNo	CHAR(14)	User Bank Account Number; display only
UserBankAccBal	NUMBER(9,2)	Organization Account Balance; display only

(iv) Table Name: SERVICE (Service Information table)

Field Name	Field Definition	Field Description
RefNo (P)	NUMBER (12,0)	Program/Service Reference Number generated by system to uniquely identify a service/program
VenueRefNo (F)	CHAR(4)	Denote the place where the program is held; reference to the VENUE table
Title	CHAR(50)	Program Title; compulsory
StartTime	CHAR(4)	Start time of the program; Compulsory with format HHMM, from 0000 to 2359; must not be earlier than open hour of venue
EndTime	CHAR(4)	End time of the program; Compulsory with format HHMM, from 0000 to 2359; must be later than close hour of venue and > StartTime
UserId(F)	CHAR(8)	User identity; who has booked the venue
TranDt	Date	Transaction Date; when the venue is booked
EnquiryTel	CHAR(10)	Enquiry Telephone No.; compulsory for enquiry purpose
Description	VARCHAR2(200)	Description of the service for advertisement; compulsory; length more than 200 will be truncated

(v) Table Name: SESSIONS (Sessions table)

Field Name	Field Definition	Field Description
RefNo (P)	NUMBER(12,0)	Program/Service Reference Number
SessNo(P)	CHAR(10)	Format: YYYYMMDDXX where YYYY represents the year, MM is the month, DD is the day and XX is the sequence no.; to identify different sessions under the same service title

(vi) Table Name: FEE (Fee Information table)

Field Name	Field Definition	Field Description
RefNo (P)	NUMBER(12,0)	Program/Service Reference Number
Zone (P)	CHAR(2)	Zone no.
Fee	NUMBER(4,0)	Fee for a seat of a program/service under a zone

(vii) Table Name: TRANSACTION (Ticket Transaction table)

Field Name	Field Definition	Field Description
RefNo (P)	NUMBER(12,0)	Program/Service Reference Number
SessNo (P)	CHAR (10)	Sessions No.
Seat (P)	NUMBER(5,0)	Seat No.
UserId(F)	CHAR(8)	Identify who reserves the ticket
TranDt	Date	Transaction Date; when the seat reserved

4.4 Development Tools

The Integrated Transaction Service system will be developed on Solaris^{TM} UNIX environment. VisiBroker for Java version 3.4 is used to build the CORBA architecture. It provides pure Java implementation of ORB and a complete IDL-to-Java language binding. In addition, it has a useful utility, the gatekeeper, which is an IIOP proxy server and can be used to warp IIOP (Internet Inter-ORB Protocol) messages, into HTTP message. The front end is written with Java Applets. The server side is also written in Java, which is deliberately chosen for its platform independence properties and its powerful Abstract Window Tools (AWT) for GUI. The Oracle8 Server, Release 8.0.5.0.0, a powerful database server, is used for storage of the persistence data. In additional to supports for traditional Relational Database, its object option supports Object-Relational Database (*). Developers can define their own data type. Varying arrays and nested table can be constructed. The server side connects to database server through Java DataBase Connectivity (JDBC). The system is designed in the Object-Oriented approach. Beside it handles well for concurrency control for multiple updates and failures recovery. The client can be run on any web browser on Internet.

Remark:

Visibroker^{TM} and Visigenic^{TM} are registered trademarks of Visigenic Software, Inc.
Solaris^{TM} and Java ^{TM} are registered trademark of Sun Microsystems.
Oracle 8^{TM} is a registered trademark of Oracle Inc.
(*) Object-Relational feature is only exists in the “i” version and not used in this project

5. Interface Definition Language

5.1 Introduction

Interface Definition Language (IDL) is a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language. In distributed object technology, it is important that new objects be able to be sent to nay platform environment and discover how to run in that environment.[1] The purpose of an IDL is to define a protocol between client and server processes so that they can communicate with each other at a level higher than simple byte strings in a heterogeneous networking environment. Each IDL has a specification, typically specified in BNF form, and a compiler. The compiler or the generator takes as its input an IDL file and generates code, usually C/C++/Java. The compiler also generates the necessary client and server stubs.

The OMG IDL is the language used to describe the interfaces that client objects call and object implementations provide. An interface definition written in OMG IDL completely defines the interface and fully specifies each operation's parameters. An OMG IDL interface provides the information needed to develop clients that use the interface's operations. Clients are not written in OMG IDL, which is purely a descriptive language, but in languages for which mappings from OMG IDL concepts have been defined. The mapping of an OMG IDL concept to a client language construct will depend on the facilities available in the client language. For example, an OMG IDL exception might be mapped to a structure in a language that has no notion of exception, or to an exception in a language that does. As a core part in CORBA applications, IDL must be carefully designed.

5.2 Design Details

The following is the IDL for the ITSS. It contains two modules: the *Client* and the *InterenetTran*. For the Client module, it has only one interface with one operation for callback purpose. The server-side can use the '*inform*' operation to notify the client. The *InterenetTran* module is the main part of the system. It populates the interface with operations for the client to invoke. It has two interfaces: *BookDispenser* and *Book*. The *BookDispenser* object is responsible for validation of the users' ID and password and registration of valid clients. It also allocates and releases the Book object to serve the client. The Book object provides a list of functions to serve the client. Besides, several s*tructs* and *sequences* are defined for the ease of manipulation.

```
// Book.idl
module Client
{
   interface ClientControl
   {
            // to inform the client if new changes
            boolean          inform(in string message);
   };
};

module InternetTran
{
   exception BookException
   {
            string reason;
   };

   typedef string SessList[10]; // Dates (sessions) booked,
                                // at most 10
   typedef unsigned long FeeList[99]; // at most 99 zones
   typedef unsigned long SeatList[20]; // at most 20 seats

   struct VenueStruct
   {
        string VenueRefNo;
            string Venue;
            string Location;
   };

   struct VenueDetailStruct
   {
            string VenueRefNo;
            string Open;
            string Close;
            string Venue;
            string Location;
            string Description;
            unsigned long VenueFee;
            unsigned long Seat;
            string Zone;
            unsigned long ZRow;
            unsigned long ZColumn;
            string Unit;
   };

   typedef sequence<VenueStruct> VenueSeq;

   struct ZoneStruct
   {
      string Zone;
            unsigned long SeatMarker;
   };
   typedef sequence<ZoneStruct> ZoneSeq;
   struct UserStruct
   {
            string UserId;
            string UserPass;
            string UserName;
            string UserTel;
            string UserAdd;
        string UserEmail;
            string UserBankAccNo;
            double UserBankAccBal;
   };

   struct ServiceStruct
   {
            long long RefNo;
        string SessNo;
            string VenueRefNo;
            string Title;
   };

   typedef sequence<ServiceStruct> ServiceSeq;

   struct ServiceDetailStruct
   {
            long long RefNo;
        string SessNo;
      string VenueRefNo;
            string Title;
            string StartTime;
            string EndTime;
            string Organizer; // held by which organization
      string EnquiryTel;
            string Description;
   };

   struct SeatDetailStruct
   {
            unsigned long Seat;
            unsigned long ZRow;
            unsigned long ZColumn;
   };

   struct FeeStruct
   {
            string Zone;
            unsigned long Fees;
   };

   typedef sequence<FeeStruct> FeeSeq;

   struct TranStruct
   {
            long long RefNo;
        string SessNo;
        string Title;
            unsigned long Seat;
            string TranDt;      // which date the user reserve
   };
   typedef sequence<TranStruct> TranSeq;

   struct SeatStruct
   {
            unsigned long Seat;     // the seat no.
   };

   typedef sequence<SeatStruct> SeatSeq;

   struct SessNoStruct
   {
            string SessNo;
   };

   typedef sequence<SessNoStruct> SessNoSeq;

   interface Book
   {
            VenueSeq        getVenueList();
            VenueDetailStruct   getVenue(in string VenueRefNo);
            ZoneSeq               getZoneList(in string VenueRefNo);
            UserStruct      getUser(in string UserId);
            double                getUserBankAccBal(in string
UserId);
            ServiceSeq      getServiceList(in string VenueRefNo);
            ServiceSeq      getServiceListAll();
            ServiceSeq      getServiceListUserId(in string UserId);
        ServiceDetailStruct getService(in long long RefNo, in string
            SessNo);
            SeatDetailStruct    getSeat(in string VenueRefNo);
            SeatSeq               getSeatList(in long long RefNo, in
string
            SessNo);
            FeeSeq                getFeeList(in long long RefNo);
            TranSeq               getTranList(in string UserId);
            TranSeq               getTranListAll();
        SessNoSeq     getSessNoList(in string VenueRefNo);
            boolean               setUser(in UserStruct User);
            boolean               setPass(in string UserId, in string
Password)
                        raises (BookException);
        boolean         addService(in string VenueRefNo,

                    in string Title,
                          in string StartTime,
                        in string EndTime,
                        in string UserId,
                        in unsigned long VenueFee,
                        in string EnquiryTel,
                        in string Description,
                        in SessList sslist,
                        in FeeList flist, out string resaon);

            boolean             addTran(in long long RefNo, in string
SessNo,
                    in string UserId, in SeatList stlist,
                    in unsigned long Total, out string reason);
            boolean     cancelBook(in long long RefNo,in string
SessNo,
                    out string reason);
            boolean     cancelRes(in long long RefNo, in string
SessNo,
                        in unsigned long Seat, out string reason);

   };
   interface BookDispenser
   {
            boolean     register(in string UserId,
            in Client::ClientControl clientObjRef)
                      raises (BookException);
            void notifyOther(in string UserId, in string Happen )
                raises (BookException);
            boolean isValid(in string UserId, in string UserPass,
                out string message )
                      raises (BookException);
            Book reserveBookObject(in string UserId )
                raises (BookException);
            void releaseBookObject(in Book BookObject)
                raises (BookException);
   };

};
```

Reference:

[1]   http://whatis.com/idl.htm

6. Implementation

6.1 System Overview

As discussed before, the ITSS is a 3-tier Client/Server application, a special type of client/server architecture consisting of three well-defined and separate processes, each running on a different platform:

a. The user interface, which runs on the user's computer (the client).

b. The functional modules that actually process data. This middle tier runs on a server and is often called the application server.

c. A database management system (DBMS) that stores the data required by the middle tier. This tier runs on a second server called the database server.

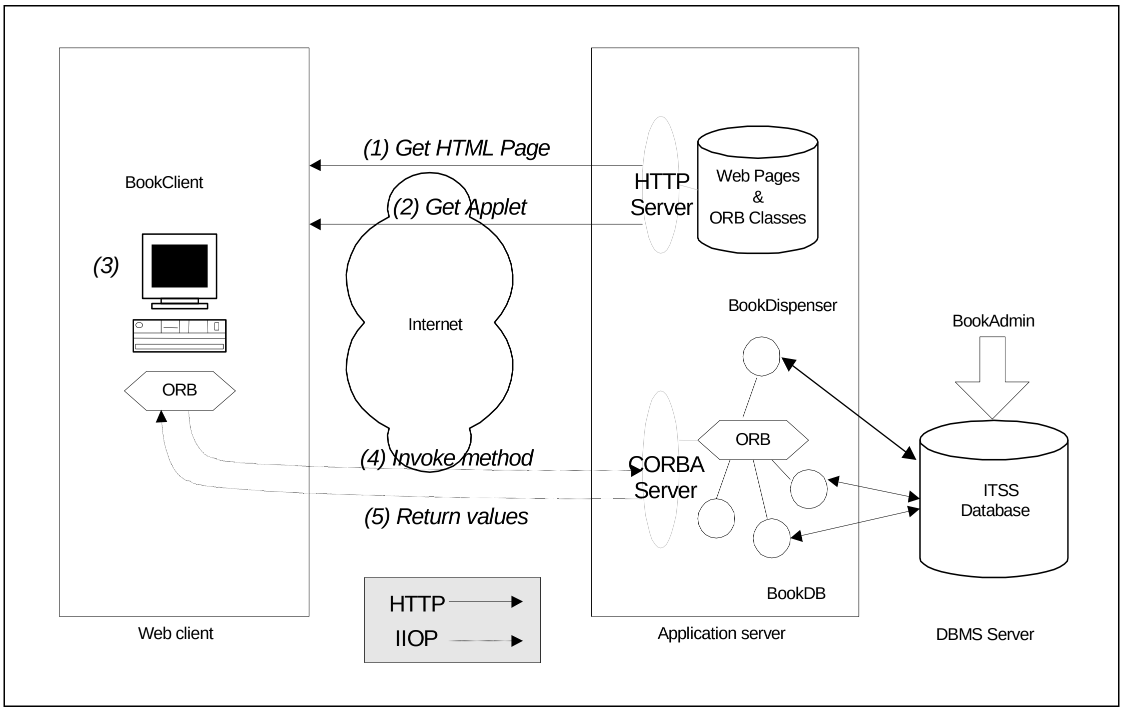


Figure 6-1 The top-level architecture of the 3-tier ITSS

(For simplicity, the callback service for the server to inform the client is not shown)

The figure above shows the overall architecture of the system. It consists of the following objects. The front tier is a client called *BookClient*, which is responsible for handling the user interface. It contains a *ClientControl* thread to listen for any notification from the server. It starts to run after the user has successfully logon. The middle-tier server consists of four main classes: 1) a *BookDispenser* that manages a pool of server objects, 2) a helper object of the class SelectIDBookDB which is dedicated for *BookDispenser* to communicate with the database server (only for validation of the login ID), 3) a pool of server objects of the class *Book*, and 4) a pool of helper objects of the class BookDB – these are the worker objects for *Book* objects to communicate with the database server through persistent JDBC connections. The third tier consists of the JDBC databases – this is where the persistent state is stored.

