Department of Computer Science and Engineering

The Chinese University of Hong Kong

LYU 9901

TravelNet

Final Year Project Individual Report 1999-2000

Supervisor
Professor Michael R. Lyu

Marker
Professor M.C. Lee

Group Member
Lau Chi Ho Arthur
Ho Chi Ho Malcolm

Prepared by
Lau Chi Ho Arthur (Student ID: 97590853)

Date of Submission: 18th April, 2000

Table of Content

Abstract

No one can deny the rapid development of Internet. It is a trend that many kinds of business are now taking the form of operation from traditional mode to the new e-commerce model. A great success has been seen on different fields of business by adapting to the Internet world. In this project, we will investigate the requirement of building an online e-commerce application – TravelNet, which is a typical e-commerce application for travel agency. In this report, we will first provide an overview of the project and a brief discussion on nowadays e-commerce applications. Then, we will briefly describe the facilities and functions provided by TravelNet, followed by a chapter, which discusses the basic system design and implementation details as a reference of work done in the previous semester. Next, we will briefly address the enhancement of the system over the original one. The enhancement details, including the new payment methods supported, the cooperation of CORBA and TravelNet for distributed application development and simplification of system modules using JSP, are explained in the following three chapters respectively. Finally, we will present a conclusion on this report and our project at the end.

1. Introduction

1.1 Project Objectives

In this project we focus on application -level programming to deve lop a database transaction service: TravelNet. TravelNet allows users to reserve flights over the Internet. It also allows users to shop and purchase travel accessories online be means of a web browser.

We use Java, particularly by the Servlet technolog y, as the main programming language to develop the project. All necessary information is stored in different databases, which consist of both local and remote. The programs will try to collect information among all the databases, then search for the best item that meet clients' needs.

The project will include the integration of payment system, as it is an unavoidable part of an e -commerce application. Payment system in research project and real life may be integrated in the system built.

On the large co llection of components (databases, payment system), it is effectively useful for such components to be distributed. Another objective of this project is to develop this application in a distributed manner. CORBA technology will be used for developing the distributed components.

1.2 Online Travel Agency : TravelNet

Internet is growing every day. Different companies have already started their e-business in the net as attracted by the potential great sales and profit in this fast growing environment.

Online Travel Agencies are perhaps one of the most popular e-commerce applications over the Internet. Large amount of such agencies, like Expedia, Travelocity, LowAirFar.com and Preview Travel are readily available around the Internet. This type of service provides a great convenience for individual or family to buy ticket online for their vacation. It's not convenient for the travellers to check the price by consulting the airline companies and real life travel agency. Online Travel agency can help them to collect and compare price instantly in order to give them a comfortable trip.

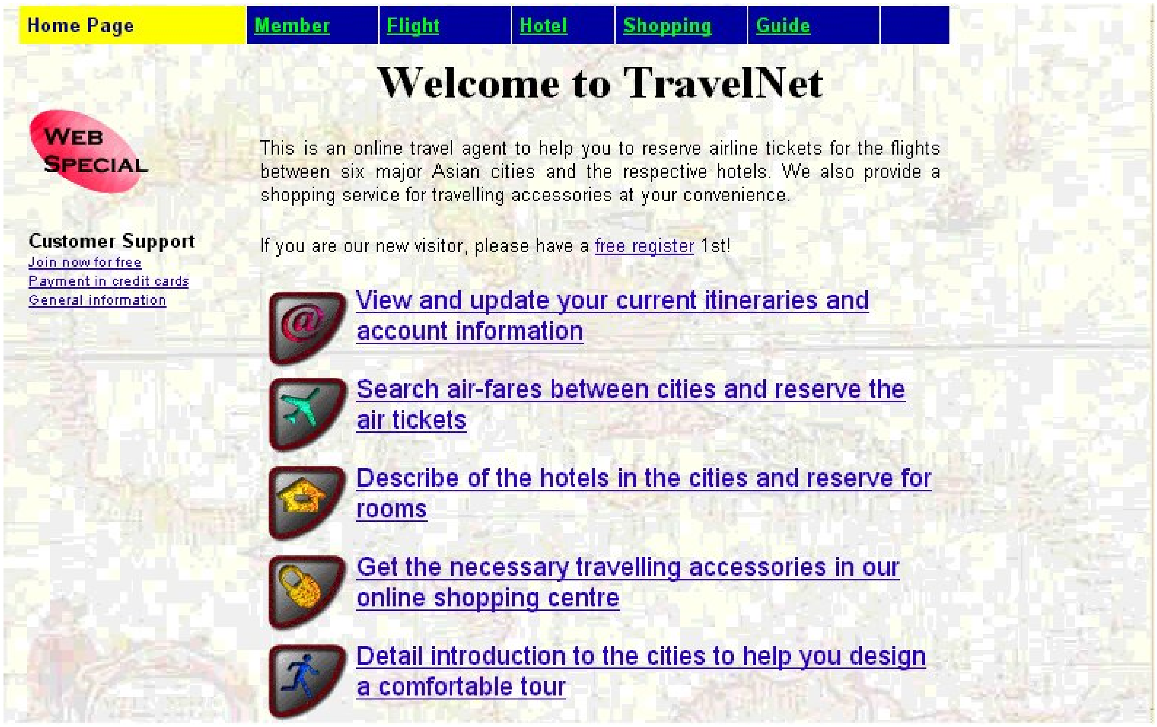
TravelNet is just like other online travel agencies in terms of functionality. However, due to the existence of new technology, we use a relatively new approach to develop our system – the Java Servlet technology that outperforms the old style CGI implementation of providing online applications. Java Servlet is also good in porting application on different platforms with the help of the portability nature of Java.

Besides the centralized approach in TravelNet, distributed approach using CORBA is also developed. In the 1st term report, the centralized version has been discussed in details. In this report, we will first give a brief description over the centralized version and then the distributed approach will be discussed in detail. We will also explain other enhancement made from the original version in the report.

2.Features

2.1 Introduction

TravelNet is an online travelling agency. It is necessary to provide enough faci lities and function such that it makes no difference from other existing on -line agencies. In this chapter, we will describe the facilities and functions provided in TravelNet, which includes Membership, Flight Search and Reservation, Itinerary Managemen t, Travel Accessories Shop, Hotel Information and Travel Guides. The picture below is a screen-shot from the main page of TravelNet.



Welcome to TravelNet

This is an online travel agent to help you to reserve airline tickets for the flights between six major Asian cities and the respective hotels. We also provide a shopping service for travelling accessories at your convenience.

If you are our new visitor, please have a free register 1st!

View and update your current itineraries and account information

Search air-fares between cities and reserve the air tickets

Describe of the hotels in the cities and reserve for rooms

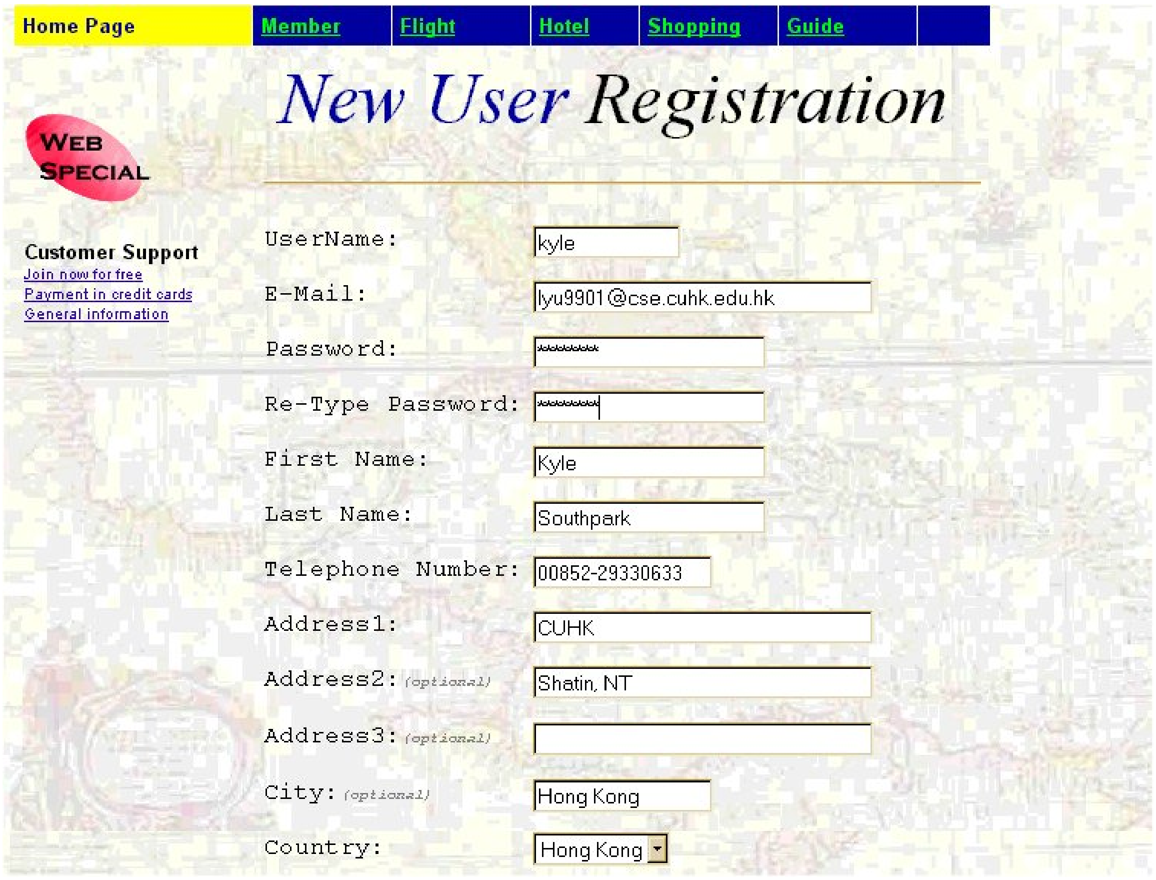
Get the necessary travelling accessories in our online shopping centre

Detail introduction to the cities to help you design a comfortable tour

Figure 2-1: Main page of TravelNet

All the service of TravelNet is listed in this page for users to choose and use.

2.2 Membership



New User Registration

UserName:	kyle
E-Mail:	lyu9901@cse.cuhk.edu.hk
Password:	********
Re-Type Password:	********
First Name:	Kyle
Last Name:	Southpark
Telephone Number:	00852-29330633
Address1:	CUHK
Address2: (optional)	Shatin, NT
Address3: (optional)	
City: (optional)	Hong Kong
Country:	Hong Kong

Figure 2-2: User registration page

In order to use the service of TravelNet, users are required to register for a user account in our system. New users that haven't got a user account can apply for a free user account from us. Once the application is successful, they can enter our system as soon as possible.

The registration for a user account is simple and straightforward. Users are required to input username, e -mail address, password, their real name, telephone number, and address. Since the username should be unique in our system, checking will be carried out to ensure the uniqueness. If the username which is stored in our database already exists in the system, warning will be given out and user should re -enter the username that match the requirement. Any successful registration will be confir med to users by e-mail sending confirmation.

Once users get their user account, they can login to our system to enjoy all services provided. In order to provide enough security to transmitting user password over the network, security feature (SSL) has been implemented for such purpose. The detail of the security feature will be discussed in chapter 3.

The following picture shows the login screen of TravelNet.

