TESTING UML DESIGN MODEL FOR WEB-BASED RESERVATION SYSTEM

MAZNI OMAR

NORLIZA KATUK

FACULTY OF INFORMATION TECHNOLOGY

UNIVERSITI UTARA MALAYSIA

2005

DISCLAIMER

We are responsible for the accuracy of all opinion, technical comment, factual report, data, figures, illustrations and photographs in this report. We bear full responsibility for the checking whether material submitted is subject to copyright or ownership right. UUM does not accept any liability for the accuracy of such comment, report and other technical and factual information and the copyright or ownership rights claims.

PROJECT LEADER:

……
Name: Mazni Omar

MEMBER:

…….
Name: Norliza Katuk

ACKNOWLEDGEMENT

We wish to express our gratitude to the Faculty of Information Technology for the financial support under the faculty grant for this research project. We would also like to thank all our colleagues and individual for their support and assistance.

ABSTRACT

Adequate testing is essential to guarantee the quality of a software system and satisfy the user requirements. In today's environment, web-based applications become critical to a company's success. Therefore to ensure the web-based application works correctly, the application's part must be thoroughly tested with appropriate testing technique. A systematic procedure in developing set of test cases is needed to ensure the quality of the application.

Thus, this study is conducted to construct test cases using Design Class Diagram (DCD) criteria and study the suitability of this criterion in testing the web-based application class diagram. The DCD criteria contain three (3) criterions which are, `Association End Multiplicity (AEM)`, `Generalization (GN)` and `Class Attributes (CA)`. UML for Web Application Extension (WAE) was adapted in designing the web-based application class diagram.

In order to test a web-based application, a web-based Hotel Reservation System (W-HReS) was developed using Active Server Page (ASP) technology. The application consists of five use cases, which are `Make Registration`, `Make Reservation`, `Search for Room Availability`, `Login` and `Cancel Reservation`. A set of test cases derived for DCD criteria has been constructed to test the application. The findings suggest that DCD criteria, in total, are not appropriate for testing web-based application. This is due to the fact that AEM cannot be used to represent all relationships in WAE class diagram. In addition, since there is no generalization association in WAE class diagram, GN cannot be used to test the diagram. However CA seems to be appropriate to test the diagram since all criteria are apparent and well-defined in the diagram.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter contains an overview the study, problem statement, objective, scope and significance of the study.

1.1 An Overview of the Study

The Unified Modeling Language (UML) from Object Management Group (OMG) has attracted great attention recently. UML has become the current standard for modeling software-intensive system (Nilawar, 2003). Besides, UML has also received a great deal of attention from the software design and development communities, and work is on going to enhance and expand its capabilities (Williams, 1999).

Adequate testing is essential to guarantee the quality of a software system and to ensure that the software satisfy the user requirement. Driven by the extreme demands of business world and enthusiasm of the public, more and more businesses being conducted through the web. Thus, testing web application becomes more challenging than conventional software (Jia & Liu, 2002). Therefore to ensure the web application works correctly, the web functionality must be thoroughly tested using an appropriate testing technique.

According to Andrews et al. (2003), testing executable forms of model is analogous to program testing and involves creation of test cases, the execution of the artifact using the test cases and the analysis of test results to determine correctness of the tested behaviour. Test criteria are important to define the testing objectives while performing software testing. Therefore, the UML test criteria proposed by Andrews et al. (2003) are used in testing UML design models.

1.2 Problem Statement

UML has been widely used as a modeling tool in software development. Software developed with UML has to be tested to assure its quality and to prevent faults (Offutt & Abdurazik, 1999). Current practice in UML design evaluation consists of walkthrough and inspections. However, these techniques are too complex and tedious because the reviewer needs to track large amount of information (Trong, 2003). Furthermore, the lack of assessment of design quality, and deficiency in detecting and correcting design fault in the model can increase the total software development costs and time to market (Ghosh et al., 2003). This indicates that the need of testing and validating design model for web-based applications is high. Therefore, this study intends to justify whether the DCD criteria can be used to derive suitable test cases for web-based application UML design model.

1.3 Objective

The main objectives of this study are:

i. To construct test cases from UML design model of web-based application using DCD criteria.

ii. To determine the suitability of DCD criteria in testing web-based application design model.

1.4 Scope of the Study

The scopes of the study are:

i. Testing only focuses on UML Class Diagram design model.

ii. The UML design model is tested based on UML Design Class Diagram criteria (DCD).

iii. Hotel Reservation System as domain environment.

1.5 Significance of the Study

The significances of this study are:

i. To provide justification as to whether DCD criteria can be used to derive test cases for web-based application based on UML design model.

ii. The results obtained from this study can contribute towards software testing process.

1.6 Summary

This chapter provides an overview of the study including problem statement, objective, scope and significance of the study. Related studies on web-based application and UML-based testing will be explored in the following chapter.

CHAPTER 2

LITERATURE REVIEW

This chapter focuses on reviews on web-based application, software testing, UML, UML testing criteria and related works for this study.

2.1 Introduction

Testing of web applications is a specialized area of software testing and it is quite a new area. Due to the unique characteristic of web applications, conventional software testing tools are not adequate in dealing with web applications. It is critical to develop effective methodologies and tools for testing web applications (Jia and Liu, 2002). This section provides information about testing UML models for web-based applications.

2.2 Web-based Application

Shklar and Rosen (2003) define web application as a client/server application that uses a web browser as its client program, and perform an interactive service by connecting with servers over the internet (or Intranet). A web site simply delivers content from static files while, a web application presents dynamically tailored content based on request parameters, tracked user behaviours, and security considerations.

Web applications are also defined as software programs or applications that receive input and deliver output through the web, usually in the form of HTML or XML. Web applications are dynamic, interactive, often serve as the front-end of complicated applications that often involve database at the back-end (Jia and Liu, 2002).

There is also another definition by Wu and Offet (2002). They define web applications as programs that share some characteristic of client server, distribute and traditional program. However, there are a number of novel aspects of web applications. These include the fact that web applications are dynamic, due to factors such as the frequent changes of the application requirement as well as dramatic changes of the web technologies. According to Neise et al. (2002), a typical web-based architecture contains a web browser which is located on a client and interacts with a web server. The web server will communicate with an application server that builds the requested web pages dynamically through an interaction with a database and (several) back-end services.

2.3 Software Testing

Software testing is crucial in software development process. Testing aims to finds error in software. It involves executing a program with a set of test cases and comparing the actual results with the expected output (Offutt & Abdurazik, 1999). A test case is a sequence of inputs (test data) that determines the behaviour of the tested system. The test data can either be manually devised or automatically generated to test the system. A test is successful if the observed behaviour matches the expected behaviour; otherwise the test fails (Andrews et al., 2003).

In general, a system is tested to reveal the system defect before it is being delivered and thus termed defect testing (Sommerville, 2001). Figure 2.1 depicts a general model of a defect testing process.

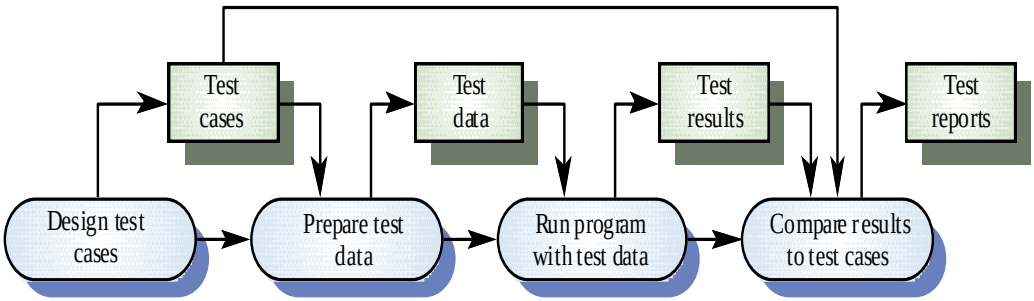


Figure 2.1: The Defect Testing Process (Sommerville, 2001)

According to Sommerville (2001), there are two categories of dynamic testing, which are black-box testing and white-box testing. Dynamic testing will execute the software by executing the software with the generated test data (Roper, 1994). Black-box testing focuses on the functional requirement of the software (Pressman, 2001). This indicates that black-box testing enables the software engineer to derive sets of input conditions which fully exercise the functional requirements of a program. On the other hand, white-box testing concerns with the internal structure or the program's code. The white-box testing is conducted based on knowledge of the internal logic of an application's code (Futrell, 2001).

According to Nguyen (2001), the software testing techniques that are applied to other application are same that are applied to web-based applications. Therefore, the black-box testing technique is adopted to test the web-based functionality. The next section will discuss about concepts of UML models and UML-based testing.