The interactions between the client and application server and database server is illustrated as follows:

1. **Web browser downloads HTML** — the page includes reference to embedded the client Java applet.
2. **Web browser retrieves Java applet and ORB classes from HTTP server** — The HTTP server delivers the applet and ORB classes into the browser in the form of Java bytecodes.
3. **Web browser loads and starts applet** — The applet is first run through the Java run-time security gauntlet and then loaded into memory.
4. **Applet invokes CORBA server objects** — The Java applet includes IDL-defined objects. Invoking methods on these objects will be directed to the server implementation through IIOP. In fact, the client does not communicate with the object server directly due to the Java sandbox model restriction. The Gatekeeper makes it possible for the client to invoke an object server on a host other the one from which the applet originated. The Gatekeeper also wraps IIOP messages into HTTP message while transmitting through Internet.
5. **CORBA server return result values** — The return value of the method and values of parameters defined to be "out" type are sent to client through IIOP. Again, the IIOP messages are wrapped in HTTP messages during the transmission.

6.2 Client side of ITSS

Written in Java, the *BookClient*, which provides a user-friendly interface, acts as the front-end of ITSS. It extends the **java.applet.Applet** and implements **ActionListener** and **ItemListener** to trap the event created due to the users' action. It consists of ten different *Panels* objects for display of different layouts on each screen, a control thread to receive commands from the server and a dialog to display error or notification messages.

6.2.1 Class Hierarchy and Methods Implementation of Client

The *init()* method of the applet performs the following functions: 1) creates a new **CardLayout** object, 2) creates ten new panel object, 3) adds the panels to the **CardLayout**, 4) initializes the ORB, 5) locates a *BookDispenser* object on the server.

When there is a click on the button which is trapped by the **Actionlistener**, an **ActionEvent** is created. The method **actionPerformed** is implemented to process the actions appropriately. Similarly, the **ItemEvent** created by a click on **Choice** or **List** objects are trapped by the **ItemListener**. Correspondingly, the method **itemStateChanged** is implemented to process the actions.

If the user has successfully logon the ITSS, the client will invoke *reserveBookObject* to obtain a reference to a CORBA Book server object. If it succeeds, the control thread will run immediately, with a higher priority than the normal one. It acts as the server for the *BookDispenser* on the client side. It initializes the ORB and the BOA as other servers. It then registers itself to the *BookDispenser*.

The *BookPanels* is the superclass of all the panel classes. It sets the background color, automatically creates the **GridBagLayout** and **GridBagConstraints** and provides a generic *addGBComponent* method that places an AWT component inside a grid.

The client side is only responsible for the display of information to the client and processes all users' actions. All interactions with the database as well as the processing of data are left for the application server to handle so as to maintain a thin client. The client simply invokes the operations defined on the IDL to get the work done by the server.

Finally, the browser calls the *destroy* method when the applet terminates. This method in turn invokes the *releaseBookObject* on the *BookDispenser* server. If the user has been allocated a *Book* object, the resources will then be released to the pool of available objects, which is part of the OLTP etiquette.

6.2.2 System Requirement for client

To start the ITSS client, one needs the followings:

a. Java-enabled Web browser, preferably Netscape Communicator 4.X or Internet Explorer 4.X or above, may be run on either a Windows NT or a Unix Workstation
b. Java plug-in, version 1.1.2 or above, which depends on browser
c. Internet connection for network communication

6.3 Server side of ITSS

Again written in Java, the CORBA server objects, being the heart of the system, provide the middle tier. As shown in Figure 6-1, they interact with both the client applets on the front-end and the Oracle database on the back-end. The interfaces to the server objects are defined in CORBA IDL. The main task of the server objects is to implement the operations in the IDL.

6.3.1 Server Objects Interactions

The two main server objects are the *BookDispenser* and the *Book* object. Both are IDL-defined CORBA objects. The first one acts as application server objects that encapsulate interactions with clients and the second one is the data object that encapsulate the JDBC database.

The *BookDispenser* is a broker of server objects. It prestarts and manages a pool of server object-pairs, which it then allocates to clients on demand. Each server object (*Book*) runs in its own thread and maintains a permanent connection to a JDBC. The *BookDB* is a Java-only object; it is not a remote object. The number of object-pairs can be set at the start of the server.

The *BookServer* provides the main method for the server. It provides the following functions: 1) initializes the ORB, 2) obtains a reference to the BOA, 3) creates a new *BookDispenser* object and passes it the size of the server pool, 4) invokes *obj_is_ready* to register the newly created *BookDispenserImpl* object, 5) invokes the *impl_is_ready* to tell the ORB this server is ready for business.

The *BookDispenserImpl* implements the IDL-defined *BookDispenser* interface. It contains an array of *BookStatus* and *ClientControl*. The class constructor creates a pool of *BookImpl* objects and stores their references in an array of *Bookstatus* objects. Finally, it invokes the *obj_is_ready* to register each newly created object with the ORB. The class implements five IDL-defined methods (operation): *isValid*, *reserveBookObject*, *register*, *notifyOther* and *releaseBookObject*.

The *BookStatus* implements a structure with three fields: the *ref* as the reference to which *Book* object is serving the client, the *userId* to identify the user and the *inUse* to store the status of the *Book* object.

The *ClientContrrol* is to provide a reference to the client control thread for the *BookDispenser* to notify. The *ClientControl* on the client side register itself to tell the *BookDispenser* that it is ready for request.

The *BookImpl* implements the IDL-defined Book interface. The class constructor creates a new *BookDB* object and then connects to it. The object is preconnected to the database. The methods implemented by this class are already shown in chapter 5. Each of these methods serves a client request with a corresponding helper method on the *BookDB* object.

The *BookDB* and the *SelectIDBookDB* are database-encapsulator classes. They handle all the interactions with JDBC. They provide the flexibility for accessing different database servers. In ITSS, however, only one database server is employed.

The concurrency control in ITSS (a multi-users system) is done by employing the 2-phase locking technique provided by the database. In ITSS, it uses the table lock for an insert and the row lock for an update. When a transaction commits or rollbacks, all locks are released. The locks on resources are acquired in a particular order. With this careful design, deadlocks are avoided.

6.3.2 Server Software and Components

Comparatively, the server side is much more complicated than the client. The software includes:

a. Inprise VisiBroker for Java 3.4
b. Apache Web 1.3.3
c. Oracle8 Server, Release 8.0.5.0.0
d. Java Development Kit 1.1.4

The followings have to be done to ensure the application server works properly:

a. the Web server and database server are already up and running
b. load the necessary data to the database
c. starts osagent
d. starts Gatekeeper on the host running the web server
e. starts the object implementation server (*BookServer*)

Further details for setting up the server are written in the user guide.

The VisiBroker ORB Smart Agent (osagent) is an ORB-specific utility program provided by VisiBroker. It provides some ORB-specific functions such as ORB domains, object implementation, fault tolerance and object migration from one host to another. It is necessary to start the osagent before using any VisiBroker functions.

Because of the sandbox security model imposed by Java applet, we need an IIOP proxy server on the Web server host. The IIOP proxy server provides IIOP tunneling over a HTTP connection and routes the messages to and from other hosts on the server side network as requested by the client. Because there is only HTTP communications between the client program and the IIOP proxy, the client applet is able to run within a firewall protected network.

6.4 ITSS Client/Server Scenario

The following scenario shows how the objects of the 3-tier ITSS system interact with each other.

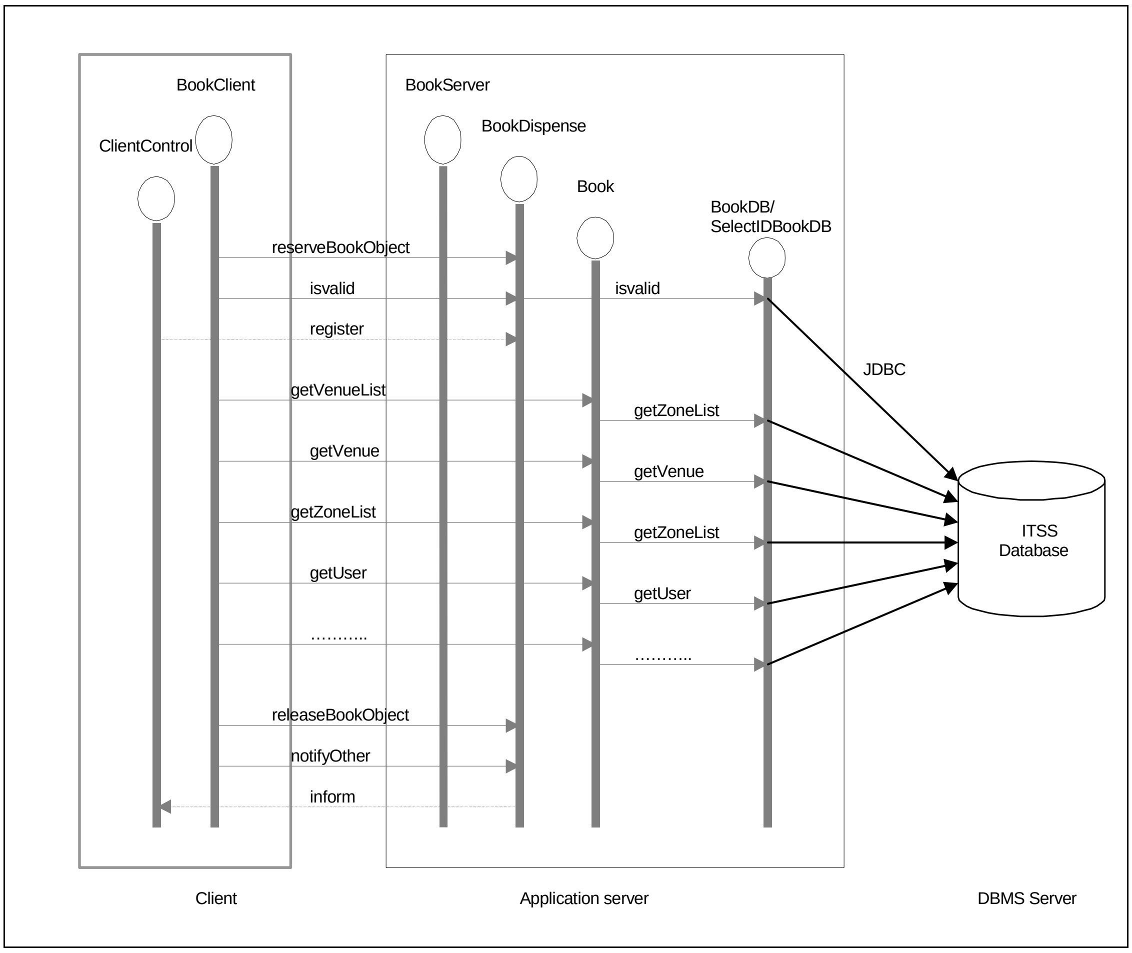


Figure 6-2 The ITSS Client/Server Scenario

1. **The Server creates a dispenser object.** The *BookServer* creates a *BookDispenserImpl* object and passes it the size of the server pool.
2. **The *BookDispenser* prestarts a pool of server objects.** The *BookDispenser* prestarts a pool of server objects and stores their references in an array of *BookStatus* objects. It also invokes *obj_is_ready* to register each newly created *BookImpl* object with the ORB.
3. **The server object creates its JDBC helper.** Each *BookImpl* object creates a *BookImpl* helper. This is the worker object that encapsulates JDBC. This object runs within the same thread as its creator.
4. **The server object connects to the database.** Each *BookImpl* object invokes connect on its *BookImpl* to open a JDBC connection with a ITSS database.
5. **The client connects to the *BookDispenser*.** The *BookDispenser* will validate the user login and check if the user Id has already used.
6. **The client requests a server object.** The client invokes

reserveBookObject on the *BookDispenser* to obtain a server object.

7. **The client creates the *ClientControl* thread.** If the allocation of a server object is success, the client will create and starts the *ClientContorl* thread.
8. **The *ClientControl* thread registers itself to the *BookDispenser*.** The *ClientControl* invoke the *register* on the *BookDispenser* to listen for any notification.
9. **The client invokes the operations provided by the server object.** When the client navigates through the screens, the IDL-defined operations on the server object are invoked.
10. **Notify other client objects.** When a new program/service is successfully added, the client invokes *notifyOther* on the *BookDispenser* to notify other concurrent users.
11. **Inform the client objects.** The *BookDispenser* invokes *inform* on the *ClientControl* to notify the client objects. The *ClientControl* object will then display the notification message on the client applet.
12. **Release the server object**. The client invokes *releaseBookObject* on the *BookDispenser* to return the server object to the pool. The resources will then be available for another client.