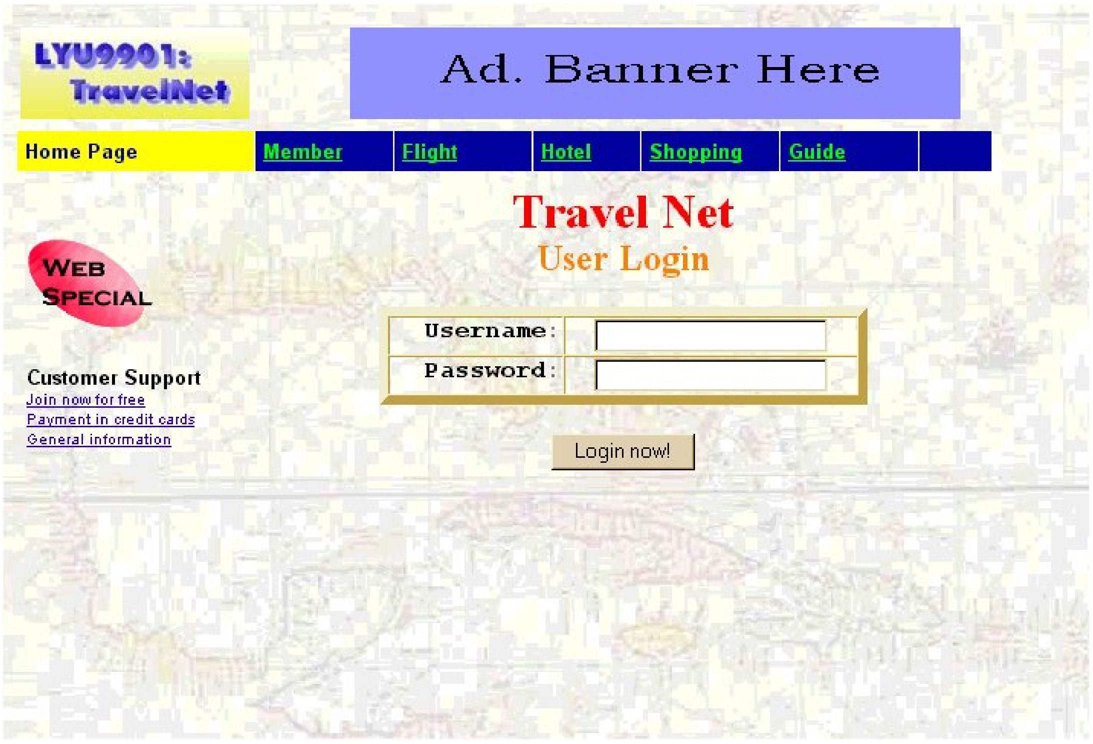


Figure 2-3: User login page

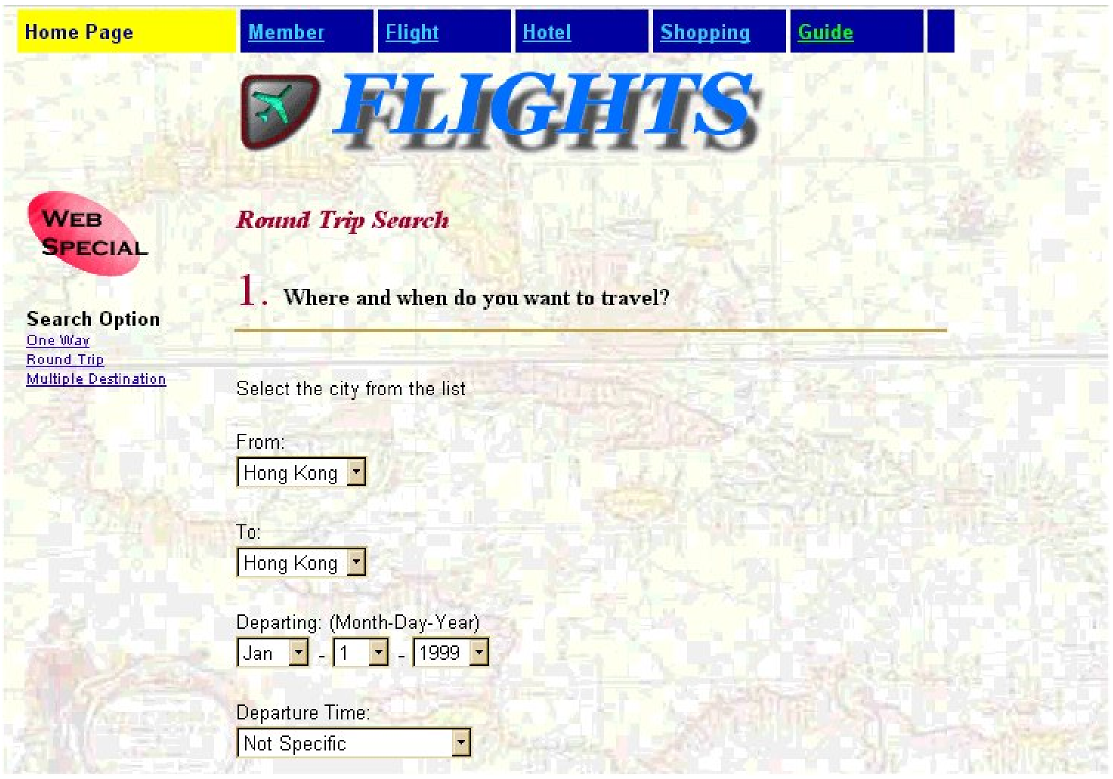
Users can access all membership -related service through the member page. At the member page, users can select to update their own profile entry and the itinerary entry. They can also choose to logout the system when they want to leave. Note that when they want to change their login password of the system, it requires them to supply the old password as a verification of user identity.

2.3 Flight Search and Reservation

Flight search is a key element in the TravelNet. With this feature, users are allowed to consult the airlines ' databases of fulfilling their own set of requirement and make reservation on the search result. The system requires users to input some basic elements on the search. The basic elements of queries includes the departure and arrival cities, the departure date, the types of flight (one way/round trip), the class of service (first class/ business class/ economy class), the age category of the ticket (below 12/adult/above 65). Possible additional requirement includes the exact range for departure time, the choice on fare (e.g. is there any penalties for refund of tickets), the airline company, etc. Usually, the optional requirement helps to lower the size of the search result while the basic method is also provided to enhance the flexibility of the search.

There are 2 types of search for different uses. They are the one way search, the round trip search. One way search is a simple search on the availability and the fare of the single flight. Round trip search is a search that queries on round-trip tour. Usually, a round-trip ticket is cheaper than 2 one-way flight. It is useful and money saving if the users have a definite plan on their trip.

Once the result is generated to users, it allows users to choose the most favourable items put it in the itinerary for further reservation.



Select the city from the list

From: Hong Kong

To: Hong Kong

Departing: (Month-Day-Year) Jan - 1 - 1999

Departure Time: Not Specific

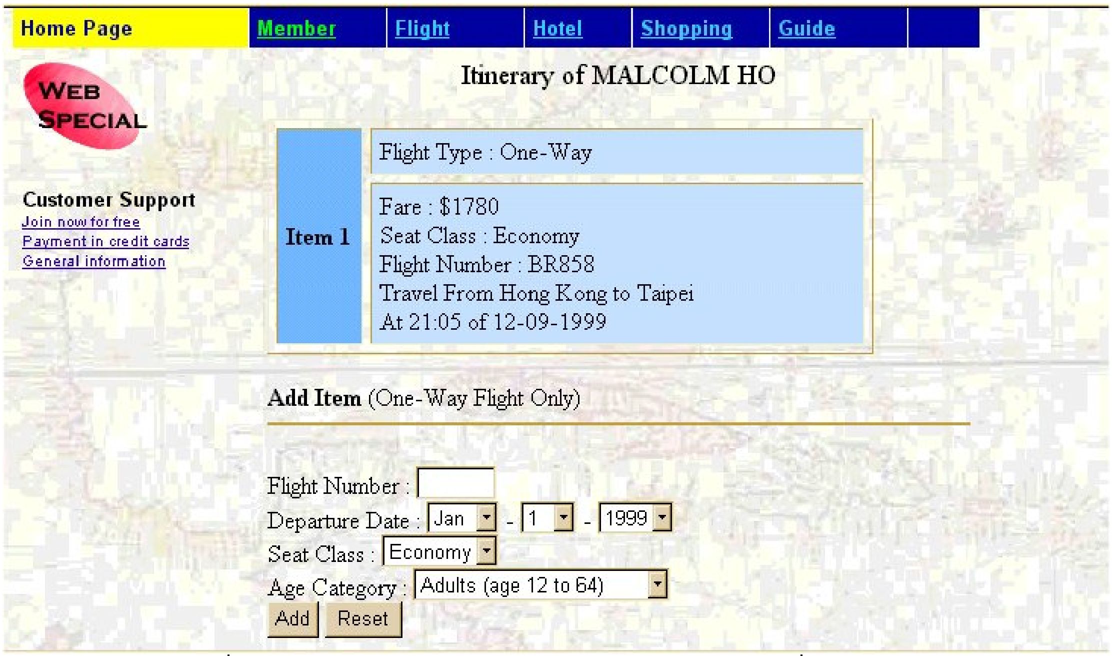
Figure 2-4: One way flight search page

The above picture is the page for one-way search. For convenience purpose, the design of the interface is made such that most of the search options are selected through simple selection of pre -defined values. This lowers the risks of for user to have typo that makes a wrong search.

2.4 Itinerary Management

Each user is associated with an itinerary to their account. It stores the items that the reservations are going to be made. Details of each items are listed clearly so that users can decide to make actual reservation or discard th e item without referring to other sources of information.

Users can edit their own set of itinerary. Normally, users add item to the list through the search result but it is also possible to add it manually by entering all necessary information like flig ht number and departure date. On the other hand, users may delete items if they found it unnecessary to keep.



Itinerary of MALCOLM HO

Item 1

Flight Type : One-Way

Fare : $1780
Seat Class : Economy
Flight Number : BR858
Travel From Hong Kong to Taipei
At 21:05 of 12-09-1999

Add Item (One-Way Flight Only)

Flight Number :
Departure Date : Jan - 1 - 1999
Seat Class : Economy
Age Category : Adults (age 12 to 64)

Figure 2-5 Itinerary Management Page

Moreover, users may choose to reserve flights from their itinerary lists. They will be informed to provide necessary payment information. Result of the operation will be shown no matter it is successful or not.

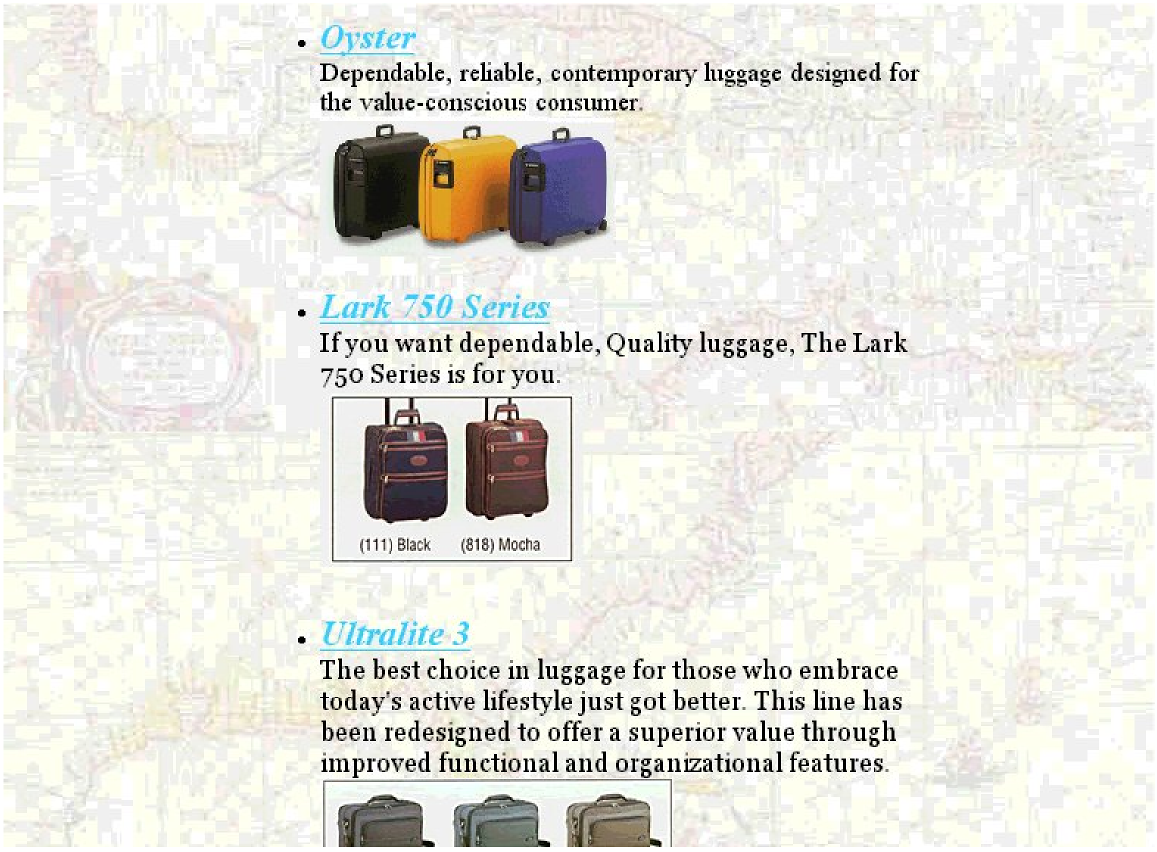
2.5 Travel Accessories Shop

In real life, travellers must have some travelling accessories to bring with them during the trip. Luggages, maps and travel guides are examples of those necessary accessories. To provide a full-integrated service to our users, TravelNet also includes an online travelling accessories shop for travellers to buy the accessories with ease.

In our travel accessories shop, users can buy luggages, maps, guides and other travel related stuffs. Users first select the product that they are interested in to purchase of appropriate amount. Then they can add the item into the shopping basket. Users can check the current content of the shopping basket easily. If they find that they have put in unnecessary thing in the shopping basket, they can remove it from the basket by a simple button.