2.4 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a visual modeling language that can be used to specify, visualize, construct and document the artifacts of software system (Booch et al., 1998). According to Eriksson & Penker (1999), UML can be applied in different phases of system development, from the requirement specification to the test of finished system. In addition, UML can also be used for business modeling, software modeling in all phases of development and for all types of system, and general modeling of any construction that has both a static structure and dynamic behaviour. Furthermore, UML has became the formal and de facto standard for creating software models.

There are many types of diagrams in UML. Each diagram in UML has a specific purpose. The following defines the types of UML models for the specific domains (Sparxsystems):

- The User Interaction or Use Case Model - describes the boundary and interaction between the system and users.
- The Interaction or Communication Model - describes how objects in the system will interact with each other to get the work done.
- The State or Dynamic Model - State charts describe the states or conditions that classes assume over time. Activity graphs describe the workflows the system will implement.
- The Logical or Class Model - describes the classes and objects that will make up the system
- The Physical Component Model - describes the software (and sometimes hardware) components that make up the system.

- The Physical Deployment Model - describes the physical architecture and the deployment of components on that hardware architecture.

As mention in Williams (1999), there are many phases in the testing process, including unit, function, system, regression and solution testing. Table 2.1 shows the differences among these phases, as well as the potential UML diagram used in each phase in terms of coverage criteria and fault model.

Table 2.1: The Differences Among Testing Phases, Coverage Criteria, Fault Model and Potential UML Diagram to Be Used

Test Phase	Coverage Criteria	Fault Model	UML Diagram
Unit	Code	correctness, error handling, pre or post condition, invariants	class and state diagram
Function	Functional	functional and API behaviour, integration issues	interaction and class diagram
System	Operational behaviour	workload, contention, synchronous, recovery	use case, activity and interaction diagram
Regression	Functional	unexpected behaviour from new or changed function	SAME AS FUNCTION
Solution	Inter-system communication	interoperability problems	use case and deployment diagram

2.5 UML Testing Criteria

Andrews et al. (2003) define a family of test adequacy criteria for Class Diagrams and Collaboration Diagrams. Test adequacy criteria are set of properties that must be covered during the test. A test is considered adequate if all test adequacy criteria are covered. These criteria are used to guide the selection of test cases and measurement of test adequacy.

Design Class Diagram (DCD) criteria include the following:

- *Association-end Multiplicity (AEM)*: AEM is designed to ensure that configurations containing boundary and non-boundary occurrences of links between objects are tested. AEM specifies how many instances of a class at the opposite end of association link can be associated with a single instance of a class at the association end.

  In AEM, for given a test set *T*, and a system model *SM*, *T* will represent multiplicity-pair for *SM*. For a relationship between a *Lecturer* and *Department* classes in Class Diagram, an example of representative in association multiplicity can be {(0,0), (0,1)}. The first number in each pair denotes how many *Lecturer i*nstances are associated with a *Department* instance. The second number denotes how many *Department* instances are associated with one *Lecturer* instances.

- *Generalization (GN)*: GN is designed to ensure that each specialized class is instantiated and used during the tests. The GN criterion defines the

representatives set of specialization types that must be created from DCD's super classes during the system model test.

For a test set *T* and a system model *SM*, every specialization defined in generalization relationship will be represented in *T*.

- *Class Attribute (CA)*: CA is designed to ensure the testing of behaviours using combinations of representative class attribute values. For a test set *T*, a system model *SM*, and a class *C*, *T* will represent a set of attributes values in each instance of class *C*.

  In order to establish the set of representative values, a form of category-partitioning technique is adapted. Using this technique, the value domain is partitioned into equivalence classes, which is invalid and valid class.

2.6 Related Works on UML-based Testing

Briand & Labiche (2002) describe the TOTEM (Testing Object OrienTed systems with the Unified Modeling Language) functional system test methodology. In TOTEM, system test requirements are derived from the UML artifacts such as use cases, their corresponding sequence and collaboration diagrams, class diagrams and the use of Object Constraint Language (OCL) expressions across all the artifacts. The goal is to transform the test requirements into code-level test cases, test oracles and test drivers using more detailed information.

Offutt and Abdurazik (1999) developed a technique for generating test cases for code from UML state diagrams. They also developed test criteria based on collaboration diagram for static and dynamic testing. The goals of both approaches are to test design models and use information from different types of UML diagrams (class and collaboration diagrams) during testing.

Binder (1999) describes generic test requirements derived from UML models and introduces test design patterns. These patterns focus on determining appropriate test strategies, faults that may be detected, test case development from model and the test oracles.

In this study, we focus on testing the UML design model by applying the DCD test criteria defined by Andrews et al. (2003) to generate test cases and test the models.

2.7 Summary

This chapter has briefly highlighted the concept of web-based application, UML and reviewed the detail aspects of software testing and UML testing criteria. In addition, the related works of UML-based testing have also been discussed. The following chapter will explain the methodology used in this study.

CHAPTER 3

METHODOLOGY

This chapter introduces the methodology used throughout the study. This study is conducted in four main phases:

i. Requirements Analysis
ii. Design the Unified Modeling Language (UML) Design Model
iii. Prototype Development
iv. Testing UML Design Model

3.1 Requirements Analysis

According to Braude (2000), requirement analysis is a process of understanding and documenting what an application meant to do. The objective of the requirements analysis is to identify what the user requires from the software elements of the system (Bennett et al., 2002). These requirements can be identified using fact finding techniques such as background reading, interviewing, observation and questionnaires.

During this Requirements Analysis phase, requirements on web-based Hotel Reservation System (W-HReS) for this study are modeled using UML. Further information for this phase will be provided in Chapter 4: Requirements Analysis.

3.2 Design the UML Design Model

The aim of this phase is to develop Class Diagram for web-based reservation system from the requirements identified in previous phase. For this study, modeling Class Diagram for web-based application proposed by Conallen (1999) will be adopted to model the web design architecture. Further information for this will be provided in Chapter 5: Design Modeling.

3.3 Prototype Development

A prototype design is required to check the user requirements. During this phase, the generic requirements defined in the previous phase will be transformed into codes. In this study, a Web-based Hotel Reservation System or W-HReS is developed as a prototype and used to test its design model. The details about prototype development will be discussed in Chapter 6: Prototype Development.

3.4 Testing UML Design Model

Testing phase is the most crucial part in validating the design process. Prototype testing is the art of executing software on individuals input values to learn about its behaviour. Testing UML design model has been performed in order to measure and validate the prototype designed. Test case design using DCD criteria as described in Section 2.5 has been used. User will input real data and the developer will confirm whether system is reliable or not. The test focuses only on DCD criteria to test and validate the UML Class Diagram for web-based reservation system. Testing analysis using DCD criteria will be discussed in details in Chapter 7: Testing Analysis.

3.5 Summary

This chapter explained the methodology used in the study. This study is carried out in four main phases; requirement analysis, design UML model, prototype development and testing UML design model. UML model has been adopted to analyze, design and model the prototype requirements. DCD criteria are used to design the test cases and validate the UML Class diagram for web-based reservation system. The next chapter will discuss the requirements analysis phase in this study.

CHAPTER 4

REQUIREMENTS ANALYSIS

This chapter intends to detail out the requirements analysis phase for web-based Hotel Reservation System (W-HReS). There are two main activities involved in order to understand the requirements and required properties of the study, which are:

- Gathering information
- Modeling requirements

4.1 Gathering Information

Gathering information is an initial activity that will define the requirements of the study. The information, issues and problems related to the study domain, which the web-based reservation system, is gathered. This has been conducted through:

- Literature reviews of existing journal, proceedings and web site that cover about testing Unified Modeling Language (UML) model and reservation application domain.
- Observation of reservation normal business activities.
- Review of existing web-based reservation system. Using the selected commercial reservation web-site for example **www.marimari.com** to gain hotel reservation process understanding and experience.

4.2 Requirements Modeling

Object-oriented Analysis and Design (OOAD) technique with Unified Modeling Language (UML) has been adopted to model the requirements. UML which is created by Booch et al. (1998) is a language for specifying, visualizing, constructing and documenting the deliverables of software product. It enables stakeholders that are involved in software development to document and to describe the software in a standard way. In UML modeling, use case is the basic diagram to model the requirements. Use case is a description of set of sequences of actions that a system performs that yields an observable result of value to a particular actor (Booch et al., 1998).

4.2.1 Use Case Diagram

A use case diagram represents a set of use cases and actors, and the relationships among them. The W-HReS consists of five use cases which are:

- Make Registration
- Search for Room Availability
- Make Reservation
- Login and
- Cancel Reservation.