7. Testing and Result

7.1 Test Data

The following test data are inserted before the testing procedure. They include the venue information and their corresponding zone information. Beside, twelve user records are created. For those fields which are not significant and too long to be shown are marked with (*).

(i) Table Name: VENUE (Venue Information table)

Field Name	**Record 1**	**Record 2**	**Record 3**	**Record 4**	**Record 5**
VenueRefNo	0001	0002	0003	0004	0005
Open	1200	1200	1200	1200	1200
Close	2330	2300	2300	2300	2300
Venue	Hong Kong Coliseum	Queen Elizabeth Stadium	Ko Shan Theatre – Auditorium	Hong Kong Cultural Centre – Grand Theatre	Hong Kong Cultural Centre - Concert Hall
Location	*	*	*	*	*
Description	*	*	*	*	*
VenueFee	200000	60000	34000	31000	35000
Seat	12600	3600	1200	1800	2100
Zone	**42**	**12**	**4**	**6**	**7**
Zrow	12	12	12	12	12
Zcolunm	25	25	25	25	25
Unit	D	D	D	D	D

Table 7-1 Testing Data for Venue Table

The creation of Zone record is done by a batch Java program. A total of 71 zone records are inserted, equal to the sum of all zone values above (42 + 12 + 4 + 4 + 6 + 7 = 71).

(ii) Table Name: Users (User Information table)

Field Name	Record 1	Record 2	Record 3
UserId	Superusr	00000001	00000002
UserPass	Superusr	97082721	97082722
UserName	Blank	*	*
UserTel	Blank	25102761	25102762
UserAddress	Blank	*	*
UserEMail	Blank	*	*
UserBankAccNo	Blank	*	*
UserBankAccBal	0	20,000.00	20,000.00

Field Name	Record 4	Record 5	Record 6
UserId	00000003	00000004	00000005
UserPass	97082723	97082724	97082725
UserName	*	*	*
UserTel	25102763	25102764	25102765
UserAddress	*	*	*
UserEMail	*	*	*
UserBankAccNo	*	*	*
UserBankAccBal	20,000.00	20,000.00	20,000.00

Field Name	Record 7	Record 8	Record 9
UserId	00000006	00000007	00000008
UserPass	97082726	97082727	97082728
UserName	*	*	*
UserTel	25102765	25102767	25102768
UserAddress	*	*	*
UserEMail	*	*	*
UserBankAccNo	*	*	*
UserBankAccBal	20,000.00	20,000.00	20,000.00

Field Name	Record 10	Record 11	Record 12
UserId	00010001	00010002	00010003
UserPass	97092721	97092722	97092723
UserName	Yiu Wing Ent. Co. Ltd	Rich Elite International Ltd	Hong Kong BasketBall Association
UserTel	25112766	25112767	25112768
UserAddress	*	*	*
UserEMail	*	*	*
UserBankAccNo	*	*	*
UserBankAccBal	2,000,000.00	2,000,000.00	2,000,000.00

Table 7-2 Testing Data for Users Table

7.2 Test Plan

The testing perform on each logical part as well as each functional part. Records are selected, updated, inserted and deleted. To simulate the real situation, multiple users are concurrently invoking the same server to test its accuracy and performance, including how long the response time is and how well it resolves the problem of two parties simultaneously booking the same venue or reserving the same seat.

7.2.1 Test on Functions of each Screen and Exit of the System

a) Opening Screen:
valid user Id and valid user password;
valid user Id and invalid user password;
invalid user Id;
already logon user Id;
password appears in '*'
click on 'Start Over' button;
click on 'Submit' button

b) Main Screen:
Venue List display;
Program/Service List display;

selection of the first item in Venue List at the beginning;
selection of the first item in Program/Service List at the beginning if exists;
click on items of Venue List;
click on items of Program/Service List;
click on 'Book Venue' button;
click on 'Reserve Seat' button;
click on 'List Program' button;
click on 'List Reserve' button;
click on 'Update Info.' Button;
click on 'Change Pass.' Button;

c) Venue Booking Screen:
Heading Description display;
validation of Title;
validation of Start time;
validation of End time;
validation of Description;
validation of Enquiry Telephone ;
selection of Date List;
setting of Fee List;
click on 'Submit' button;
click on 'Cancel' button;

d) Seat Reservation Screen:
Heading Description display;
Program/Service, Zone, Fee and seating plan combination;
click on available seat;
click on reserved seat;
Bank Account display;
adding of the Total Fee;
item added on Seat List;
double click on Seat List;
item removal on Seat List;
disable/enable of seats button;

click on ‘Submit’ button;
click on ‘Cancel’ button

e) Service Transaction Screen:
display of all text fields;
display of date chosen;
display of the Total Fee;
click on ‘Confirm’ button;
click on ‘Reconsider’ button ;
addition of new programs/services and subtraction from user’s Bank Account;
result of transaction displayed in notification message;
for successful transaction, callback received by all other existing users together with the refresh on the Program/Service List

f) Reservation Transaction Screen:
display of seat chosen;
display of Bank Account;
display of the Total Fee;
click on ‘Confirm’ button;
click on ‘Reconsider’ button ;
addition of new reservations and transfer from reservation party’s Bank Account to organizer’s Bank Account
result of transaction displayed in notification message

g) List Service Screen:
Program/Service List display;
enable/disable of ‘Cancel’ button;
click on ‘Return to Selection’ button;
click on items of Program/Service List;
click on ‘Cancel’ button;
cancellation of existing programs/services and payment refund to Venue-booker’s Bank Account;
result displayed in notification message

h) List Reservation Screen:
Reservation List display;
enable/disable of ‘Cancel’ button;
click on ‘Return to Selection’ button;
click on items of Reservation List;
click on ‘Cancel’ button;
cancellation of existing reservation and payment refund from Venue-booker’s Bank Account to reservation party’s Bank Account;
result displayed in notification message

i) Information Update Screen:
display of and edit on User ID;
display of and edit on User Name;
display of and edit on User Telephone;
display of and edit on User Address;
display of and edit on User Email;
display of and edit on Bank Account No.;
display of and edit on Bank Account Balance;
validation of User Name;
validation of User Telephone;
validation of User Address;
click on ‘Ok’ button;
update of user’s information;
click on ‘Cancel’ button

j) Change Password Screen.
click on ‘Ok’ button;
update of user’s password;
combination of correct and incorrect Old Password, New Password and Retype Password;
click on ‘Cancel’ button

k) Others
maximum number of user reached;
release of book object on exit;

first come first serve policy for simultaneous booking/reservation

7.2.2 Test on Performance (Windows NT and Unix Workstation)

Response times for the following items are measured.

1. initial loading of Opening Screen
2. validation of User ID and Password
3. initial loading of Main Screen
4. display of Main Screen (return from other screens)
5. refresh of Program List on Main Screen
6. display of Venue Booking Screen (from Main Screen)
7. display of Venue Booking Screen (from Service Transaction Screen)
8. display of Seat Reservation Screen (from Main Screen)
9. display of Seat Reservation Screen (from Reservation Transaction Screen)
10. display of Service Transaction Screen (ignored as no network involved)
11. service transaction
12. display of Reservation Transaction Screen (ignored as no network involved)
13. reservation transaction
14. display of List Service Screen
15. cancellation of Service
16. display of List Reservation Screen
17. cancellation of Reservation
18. display of Information Update Screen
19. update of users’ information
20. display of Change Password Screen
21. update of users’ password
22. Callback/Notification time

7.3 Test Result

All the functions listed in 7.2.1 are performed precisely for situations of both single user and multiple users (up to ten) concurrently invoke the same server. The result for 7.2.2 is listed on Table 7-3 below. The application server process is run on any Solar/Sparc Workstation while the client works on any other Workstations. With a total of twelve concurrent users, each having a minimized workload on the client computer, the average of ten times for each item is taken.

Item	Response Time	Unix Workstation	Windows NT
1	initial loading of Opening Screen	30	32
2	Validation of User ID and Password	5	< 2
3	initial loading of Main Screen	3 to 4 (*)	2 to 3
4	display of Main Screen (return from other screens)	< 1	< 1
5	refresh of Program List on Main Screen	< 2	< 1
6	display of Venue Booking Screen (from Main Screen)	< 2	< 1
7	display of Venue Booking Screen (from Service Transaction Screen)	< 1	< 1
8	display of Seat Reservation Screen (from Main Screen)	10	3
9	display of Seat Reservation Screen (from Reservation Transaction Screen)	2 to 3	1
10	display of Service Transaction Screen (ignored as no network involved)	N/A	N/A
11	service transaction	2	< 1
12	display of Reservation Transaction Screen (ignored as no network involved)	N/A	N/A
13	reservation transaction	2	< 1
14	display of List Service Screen	< 1	< 1
15	cancellation of Service	< 1	< 1
16	display of List Reservation Screen	< 2	< 1

17	cancellation of Reservation	< 2	< 1
18	display of Information Update Screen	< 1	< 1
19	update of users’ information	< 1	< 1
20	display of Change Password Screen	< 1	< 1
21	update of users’ password	< 1	< 1
22	Callback/Notification time	< 2	< 1

Table 7-3 Testing Result for Response time (measure in seconds)

The result shows that the initial loading of Opening Screen (Item 1) and the display of Seat Reservation Screen from Main Screen (Item 8) take a longer time. The other parts of the system work much better, with an average response time within 1 to 2 seconds. In general, the performance on Windows NT seems better than that of Unix Workstation.

Besides, the performance on Unix Workstation fluctuates in a large range than that of Windows NT. The validation of User ID and Password (Item 2) is the most significant, ranging from 1 second to 17 seconds. It is most likely due to the network traffic. Moreover, there may be some background process being run by other users during the testing.

8. Discussion

The chapter is divided into ten sessions. The accuracy and performance of ITSS will be discussed first, followed by design strategy and enhancement. It then comes to the reusability, scalability, maintainability and interoperability with other Systems. At last, issues on the ease of use, advantages over other alternatives and further enhancement on CORBA are addressed.

8.1 Accuracy and Performance

With a total of six thousand lines of code, the transaction system (ITSS) is still relatively small as compared to a real-life commercial system but it is implemented thoroughly at least. It does precisely perform what has been defined in Chapter 4, the system description. The testing result on 7.2 shows that most of the basic functions for data manipulation, including selection, deletion, insertion and modification of records in a database can be successfully done through the Internet with CORBA as Interface. Furthermore, multiple updates/inserts are skillfully handled in the system and callback service is also introduced.

The performance of the response time in Table 7-3 shows that the result is satisfactory. Most of them are less than 1 second, which is excellent for real-time transaction. However, the starting time (Item 1) is quite long, up to about 30 seconds. This is expected because of the loading of the Java applet to the client web browser. Beside, the structs used by the client defined in IDL are transferred as well.

The display of Seat Reservation Screen from Main Screen (Item 8), is the second bottleneck. It is because of the removal and creation of seat button objects severely lengthens the display time. Again, this is due to the nature of Java applet. Another point to notice is that it takes 10 seconds long for the Unix Workstation to load the screen, while its counter part, the Windows NT, takes only 3 seconds. Both use the netscape web browser. The most likely reason is that the time for creation of peers is longer in the Sun Sparc than in the Windows NT. Another reason may be due to the busy network traffic. In general, the response time on Unix Workstation is longer than the Windows NT.

It is supposed that the transaction time for delete and insert of new service or reservation should be longer than the selection time because they involves more table updates and inserts. However, they are more or less the same. Actually, the most crucial factors are the number of times of transfer through the network and the nature of the protocol. It can be suggested by the initial loading of Main Screen (Item 3), involving two method calls, namely a) the selection of Venue and b) the selection of the service corresponding to the selected venue, which takes nearly twice the times as other selection. The ORBs which the objects communicate with are interconnected through IIOP, a protocol based on TCP/IP, which is very efficient for data transfer through the network.

8.2 Design Strategy

In the design architecture, the *BookDispenser* acts as a Transaction Processing monitor (TP monitor) to maintain a pool of server objects, which are assigned to the client on demand. TP monitor is a program that monitors a transaction as it passes from one stage in a process to another. The TP monitor's purpose is to ensure that the transaction processes complete or, if an error occurs, appropriate action is taken. TP monitors are especially important in three-tier architectures that employ load balancing because a transaction may be forwarded to any of several servers. In fact, many TP monitors handle all the load balancing operations, forwarding transactions to different servers based on their availability. In ITSS, a pool of server objects has been prestarted to save the connection time to the remote database. Resources are reallocated after a client has left. In ITSS, however, the number of concurrent users is limited to 20 for a server. To make it prefect, it should be able to start a new server on another machine and direct the client to connect it. Also the inter-server communication must be handled.