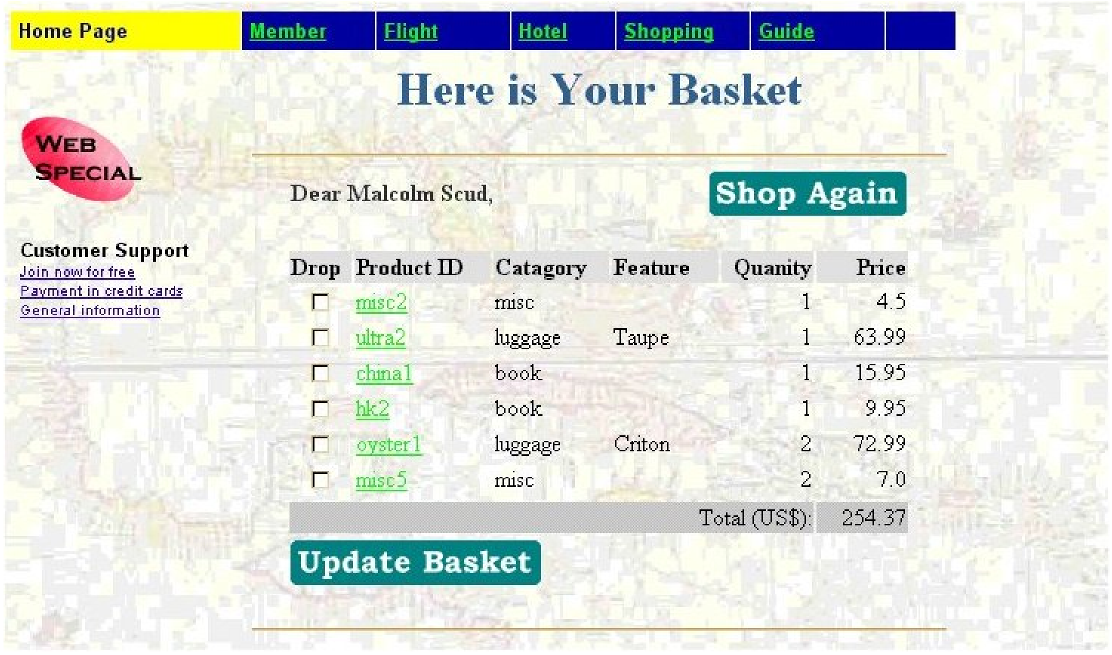
After they have finished shopping, they can check out the items. Originally, the supported payment method is credit card. However, after the enhancement of the system, we now support both payment by credit card and by mondex card. In the case of credit card payment, users are required to enter the name of the cardholder, the expiry-date of the card, the type of the card (Visa/Master) and the corresponding card number for payment. In the case of Mondex payment, users are required to have the mondex card reader and the corresponding plugin software ready before the payment. Both methods will be discussed in detail in chapter 5 of the report.

The picture below shows the shopping picture for luggages. Users can easily add the item by selecting the appropriate quantity of the chosen products and click the "Add to Basket" label.



- Oyster
  Dependable, reliable, contemporary luggage designed for the value-conscious consumer.
- Lark 750 Series
  If you want dependable, Quality luggage, The Lark 750 Series is for you.
- Ultralite 3
  The best choice in luggage for those who embrace today's active lifestyle just got better. This line has been redesigned to offer a superior value through improved functional and organizational features.

Figure 2-6: The snap-shot of part of the travel acessories shop(luggage).



Here is Your Basket

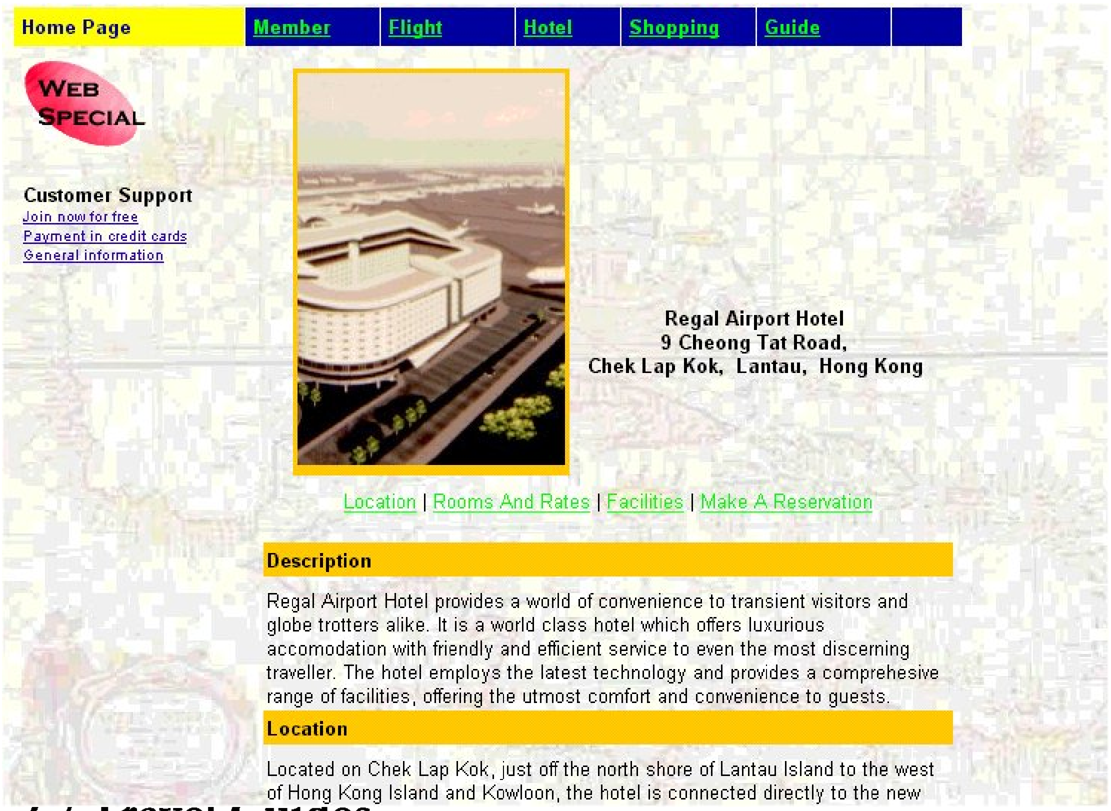
Dear Malcolm Scud,

Drop	Product ID	Catagory	Feature	Quanity	Price
☐	misc2	misc		1	4.5
☐	ultra2	luggage	Taupe	1	63.99
☐	china1	book		1	15.95
☐	hk2	book		1	9.95
☐	oyster1	luggage	Criton	2	72.99
☐	misc5	misc		2	7.0
				Total (US$):	254.37

Figure 2-7: page for viewing shopping basket

2.6 Hotel Information

TravelNet provides hotel information on the different Asian cities. It provides a descriptive information on hotels of their location, fares and service. The basic aim for this function is to help user to choose hotels in their journey.



Regal Airport Hotel
9 Cheong Tat Road,
Chek Lap Kok, Lantau, Hong Kong

Description

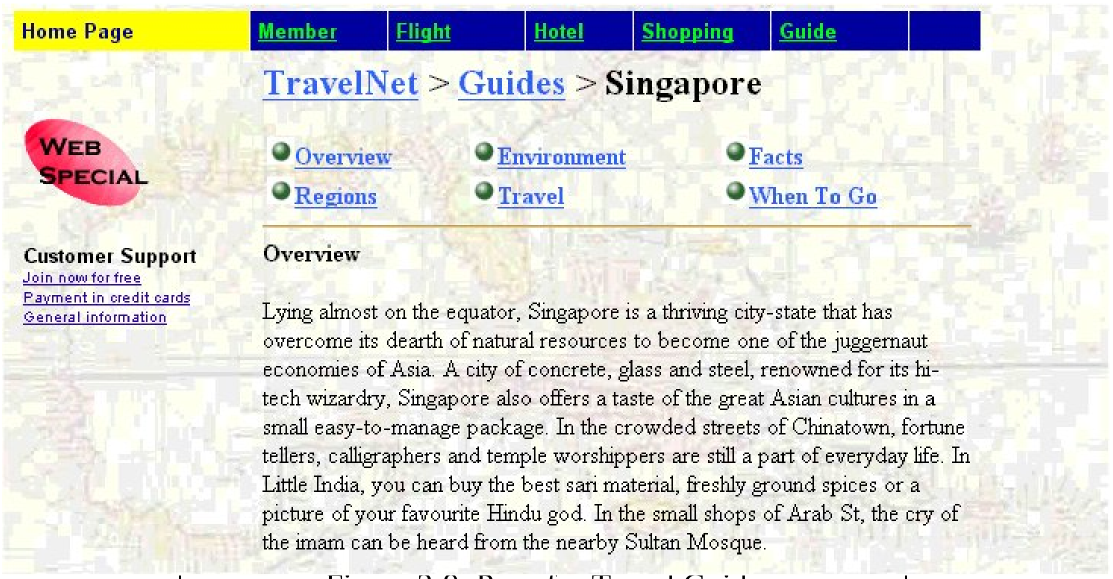
Regal Airport Hotel provides a world of convenience to transient visitors and globe trotters alike. It is a world class hotel which offers luxurious accomodation with friendly and efficient service to even the most discerning traveller. The hotel employs the latest technology and provides a comprehesive range of facilities, offering the utmost comfort and convenience to guests.

Location

Located on Chek Lap Kok, just off the north shore of Lantau Island to the west of Hong Kong Island and Kowloon, the hotel is connected directly to the new

2.7 Travel Guides

TravelNet also provides the online travel guide on different cities. Information likes basic description of the cities, map of the cities, introduction of some famous spot and the c urrency. The basic aim of travel guides is to help users to know the basic information of each cites so that they can plan trips in a convenient way.



TravelNet > Guides > Singapore

Customer Support
Join now for free
Payment in credit cards
General information

Overview

Lying almost on the equator, Singapore is a thriving city-state that has overcome its dearth of natural resources to become one of the juggernaut economies of Asia. A city of concrete, glass and steel, renowned for its hi-tech wizardry, Singapore also offers a taste of the great Asian cultures in a small easy-to-manage package. In the crowded streets of Chinatown, fortune tellers, calligraphers and temple worshippers are still a part of everyday life. In Little India, you can buy the best sari material, freshly ground spices or a picture of your favourite Hindu god. In the small shops of Arab St, the cry of the imam can be heard from the nearby Sultan Mosque.

Figure 2-9: Page for Travel Guide

3. Basic System Design

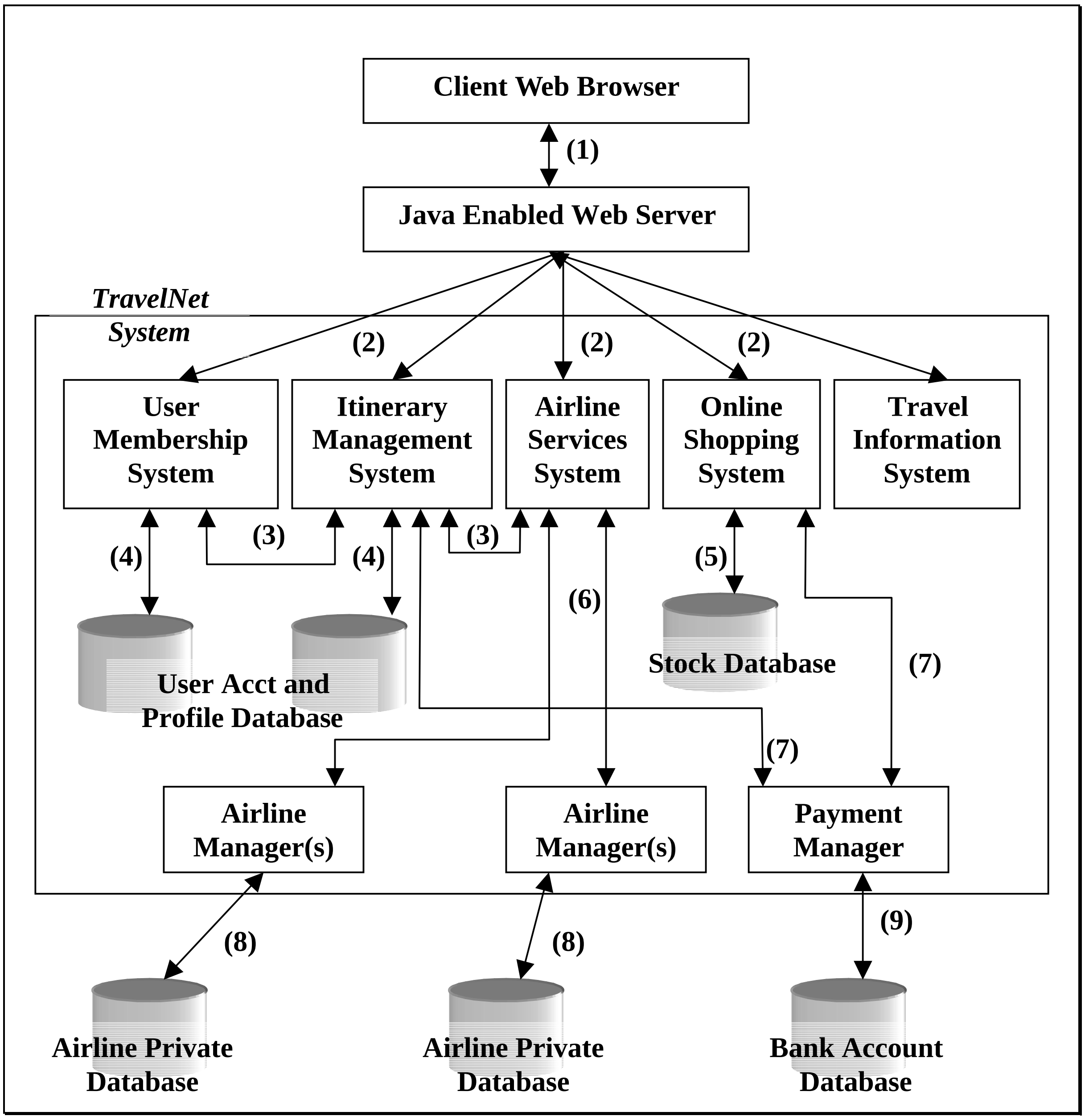
3.1 Introduction

In this chapter, it will cover t he basic system design issue of the TravelNet. The basic design of the TravelNet can be referred as the centralized version of the system without any enhancement. The enhanced version will be discussed in the next chapter.