The use case diagram for W-HReS is depicts in Figure 4.1. The description of each use case is provided in Section 4.2.2.

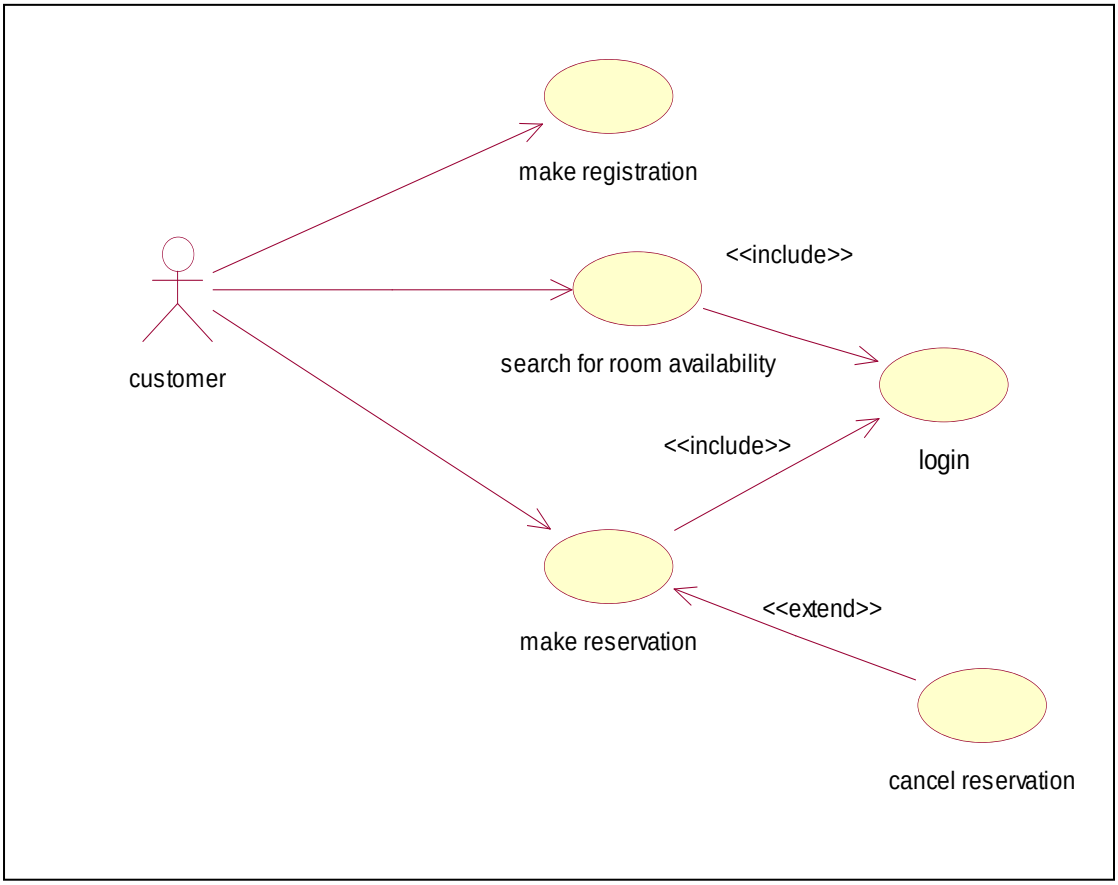


Figure 4.1: Use Case Diagram for W-HReS

There is only one actor, which is the customer, interacts with the system. The customer has to register before they can login into the system. Once the customer logs in, the customer can search for available room and make room reservation. Apart from this, the system also allows the customer to cancel his/her reservation.

4.2.2 Use Case Description

Use case descriptions provide detail descriptions of the interaction between the actors and the system. The following are the use case descriptions for each use case identified for W-HReS.

a. Use case "Make Registration"

The use case description for use case **`Make Registration`** is illustrated in Table 4.1.

Table 4.1: Use Case Description: `Make Registration`

Goal in context	The users have to register themselves in order to search for room availability or to make a room reservation.		
Pre-conditions	N/A		
Success-end conditions	The user can successfully register with the system and received ID and password.		
Failed-end conditions	The system will display a message if: 1. The length of chosen ID is less than 5 characters 2. The length of chosen password is less than 5 characters 3. The format of IC Number is invalid 4. The format of Phone Number is invalid 5. The format of e-mail is invalid 6. The format of Expiry Date is invalid 7. The format of Credit card Number is invalid 8. One or more fields are empty 9. The chosen ID has been used by other users 10. The IC Number has been recorded in database		
Actor	Customer		
Description	**Step**	**Actor Action**	**System Response**
	1	The user click on "`Search Room (for Non-Member)`"	'`Registration Form`' will be displayed.
	2	The user has to enter the information about: `ID, password, name, IC number, phone number, address,`	

		e-mail, credit card number, credit card expiry date and type of credit card.	
	3	The user has to click the 'Submit' button.	The system will record the user's information in the database. User has to remember the ID and password for future login.

b. Use case "Search for Room Availability"

The use case description for use case **Search for Room Availability** is illustrated in Table 4.2.

Table 4.2: Use Case Description: Search for Room Availability

Goal in context	The user need to find available room for reservation
Pre-conditions	The user's information must be available in the database.
Success-end conditions	The user can successfully search available room at certain date.
Failed-end conditions	The system will display a message if: 1. Check in date is less than current date 2. Check out date is less than current date 3. Check out date is less than check in date
Actor	Customer

Description	Step	Actor Action	System Response
	1	The user click 'Calendar' image beside the Check In Date text box.	'Calendar Window' will be displayed.
	2	The user selects the check in date.	The selected date will be displayed in the check in date text box
	3	The user click 'Calendar' image beside the Check Out Date text box.	'Calendar Window' will be displayed.
	4	The user selects the check out date.	The selected date will be displayed in the check out date text box
	5	The user selects the number of guest drop-down list. The number of guest is limited to 6 persons.	The default value of the number of guest is 1 person.
	6	The user clicks the 'Search Room' button.	The system will find available room on the required date.

c. Use case "Make Reservation"

The use case description for use case **`Make Reservation`** is illustrated in Table 4.3.

Table 4.3: Use Case Description: `Make Reservation`

Goal in context	The user needs to reserve room on requested date.		
Pre-conditions	1. The user's information must be available in the database. 2. The user must check for room availability on the requested date.		
Success-end conditions	The user can successfully reserve room on the requested date.		
Failed-end conditions	The system will display a message if: 1. No available rooms at the requested date.		
Actor	Customer		
Description	**Step**	**Actor Action**	**System Response**
	1	The user marks the check box of required `room type`.	
	2	The user clicks the 'Reserve Room' button.	The system will generate a random `booking code` and record the reservation to the database.

d. Use case "Login"

The use case description for use case **`Login`** is illustrated in Table 4.4.

Table 4.4: Use Case Description: `Login`

Goal in context	The user needs to log into the system. Users can make reservation or cancel available reservation.		
Pre-conditions	1. The user's information must be available in the database		
Success-end conditions	The user can successfully log in to the system.		
Failed-end conditions	The system will display a message if: 1. User's ID was incorrect. 2. User's password was incorrect. 3. The ID length is less than 5 characters. 4. The password length is less than 5 characters. 5. One or more input box is empty.		
Actor	Customer		
Description	**Step**	**Actor Action**	**System Response**
	1	The user has to type the `ID` and `password`.	
	2	The user clicks the '`Log In`' button.	The system will validate the `ID` and `password`. User fail to login if he/she gave the wrong `ID` or `password`.

e. Use case "Cancel Reservation"

The use case description for use case **`Cancel Reservation`** is illustrated in Table 4.5.

Table 4.5: Use Case Description: `Cancel Reservation`

Goal in context	The user needs to cancel room reservation on requested date.		
Pre-conditions	1. The user's information must be available in the database 2. The user has made a room reservation.		
Success-end conditions	The user can successfully cancel the room reservation on the requested date.		
Failed-end conditions	The system will display a message if: 1. The user typed incorrect booking code. 2. The booking code is less than 6 characters.		
Actor	Customer		
Description	**Step**	**Actor Action**	**System Response**
	1	The user has to marks the `room reservation` check box.	
	2	The user has to type the `booking code` for the selected room reservation.	
	3	The user clicks the '`Cancel Reservation`' button.	The system will validate the `booking code` and delete the reservation record.