Although the number of concurrent users in this test is only 12, it is sufficiently large to support the real situation. In real case, there may be hundreds of users simultaneously logon the system and invoke the functions on the server. However the system is able to maintain its high performance because each client's invocations can be handled by a different server on different machines for load balancing. In addition, most of times they are browsing the information rather than inserting/updating the same records. The probability of conflict to occur is very small. Besides, as each server only needs to serve one client, no queuing is required. Under the multi-tiered architecture, the thin client – fat server concept can be employed. Clients handle only the user-interface and leave the work of data processing to the application servers, which can be run on different machines to share the workload. Therefore the overall efficiency can be enhanced. Besides, the concept has a major impact on portability as it essentially reduces the client to be cheaper to port to different platforms. Any changes of complicated transaction logic will need only to modify the application server.

All the messages passing are two-way except for the callback service. The callback/notification time (Item 22 on Table 7-3) is taken after the 1st party (who adds a new service) has received the resultant message of its booking. Since this is a Uni-directional communication, the sender need not wait for acknowledgement. The *BookDispenser* receives the message and then sends it to all the concurrent users. Again this is a one-way communication. The performance is nearly the same for hundred of concurrent users. For the case of multiple servers, it has to notify other servers on other machines and this may double the response time.

8.3 Enhancement

Beside the multiple servers discussed above, there are two more areas for improvement, a) use of multiple databases on different servers and b) use of Object-Oriented Database.

For reliability purpose, it should have replicas of one database on a different machine. Simultaneous updates of two databases can be facilitated by the transaction service of CORBA, which implements the two-phase commit protocol. The use of Object-Oriented Database can complement the object-oriented design approach of CORBA so that an object can be saved in as a whole instead of breaking it down. This is especially important for the extent of scale of a system through inheritance. Another advantage is the decrease in locking granularity. Locking on several tables can be replaced by locking on a single object. In ITSS, with traditional relational database, insert or update may involve locking on several tables. The throughput is significantly lowered when the number of concurrent users increased to a thousand.

8.4 Reusability

One of the main aims of this project is to demonstrate the reusability. Although another system based on ITSS has not been implemented, it is foreseeable and can be trivially deduced. The inherence property of interface can extend the system to support various functions. For example, a stock center can be built. The *Book* interface can be extended to *Stock* for buying and selling of stock shares with the override of some operations. For the client side, little modification on objects can be made to accommodate the change such as the functional screens.

8.5 Scalability

As a key characteristic of distributed system, it should be accommodate more users and/or to improve the corresponding responsiveness of the system. In ITSS, with the use of multiple servers as discussed in 8.2, nearly no component has to be changed when the scale of a system increases, such as the number of users. Even if other kinds of databases are used instead of Oracle, only the *BookDB* class, which is responsible to manage transactions, needs modification.

8.6 Maintainability

The ease of maintenance of the system comes from the virtue of CORBA. No or little network maintenance is required. It is handled solely by the ORB. The developers on the client side need not worry about what kind of programming languages or operating system the server run on and where the server locates.

Similarly, the server side developers need not care the similar problems. The effect of change of vendor's product can be minimized. As a result, the possibility of teamwork is increased and the overall administration for large system becomes easier.

8.7 Interoperability with other Systems

Interoperability is defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [1]. For any other systems with CORBA features (CORBA objects), they can communicate with each other through the IDL. As a good example, in ITSS, the *ClientControl* interface in the client side allows the server to callback. In CORBA world, it is possible for an object to interoperate with other objects simply by populating its interface.

8.8 Ease of use

The popularity of a programming language or architecture depends on its ease of use. CORBA is a complex specification, and considerable effort may be required to develop expertise in its use. A number of factors compound the inherent complexity of the CORBA specification. a) While CORBA defines a standard, there is great latitude in many of the implementation details. ORBs developed by different vendors may have significantly different features and capabilities. Thus, users must learn the way by which the vendors implement the specification and their value-added features (which are often necessary to make a CORBA product usable). b) While CORBA makes the development of distributed applications easier than with previous technologies, this ease of use may be deceptive. The difficult issues involved in designing robust distributed systems still remain (e.g., performance prediction and analysis, failure mode analysis, consistency and caching, and security). c) Facility with CORBA may require deep expertise in related technologies, such as distributed systems design, distributed and multi-threaded programming and debugging; inter-networking; object-oriented design, analysis, and programming. In particular, expertise in object-oriented technology may require a substantial change in engineering practice, with all the technology transition issues that imply.

Programming language support. IDL is a "least-common denominator" language. It does not fully exploit the capabilities of programming languages to which it is mapped, especially where the definition of abstract types is concerned.

The implementation code usually written in a low-level language (such as C++ or Java) is another limitation. Currently, no popular visual toolset (such as PowerBuiler or Visual Basic) for CORBA is widely available yet. The complexity increases with the size of system. As a result, it raises the development cost.

Nevertheless, a large and growing number of implementations of CORBA are available in the marketplace, including implementations from most major computer manufacturers and independent software vendors. CORBA ORBs are also being developed by university research and development projects, for example Stanford's Fresco, XeroxPARC's ILU, Cornell's Electra, and others.

8.9 Comparative Advantages over other alternatives

A CORBA object reference is a very powerful unit of distributed service negotiation. It points to an object interface, a set of related methods (attributes can be replaced by a pair of set() and get() operations) that operate on an individual object. In contrast, an RPC only returns a reference to a single function. Furthermore, CORBA interfaces can be aggregated via multiple-inheritance. Also CORBA objects are polymorphic, i.e. the same call behaves differently depending on the object type that receives it. RPC does not support them at all.

CORBA integrates excellent with Java, a purely OO programming language which is powerful for building multi-thread systems. In ITSS, adding a thread running on the client to listen for the callback from server is simple. Besides, CORBA also works well with C++, with which many existing objects built.

CORBA objects are self-describing and introspective. In ITSS, the dynamic invocation is avoided for its performance. In fact, CORBA’s dynamic facilities including Naming Service, Trading, Service, DII, DSI, and Interface repository provide a solid foundation for the dynamic discovery and invocation of services on the intergalactic network so that flexible and agile system can be created. DCOM [2] is the only other alternative to support this property.

The major computing companies including Sun, JavaSoft, IBM, Netscape, Apple, Oracle, Sybase and HP have chosen CORBA IIOP as the common way to connect distributed objects across the Internet and Intranets. By the end of 1998, CORBA became almost as ubiquitous as TCP/IP.

Unlike DCOM, CORBA is not controlled by a single vendor. Consequently, it is always able to obtained CORBA's ORB from more than one vendor. Other than Visibroker, Iona Orbix is also another provider. Besides, CORBA is not platform-specific. It can be run on all major hardware platforms and different operating systems. Its open standard makes it in advantage over DCOM [3].

8.10 Further Enhancement on CORBA

The OMG is still enriching CORBA [4]. CORBA 3, the first major addition to the Common Object request Broker Architecture form OMG since the IIOP protocol added interoperability in 1996, will be released in the coming months. The specification included in the designation CORBA 3 divided nearly into three major categories: a) Java and Internet Integration, b) Quality and Service Control and c) The CORBAcomponent architecture.

For Java and Internet Integration, three specifications enhance CORBA integration with the increasingly popular language and the Internet. First of all, CORBA 3 adds a Java-to-IDL mapping to the traditional IDL-to-Java mapping. This new mapping defines IDL interfaces for a Java objects with two effects: It lets Java programmers use the OMG standard protocol IIOP for their remote invocations and it allows Java servers to be invoked by CORBA clients written in any CORBA-supported programming language. The second one is the firewall specification. The CORBA 3 firewall Specification defines transport-level firewalls, application-level firewalls and a bi-directional GIOP connection useful for callbacks and event notifications. Because standard CORBA connections carry invocations only one-way, a callback typically requires the establishing of a second TCP connection for this traffic heading in the other direction, which is a no-no to virtually every firewall in existence. Under the new specification, an IIOP connection is allowed to carry invocations in the reverse direction under certain restrictive conditions that do not compromise the security at either end of the connection. The third one is the Interoperable Naming Service, which defines one URL-format object reference, *iioploc*, that can be typed into a program to reach defined services at a remote location, including the Naming Service. A second URL format, *iiopname*, actually invokes the remote Naming service using the name that the user appends to the URL, and retrieves the named object. For example, an *iiopLoc* identifier *iioploc://www.omg.org/NameServiece* would resolve to the CORBA Naming Service running on the machine whose IP address corresponded to the domain name www.omg.org.

Though they have not yet been voted by the time of writing this document, these specifications taken together add a new dimension of capability and ease-of-use to CORBA, which will ensure that CORBA continues to play an ever-increasing role in computing world of future.

References:

[1] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

[2] DCOM (Distributed Components Object Model), a product of Microsoft

[3] Client/Server Programming With Java and CORBA by Robert Orfali and Dan Harkey P.328-329

[4] OMG in Motion – June '99 edition (http://www.omg.org/)

9. Conclusion

Through the work of our project, we proved that CORBA is undoubtedly the best architecture to provide the bridge for connection between the client side on the Web and the application server on host. Its formidable power comes from its independence of hardware platform, free choice of operating system, neutrality in language implementation and open standard. A sample application, ITSS, is implemented to successfully demonstrate distributed transaction on CORBA. Moreover, we make some suggestions for improvement of system as well as the extension of the system to accommodate all kinds of transaction.

In fact, CORBA can result in distributed systems that can be rapidly developed and can reap the benefits that result form using high-level building blocks provided by CORBA, such as maintainability and adaptability. As an industry standard, it also has the advantage of flexibility in response to changes in market conditions and technology advances.

On the contrary, the complexity of CORBA has some drawbacks. Suggested by research and development team leaders, training is essential even for the already experienced programmers [1]. Besides, they must have distributed computing concepts and have to confront with the lack of visual tools for development.

However, the benefit from CORBA is still over its disadvantages by a great deal. In the near future, it is expected to be the leading standard architecture for distributed system development.

Reference:

[1] Mowbray, T.J. & Brando, T. "Interoperability and CORBA-Based Open Systems." Object Magazine 3, 3 (September/October 1993): 50-4.

I. Reference

Though not quoted in previous chapters, the following list of papers, books and web sites have given valuable ideas and inspiration for the author to accomplish this project.

References :

[1] Baker, S. "CORBA Implementation Issues." IEEE Colloquium on Distributed Object Management Digest 1994 7 (January 1994): 24-25.

[2] Brando, T. "Comparing CORBA & DCE." Object Magazine 6, 1 (March 1996):
52-7.

[3] Pineapplesoft Link, January 1998
http://www.pineapplesoft.com/newsletter/archive/19980101_3tier.html

[4] OMG archive for list: experts (by date)
http://www.omg.org/archives/experts/

[5] CORBA http://mordor.cs.hut.fi/~cls/corba_basics/
Casper Lassenius HUT

[6] Software Technology Review
http://www.sei.cmu.edu/str/descriptions/corba_body.html

[7] Deng, R.H., et al. "Integrating Security in CORBA-Based Object Architectures,"
50-61. Proceedings of the 1995 IEEE Symposium on Security and Privacy. Oakland, CA, May 8-10, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.

[8] Foody, M.A. "OLE and COM vs. CORBA." UNIX Review 14, 4. (April 1996): 43-45.

[9] Jell, T. & Stal, M. "Comparing, Contrasting, and Interweaving CORBA and OLE," 140-144. Object Expo Europe 1995. London, UK, September 25-29, 1995. Newdigate, UK: SIGS Conferences, 1995.

[10] Kain, J.B. "An Overview of OMG's CORBA," 131-134. Proceedings of OBJECT EXPO `94. New York, NY, June 6-10, 1994. New York, NY: SIGS Publications, 1994.

[11] Mowbray, T.J. & Brando, T. "Interoperability and CORBA-Based Open Systems." Object Magazine 3, 3 (September/October 1993): 50-4.

[12] Roy, Mark & Ewald, Alan. "Distributed Object Interoperability." Object Magazine 5, 1 (March/April 1995): 18.

[13] Steinke, Steve. "Middleware Meets the Network." LAN: The Network Solutions
Magazine 10, 13 (December 1995): 56.

[14] Tibbets, Fred. "CORBA: A Common Touch for Distributed Applications." Data Comm Magazine 24, 7 (May 1995): 71-75.

[15] Wallnau, Kurt & Wallace, Evan. "A Situated Evaluation of the Object Management Group's (OMG) Object Management Architecture (OMA)," 168-178. Proceedings of the OOPSLA'96. San Jose, CA, October 6-10, 1996. New York, NY: ACM, 1996. Presentation available [online] FTP. <URL: ftp://ftp.sei.cmu.edu/pub/corba/OOPSLA/present> (1996).

[16] Watson, A. "The OMG After CORBA 2." Object Magazine 6, 1 (March 1996): 58-60.

[17] Mowbray, T.J. & Brando, T. "Interoperability and CORBA-Based Open Systems." Object Magazine 3, 3 (September/October 1993): 50-4.