The content of the chapter is organized in the following way:

- Overview of the Architecture: A broad view on the infrastructure and the data flow between the components
- Details of the main components: A detailed discussion of the main components, communication interfaces and its databases.

3.2 Overview of System Architecture



(1) Communication between Web Server and Client Browser
(2) Communication between Web server and Components of TravelNet
(3) Local access between components
(4) Components access to local user profile and itinerary databases
(5) Component access to local stock database
(6) Consulting airline(s) on flight queries and reservation
(7) Payment request to payment manager through bank interface
(8) Airline own access to its local database
(9) Bank own access to its local database

Figure 3-1:
Architecture of TravelNet

Description of the data flows:

(1) The Client web browsers will retrieve information and generate requests to the web server, which TravelNet is hosted on by means of standard HTTP protocol. In normal situation, the information transmitted between is not confidential data that the data will not be encrypted. This can ensure a faster response. However, in some occasion that user's private information, like password and credit card number, are transmitted, SSL connection are provided that it can lower the risk of data being captured and interpreted by third parties.

(2) Web server that is Java enabled will direct request and call the appropriate components of TravelNet to provide services. Requests can be divided into 2 types: a) requests for static pages like travel guides, and b) requests for dynamic service, which involves servlet invocation.

(3) There is the possibility that one particular service is done together by the cooperation of different components in the system. For example, itinerary list is updated once the reservation of flight succeeds. There should be a communication channel between these components for such cooperation to exist. In Java, calling corresponding object's method, which is a general strategy of message passing in object oriented programming environment, can easily do this.

(4) There is a database to store the user account information, which includes the user profile and their itinerary list. Since they are local to the system, all access to these databases is done by direct connection using Java Database Co nnectivity (JDBC).

(5) Again, for the online travel accessories shop, it has a stock database to keep track of the stock information. It is similar to the situation of user account database that they are local to the system and can be accessed directly using JDBC.

(6) Flight related operations are needed for booking flight and queries. Since TravelNet should not have right for direct access to the databases of each airline. Therefore, all the operations are provided abstractly by Airline Manager, which serves as a dealer to the particular airline. These Airline Managers should be act as a coded client provided by each airline to support such operations.

(7) Payment request will be generated during transactions. Similar to the case of flight operations, all banking operations are done through the payment manager via the Bank Interface.

(8) This is the internal access between the Airline Manager and the own set of databases. It is outside of TravelNet system

(9) This is the internal access between the Payment Manager and the own set of bank databases, It is outside of TravelNet system

3.3 User Membership System

The User Membership System is responsible for managing user accounts in TravelNet. The functions provided in this module includes Login/Logout, Account registration, and Profile management. It also consists of a database to store the user information.

Database Structure:

- ***USER_PROFILE***

Name	Type	Nullity	Integrity
USERNAME	VARCHAR2(12)	NOT NULL	PRIMARY KEY
EMAIL	VARCHAR2(30)	NOT NULL	
PASSWORD	VARCHAR2(20)	NOT NULL	
FIRSTNAME	VARCHAR2(20)	NOT NULL	
LASTNAME	VARCHAR2(20)	NOT NULL	
TELENUM	VARCHAR2(15)	NOT NULL	
ADDRESS	VARCHAR2(90)	NOT NULL	
CITY	VARCHAR2(15)		
COUNTRY	VARCHAR2(5)		
CREDITNO	VARCHAR2(16)		

3.4 Itinerary Management System

Itinerary Management System is responsible for managing registered users' itinerary in TravelNet. It works closely with the user management system and the airline service system in its operations. Only registered users have rights to using the itinerary management system and reservation of flights are only possible when it is in the itinerary. The function provided in this module includes view itinerary, add item to itinerary and remove item from itinerary.

Communication interface: Itinerary Manager

Itinerary Manager is the general controller of handling all itinerary related operation.

<u>Add item</u>

BOOLEAN ADD_ITEM (USER_ID, FLIGHT_INFO)
THROWS (FLIGHT_NOT_EXIST)

Input:

USER_ID: the specific user
FLIGHT_INFO: information that can be used to identify a flight. e.g. date of flight, flight number, etc.

Output:

BOOLEAN: True if success, else false

Exception:

FLIGHT_NOT_EXIST: No such flight available.

<u>View item</u>

ITEM_INFO_LIST GET_ALL_ITEMS (USER_ID)

Input:

USER_ID: the specific user

Output:

ITEM_INFO_LIST: the itinerary list that associate with the specific user

<u>Remove item</u>

BOOLEAN REMOVE_ITEM (USER_ID, ITEM_NO)
THROWS (ITEM_NOT_EXIST)

Input:

USER_ID: the specific user
ITEM_NO: the number of item that is going to be deleted

Output:

BOOLEAN: True if success, else false

Exception:

ITEM_NOT_EXIST: No such item in the itinerary

Database Structure

- ***TN_USER_ITINERARY***
  A table storing all users' itinerary in TravelNet

Name	Type	Nullity	Integrity
USER_ID	VARCHAR2(12)	NOT NULL	
FLIGHT_INFO	VARCHAR2(100)	NOT NULL	

3.5 Airline Service System

Airline Service System is responsible for flight search and reservation. It makes use of the provided Airline Manager to consult the databases of different airlines.

Communication Interface: Airline Manager

<u>Flight information query</u>

```
FLIGHT_ID FLIGHT_QUERY (DEPARTURE_DATE, DEPARTURE_TIME
            SOURCE, DESTINATION, TYPE_OF_FLIGHT, CLASS_OF_SEAT,
            AGE_GROUP, USER_REQUIREMENT)
        THROWS  (NO_FLIGHT_MATCH)
```

Input:
DEPARTURE_DATE = the desired departure date of the flight
DEPARTURE_TIME = the desired departure time of the flight (Optional)
SOURCE = the source city for the customer to take off
DESTINATION = the destination city for the customer
TYPE_OF_FLIGHT = one-way and round trip
CLASS_OF_SEAT = Economy, Business, 1st Class
USER_REQUIREMENT = terms of tickets
AGE_GROUP = age group of the customer

Output:
FLIGHT_ID = the flight ID of the specific flight in the airline company

Exception:
NO_FLIGHT_MATCH = This airline doesn't provide the tickets match the specified requirement.

Flight booking request

```
BOOLEAN FLIGHT_BOOK (DEPARTURE_DATE, FLIGHT_ID
          TYPE_OF_FLIGHT, CLASS_OF_SEAT, AGE_GROUP,
          USER_REQUIREMENT, USER_INFORMATION)
    THROWS (NO_FLIGHT_MATCH, BOOKING_FULL)
```

Input:
DEPARTURE_DATE = the desired departure date of the flight
FLIGHT_ID = the flight ID of a specific flight
TYPE_OF_FLIGHT = one-way and round trip
CLASS_OF_SEAT = Economy, Business, 1st Class
USER_REQUIREMENT = terms of tickets
AGE_GROUP = age group of the customer
USER_INFORMATION = the information of the customer who book the ticket.

Output:
BOOLEAN = true if success, else false

Exceptions:
NO_FLIGHT_MATCH = This airline doesn't provide the tickets match the specified requirement.
BOOKING_FULL = the specified booking is already full

Database Structure

- ***FLIGHT_INFO***
  A database stores all the flights operated by the airline company.

Name	Type	Nullity	Integrity
FLIGHT_NUM	VARCHAR2(6)	NOT NULL	PRIMARY KEY
SRC_PLACE	VARCHAR2(3)	NOT NULL	
DEST_PLACE	VARCHAR2(3)	NOT NULL	
DTIME	DATE	NOT NULL	
ATIME	DATE	NOT NULL	
AIRCRAFT	VARCHAR2(4)	NOT NULL	

- ***FLIGHT_SCHEDULE***
  A database for weekly schedule of specific flights

Name	Type	Nullity	Integrity
FLIGHT_NUM	VARCHAR2(6)	NOT NULL	PRIMARY KEY
SUN	VARCHAR2(1)	NOT NULL	
MON	VARCHAR2(1)	NOT NULL	
TUE	VARCHAR2(1)	NOT NULL	
WED	VARCHAR2(1)	NOT NULL	
THU	VARCHAR2(1)	NOT NULL	
FRI	VARCHAR2(1)	NOT NULL	
SAT	VARCHAR2(1)	NOT NULL	

- ***FARE_INFO***
  A database stores the fare list of each class of tickets in terms of one-way flights and round-trip flights.

Name	Type	Nullity	Integrity
FLIGHT_NUM	VARCHAR2(6)	NOT NULL	PRIMARY KEY
OW_FCLASS	FLOAT(10)	NOT NULL	>0
OW_BCLASS	FLOAT(10)	NOT NULL	>0
OW_ECLASS	FLOAT(10)	NOT NULL	>0
RT_FCLASS	FLOAT(10)	NOT NULL	>0
RT_BCLASS	FLOAT(10)	NOT NULL	>0
RT_ECLASS	FLOAT(10)	NOT NULL	>0

- ***PLANE_SIZE***
  A database stores the capacity of each plane of 3 classes of service(first class/business class/economy class).

Name	Type	Nullity	Integrity
AIRCRAFT	VARCHAR2(4)	NOT NULL	PRIMARY KEY
FCLASS	NUMBER(3)	NOT NULL	
BCLASS	NUMBER(3)	NOT NULL	
ECLASS	NUMBER(3)	NOT NULL	

- ***TICKET***
  A database stores the capacity of each plane of 3 classes of service(first class/business class/economy class).

Name	Type	Nullity	Integrity
FLIGHT_ID	VARCHAR2(6)	NOT NULL	PRIMARY KEY
DDATE	DATE	NOT NULL	PRIMARY KEY
FCLASS	NUMBER(3)	NOT NULL	
BCLASS	NUMBER(3)	NOT NULL	
ECLASS	NUMBER(3)	NOT NULL	

- ***USER_ITINERARY***
  A database which stores the sold ticket for internal usage.

Name	Type	Nullity	Integrity
TICKET_NUM	VARCHAR2(12)	NOT NULL	PRIMARY KEY
FLIGHT_NUM	VARCHAR2(6)	NOT NULL	
NAME	VARCHAR2(40)	NOT NULL	

Note: The above is the database schema for each airline company. Since it is not available to have multiple database for us to use, we simply simulate the situation by append a code as a prefix to the database table to represent the ownership of the table. For example, the code for Cathay Pacific Airways is CX, so all the tables that belongs to the company are started with CX_, like CX_TICKET and so on.

3.6 Online Shopping System

Online Shopping System is the system to provide selling servic e of travel accessories. It mainly consists of a stock database and a shop basket system. For the payment part, it will link to the payment system that will be discussed later in the section.