4.3 Summary

The chapter discussed the requirements analysis phase in this study. There are two main activities in this phase which are information gathering and requirements modeling. UML is adopted to model the requirements. Use case diagram and use cases description for W-HReS has been provided. W-HReS has one actor or user namely, the customer and five use cases that are: Make Registration, Search for Room Availability, Make Reservation, Login and Cancel Reservation. The following chapter will discuss the design modeling phase for this study.

CHAPTER 5

DESIGN MODELING

This chapter discusses the UML design model developed for web-based Hotel Reservation System. The Web Application Extension (WAE) for UML proposed by Conallen (1999) has been adopted to model the class diagram.

5.1 Web Application Extension (WAE)

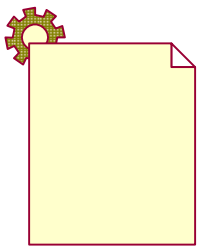
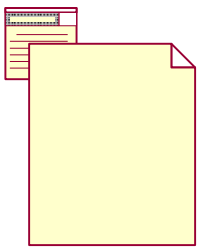
UML is the standard modeling language for modeling software intensive systems. However, Conallen (1999) claims that the standard UML is not a perfect fit to model all types of applications. Web applications represent one of these applications. Therefore Conallen defines a formal way to extend UML to meet the needs of web-based application. This extension is named UML extension for Web Applications (WAE).

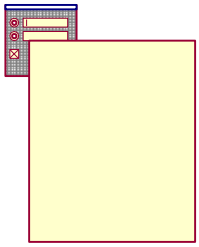
In design modeling using WAE, there are three main class stereotypes were used in building the class diagram for this study, which are:

- `Server page`
- `Client page`
- `Form`

Table 5.1 illustrates the class stereotypes used.

Table 5.1: Class Stereotypes

Name	Server Page
Description	A server page represents a web page that has scripts that are executed by the server. It also interacts with server-side resources such as databases, business logic components, and external system and so on.
Icon	
Name	Client Page
Description	An instance of a client page is a Hypertext Markup Language (HTML) formatted web page with a mix of data, presentation and even logic. Client pages are rendered by client browsers and may contain scripts that are interpreted by the browser.
Icon	

Name	Form
Description	A class stereotyped as a <<form>> is a collection of input fields that are part of client page. This class maps directly to the HTML <form> tag. Its attributes represent the HTML form's input fields: such as input boxes, text areas, radio buttons, check boxes and hidden fields. A <<form >> has no operations since operation can't be encapsulated in a form.
Icon	

In addition, there are four association stereotypes used in this study. They are:

- `Link`
- `Submit`
- `Build`
- `Redirect`

Table 5.2 illustrates the class association stereotypes used.

Table 5.2: Association Stereotypes

Association Stereotypes	Description
<<link>>	A <<link>> is relationship between a client page and another Web page. In a class diagram a link is an association between a «client page» and either another «client page» or a «server page». A Link association maps directly to the HTML anchor tag, where the href attribute is defined.
<<submit>>	A «submit» association is always between a «form» and a «server page». Forms *submit* their field values to the server through «server pages» for processing. The web server processes the «server page», which accepts and uses the information in the submitted form.
<<build>>	The «builds» relationship is a special relationship that bridges the gap between client and server pages. Server pages only exist on the server. This relationship identifies the HTML output of a server's page execution.
<<redirect>>	A «redirect» relationship is a directional association with another web page. It can be directed both from and to client and server pages. This association indicates a command to the client to request another resource.

5.2 Class Diagram for W-HReS

Class diagram for W-HReS has been developed using WAE for UML. As described in Section 5.1, three class stereotypes and four association stereotypes are used. Figure 5.1 shows the class diagram for W-HReS.

As described in Chapter 4, W-HReS consists of five main use cases. They are `make registration`, `make reservation`, `login`, `search for room availability` and `cancel reservation`. Section 5.3 will discuss in detail W-HReS class diagram in terms of system functionality.

5.3 Detailed Class Diagram for W-HReS

Detailed class diagram for W-HReS has been divided based on the customer roles, registered or unregistered customer. In this logical view, attributes and operation for each class was identified.

5.3.1 Detailed Class Diagram for Make Registration (Non- Member)

Customer has to register first before he/she can search or reserve the hotel room. Figure 5.2 shows the detailed class diagram for use case `Make Registration`.

In Figure 5.2, the `<<Client Page>>`, which is `index.html` is associated with a `<<link>>` stereotyped association to another `<<Client Page>>`, `newRegistration.html`. One customer can only make one registration. The `<<Form>>` stereotyped class is modeled as a contained class of the `<<Client Page>>`, `newRegistration.html`. The attributes represents the form input fields such as input boxes, text areas, selection area and hidden fields.