[18] INPRISE VisiBroker Product Documentation
http://www.inprise.com/techpubs/books/vbj/vbj33/pdf_index.html

[19] TP Lite vs. TP Heavy
http://www.byte.com/art/9504/sec11/art4.htm

[20] Dickman, A. "Two-Tier Versus Three-Tier Apps." Informationweek 553 (November 13, 1995): 74-80.

[21] Bernstein, Philip A. "Middleware: A Model for Distributed Services." Communications of the ACM 39, 2 (February 1996): 86-97.

[22] "Middleware Can Mask the Complexity of your Distributed Environment." Client/Server Economics Letter 2, 6 (June 1995): 1-5.

[23] Brando, T. "Comparing CORBA & DCE." Object Magazine 6, 1 (March 1996):52-7.

II. User Guide

The purpose of this user guide is for those who would use the booking service of the *Integrated Transaction Service System* (ITSS), both venue booking and seat reservation. The system must be set up according to the readme.txt before use.

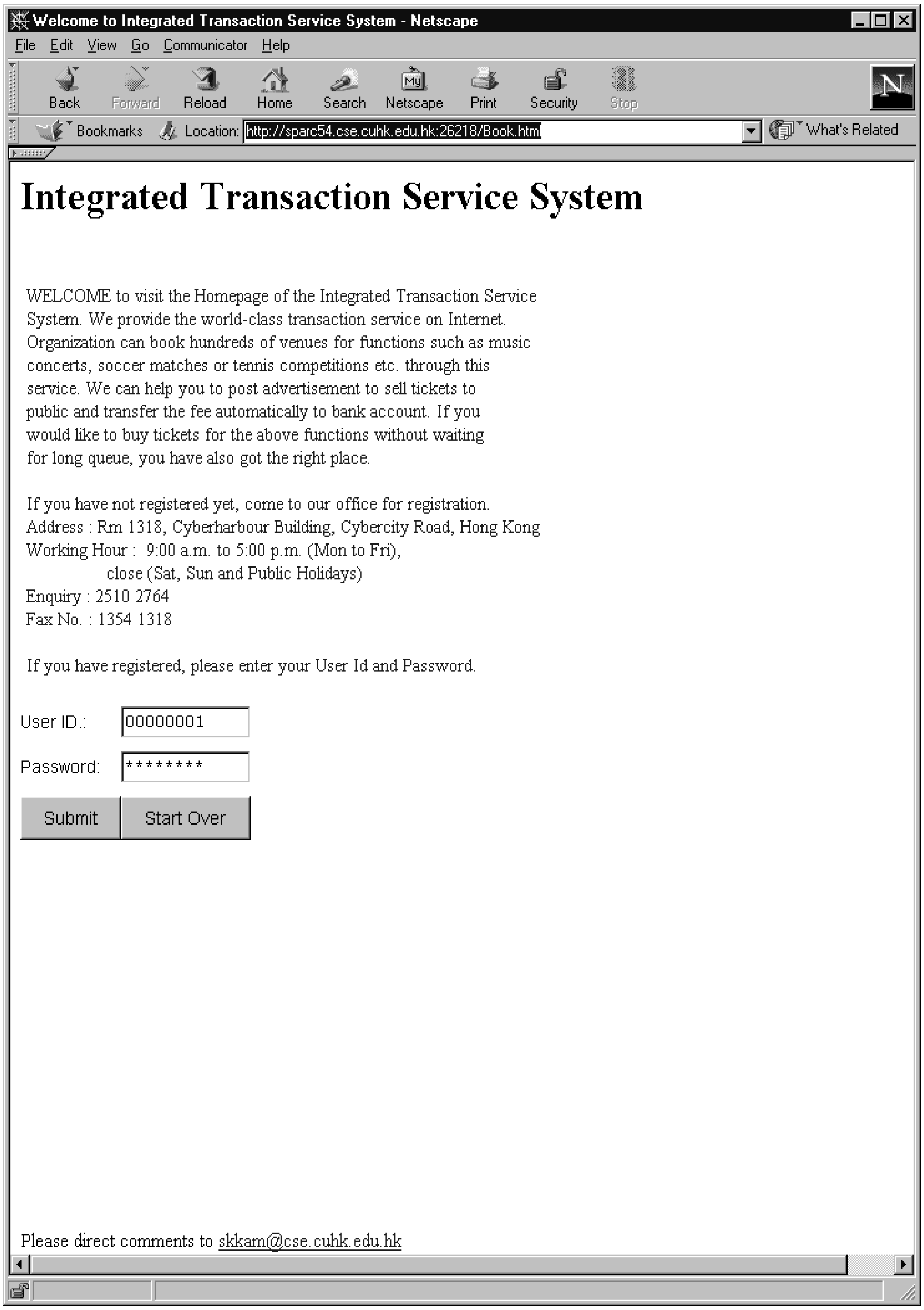
Operation Details

All the users are eligible to book venue, reserve seat and browse the venues they book and seats they reserve. Only the super user has the right to browse the all the bookings and the reservations. Moreover, he has the right to cancel bookings and reservations. There is a limit on maximum of number of concurrent users.

Error/Notification message dialog boxes will be displayed to notify the user. Press ‘Ok’ to close the message boxes.

A. Opening Screen

When you access the site, an opening screen is shown as in Figure 1. Read the fancy description about the system before you enjoy using the system. Type in the User ID and the Password and then press ‘Submit’ to logon the screen. You can press ‘Start Over’ to clear the User ID and the Password. A User ID can be used to logon once a time. There is a limit on maximum number of users. If it exceeds, you have to wait until another user logout. Valid User ID and Password will lead you to the Main Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

WELCOME to visit the Homepage of the Integrated Transaction Service System. We provide the world-class transaction service on Internet. Organization can book hundreds of venues for functions such as music concerts, soccer matches or tennis competitions etc. through this service. We can help you to post advertisement to sell tickets to public and transfer the fee automatically to bank account. If you would like to buy tickets for the above functions without waiting for long queue, you have also got the right place.

If you have not registered yet, come to our office for registration.
Address : Rm 1318, Cyberharbour Building, Cybercity Road, Hong Kong
Working Hour : 9:00 a.m. to 5:00 p.m. (Mon to Fri),
close (Sat, Sun and Public Holidays)
Enquiry : 2510 2764
Fax No. : 1354 1318

If you have registered, please enter your User Id and Password.

User ID.: 00000001
Password: ********

Please direct comments to skkam@cse.cuhk.edu.hk

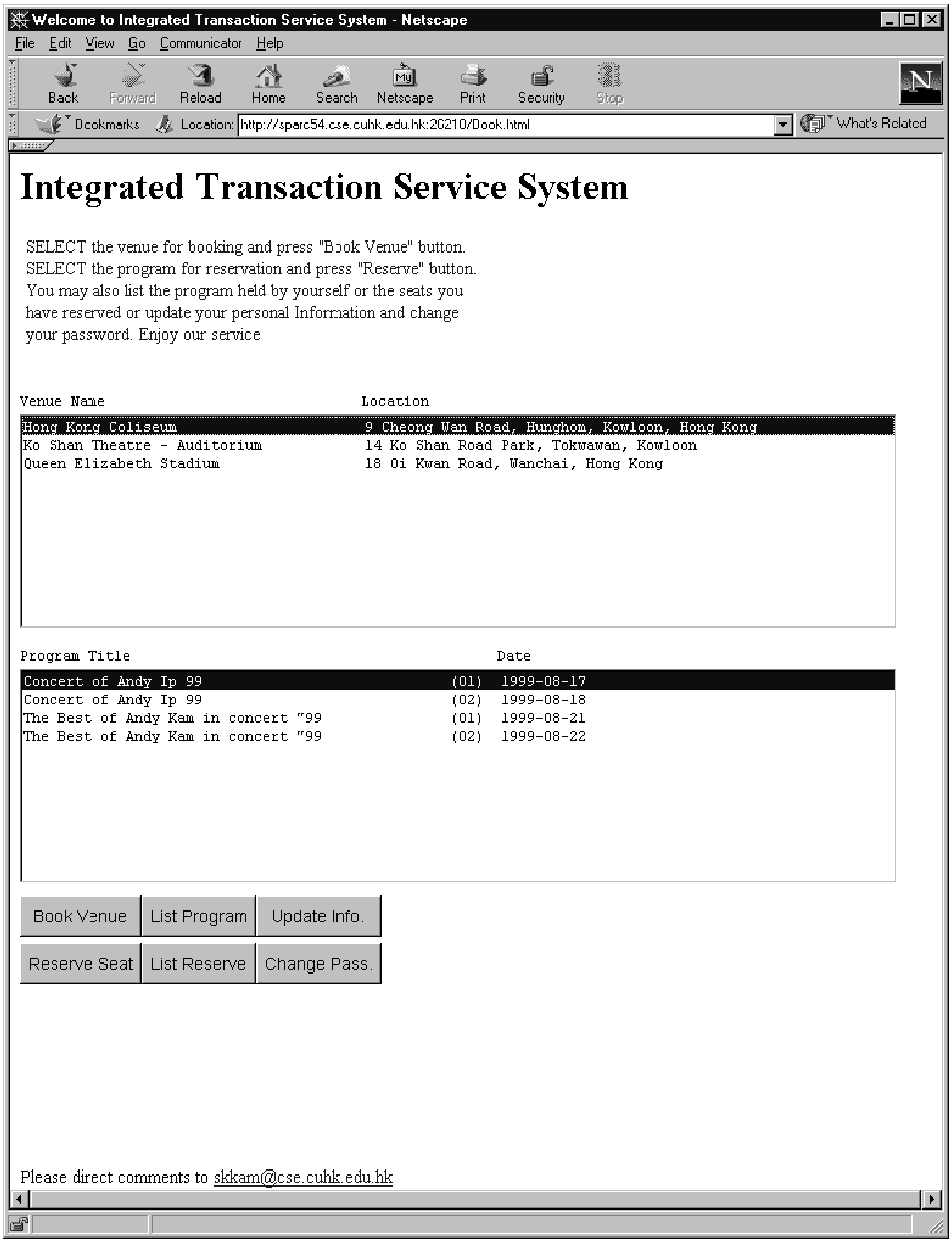
Figure 1 Opening Screen

B. Main Screen

The screen shows the Venue List and Program List [Figure 2]. The Venue List shows all the venues for booking while the Program List shows all the corresponding programs hold on that venue. In the Venue List, the venues are displayed in alphabetical order of the venue name. In the Program List, you can see the program title and the date when it holds. The number in parentheses is the session number. The programs are displayed in alphabetical order of the program title and the date. Click on a venue item will show the all the corresponding programs in the Program List. Any newly added programs by other user will be shown immediately.

Press ‘Book Venue’ to book the venue selected or press ‘Reserve Seat’ to reserve the seat for the program selected. It has no effect if there is no program in the list. Press ‘List Program’ or ‘List Reservation’ to the browse the program you hold or the seat you reserve.

To modify your personal information, you can press ‘Update Info.’ or if you want to get a new password, press ‘Change Pass.’.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html What's Related

Integrated Transaction Service System

SELECT the venue for booking and press "Book Venue" button.
SELECT the program for reservation and press "Reserve" button.
You may also list the program held by yourself or the seats you have reserved or update your personal Information and change your password. Enjoy our service

```
Venue Name                             Location
Hong Kong Coliseum                     9 Cheong Wan Road, Hunghom, Kowloon, Hong Kong
Ko Shan Theatre - Auditorium           14 Ko Shan Road Park, Tokwawan, Kowloon
Queen Elizabeth Stadium                18 Oi Kwan Road, Wanchai, Hong Kong
```

```
Program Title                                          Date
Concert of Andy Ip 99                         (01)  1999-08-17
Concert of Andy Ip 99                         (02)  1999-08-18
The Best of Andy Kam in concert "99           (01)  1999-08-21
The Best of Andy Kam in concert "99           (02)  1999-08-22
```