Shop Basket

The basket contains a list of shopping items. It provides operations to add, remove and get related information of the basket. Operations are listed below:

Put a shop item into basket:
VOID PUT_SHOP_ITEM (PRODUCT_ID, PRICE, QUANITY, PRODUCT_TYPE, OTHER_DETAIL)

***Remove an item from basket*:**
ITEM REMOVE (PRODUCT_ID)

Get the price of an item in the basket:
FLOAT GET_PRICE(PRODUCT_ID)

Get the quantity of an item in the basket:
INT GET_QUAN(PRODUCT_ID)

Get the other detail of an item in the basket:
STRING GET_DETAIL(PRODUCT_ID)

Get the total amount of all items in the basket:
FLOAT GET_TOTAL()

Database Structure

- ***STOCK***
  Inventory stock will be stored in this database. It reveals the actual stock of TravelShop.

Name	Type	Nullity	Integrity
PRODUCT_ID	VARCHAR2(10)	NOT NULL	PRIMARY KEY
PRICE	FLOAT(126)	NOT NULL	>0
STOCK	NUMBER(38)	NOT NULL	>0

- ***TRANSCATION_RECORD***
  Payment transactions will be recorded in here. For later reference or complain from users.

Name	Type	Nullity	Integrity
TRANS_NO	NUMBER(38)	NOT NULL	PRIMARY KEY
CARD_NO	VARCHAR2(16)	NOT NULL	
AMOUNT	FLOAT(126)	NOT NULL	>0
TRANS_TIME	DATE	NOT NULL	

3.7 Travel Information System

Travel Information System is responsible for providing travelling related information, which consists of Hotel Information and the Travel Guides. They are static html pages resided in the web server.

3.8 Payment System

Payment System is responsible for managing payment related service to components in the TravelNet. Both airline reservation and online travel shop will make use of this system. At the basic approach, the payment system is simple and no security issue is concerned. Also, it can only accept credit card payment. An enhanced version is thus developed which will be discussed in other parts of this report.

Communication Interface: Payment Manager

Visa card validation interface
VALIDATE_VISA (VISA_NUMBER, CARD_HOLDER_NAME, EXPIRE_DATE)
    THROWS (INVALID_VISA)

This interface allows client (TravelNet) to check whether the corresponding visa card information is valid according to the bank database.

Input:
VISA_NUMBER = the visa card number to be checked
CARD_HOLDER_NAME = the name written on the visa card
EXPIRE_DATE = the expire date of the visa card

Exception:
INVALID_VISA = Invalid visa card information. It may be card number integrity error or expire date / holder name not match the specific card.

Visa card debit credit interface

```
DEDUCT_CREDIT_FROM_VISA_CARD (VISA_NUMBER,
                CARD_HOLDER_NAME, EXPIRE_DATE, DEBIT_AMOUNT,
                CREDIT_ACCOUNT)
        THROWS (INVALID_VISA, NOT_ENOUGH_CREDIT,
                CREDIT_ACCOUNT_NOT_EXIST)
```

Input:
VISA_NUMBER = the visa card number to be checked.
CARD_HOLDER_NAME = the name written on the visa card.
EXPIRE_DATE = the expiry date of the visa card.
DEBIT_AMOUNT = the amount to be debit from the visa card.
CREDIT_ACCOUNT = the bank saving account the amount to be credited to.

Exception:
INVALID_VISA = Invalid visa card information. It may be card number integrity error or expire date / holder name not match the specific card.
NOT_ENOUGH_CREDIT = the credit for this credit card is not enough for this amount of payment.
CREDIT_ACCOUNT_NOT_EXIST = the credit saving account did not exist at all.

Database Structure

- ***BANK_VISA***
  A database for all the credit cards information that will be used in our community. This database can't be accessed directly by TravelNet. All the accesses of this database are through the Payment manager.

Name	Type	Nullity	Integrity
NAME	VARCHAR2(30)	NOT NULL	
VISANUM	VARCHAR2(16)	NOT NULL	PRIMARY KEY
CREDIT	FLOAT(126)	NOT NULL	
EXPIRE	DATE	NOT NULL	

- ***BANK_SAVING***
  This database stored saving accounts of the bank.

Name	Type	Nullity	Integrity
ACC_NUM	VARCHAR2(20)	NOT NULL	PRIMARY KEY
NAME	VARCHAR2(40)	NOT NULL	
AMOUNT	FLOAT(126)	NOT NULL	>0

3.9 Web Site Map

The web site is well structured using the functions provided in TravelNet. Each branch corresponds to a module of TravelNet system

The figure followed shows the hierarchy of TravelNet

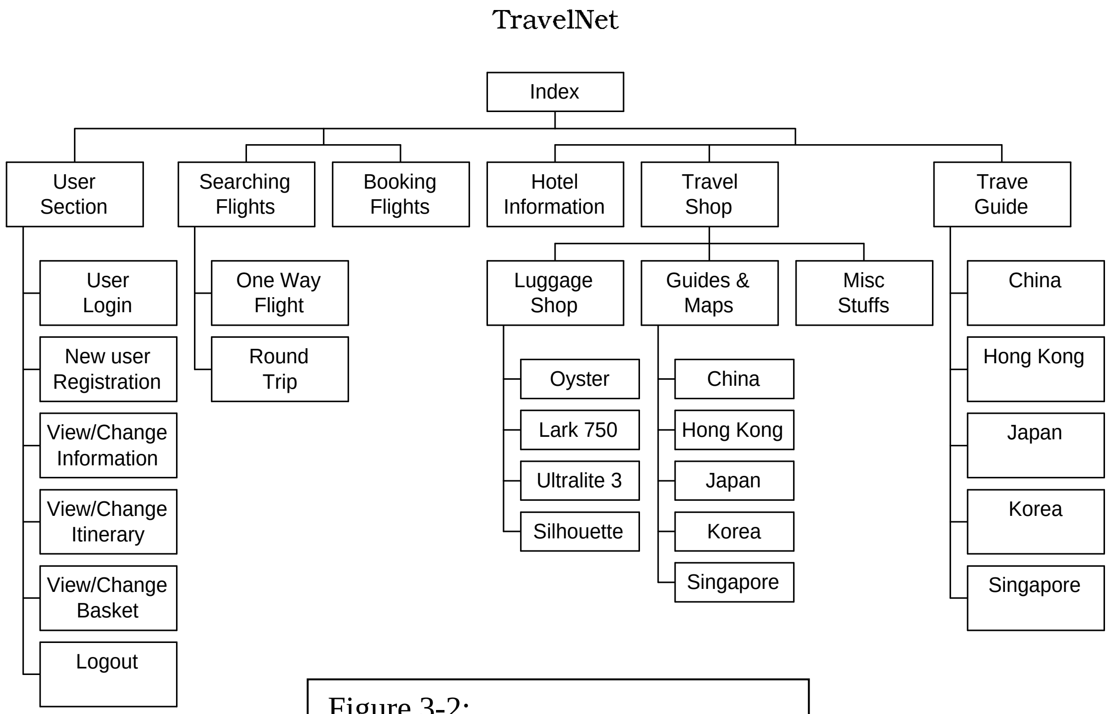


Figure 3-2:
Site hierarchy of Travel Net

4. Enhancement of TravelNet

4.1 Introduction

System is difficult to be perfect in its first built. The development of TravelNet has no exception to this. After we have built the original centralized version, we evaluate it and encounter some of its discrepancies. In this chapter, we will give an overview to the enhancement made to the TravelNet from the basic system design stated in the last chapter. Later in this report, each enhancement will be discussed in full detail.

4.2 Overview of Enhancement

The enhancements are made basically on the basic architecture of the system. This can ensure that it can achieve a high level of compatibility of the system without the need to rewrite a large amount of code. The enhancement made is mainly on three different ways:

a) Payment Methods
b) Distributed Components
c) Simplification of Components

Note that most of these e nhancements are made independent of the user interface. Therefore, from the view of users, there is almost no difference between the basic version and the enhanced version. It is an important concern for an application that the internal design of the sys tem and its user interface should be made separated. Users should not aware of any changes in internal design in the interface.

4.2.1 Payment Methods

In the original design of our application, we have included a payment manager for the payment operation to the TravelNet. We have also built a simple bank to simulate the credit and debit operation between users and TravelNet. Although the system works fine with this implementation, a lack of concern on the security issue of this credit card based payment method makes it inappropriate and impractical in the e-commerce environment. A secured payment method is always one of the key elements for the success of an e-commerce online application.

In order to deal with the situation, we cooperate with Mr. Steve K.L.Chong[1] on implementing a more secured channel for credit card based payment. The detail of this payment method will be discussed in the next chapter.

Besides the enhancement of the original credit card based payment method, we also like to include ne wer payment method as well. Thus we have made the system to support another payment method – the mondex card. Mondex is one of the leading technologies in micropayment. The detail of this payment method again, will be discussed in the next chapter.

4.2.2 Distributed Components

The original basic design of the system is a centralized one that most of the operations are done on the server of TravelNet. Even some of these components should be accessed in a more distributed way, they are mainly developed a nd run in a centralized manner. This is an unfavourable act on these components. Thus, we have identified these components and modified it to work in a distributed manner.

The technique we used is CORBA, which is a general standard on developing distributed application. With the help of Java, CORBA integration in the system is made possible. In fact, building a distributed version is one of our objectives in the project.

4.2.3 Simplification of Components

Simplification of components is an important process of building applications, especially that they are large -scaled. The application system is expanding when it undergoes its development phase. It will be much chance that some of the components are redundant and over -complex. By simplifying these components,

[1] Mr Chong is a postgratuate student of CSE CUHK.

maintenance of the system is made easier and it can benefit the further development of the next system upgrade.

In our system, we have made use of the Java Server Pages (JSP) technology to simplifying such components in our system. The issue   of this part will be discussed in the chapter after the part of Distributed Components in the report.

5. Payment Methods

5.1 Introduction

In this chapter, we will focus on the detail of the two payment methods provided as enhancement in TravelNet.

5.2 Secured Credit Card Payment

Introduction

Credit card payment perhaps is the most popular form of payment method used in Internet. Most of the online merchants can support credit card as the payment method for service and goods. Security is a major co ncern in the payment process as private information like credit cards number are transmitted during the process. Any insecure channel of transmission of this kind of information gives a high -risk exposure of the secret. Customers will bear a high risk of loss in this way. In order to increase the confidence of customers to obtain service and buy goods on the Internet, a secured channel of credit card payment must be provided.

There exist many different electronic payment protocols to tackle the situatio n. In TravelNet, we make use of the one developed by Steve K.L Chong which can achieve a certain degree of security without a great affection to the performance.

Payment Model

There are four major entities involved in payment system. They are customers, merchants, a payment gateway and banks. The Certificate Authority will manage the certificate and those public keys required for the entities. RSA public -key cryptography is used for authentication and encryption purposes. A pair of private/public keys is generated by the customer or by a trusted third party, i.e. the Certificate Authority.

Before the description of the payment system, we introduce the conventions that are used in the message content.

- address: The mailing address of the customer.
- amt: The total amount of the purchased goods.
- card_name: The name of the credit card holder.
- card_no: The credit card number of the customer.
- card_type: There are three types of credit card: MasterCard (MC), VISA (VS), and American Express (AE).
- e_date: The expiry date of the customer's credit card.
- p_opt: There are two payment options: using credit card (CC), and using electronic coins (EC).
- prod_id: An identification number for different products.
- quan: The total quantity of the purchased goods.
- receipt: An unique number recording the transaction for future retrieval when needed.
- RESULT: An acknowledgement from acquirer to merchant, and also from merchant to customer, stating whether the transaction is completed or aborted.
- SIG: The digital signature of a message. It uses the sender's private key to sign on message digest.
- X_cert: A public-key certificate of different parties, denoted by X. It is composed of the acquirer's name, the public-key, trusted third party's name. X = Payment Gateway (pg) or bank (bank).
- X_id: An 8-digit unique number for different parties X. X = bank (bank) or merchant (m).
- X_name: The name of party X. X = customer (cust), or merchant (m).
- X_priv: The private key of party X. X = PG (pg), bank (bank), customer (cust), or merchant (merc).
- X_pub: The public key of party X. X = PG (pg), bank (bank), customer (cust), or merchant (merc).