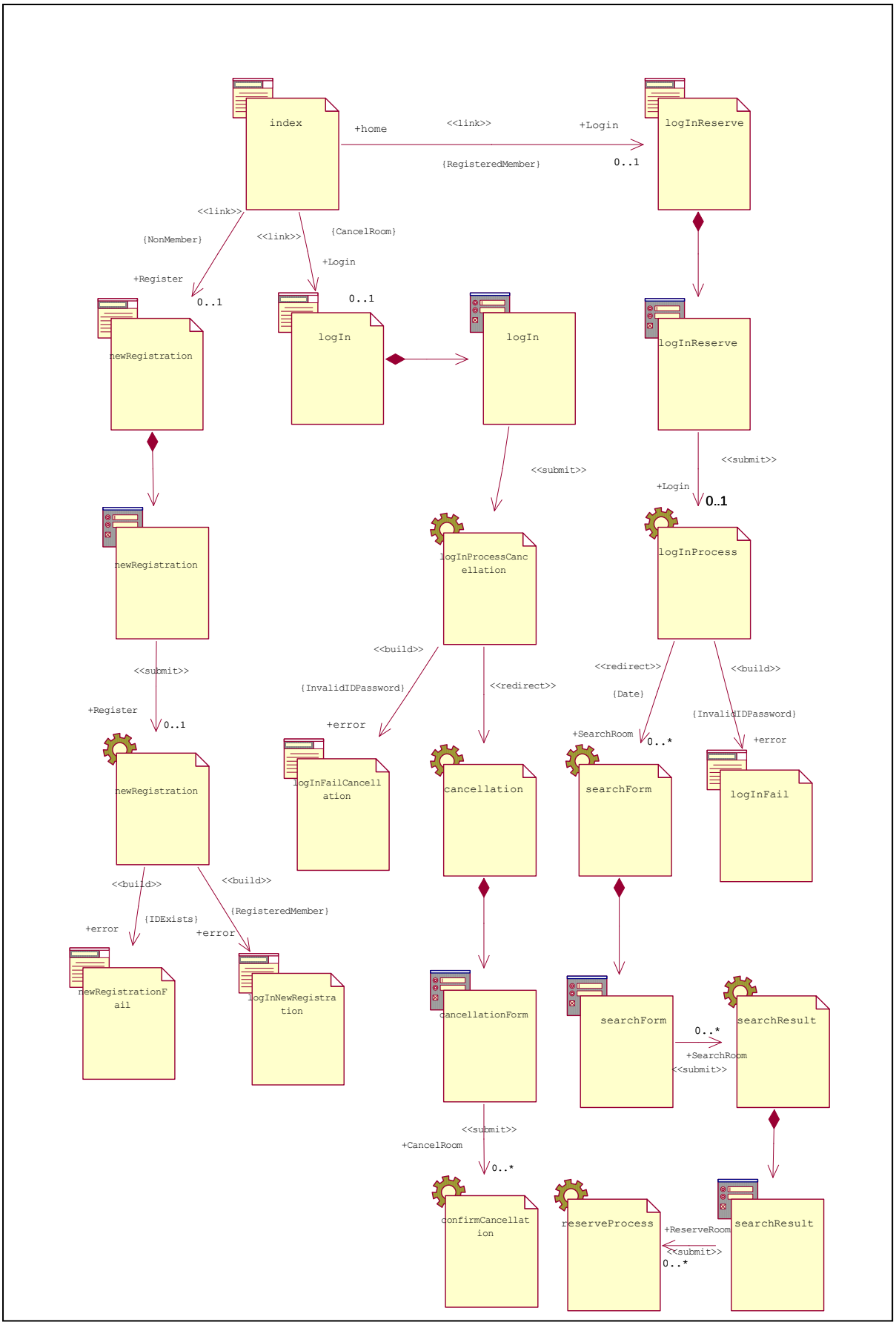


Figure 5.1: Class Diagram for W-HReS

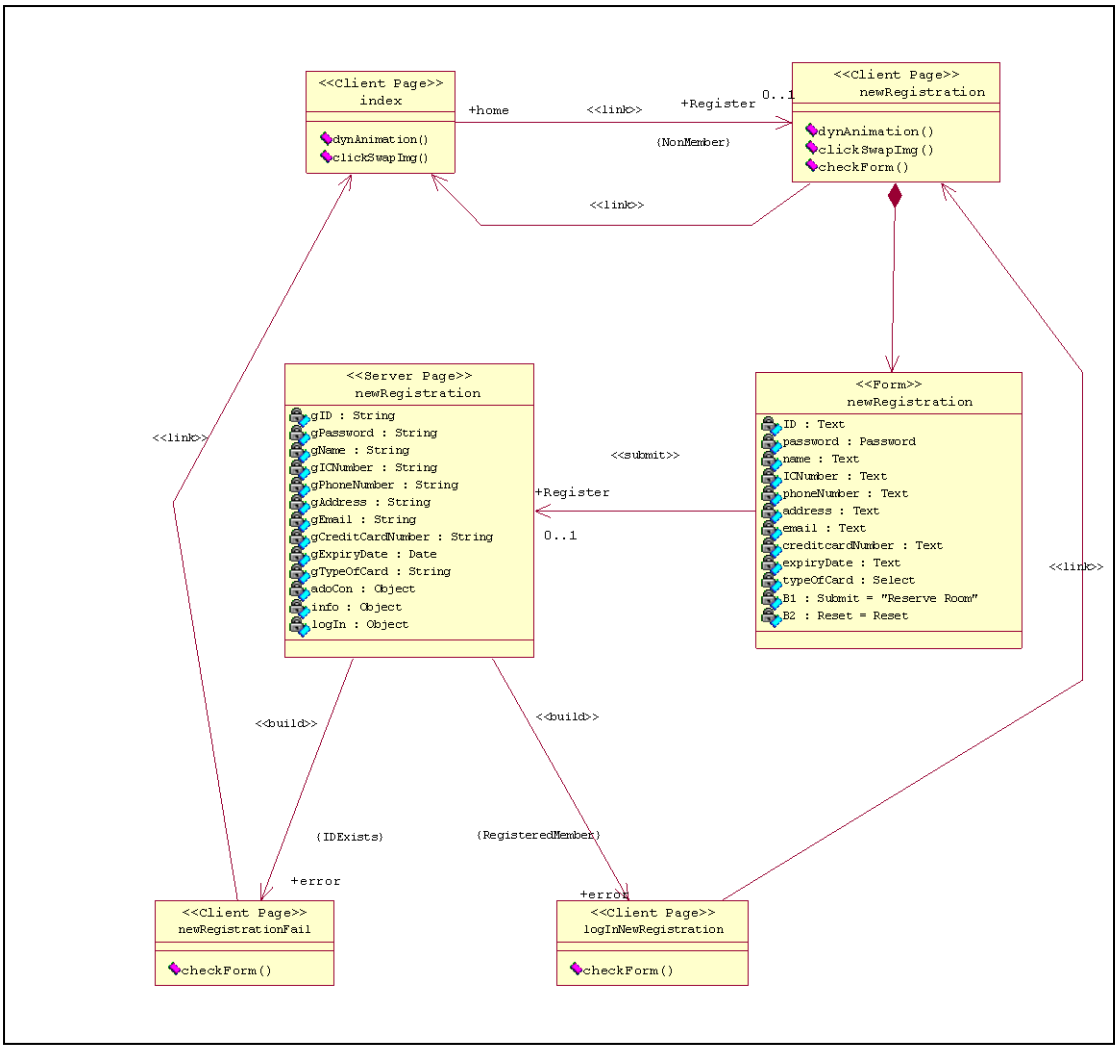


Figure 5.2: Detailed Class Diagram for Make Registration (Non-Member)

The `newRegistration` form is associated with the `<<Server Page>>`, `newRegistration.asp`. The `<<Server Page>>` will process all the request and form's field attributes submitted by the `<<Form>>`. Two client pages which are the `newRegistrationFail.html` and `logInNewRegistration.html` are associated using a `<<build>>` relationship with the `<<Server Page>>`. Both `<<Client Pages>>` will handle error exception from the `<<Server Page>>`. For example, they will give an error message if the registration `ID` has been used and if the customer has already registered.

5.3.2 Detailed Class Diagram for Registered Member

Customer who is already a registered member can search for available room and make room reservation. In order to login to W-HReS, the customer has to use valid `ID` and `password`. Figure 5.3 illustrates the detailed class diagram for registered member to search and reserve the room.

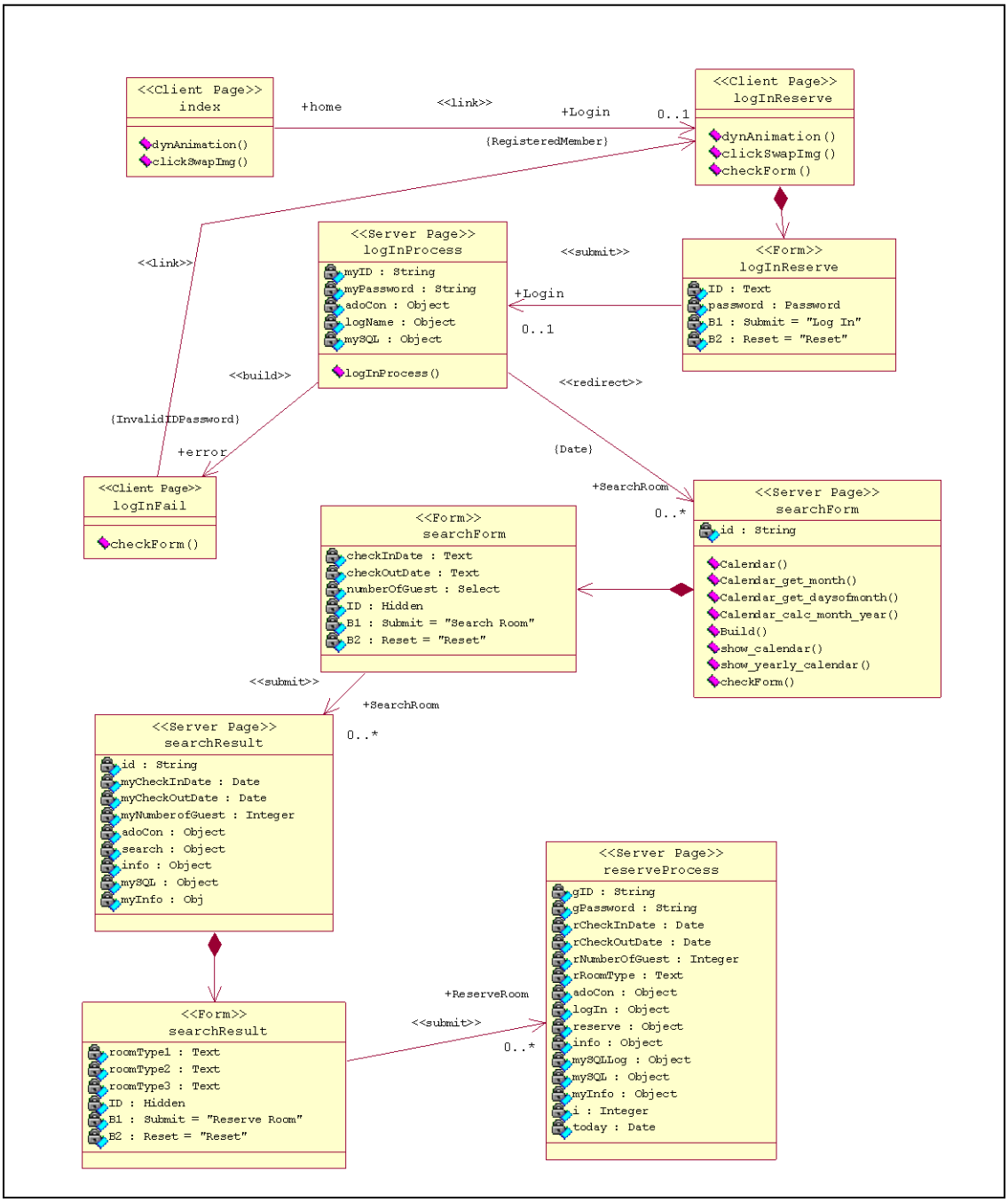


Figure 5.3: Detailed Class Diagram for Search and Room Reservation

In Figure 5.3, the <<Client Page >>, index.html acts as the home page of W-HReS. It is associated with <<link>> relationship to another <<Client Page>>, logInReserve.html. The registered customer member can log to reserve the room as many times as they want. The <<Form>> stereotyped class is modeled as a contained class of the <<Client Page>>. There are four attributes of logInReserve form which are ID, password, Log In and Reset input fields.