Please direct comments to skkam@cse.cuhk.edu.hk

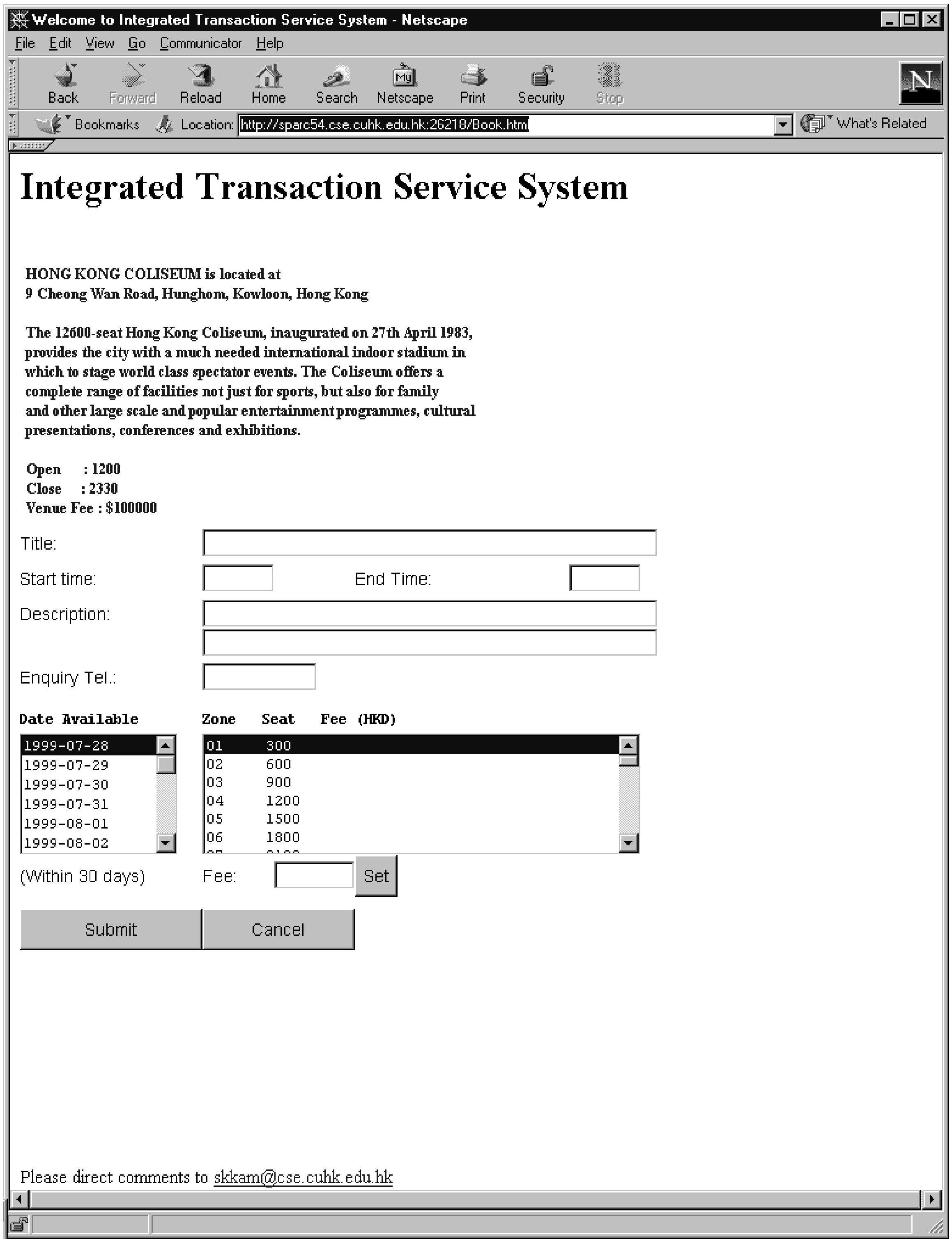
Figure 2 Main Screen

C. Venue Booking Screen

You will see the screen as shown in Figure 3. The name, location and other details of the venue are displayed. You are invited to enter the program title, the start time and end time. The time format must be in HHMM, where HH repersnets the hour and MM is the minute. The start time must not be earlier than the open hour of the venue and the end time must not be later than the close hour. The end time must be later than the start time. Also, they must be in correct hour/minute format. Add the descriptions for your program. The maximum length is 200. If exceeds, it will be truncated. The Enquiry Telephone is compulsory. It must consist only digits. The above details will be posted in the advertisement.

The Date List shows all the available dates within 30 days. By default the first date is selected. You can select or deselect the date by a click on the item. At least one date must be chosen and at most ten dates is allowed at a time. The Fee list on the right allows you to set the fee for the zones. Enter the fee and then press ‘Set’ to set the fee for that zone selected. You must at least set the fee for the first zone. Those fees that are not set will follow the one above.

Finally, press ‘Submit’ to confirm. It will lead you to the service transaction screen. You can press ‘Cancel’ to go back to the Main Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html What's Related

Integrated Transaction Service System

HONG KONG COLISEUM is located at
9 Cheong Wan Road, Hunghom, Kowloon, Hong Kong

The 12600-seat Hong Kong Coliseum, inaugurated on 27th April 1983, provides the city with a much needed international indoor stadium in which to stage world class spectator events. The Coliseum offers a complete range of facilities not just for sports, but also for family and other large scale and popular entertainment programmes, cultural presentations, conferences and exhibitions.

Open : 1200
Close : 2330
Venue Fee : $100000

Title:
Start time: End Time:
Description:
Enquiry Tel.:

```
Date Available      Zone   Seat   Fee (HKD)
1999-07-28          01     300
1999-07-29          02     600
1999-07-30          03     900
1999-07-31          04     1200
1999-08-01          05     1500
1999-08-02          06     1800
```

(Within 30 days) Fee: Set

Submit Cancel

Please direct comments to skkam@cse.cuhk.edu.hk

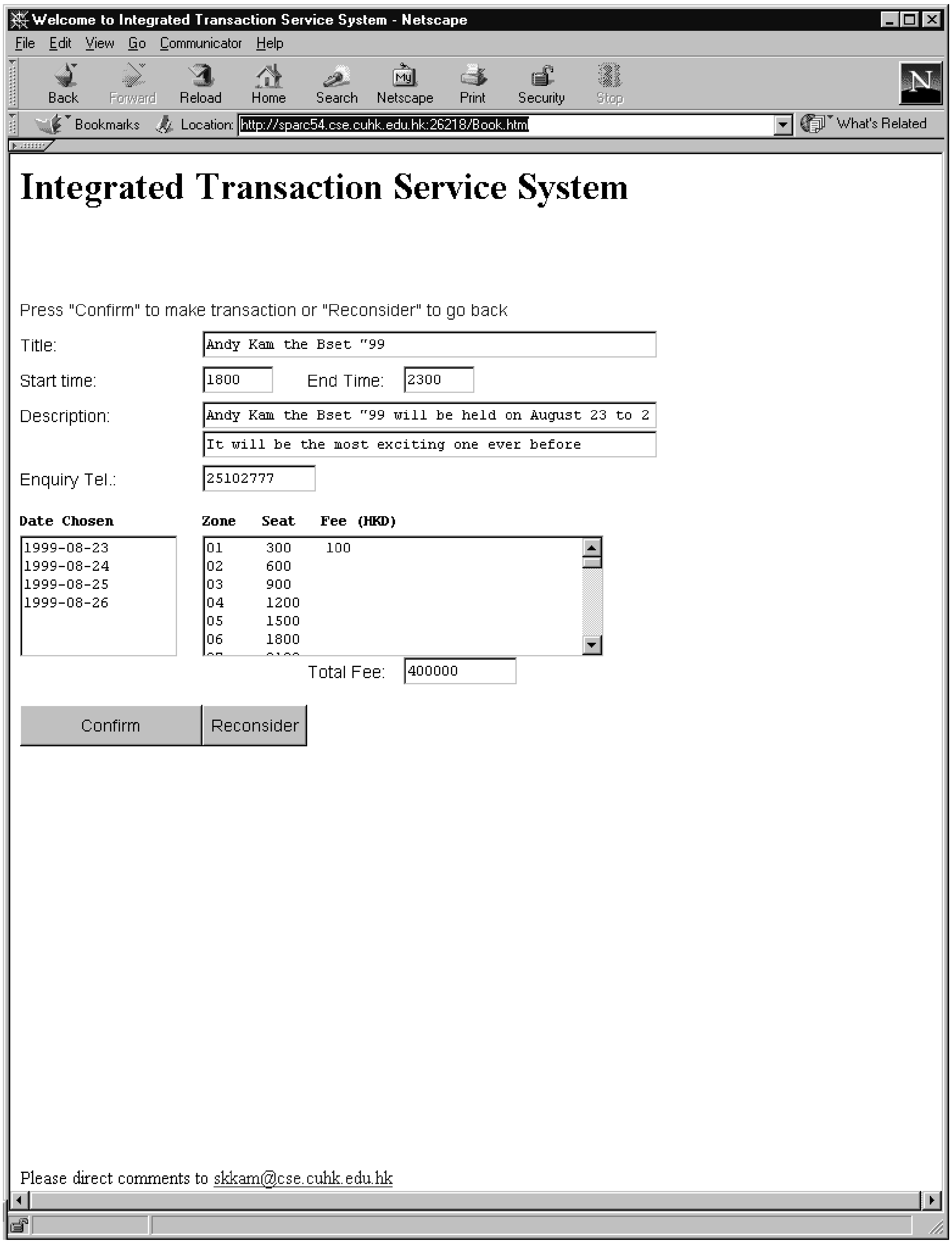
Figure 3 Venue Booking Screen

D. Service Transaction Screen

The screen [Figure 4] shows the details you have previously entered in the Venue Booking Screen. User can double-check for the correctness before confirmation. Note that the Date List shows only those chosen.

If you think it is ok, you can press ‘Confirm’ to make transaction. It is your responsibility to ensure that you have money for the transaction. If the you do not has enough money in your bank account, it will be rejected. If some other user has booked the venue of same dates at almost the same time, it will again be rejected.

It will return to the Main Screen. If the transaction is success, the users’ bank account balance will be shown. All other concurrent users will be notified. If it fails, the corresponding reason will be shown. Press ‘Reconsider’ to go back to the Venue Booking Screen for reconsideration.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

Press "Confirm" to make transaction or "Reconsider" to go back

Title:	Andy Kam the Bset "99		
Start time:	1800	End Time:	2300
Description:	Andy Kam the Bset "99 will be held on August 23 to 2		
	It will be the most exciting one ever before		
Enquiry Tel.:	25102777		

Date Chosen
1999-08-23
1999-08-24
1999-08-25
1999-08-26

Zone	Seat	Fee (HKD)
01	300	100
02	600	
03	900	
04	1200	
05	1500	
06	1800	

Total Fee: 400000

Please direct comments to skkam@cse.cuhk.edu.hk

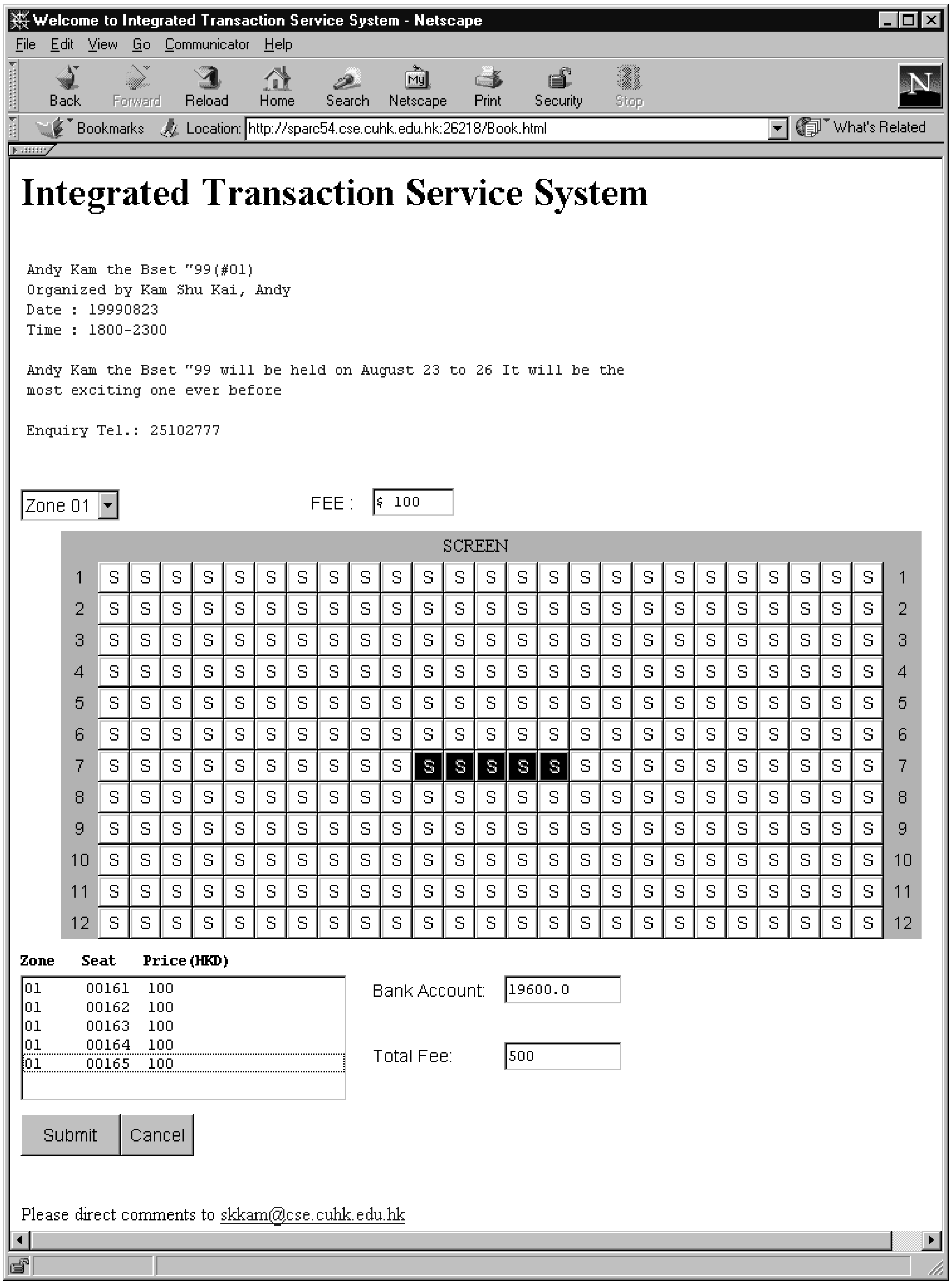
Figure 4 Service Transaction Screen

E. Seat Reservation Screen

The fancy screen is shown as in Figure 5. The program title, the organizer and the other program detail are displayed. By default, the first zone is chosen. The corresponding fee and the seating plan are displayed.