The mechanism of the payment model is shown in Figure 5-1. The payment process is described in four steps, and the details of the information flows are as follows:

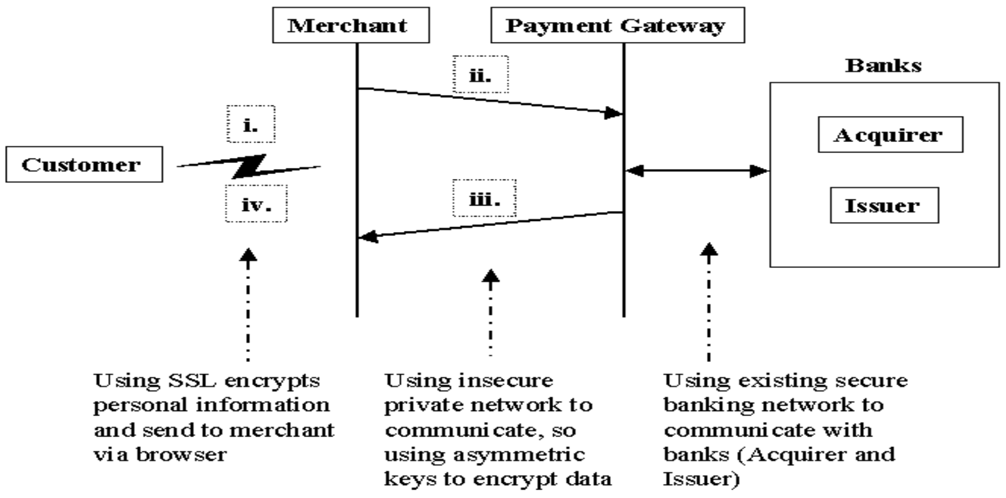

Merchant
Payment Gateway
Banks
Acquirer
Issuer
Customer
i.
ii.
iii.
iv.
Using SSL encrypts personal information and send to merchant via browser
Using insecure private network to communicate, so using asymmetric keys to encrypt data
Using existing secure banking network to communicate with banks (Acquirer and Issuer)

Figure 5-1: The Payment System and Its Payment Process Flows

i. The customer first goes to the merchant's homepage and browses products, and puts the selected goods into a virtual basket. After the customer finishes choosing the products, the payment process is triggered by clicking a button. A secure connection between the customer and the merchant is established using SSL protocol for communications. The customer then enters personal information and credit card information into the browser. In addition, the product information and the total amount will be included in the message which is sent to the merchant. The message content (MC1) in this step is

MC1: $\{card\_name, card\_no, e\_date, card\_type, address, prod\_id, quan, amt, p\_opt\}_{by\ SSL}$

ii. Upon the receipt of message MC1, the merchant can get the personal information and credit card information of the customer. The merchant then requests payment authorization and validation of credit card from cardholder's financial institution by composing a message (MC2) which consists of the customer's personal and credit card information, together with the total amount and the merchant's name. This message will be encrypted by the merchant's private key to serve as an authentication. A header, which contains the merchant identification number and a number, denoting the payment option the customer chose, is attached to the message. The whole message is encrypted with the payment gateway's public key to prevent eavesdropping and message tampering. At this step, the merchant will send out the message packet to the PG as

MC2: $\{\{card\_name, card\_no, e\_date, card\_type, amt, m\_name\}_{merc\_priv}, m\_id, SIG, p\_opt\}_{pg\_pub}$

iii. When the PG receives the message (MC2) from the merchant, the PG first uses the private key to decrypt the message to get a decrypted message and a header. The PG will notice the message is sent by a specific merchant but only the merchant's public key can decrypt the header message. Next, PG will communicate with the issuer (the bank issue customer's credit card) and the acquirer (the bank where merchant's account resides) thr ough an existing banking network which is assumed secure. After the PG receives the response from the issuer and the acquirer, the PG will compose a message (MC3) including the response (whether the credit card is valid and the purchase is within the credi t limit) and a receipt to the merchant for record purposes. It is then encrypted by the PG's private key for authentication. In addition to the message, the PG's certificate is adhered to the message. The whole message is encrypted by the merchant's public key for privacy and security purpose.

MC3: $\{\{RESULT, receipt, m\_name\}_{pg\_priv}, SIG, pg\_cert\}_{merc\_pub}$

iv. Upon the receipt of the PG's message, the merchant will decrypt the message using the private key and then using PG's public key to obtain the original message. After checking the result, the merchant will compose a message (MC4) to inform the customer if the purchase is successful or not. The message will be displayed as an html document for the customer. The message can be decrypted by the SSL for the privacy purpose.

MC4: $\{RESULT, receipt, prod\_id, quan, card\_name, address\}_{by\ SSL}$

Implementation in TravelNet

The first step is to replace the existing payment manager by a connection representative to the payment gateway supplied by the payment system. All necessary messages are diverted to this representative for verification and debit.

In order to carry out a payment process, user credit card information is collected through a SSL channel when users initiate a payment request. The use of SSL can ensure that there is no exposure of the private information during the transmission of data between client and TravelNet. After verifying the internal status of TravelNet system(e.g. the accessory that the user purchased exists in the shop), we will connect to the payment gateway (PG). Our server then requests a payment from a specific credit card. Message to PG will be encrypted by an agreed public key of PG and TravelNet's private key will be used for authentication (MC2). An acknowledgement of a successful or unsuccessful transaction will be encrypted by TravelNet's public key and send back from PG to TravelNet (MC3).

Communication Interface: PG representative

STATUS_ID OPERATION (CARD_NAME, CARD_NO, E_DATE, CARD_TYPE,
M_ID, M_NAME, AMT)

Input:

CARD_NAME = name of card holder
CARD_NO = credit card number
E_DATE = expiry-date of the credit card
CARD_TYPE = Visa or Master credit card
M_ID = Merchant ID (TravelNet's ID in the PG)
M_NAME = Merchant Name(TravelNet)
AMT = Amount of money to be debited to the users

Output:

STATUS_ID = indication of successness of the payment transaction

Performance Measurement

In our experiments, the server always allows concurrent users to request a payment and all the requests can be executed concurrently. The merchant, however, can specify the type of execution scenario, either sequential or concurrent. For a single request, the total checkout time in TravelNet is between 1.7 seconds and 2 seconds. The time could be as long as 10 seconds in the worse scenario. To filter out noises, we perform 5 executions to obtain the average time measure for each data point in every experiment.

The performance measurement is based on two different models: Multiple-threaded model and single-threaded model. In the multiple-threaded model, requests are processed in parallel. Each request will obtain only a portion of the server resources, which is inversely proportional to the number of requests. For example, when there are 10 concurrent users requests, each client process will be on the average 10 times slower than each executing alone, as each of them only grasp 10% of the server resources. The time of overlapping processes will consequently be longer. There is also an extra task-switching overhead that is very significant when the number of tasks becomes large. As displayed in Figure 5-2, the payment process time increases as the number of concurrent user increases. We can also see in Figure

5-2 that the total payment process time is divided into two parts: time spent on the Merchant clien t, and time spent on the Payment system server. In terms of the portion of time spent for the total checkout process, payment server contributes over 80%.

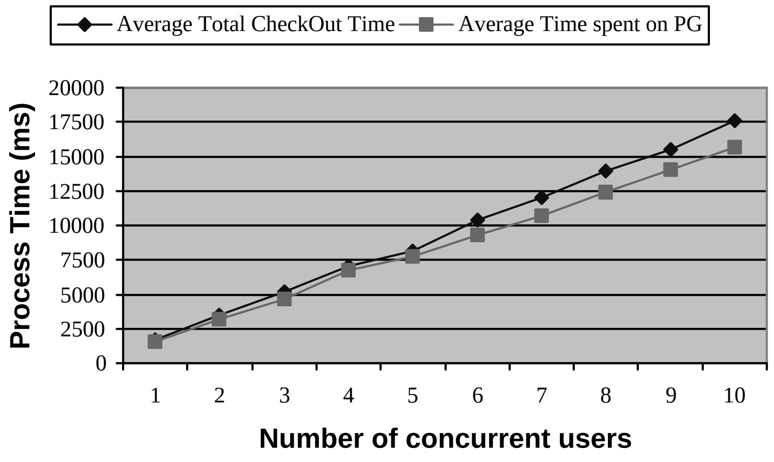


Fig. 5-2: Payment Transaction Time in Multiple-Threaded Model

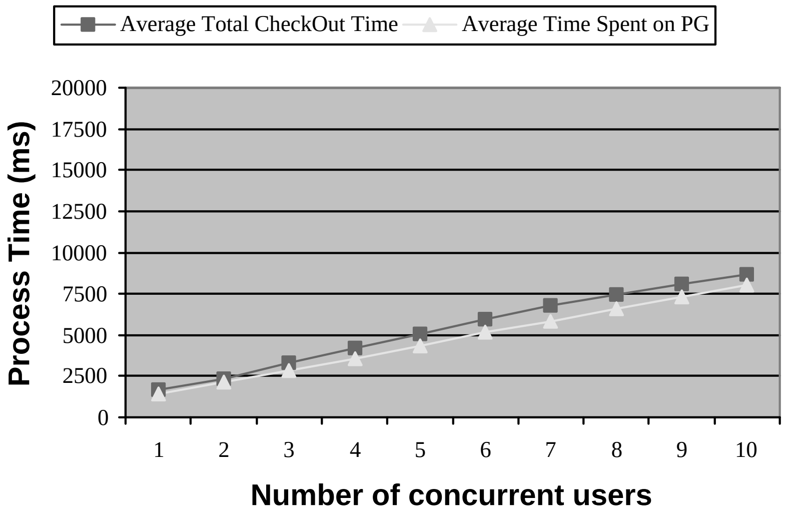


Fig. 5-3: Payment Transaction Time in Single-Threaded Model

In the single -threaded model, TravelNet clients request in a first -come-first-serve manner. Every request waits for all the previous requests to be finished before it can gain access to the server resources. Figure 5 -3 shows the average total process time and the time spent on PG for the single -threaded model. As a comparison, we can see from Figure 5 -4 that its average process time is much shorter than that of the multiple-threaded model. The main reason is due to database resource conflict for the multiple-threaded model when the multiple concurrent processes access the PG, which currently has only one merchant, namely, TravelNet. As the PG server resources have to be shared among the multiple requests, the requests will hold resource (e.g., lock a data item) and compete with each other, thus delaying the complete time. In the single-threaded model, server resources are not shared among the requests and only a task-switching time is necessary between each request. As the response time is quite important in such an interactive application, the single -threaded model behaves better than the multiple -threaded model. It is noted, however, that if we have multiple merchants in the PG which handles different requests with independent merchants, the multiple-threaded model would be significantly improved.

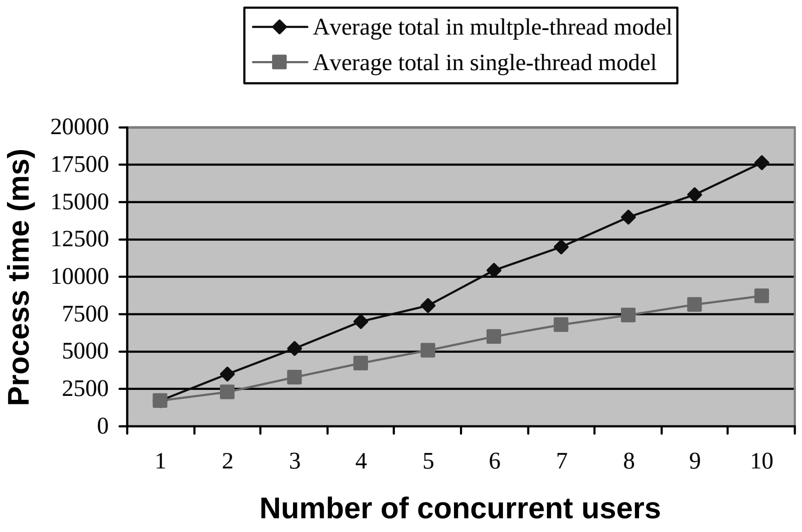


Fig.5-4: A Comparison for Single-Threaded and Multi-Threaded Model

The payment processing time ca n be divided into two parts as well: the time required to perform cryptography algorithms (including message encryption and decryption), and the time required to transmit messages and handle payments. Figure 5-5 shows the comparison on the payment process time on the PG regarding the overhead due to cryptography. We found that when the number of concurrent users increases, the gap showing the difference on the process time between using cryptographic algorithms and without using them becomes larger. This ov erhead indicates that for a more secure payment system, there is a tradeoff on the time to handle payment transactions. This tradeoff is quantitatively provided in TravelNet for a detailed analysis.

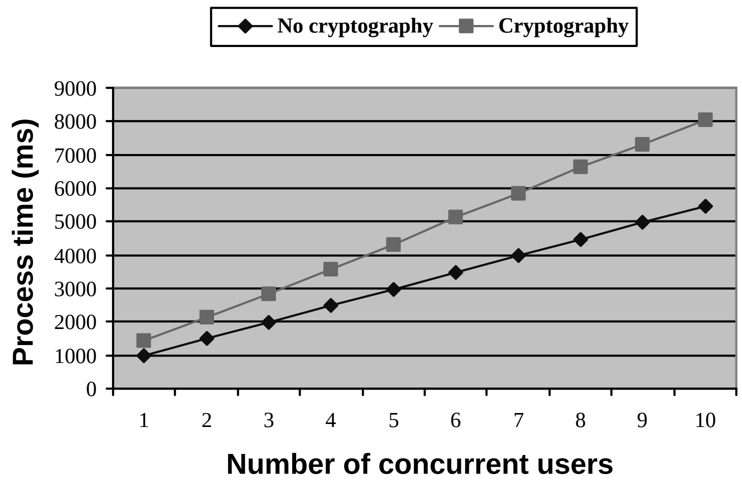


Fig. 5-5: Single-Threaded Model on the Payment Transaction Time on PG

5.3 Micropayment in Mondex

Introduction

Micropayment is the payment that only involves a small amount of transfer of money from customers to merchants. It provides an alternative revenue source for content and service providers. It is a more efficient and lower cost method than credit card in transaction, which the values of the service and products involved are low.