The logInReserve form will submit all the input requests to <<Server Page>>, logInProcess.asp to be processed. If the login process is successful then the page will be redirected to <<Server Page>>, searchForm.asp. If not, the <<Client Page>>, logInFail.html will handle the error exception.

The <<Server Page>>, searchForm.asp contains the ASP form, searchForm. The customer can search the room availability by using the form. Room searching function will be based on the check in and check out date. The information requests will be submitted to <<Server Page>>, searchResults.asp for validation. If the room for the specified date is available, the customer has to choose the type of room and submit the reservation requests to <<Server Page>>, reserveProcess.asp.

The <<Server Page>>, reserveProcess.asp will process the reservation requests and store it into the database. Finally, W-HReS will generate the random booking code number to the customer.

The customer can also cancel the room reservation. Figure 5.4 illustrates the detailed class diagram for use case Cancel Rooom Reservation.

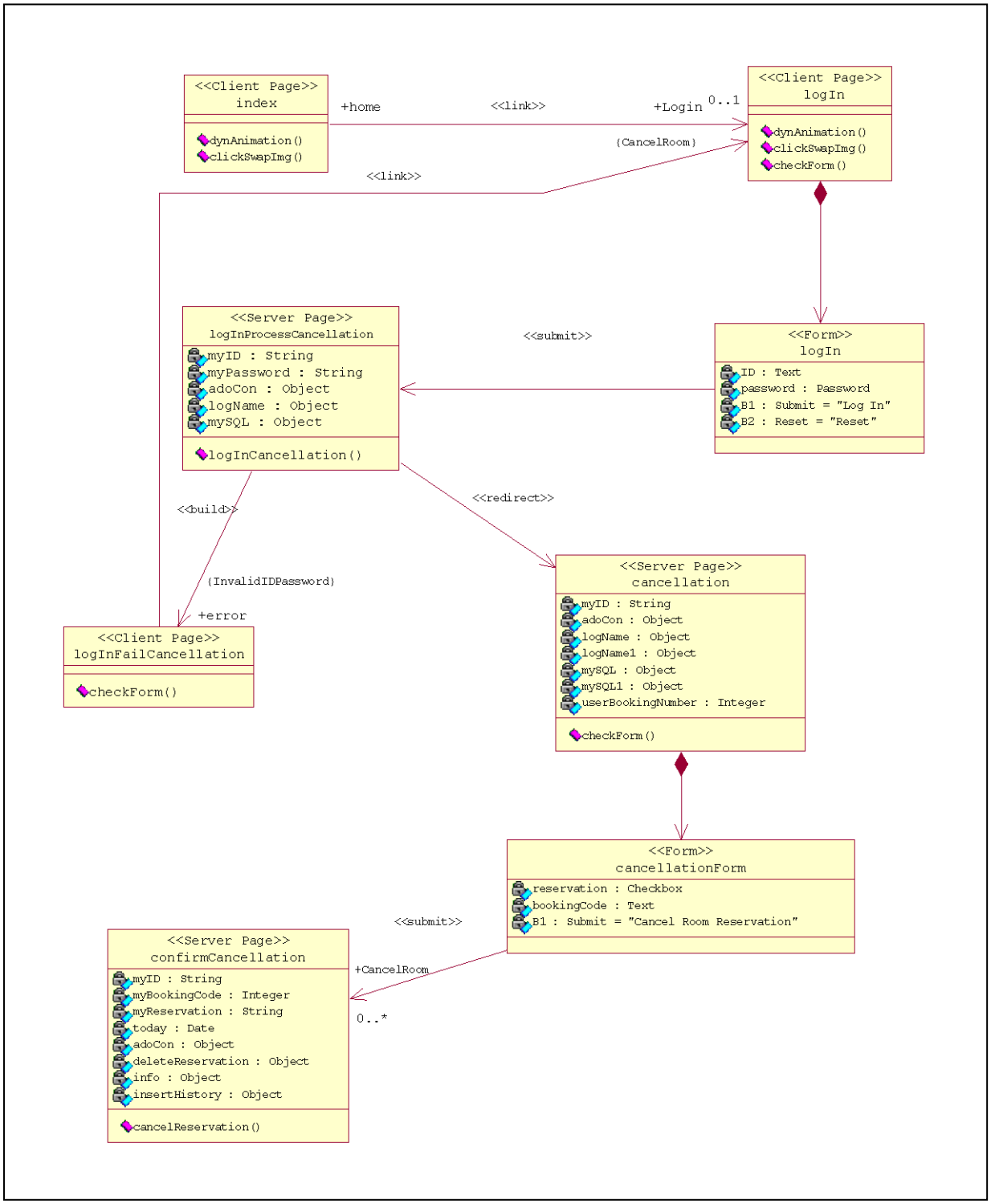


Figure 5.4: Detailed Class Diagram for Cancel Room Reservation

In Figure 5.4, the W-HReS home page which is index.html is associated with <<link>> association to another <<Client Page>>, login.html. This <<Client Page>> contains HTML login form. The <<Form>> will requests the customer to input a valid user ID and password.

<<Submit>> prototyped association will point from the login form to <<Server Page>>, logInProcessCancellation.asp to verify the input. Using logInCancellation() operation, the <<Server Page>> will make the input validation. If the input is valid, the <<Server Page>> will be redirected to another <<Server Page>>, cancellation.asp. If the input fails, the page will be pointed to <<Client Page>>, logInFailCancellation.html to prompt the error message.

From the <<Server Page>>, cancellation.asp, the customer has to input the reservation checkbox and booking code number using cancelForm to cancel the reservation. The <<Server Page>>, confirmCancellation.asp will verify the input and cancel the customer reservation information from the database.

5.4 Summary

This chapter has presented the W-HReS class diagram using WAE for UML. There are three main class stereotypes were used in building the class diagram for this study, which are: `Server page`, `Client page` and `Form`. In addition, there are four main association stereotypes were used in this study, which are: `Link`, `Submit`, `Build` and `Redirect`. The next chapter will discuss the prototype development phase.

CHAPTER 6

PROTOTYPE DEVELOPMENT

This chapter intends to explain the requirements and usage of W-HReS prototype developed. These include the requirements of hardware and software and also the prototype usage.

6.1 Hardware and Software Requirements

During the implementation phase, the system architecture which has been defined in the previous phase is transformed into codes using selected software. The hardware and software requirements used in this study are illustrated in Table 6.1 and Table 6.2.

Table 6.1: Hardware Requirements

No	Type of hardware	Purpose
1.	Intel Pentium IV Processor	Processor
2.	225 MB RAM	Memory
3.	Microsoft XP Professional	Operating System (OS)

Table 6.2: Software Requirements

No	Type of software	Purpose
1.	Internet Information Services (IIS) 5.1	Web Server
2.	Internet Explorer 6.1	Browser
3.	Microsoft Access XP	Database Management System (DBMS)

6.2 Prototype Usage

A user can use the W-HReS by using the browser to access the Hotel Reservation System home page. The home page provides the relevant hotel information. The basic functionality of W-HReS is based on the use case identified in Section 4.2.1. They are `make registration`, `make reservation`, `login`, `search for room availability` and `cancel reservation`. There are two sections for the user to use the system which are Non-Member and Registered Member section. The user has to register before they can login into the system. Once the user logs in, the user can search for available room and make room reservation. Apart from this, the system also allows the user to cancel their room reservation.

Details for prototype usage are provided in Appendix A: W-HReS User Manual.

6.3 Summary

This chapter has highlighted the prototype developed which is Web-based Hotel Reservation System (W-HReS). W-HReS is being used as an example of common web-based reservation system in this study. This prototype has been developed using ASP scripting language and MS Access as a database. For more details about the prototype usage; refer to Appendix A (W-HReS User Manual). The following chapter will explain the testing analysis phase in this study.

CHAPTER 7

TESTING ANALYSIS

This chapter discusses the testing procedures to test the UML design model. DCD criteria proposed by Andrews et al (2003) have been used to test the W-HReS class diagram.

7.1 Testing Procedure

In this phase, UML design model refers to the class diagram. DCD criteria for testing UML design model include the following:

- Association-end multiplicity criterion (AEM)
- Generalization (GN) criterion
- Class attribute (CA) criterion

The DCD criteria are used to determine test adequacy for the tested class diagram and also to guide in the selection of test cases. Details for each DCD criteria have been provided in Section 2.5.