You can click on the Zone Choice to select seats from other zones. The seats which have been reserved are in reserved color. To select a seat, place the mouse cursor on the seat followed by a click. The selected seat together with its zone number, seat number and its price will be shown on the Seat List. Also the fee will be added up to the total fee. Double click the item in the Seat List to deselect a seat.

Only 20 seats can be reserved at a time. If the total fee exceeds the balance in Bank Account, no more seats can be chosen. Finally, press 'Submit' to go to the Reservation Transaction Screen or press 'Cancel' to go back to the Main Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

```
Andy Kam the Bset "99(#01)
Organized by Kam Shu Kai, Andy
Date : 19990823
Time : 1800-2300

Andy Kam the Bset "99 will be held on August 23 to 26 It will be the
most exciting one ever before

Enquiry Tel.: 25102777
```

Zone 01 FEE : $ 100

SCREEN

1	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	1
2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	2
3	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	3
4	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	4
5	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	5
6	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	6
7	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	7
8	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	8
9	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	9
10	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	10
11	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	11
12	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	12

```
Zone   Seat   Price(HKD)
01     00161  100
01     00162  100
01     00163  100
01     00164  100
01     00165  100
```

Bank Account: 19600.0

Total Fee: 500

Submit Cancel

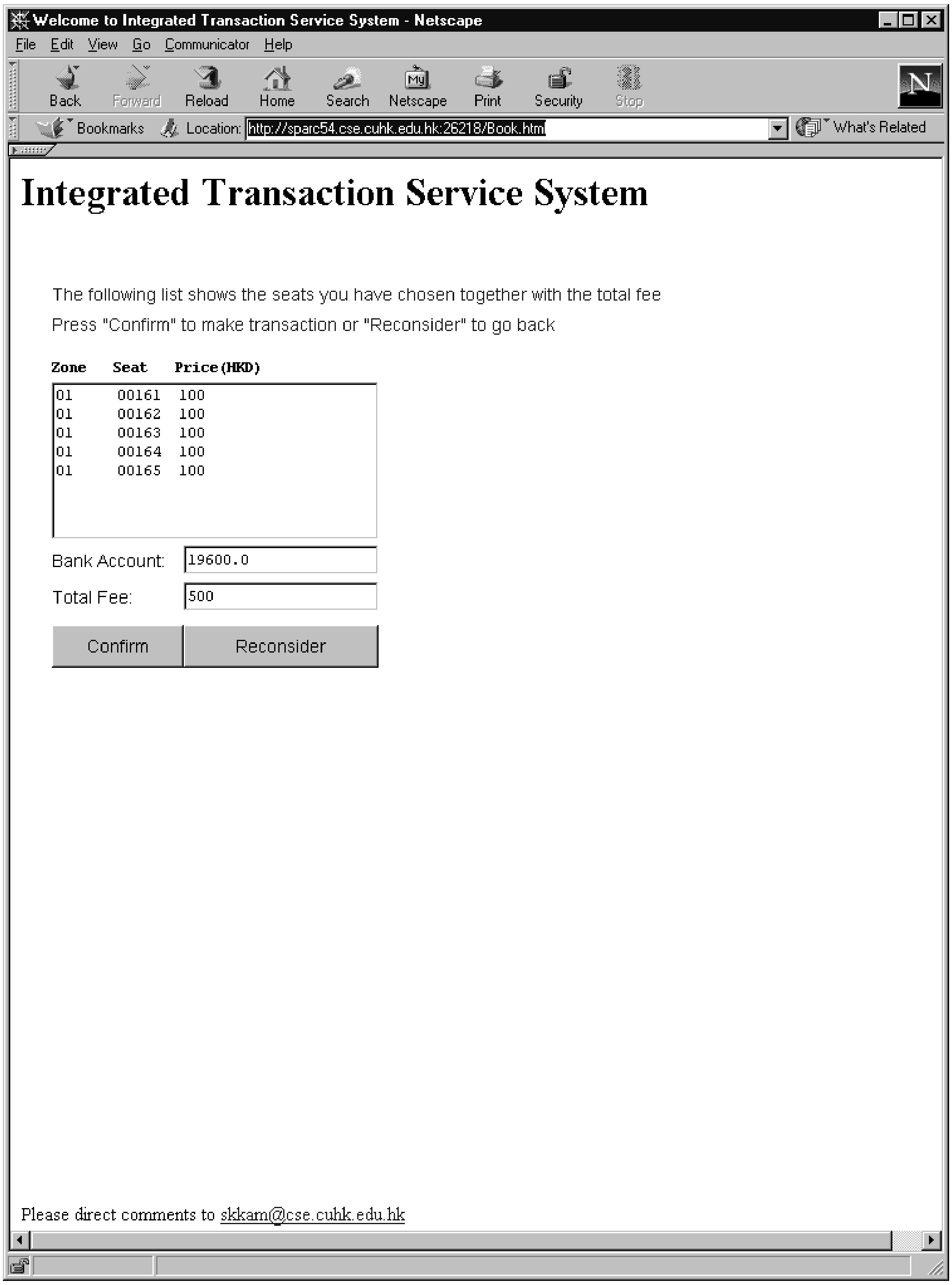
Please direct comments to skkam@cse.cuhk.edu.hk

Figure 5 Seat Reservation Screen

F. Reservation Transaction Screen

The screen shows the seat list which contain all the seats you have chosen. [Figure 6]. User can double-check before confirmation. Below the list shows your bank account and total fee.

Press ‘Confirm’ to make transaction. The transaction will be success if the no other user has reserved the seats just before. It will return to the Main Screen. If the transaction is success, your bank account balance will be shown. If it fails, the corresponding reason will be shown. Press ‘Reconsider’ to go back to the Seat Reservation Screen for reconsideration.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

The following list shows the seats you have chosen together with the total fee

Press "Confirm" to make transaction or "Reconsider" to go back

Zone	Seat	Price(HKD)
01	00161	100
01	00162	100
01	00163	100
01	00164	100
01	00165	100

Bank Account:	19600.0
Total Fee:	500

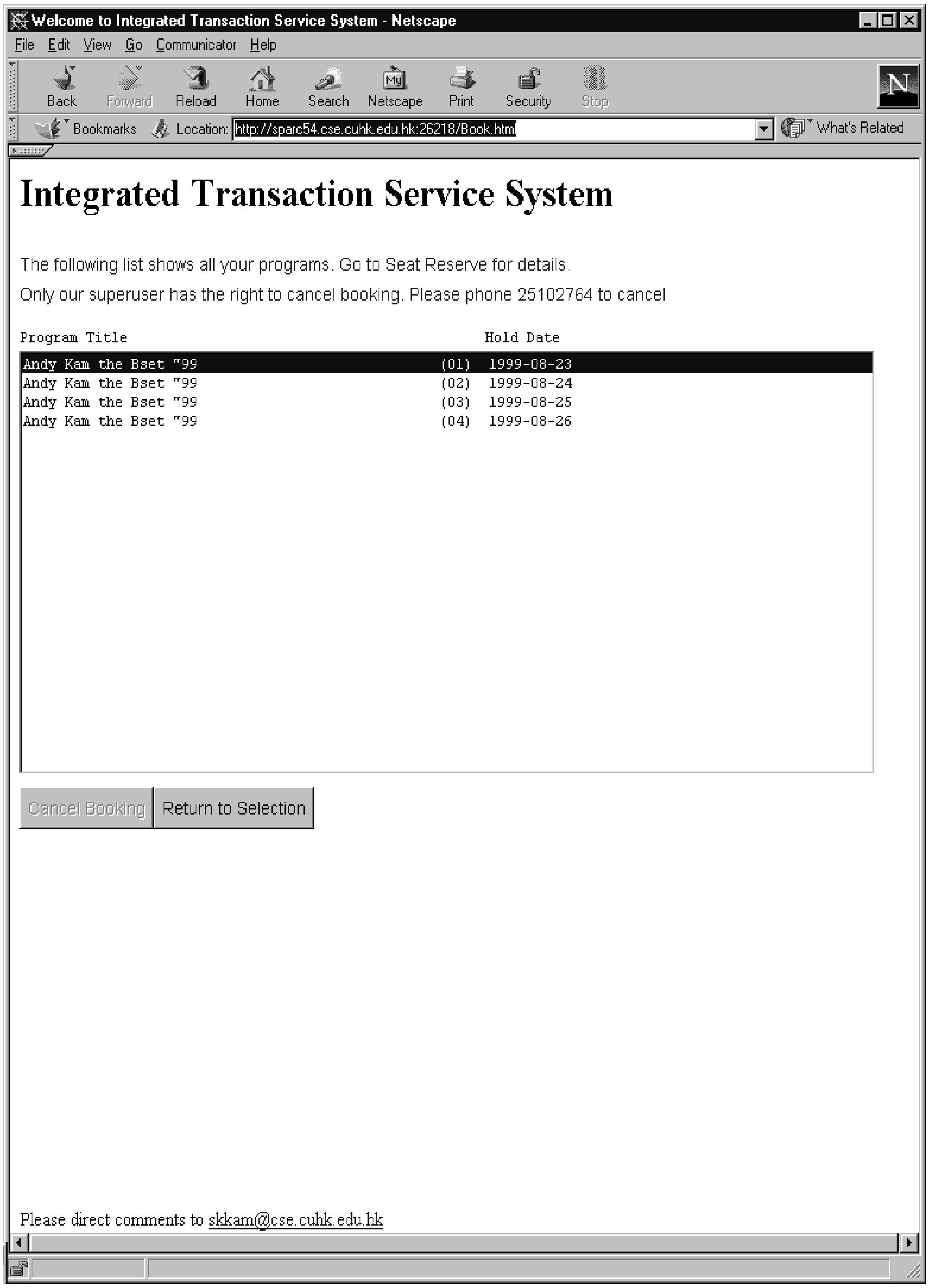
Please direct comments to skkam@cse.cuhk.edu.hk

Figure 6 Reservation Transaction Screen

G. List Service Screen

The screen [Figure 7] shows all the programs you have booked. For the super user, all the programs will be shown. The programs are listed in alphabetical order of the program title and the Hold Date. Press ‘Return to Selection’ to go back to the Main Screen.

The ‘Cancel’ button is enabled only for the super user. Select the program title and press ‘Cancel’ to cancel the booking. It will return to the Main Screen. If it is success, the User ID with its balance will be shown in a notification message. If it fails, a warning message will tell the reason.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

The following list shows all your programs. Go to Seat Reserve for details.