Mondex is one of the advanced electronic cash micropayment systems over the Internet. Its unique security architecture enables a range of functionality not offered by any other electronic cash scheme.

Mondex is preferable to be used for simple, everyday cash transactions. In TravelNet, the travelling accessories shop offers a good chance to adopt Mondex as one of the payment method for buying and selling goods. It will be a trend for supporting Mondex as one of the payment methods in online business as well.

Due to the potential cooperation of a commercial firm on Mondex products and CUHK, we have the chance to try out the device in near future. From the view of the user, it is a flexible design of payment that allows other method instead of the traditional credit card approach.

Payment Model

The figure below shows the flow of a Mondex payment using digital signature.

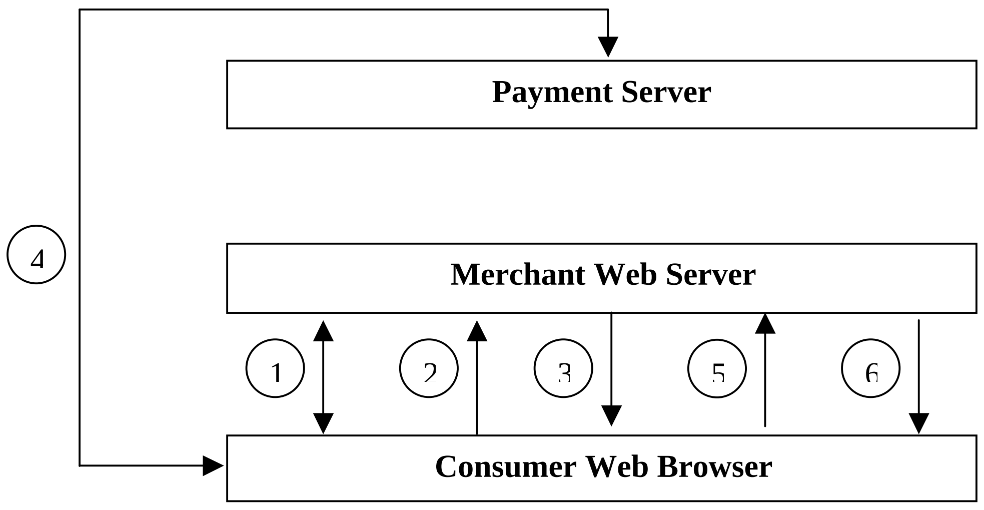


Figure 5-10 The Mondex Payment Flow Using Digital Signature

1. **Shopping.** A consumer reaches a merchant's web site, he or she either interacts with the merchant's shopping system and sel ects the desired products OR he or she wants to pay for the service charge, for example to pay for the electric bill.

   For the case of shopping, the consumer initiates the *check out* action. He or she might be then asked for more information such as the de livery address and delivery time depending on what kinds of products to be purchased. After that, a *payment confirm* web page summarize the products selected and the total payment amount is sent to the consumer.

   For the case of paying for service charge, the consumer needs to enter his or her consumer ID, then a *payment confirm* web page summarize the service charge details and the total payment amount will be sent to the consumer.

2. **Confirm the payment.** From the payment confirm web page, the consumer selects one of the available payment methods, which can be Visa, Master and *Mondex*. Finally the consumer presses the *Confirm Payment* button to confirm the payment.

3. **Send the web page embedding the plugin reference.** A web server program say called *PayByMondex* is invoked and it does the followings.
   (i) Check whether the state of payment is valid.
   (ii) Construct the payment request from database.
   (iii) Sign the payment request using the library provided (which will be described later)
   (iv) Construct a web page embedding the *Consumer Mondex Payment plugin*

reference and the corresponding plugin input agruments, and send it to the consumer. The plugin arguments contains the payment request and the merchant signature on the payment request.

4. **Connect to Payment Server and start the payme nt.** Upon the consumer receives the web page containing the plugin reference, the plugin is invoked. The plugin connects to the *payment server* via SSL. It authenicate the Payment Server and then submits the payment request to it. Payment Server first ve rifies the signature of the request, then queues it up; and eventually the Mondex payment between a merchant Mondex card and the consumer Mondex card begins. Finally, the result of payment will be signed by Payment Server and send to the consumer plugin.

5. **Submit the payment result.** The consumer plugin calls a another merchant web server program, say *Result*, to submit the signed payment result received from Payment Server.

6. **Delivery the post -payment web page.** The *Result* program first verifies the signature of Payment Server using the library provided. If it is correct, it does the post -payment processing and responses a web page to the consumer. For example it sends a web page containing the payment result and reference number to the consumer.

For secur ity, the communication between the consumer web browser and the merchant web server uses **https**.

Implementation in TravelNet

For the shopping system to use the Mondex payment service provided, it needs to do the following.

1. Modify the payment confirm web page to include Mondex as one of the payment methods. If the payment method is selected as Mondex and the confirm payment button is pressed, calls the *PayByMondex* web server.

2. Implement the *PayByMondex* web server program.

3. Implement the *Result* web server program.

The interface of signature approach similar to the approach used in phrase of iPS. They both call a web server program to generate a html page containing the plugin reference when the user confirms the payment. However, in signature approach, th e payment request is not sent from the merchant to Payment Server; instead the whole payment request is specified in the plugin input parameters, and is signed using the merchant's private key. The plugin will then handle whole payment process. Upon the payment is finished, whether it is success or not, it will call another web server program to submit the signed payment result issued by Payment Server.

There is two utility software comes with the devices. One is for the merchant side that consists of s ome development DLL library in C++. The library, which is currently on Windows Platform, is a useful tool for merchant to contact the payment server for verification and signing of the payment request. Since Java cannot call the DLL library directly, we have made a VB program as a wrapper to call the provided functions for the payment transaction and let Java to call the VB wrapper through its Runtime classes. It will be better if the library can be plugged into Java platform, but currently the VB version can work fine to demonstrate the system.

On the client side, the users have first equipped with a Mondex card reader, namely iReader, to access a Mondex card. The users also have to install the corresponding browser plugin for the linkage between the b rowser and the reader. When a user wants to start payment, he should first insert the card into the reader and then initiate the payment process. He will be informed the result of the transactions.

The general data flow of the process:

1) Client starts a payment transaction, request is sent to TravelNet.
2) The payment module calls the SignPaymentRequest() in the DLL library for the payment transaction through the VB wrapper
3) TravelNet generates the page which will initiate the browser plug -in in the client side
4) Internal checking of the card is performed and it will direct the user to the verification part of the payment
5) TravelNet receives a verification request, which will call the library again for verification of the payment by VB wrapper.
6) TravelNet will show the result of the transaction to client.

6. Distributed Components

6.1 Introduction

In this chapter, we will discuss the issue related to integrate CORBA in the existing system. We will first present a brief overview of CORBA. Then we will discuss the components that have integrated CORBA in the system. Finally, we will give a performance measurement between CORBA version and non-CORBA version.

6.2 Overview of CORBA

Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA was introduced in 1991 by Object Management Group (OMG) and defined the Interface Definition Language (IDL) and the Application Programming Interfaces (API) that enable client/server object interaction within a specific implementation of an Object Request Broker (ORB).

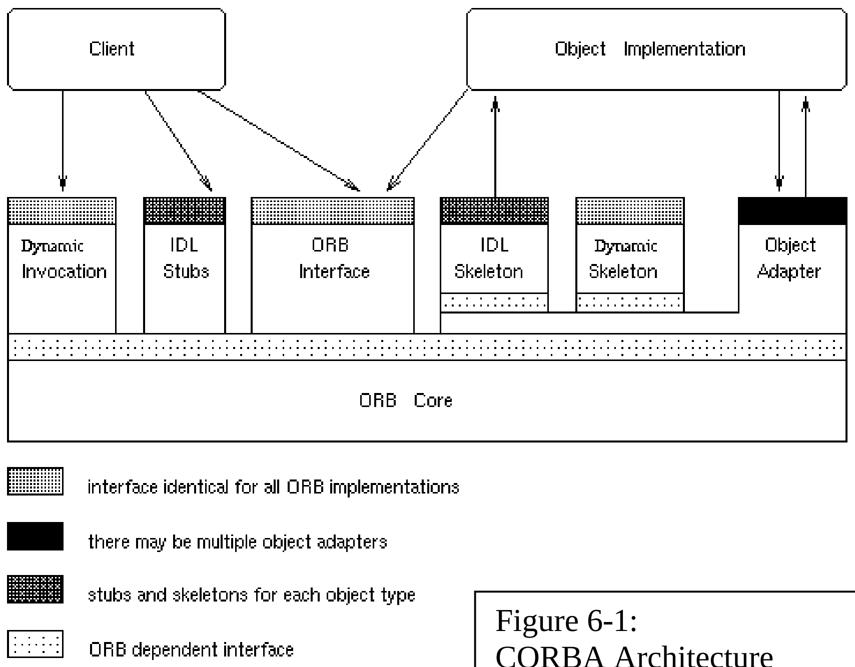


Figure 6-1:
CORBA Architecture

The (ORB) is the middle ware that establishes the client -server relationships between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. In so doing, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

6.3 CORBA in TravelNet

Although integrating CORBA in Java platform as there exist many application and applets that have used CORBA in distributed applications, it is relatively new to cooperate CORBA with Java Servlet, a main technology used in the whole TravelNet system. It creates an interesting point of combining these two technologies together.

Recall the architecture in chapter 3, the main distributed components in TravelNet is the airline manager, which is required for airline databases access. For this part, it will be definitely reasonable and favaouarble. In the centralized version of Airline Manager, it is assumed to be distributed by the airline companies which has a standard interface to allow TravelNet to call for its service provided. It creates a great problem when the airline wants to upgrade its internal design of the database, which needs to redistribute the new version of airline manager to all contracted travel agencies. With the help of CORBA, this problem can be eliminated.

Besides, CORBA also facilitates location transparency, which is a favaourable feature that TravelNet doesn't require to know the location of airline server.

Moreover, we have also made the online shopping system to work in a distributed manner. It favours the business option that TravelNet may act as a service agent to client instead of those merchants to sell their own products directly to client. If the business is of this form, it is more reasonable for the merchants to keep their own stock databases while TravelNet can consult these remote databases when necessary. Thus, we have developed the distributed version of online shopping system.

We have used the URL Naming Service pr ovided by Borland Visibroker 4.0 for object reference. It is a simple mechanism that allows a server object associate its Interoperable Object Reference(IOR) with a URL in the form a string represented in a file. Client programs can locate the object usi ng URL pointing to the file containing the stringified object on the web server. We will use this service in TravelNet.

Communication Interface: Airline Service

The following is the IDL defined for the interface between distributed component of Airline Service

```
// Exception that may exist in modules
internal_error:
```
raises when there is internal error in the Server object

// Query All() is for getting all flights information that matches the input parameter
```
string query_all(in string serv_type, in string src_place,
           in string dst_place, in string seat_class, in string dweekday,
           in long mindt, in long maxdt, in string rweekday, in long minrt,
           in long maxrt, in string dept_date, in string retr_date)
raises (internal_error);
```

Input :
serv_type : Service type (One-way / Round-Trip)
src_place: the take off place of flights
dst_place: the destination place of flights
seat_class: the seat class (First class/Business class/Economy class)
dweekday: The Week day of departure flights
mindt, *maxdt*: time range of the departure flight
rweekday: The Week day of return flights(optional)
minrt, *maxrt*: time range of the return flight(optional)
dept_date: the departure date of flights
retr_date: the arrival date of flights(optional)

Output:
a list of flight information of all matched flights

// Query one() is for getting a flight information that matches the input parameter
```
string query_one(in string flight_num, in string serv_type,
                 in string seat_class)
raises (internal_error);
```

Input :

flight_num: flight number of targeted flight
serv_type : Service type (One-way / Round-Trip)
seat_class: the seat class (First class/Business class/Economy class)

Output:

flight information of matched flight

```
// checking if the flight exist in the database
boolean is_flight_exist(in string flight_num, in string weekday,
                        in string seat_class)
raises (internal_error);
```

Input :

flight_num: flight number of targeted flight
weekday : The Week day of targeted flight
seat_class: the seat class (First class/Business class/Economy class)

Output:

boolean value

```
// checking if the seats are available for matched flight in the database
boolean is_seat_avail(in string flight_num, in string dept_date,
                     in string seat_class)
raises (internal_error);
```

Input :

flight_num: flight number of targeted flight
weekday : The Week day of targeted flight
seat_class: the seat class (First class/Business class/Economy class)

Output:

boolean value

// book(): to book a flight of input information

```
string book(in string serv_type, in string holder_name,
    in string dept_fnum, in string dept_date, in string dept_seat_class,
    in string retr_fnum, in string retr_date, in string retr_seat_class)
raises (internal_error);
```

Input :

serv_type : Service type (One-way / Round-Trip)
holder_name: The name of the ticket holder
src_place: the take off place of flight
dst_place: the destination place of flight
dept_date: the departure date of flight
dept_flight_num: flight number of departure flight
dept_seat_class: the seat class in departure flight
retr_flight_num: flight number of return flight(optional)
retr_date: the depature date of return flight(optional)
retr_seat_class: the seat class in return flight (optional)

Output:

ticket number(s) of reserved tickets

Mechanism: Airline Service

Airline Manager is still act as a repres entative of Airline. When there is request of service to the Airline Manager, it will create and bind a CORBA object of the airline server, which is, resides at the UNIX environment. They can communicate through the interface defined above.