To conduct the testing process, the following assumptions have been made:

1. The class diagram is syntactically correct. This has been ensured by the UML Computer Aided Software Engineering (CASE) tool, Rational Rose.
2. The test case design is performed by using the category partitioning technique. Using this technique the value domain is partitioned into equivalence class, which is valid and invalid class. However the `Button` such as `Submit and Reset` input type do not have the equivalence class. They have only one state.
3. Testing is performed based on the input values from the user. Therefore the test only considered the **`<<form>>` stereotypes** developed for W-HReS.
4. **Generalization (GN)** criterion was not covered in testing the class diagram since the generalization specialization relationship does not exist in the system.
5. There are **three status** used for testing results, which are:
   - `Not Applicable (NA)`

     `NA` status is considered when the input entered is invalid and no action done from the pointed class.
   - `Successful (S)`

     `S` status is considered when the input entered is valid and the system successfully interacts with the associated class.
   - `Unsuccessful (US)`

     `US` status is considered when the input entered is valid. However the input does not match with the data in the database.

6. **I** and **V** indicate invalid and valid input respectively.
7. **T** stands for `True` and **F** for `False`. `T` indicates that the input parameters can successfully pass the signal to other associated class, and `F` indicates that the input parameters not successfully pass the signal to other associated class.

The next section will discuss the approach of testing UML design model using DCD criteria. The explanations are according to use cases identified.

7.2 Make Registration

`Make Registration` activity or use case requires twelve inputs, as follows:

- `ID`
- `password`
- `name`
- `ICNumber`
- `phoneNumber`
- `address`
- `e-mail`
- `creditCardNumber`
- `expiryDate`
- `typeOfCard`
- `Submit`
- `Reset`

These can be found in the newRegistration form as illustrated in Figure 5.3.

Association multiplicities (AEM) denote the association end multiplicities between the User and newRegistration class. The AEM that need to be covered is: {(0,0) , (1,1)}. The first number in each pair denotes how many User instances are associated with a newRegistration instance. The second number denotes how many newRegistration instances are associated with one User instance.

In order to test **Class Attribute (CA)** criterion, a form of category partitioning technique is adapted for this study. Based on this technique, the following constraints have been defined for each input parameters:

- ID is not less than 5 characters and not more than 8 characters [5, 8].
- password is not less than 5 characters and not more than 8 characters [5, 8].
- ICNumber format is consists of 12 digit (xxxxxx-xx-xxxx), [12].
- phoneNumber format is consists of 10 digit (xxx-xxxxxxx), [10].
- e-mail format is xxx@xxx.xx.
- expiryDate format is consists of 6 digit (xx/xx/xx), [6].
- creditCardNumber format is consists of 16 digit (xxxx-xxxx-xxxx-xxxx), [16].

Input parameters for `name` and `address` are declared as a `String` data type. Therefore there is no equivalence class. The default input for `typeOfCard` is defined by the system using `selection box`. Therefore the `typeOfCard` will always be valid.

Table 7.1 in Appendix B shows the parameters that are passed in make registration signal (column 2), associated class and the DCD coverage elements exercised during the test. There are seven input parameters, which are `ID`, `password`, `ICNumber`, `phoneNumber`, `e-mail`, `expiryDate` and `creditCardNumber` that represent the valid and invalid class. Therefore there would be at least 128 (2^7) input combination test cases to test the class diagram. Test case 1 to 127 tests whether the user can successfully make the registration using the invalid input. Test case 128 test if the user can successfully pass all the inputs to the `newRegistration` class using valid parameters. Test case 129 and 130 were added to test whether the user can successfully make the registration using valid `ID` and `ICNumber` respectively but the parameters have been used by another user. Test case `131` tests the class using the `Reset` button.

AEM `(0,0)` indicates that no user is being registered on the system. AEM `(1,1)` indicates that a user is being registered on to the system. CA refers to the input parameters involved in particular test cases.

7.3 Login

Login activity or use case requires four main inputs, which are:

- ID
- password
- Submit
- Reset

This shows that login activity requires ID, password, submit and reset to be passed as parameters. The input can be figured out in the login <<form>> stereotypes in Figure 5.3. The form is also associated with logInProcess class.

Association multiplicities (AEM) that need to be covered between the User and loginProcess class is: {(0,0) , (1,1)}. The first number in each pair denotes how many User instances are associated with a Login instance. The second number denotes how many Login instances are associated with one User instance.

Class Attribute (CA) criterion is expressed in terms of representative values. In order to establish the set of representative value, a form of category partitioning technique is adapted. Based on the category partitioning technique, there are two inputs, which are ID and password that have the valid and invalid class. Submit and Reset is the buttons that have only one state. ID and password input constraint is not less than 5 characters and not more than 8 characters [5, 8] respectively. Therefore the input that does not meet the constraint is

considered the invalid class. `ID` and `password` can be both independently valid and invalid.

Table 7.2 in Appendix B illustrates the test cases derived from DCD for login activity. Test cases 1, 2 and 3 test whether a user can log on to the system using invalid `ID` or `password`. Test case 4 test whether the user can log on to the system using valid `ID` or `password`. Test case 5, 6 and 7 test whether user can log on to the system using valid `ID` or `password` but the input does not match with the data in the database. Test 8 repeat the same test as test case 4 but using the `Reset` button instead of `Submit` button.

AEM `(0,0)` indicates that no user is being login on to the system. AEM `(1,1)` indicates that a user is being login on to the system. CA refers to the attributes involved in particular test cases.

7.4 Make Reservation

`Make Reservation` activity or use case requires three main inputs, which are:

- `roomType`
- `Submit`
- `Reset`

The tests assume that the user is logged in and the room is available. The class form for make reservation is associated with `reserveProcess` class. The class diagram is illustrated in Figure 5.3.

Association multiplicities (AEM) that need to be covered between the `User` and `reserveProcess` class is: {`(1,0), (1, 1), (1,2)`}. The first number in each pair denotes how many `User` instances are associated with a `room reservation` instance. The second number denotes how many `room reservation` instances are associated with one `User` instance.

Class Attribute (CA) using equivalence partitioning technique for `make reservation` is considered only one input parameter, which is `roomType`. The default input for `roomType` has been defined by the system using `checkbox`. Therefore the `roomType` will always be valid.

Table 7.3 in Appendix B shows the test cases derived from DCD for make reservation activity. Test cases 1 and 2 test whether a user can make room reservation using valid `roomType` and `Submit` button. Test case 2 adds more than one room to a user. Test case 3 tests whether the user can make reservation using valid `roomType` and `Reset` button.

In coverage elements columns, AEM `(1,1)` indicates that a user can be reserved to one room, AEM `(1,2)` indicates that a user can be reserved to two rooms and AEM `(1,0)` indicates that a user being reserved to non-existing rooms. CA parameters include the `roomType`, `Submit` and `Reset` button.

7.5 Search for Room Availability

`Search for Room Availability` activity or use case requires five main inputs, which are:

- `checkInDate`
- `checkOutDate`
- `numberOfGuest`
- `Submit`
- `Reset`

The tests assume that the user is successfully logged in. The class form for room searching is associated with `searchResult` class. This is shown in Figure 5.3.

Association multiplicities (AEM) that need to be covered between the `User` and `searchResult` class is: {`(1,0), (1, 1), (1,2)`}. The first number in each pair denotes how many `User` instances are associated with a `room searching` instance. The second number denotes how many `room searching` instances are associated with one `User` instance.

Class Attribute (CA) using equivalence partitioning technique for `search room reservation` consists of three input parameter, which are `checkInDate, checkOutDate and numberOfGuest`. `Submit` and `Reset` button does not have any constraint. The default input for `numberOfGuest` has been defined by the system using `selection box`. Therefore the `numberOfGuest` will always be valid.

The checkInDate input constraint is the checkInDate parameter must be more than the current date and must be less than checkOutDate (currentDate < checkInDate < checkOutDate). The checkOutDate constraint is that the date must be more than the current date (checkOutDate > currentDate) and checkInDate (checkOutDate > checkInDate). Other input is considered invalid.

Table 7.4 in Appendix B shows the test cases derived from DCD for search room availability activity. Test cases 1, 2 and 3 test whether the user can search room availability using invalid checkInDate or checkOutDate input. Test case 4 and 5 test whether the user can search room available using valid checkInDate or checkOutDate input. Test case 5 is added to test that a user can search room more than one room. Test case 6 tests whether the user can search room using valid checkInDate , checkOutDate and Reset button.