Only our superuser has the right to cancel booking. Please phone 25102764 to cancel

```
Program Title                                       Hold Date
Andy Kam the Bset "99                          (01)  1999-08-23
Andy Kam the Bset "99                          (02)  1999-08-24
Andy Kam the Bset "99                          (03)  1999-08-25
Andy Kam the Bset "99                          (04)  1999-08-26
```

Cancel Booking
Return to Selection

Please direct comments to skkam@cse.cuhk.edu.hk

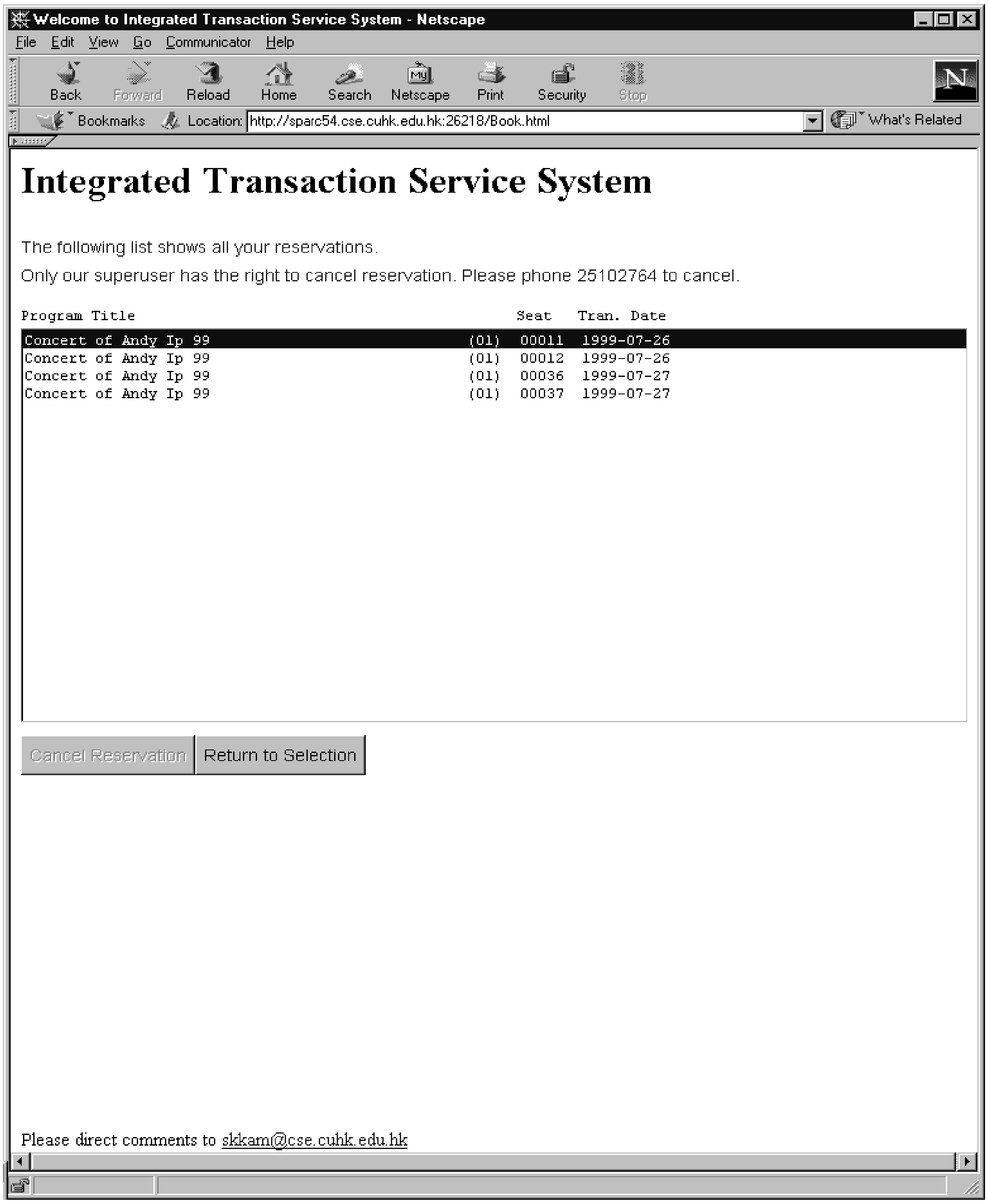
Figure 7 List Service Screen

H. List Reservation Screen

The screen [Figure 8] shows all the reservations you have reserved. For the super user, all the reservations will be shown. The reservations are listed in alphabetical order of the program title, seat number and transaction date. Press ‘Return to Selection’ to go back to the Main Screen.

The ‘Cancel’ button is enabled only for the super user. Select the reservation and press ‘Cancel’ to cancel the reservation. It will return to the Main Screen. If it is success, the organizer’s balances and the ticket-buyer’s balance will be shown in a notification message. If it fails, a warning message will tell the reason.

.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

The following list shows all your reservations.

Only our superuser has the right to cancel reservation. Please phone 25102764 to cancel.

```
Program Title                                         Seat   Tran. Date
Concert of Andy Ip 99                           (01)  00011  1999-07-26
Concert of Andy Ip 99                           (01)  00012  1999-07-26
Concert of Andy Ip 99                           (01)  00036  1999-07-27
Concert of Andy Ip 99                           (01)  00037  1999-07-27
```

Cancel Reservation
Return to Selection

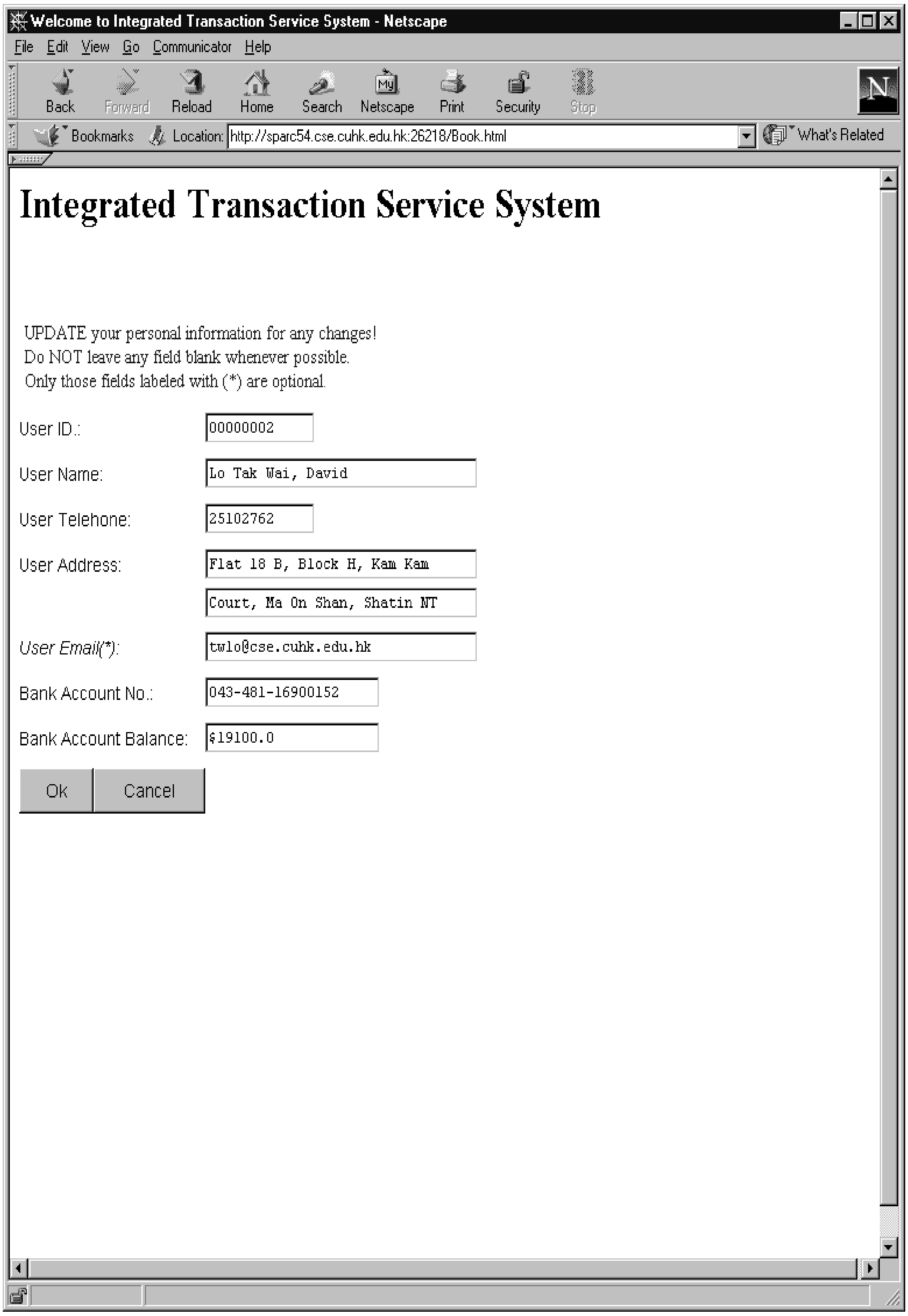
Please direct comments to skkam@cse.cuhk.edu.hk

Figure 8 List reservation Screen

I. Information Update Screen

The screen [Figure 9] shows all the users' personal details (except the password. You can modify only the User Name, user Telephone, User Address and User Email. They are compulsory except the User Email.

Press 'Ok' to make modification. It will return to the Main Screen or press 'Cancel' to ignore modification. It will also return to the Main Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

UPDATE your personal information for any changes!
Do NOT leave any field blank whenever possible.
Only those fields labeled with (*) are optional.

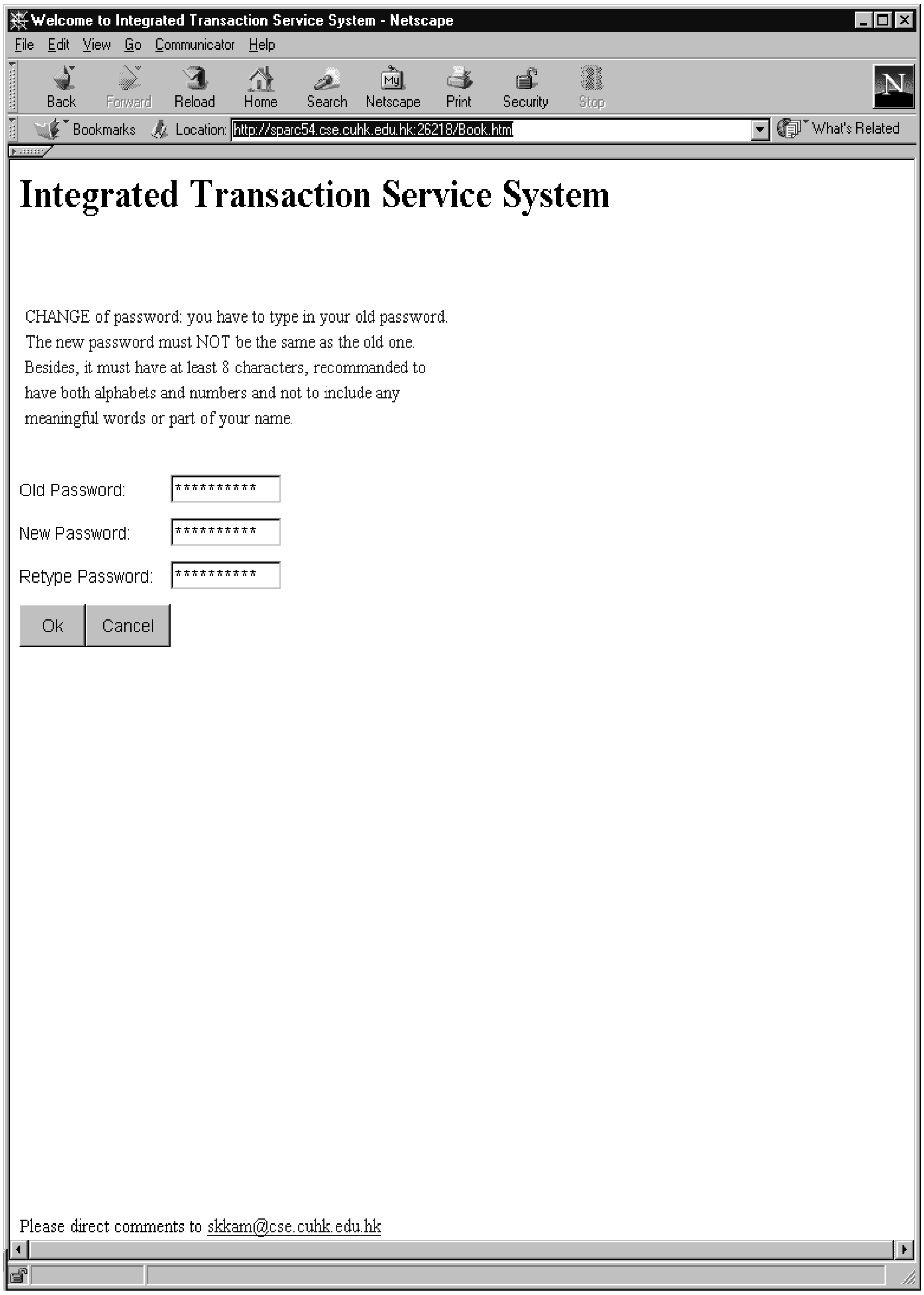
User ID.:	00000002
User Name:	Lo Tak Wai, David
User Telehone:	25102762
User Address:	Flat 18 B, Block H, Kam Kam
	Court, Ma On Shan, Shatin NT
User Email(*):	twlo@cse.cuhk.edu.hk
Bank Account No.:	043-481-16900152
Bank Account Balance:	$19100.0

Figure 9 Information Update Screen

J. Password Change Screen

The screen shows as in Figure 10. To change the password, you have to type in the old password, the new password and retype the new password for confirmation.

The old password must be one you logon the system. The new password must not be the same as the old one and the retyped one must match with the new password. Press ‘Ok’ to make change. It will return to the Main Screen or press ‘Cancel’ to ignore change. It will also return to the Main Screen.


Welcome to Integrated Transaction Service System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://sparc54.cse.cuhk.edu.hk:26218/Book.html
What's Related

Integrated Transaction Service System

CHANGE of password: you have to type in your old password. The new password must NOT be the same as the old one. Besides, it must have at least 8 characters, recommanded to have both alphabets and numbers and not to include any meaningful words or part of your name.

Old Password:	***********
New Password:	***********
Retype Password:	***********

Please direct comments to skkam@cse.cuhk.edu.hk

Figure 10 Change Password Screen