Communication Interface: Online Shopping System

The following is the IDL defined for the interface between distributed component of shopping system

// Exception that may exist in the module

`out_of_stock:` Stock is not available
`internal_error:` Internal error of the server

// interface Stock

```
float check_price(in string pid, in long quanity)
raises (out_of_stock, internal_error);
```

Input:
pid : product id
quantity: quantity of selected product

Output:
total amount of selected product of given quantity

```
boolean order(in string pid, in long quanity)
raises (internal_error);
```

Input:
pid : product id
quantity: quantity of selected product

Output:
result of order product request

```
// restore the stock database
boolean reset() raises (internal_error)
```

// Interface StockMgr

```
Stock open(in string name);
```

Input:
name : Name of the stock database

Output:
The stock database object of given name

Mechanism: Online Shopping System

The shop basket system will create and bind a CORBA object of the Stock Manager which is resides at the UNIX environment. Through the Stock Manager interface, it can retrieve the required database, which is also represented as a CORBA object. Then through the Stock interface, it can check price and order the stock resides in the associated database.

6.4 Performance Measurement

A simple performance measurement has been carried out to evaluate the performance of the distributed CORBA version versus the centralized version. The measurement is based on two experiments: 1) one -way flight search and 2) round -trip flight reservation. In each experiment, there is three run using the same set of data in measuring of the time.

The result of the experiments are listed below:

Experiment 1: One-way flight search between Hong Kong and Taipei

Run	Distributed version in CORBA Time(ms)	non-distributed version Time(ms)
1	19010	13139
2	15883	11146
3	16364	11878
Average	17086	12054

Experiment 2: Round-trip flight reservation between Hong Kong and Beijing

Run	Distributed version in Corba Time(ms)	non-distributed version Time(ms)
1	5668	5819
2	5828	4877
3	5734	5051
Average	5743	5249

The difference between experiment 1 and experiment 2 is that experiment 2 only involves communication with one CORBA airline object, while experiment 1 requires to communicate to mo re than ten CORBA airline objects. From the experiment 1, we observe that calling and binding CORBA objects produces around 0.5 s overhead. And from experiment 2, we observe that the overhead is accumulative. However, it may be possible to reduce the ove rhead by allowing parallel creation and access to different CORBA objects which is not implemented in the current system.

Although, using of CORBA creates certain overheads in operation, it is still beneficial to design through it availability in location transparency, access transparency, migration transparency and scaling transparency.

7. Simplification of Components

7.1 Introduction

In this chapter, we will discuss the issue on simplification of components be means of Java Server Pages(JSP). We will present an overview of the system and how to cooperate with the existing system.

7.2 Overview of JSP

JSP technology allows Web developers and designers to rapidly develop and easily maintain, information -rich, dynamic Web pages that leverage existing bu siness systems. As part of the Java family, the JSP technology enables rapid development of web -based applications that are platform -independent. JavaServer Pages technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content.

JSP uses XML -like tags and scriptlets written in the Java programming language to encapsulate the logic that generates the content for the page. Additionally, the application logi c can reside in server -based resources (such as JavaBean component architecture) that the page accesses with these tags and scriptlets. Any and all formatting (HTML or XML) tags are passed directly back to the response page. By separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier to build web-based applications.

JSP are an extension of the Java Servlet. Together, JSP and Servlets provide an attractive alternative to other types of dynamic web scripting/programming that offers platform independence, enhanced performance, separation of logic from display, ease of administration, extensibility into the enterprise and most importantly, ease of use.

7.3 JSP in TravelNet

We have used JSP in 3 parts of TravelNet They are:

a) *Login Page*: With the help of JSP, all programming logic of user login, login error and direct logined users to correct pages can be made into one single JSP file with much simplification

b) *Shopping Bas ket*: Originally a shop basket is consists of a combination of 3 servlets – AddBasket, ViewBasket and UpdateBasket. By using JSP and the corresponding Bean technology, it can be made all these into one JSP file which is easier to maintain any code and design changes.

c) *Hotel Information* : Instead of managing a large number of static pages, JSP is helpful in organizing these informations and select the corresponding page on demand.

8. Conclusion

In the project, we attempt to build an online travel agency, whi ch provides travelling related service to possible users. We start our work from information collection, then the initial system design and the completion of the basic system, which is a centralized one. Then we keep on improving and enhancing the curren t system by developing some distributed components using CORBA, more sophisticated payment methods using credit cards and mondex, and the simplification of redundant components.

In this report, we have presented our own design of the whole system, startin g from the original one the enhanced version. We have introduced the features of the TravelNet and its internal design. We have explained how security can be achieved in credit card payment and give a performance measurement. We have discussed the other payment method - Mondex and how it can be used in TravelNet. On the other hand, we have explored the ways that how CORBA can interact with Servlet to form a distributed system with a simple performance measurement to it. Finally, we have described the us e of JSP, which can simplify the system on top of our Servlet implementation.

Building of a large -scale online application is not an easy task. We have gained invaluable experience on this by working on our project – TravelNet. We have researched differ ent approaches on developing online application and particularly experienced on using Java(Servlet and JSP) and CORBA. Also, we have the chance on implementing different payment methods using TravelNet as a sample application. Moreover, we have realized that no matter how the system was built, the following four elements: Security, Performance, User -Interface and Ease of modular design for maintenance, are all essential for a successful online e -commerce business to be developed.

9. References

[1] B.Eckel. *Thinking in Java*, Prentice Hall Inc. 1998.

[2] “JDK^{TM} 1.1.8 Documentation”.
http://java.sun.com/products/jdk/1.1/docs/index.html

[3] “The Java Tutorial”.
http://java.sun.com/docs/books/tutorial/

[4] Victor Wolters. *Introducing Internet Information Server*, Que. Oct 14, 1996

[5] “Security in Internet Transaction”.
http://www.holt.ie/text/security.html

[6] “Web Application Development”.
http://www.winwinsoft.com/articles/wad.html

[7] C. Darby , “Developing 3 -Tier Database Apps w/ Java Servlets” , *Java Developers Journal*, Feb 1998

[8] IBM C orporation. “The Web Application Programming Model” . *IBM Application Framework for e-business*. IBM Corporation.

[9] Z.Yang, K.Duddy. “CORBA: A Platform for Distributed Object Computing ”. *Operating Systems Review, 30(2):4-31*. ACM SIGOPS, Apr. 1996.

[10] Robert Orfa li & Dan Harkey. Client/Server Programming with JAVA and CORBA, 2nd Edition, *John Wiley & Sons, Inc*., 1998

[11] K.L.Chong, C.H.Ho, C.H.Lau, Micheal R.Lyu, Y.S.Moon, “The Design, Implementation and Evaluation of an Internet Payment System”, paper to appear in World Computer Congress 2000, ITBM2000., Aug. 21-25, 2000.

[12] “JSP^{TM} 1.0 Documentation”.
http://java.sun.com/products/jsp/index.html

[13] “Mondex Official Homepage”
http://www.mondex.com

10. Acknowledgement

We would like to thank the following people for their ki nd assiatance and effort. in completing the project

Professor Michael R. Lyu (Our project supervisor)
Professor M.C. Lee (Our project marker)
Mr. Steve K.L. Chong (for secured payment method)
Mr. Edmund Chiu (for Mondex payment method)
Mr. Malcolm Ho (My partner)
and CSE System Administrators

Appendix

A. Server Software

Java API 1.1.8.

Java is an object -oriented language, which is poplar all around the world today. Because of its portability, it grows along with the Internet related technologies. Its complete and robust API brings programmer and software developer a convenient developing environment. Since it is slower than native programming language, Java is not suitable for low level programming or real time processing. On the other hand, it is perfect for net working application programming. One of the most critical factors determining the performance of network application is the connection speed. So it compromise slow execution speed of Java.

Java Servlet API 2.1

Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component -based, platform -independent method for building web -based applications, without the performance limitations of CGI programs. And unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), Servlets are server- and platform-independent.

Written in Java, Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets also acces s library of HTTP-specific calls, and all the benefits of the mature Java language, including portability, performance, reusability, and crash protection.

Windows NT Server 4.0 with IIS 4.0

Windows NT Server is a quite common commercial product Microsoft Windows NT Server 4.0 is a multipurpose operating system specialized on Server operations . A multipurpose operating system does more for less because it integrates a variety of network services that you need to run your business. The services it provides are designed to address customer requirements in every category.

The Internet Information Server is a popular web server providing Internet services like web, mail and news. Its functionality can be extended by install suitable ISAPI.

Servlet Exec 2.2

ServletExec is a Servlet engine. It is a high -performance, reliable, inexpensive web application server and Servlet engine that implements the Java Servlet API and JavaServer Pages (JSP) standards, components of the Java 2 Platform, Enterprise Edition (J2EE) suite of standards defined by Sun Microsystems. ServletExec runs on all major web servers and operating systems.

Oracle8i

Oracle8i is the database server we used in the project. It is installed in our department and it includes a set of Java JDBC drivers for database access.

Borland Visibroker for Java

Visibroker is a tool to developed CORBA based application particularly in Java platform. It includes a full set of ORB classes for the implementation of distributed programming. It runs on Unix and Wi ndows.

Mondex Merchant Utility

It includes a Windows Dynamic Library DLL for payment process on the server side. It is written in C++.

B. Server Hardware

Host machine:

Pentium II 300MHz, 96 MB memory

A mid-end machine is needed for a web server to handle requests concurrently especially our system request handler is Java Servlet. A Pentium 2 300MHz is just meet our demand. It is a server with a static Internet address. The Internet name is ntsvr4.cse.cuhk.edu.hk.

Distributed Component and Payment Gateway:

Unix Sparc Station

All distributed components are located on the Unix Sparc Station.

C. Client-side Requirement

Netscape 3.0+ or Internet Explorer 4.0+

Travel Net client only needs a simple web browser. It is recommended that client browser is SSL en able because the client will submit critical information through the Internet. This unprotected transmission is very insecure. If information is being hacked, hacker may use this information for illegal shopping.

Mondex card reader and plugin software(optional)

TravelNet supports payment by Mondex as well. In order to use this method, client should be equipped with Mondex card reader with the plug-in software installed.

D. Program Listing

Module	Operation	Number of Lines	Number of Characters
User Management	Login.jsp	90	3518
	LoginBean	110	2428
	UserSessionBean	53	1003
	Register	238	8981
	ViewUserInfo	178	7036
	UpdateInfo	153	5582
	Logout	20	464
Online Shopping System and Stock Management	Shop.jsp	120	3427
	ShopBasketBean	44	1383
	ViewBasket.jsp	152	5730
	Checkout	250	9327
	mondex.jsp	90	3580
	Mondex	78	1930
	Result	265	9248
	mondex.bas	72	2299
	Stock.idl	17	391
	StockMgrImpl	20	547
	StockServer	25	747
	StockImpl	107	2931
	StockBean	62	1841
Hotel Information	hotelresv.jsp	175	7881
	Hotel.jsp	60	1748
Airline Service	AirlineManager	498	13716
	SearchFlight	510	21009
	RserveFlight	353	13596
	AirlineServer	53	1843
	AirlineServiceImpl	484	14124
	AM.idl	27	1144
Itinerary Management	ItineraryManager	482	13211
	AddItinerary	84	2539
	ViewItinerary	293	14074
	RemoveItinerary	43	1236
Supplemetary	Database	45	1373
	Mail	39	1471
	Html	20	523
	Total:	5310	181881