In Table 7.4, in coverage elements column, AEM (1,1) indicates that a user can search to one room, AEM (1,2) indicates that a user can search to two rooms, and AEM (1,0) indicates that a user being searching to non-existing rooms. CA parameters include the checkInDate , checkOutDate, numberOfGuest, Submit and Reset button.

7.6 Cancel Reservation

Cancel Reservation activity or use case requires three main inputs, which are:

- bookingCode
- reservation
- Submit

The tests assume that the user is logged in and the room has been reserved. The class form for cancel reservation is associated with confirmCancellation class. The class diagram is illustrated in Figure 5.4.

Association multiplicities (AEM) that need to be covered between the User and cancellation class is: {(1,0), (1, 1), (1,2)}. The first number in each pair denotes how many User instances are associated with a room cancellation instance. The second number denotes how many room cancellation instances are associated with one User instance.

Class Attribute (CA) using equivalence partitioning technique for cancel reservation has one input parameter, which is bookingCode. The default input for reservation has been defined by the system using checkbox. Therefore the reservation will always be valid. The valid class for bookingCode parameter must consist of 6 characters, [6]. Input less or more than 6 characters is considered invalid.

Table 7.5 in Appendix B shows the test cases for cancel reservation activity. Test case 1 tests whether a user can cancel room reservation using invalid bookingCode number. Test case 2 tests whether a user can cancel room reservation using valid

`bookingCode` number. Test case 3 adds if the user can cancel room more than one room. Test case 4 tests whether the user can cancel reservation using valid `bookingCode` number but the parameter does not match with the data in the database.

AEM `(1,1)` indicates that a user can cancel to one room. AEM `(1,2)` indicates that a user can cancel to two rooms. AEM `(1,0)` indicates that a user being cancel to non-existing rooms. CA parameters include the `bookingCode`, `reservation`, and `Submit b`utton.

7.7 Summary

This chapter has described the testing analysis phase using the DCD criteria. The testing procedures in conducting the testing process have been highlighted. The test cases have been constructed based on the use case identified. In addition, the coverage elements criterion for each test case also has been stated. Discussion on findings and future works of this study will be covered in the following chapter.

CHAPTER 8

DISCUSSION AND FUTURE WORKS

As mentioned in Section 2.5, DCD criteria for testing W-HReS class diagram is divided into three criteria; AEM, GN and CA. However, we have found that:

i. Not all criterions in DCD can be applied to test the W-HReS class diagram using WAE. The following explains the findings for each criterion.

   - AEM – Multiplicity in web-based application design model is not apparent. Thus AEM cannot be used to represent all relationships in WAE class diagram.
   - GN – GN criterion is not applied because there is no generalization association in designing web-based application model. So, GN is totally not applicable.
   - CA – CA criterion is apparent and well-defined in WAE class diagram. All attributes are defined in `<<form>>` class stereotypes. Since these criteria can be obtained from these stereotypes, CA is appropriate to test the diagram.

ii. A class consists too many inputs will generate complex test cases. It involves a large number of test input parameters combination.

8.1 Future Works

This study can be extended to test other UML diagrams such as sequence diagram, collaboration diagram, state chart diagram, sequence diagram and activity diagram for web-based application. In addition the W-HReS can be enhanced using others technologies such as Hypertext Preprocessor (PHP) or Extensible Markup Language (XML) technology. This application can also be integrated with others DBMS like Oracles or MySQL. Besides using the web-based hotel reservation as domain environment, this study can be extended to test applications from other domain such as inventory or procurement management system.

8.2 Conclusion

In this study, the designing process of test cases from UML design model for web-based application using DCD criteria was successfully implemented as described in Chapter 7. However, the study found that DCD criteria, in total, are not appropriate for testing web-based application design model. This is due to the fact that AEM cannot be used to represent all relationships in WAE class diagram. In addition, since there is no generalization association in WAE class diagram, GN cannot be used to test the diagram. However CA seems to be appropriate to test the diagram since all criteria are apparent and well-defined in the diagram.

REFERENCES

Andrews, A., France, R., Ghosh, S. & Craig, G. (2003) Test Adequacy Criteria for UML Design Models. *Journal of Software Testing, Verification and Reliability*, 13(2):95–127

Bennett, S., McRobb, S. & Farmer, R. (2002). *Object-Oriented Systems Analysis and Design 2nd Edition*. United Kingdom, McGrawHill

Binder, R., V. (1999). *Testing Object-Oriented Systems Models, Patterns, and Tools*. Object Technology Series. Addison Wesley, Reading, Massachusetts

Booch, G., Jacobson, I. & Rumbaugh, J. (1998). *The Unified Software Development Process*. Massachusetts, Addison Wesley

Braude, E.J. (2000). *Software Engineering: An Object-Oriented Perspective.* United States of America: John Wiley & Sons, Inc

Briand L., & Labiche , Y. (2002) A UML-based approach to system testing. In *4th International Conference on the UML*, pp. 194-208

Conallen, J. (1999). Modeling Web Application Architectures with UML. *Comm. of the ACM, vol. 42, no. 10,* pp. 63-70

Elbaum, S., Karre, S., and Rotheremel, G. (2003). Improving Web Application Testing with User Session Data. *Proceedings of the 25th International Conference on Software Engineering*. May, 2003, (pp. 49-59). IEEE-ACM.

Eriksson, H., & Penker, M. (1998). *UML Toolkit*. United States of America, John-Wiley & Sons, Inc.

Futrell, R. T., Shafer, D. F., & Shafer L. I. (2001). *Quality Software Project Management*. Prentice Hall

Ghosh, S., France, R. B. , Braganza, C., Kawane, N., Andrews, A. and Pilskalns, O. (2003) Test Adequacy Assessment for UML Design Model Testing in *Proceeding of The International Symposium on Software Reliability Engineering*, ISSRE 2003, Denver, CA,USA, November 17-20, 2003**.**

Jia, X. and Liu, H. (2002). *Rigorous and Automatic Testing of Web Application*. Retrieved April 28, 2004 from http://citeseer.ist.psu.edu/552920.html

Kaner, C., Falk, J. & Nguyen, Q.H. (1999). *Testing Computer Software*. 2nd Edition.United States of America, John-Wiley & Sons, Inc.

Nguyen, H., Q. (2001). *Testing Applications on the Web: Test Planning for Internet-based Systems*. United States of America, John-Wiley & Sons, Inc.

Neise, O., Margaria, T. and Steffen, B. (2002). Automated Functional Testing of Web-based Applications. *In Proceedings of 5th International Conference on Software Quality Week Europe*, March 2002 (pp. 157-166). Brussles.

Nilawar, M (2003). A UML-based Approach for Testing Web Applications. *Master Thesis*, University of Nevada, Reno

Nunamaker, J., Chen, M., and Purdin, T. (1990-91). System Development in Information Research. *Journal of Management Information System*, 7(3), 89-106.

Offutt, J., & Abdurazik, A.(1999) Generating test from UML specifications. In *2nd International Conference on the UML*, pp. 416–429

Pressman, R. (2001). *Software Engineering A Practitioner's Approach*. 5th Edition. Singapore, McGraw-Hill International Edition

Shklar, L. and Rosen, R. (2003). *Web Application Architecture: Principle, Protocol and Practices*. West Sussex: John Wiley and Sons

Sommerville, I. (2001). *Software Engineering*. 6th Edition. England, Addison Wesley.

Ricca F., &. Tonella, P. (2001). Analysis and Testing of Web Applications. *Proceedings of the 23rd International Conference on Software Engineering (ICSE2001)*, pp. 25-34, 20

Ricca, F., & Tonella, P. (2002). Testing Processes of Web Applications. *Annals of Software Engineering, vol. 14, no. 1-4*, pp. 93-114

Roper, M (1994). *Software Testing*. International Software Quality Assurance Series. McGraw-Hill

Sparxsystems's UML Tutorial Page (n.d.). Retrieved May 3, 2004, from: http://www.sparxsystems.com.au/UML_Tutorial.htm

Trong, T. T. D. (2003). A Systematic Procedure for Testing UML Design. Presented in the *International Symposium on Software Reliability Engineering, ISSRE 2003, Denver, USA, November 17-20, 2003*. Retrieved 20 March, 2004 from: http://www.cs.colostate.edu/~trungdt/uml_testing/uml_testing.html

Williams, E., C.(1999) Software Testing and the UML (on-line). Retrieved 20 March, 2004 from: www.chillarege.com/fastabstracts/issre99/99119.pdf

Wu, Y., & Offutt, J. (2002). Modeling and Testing Web-based Applications. *GMU ISE Technical ISE- TR- 02-08*

APPENDIX

APPENDIX A:

W-HReS USER MANUAL

APPENDIX A: USER MANUAL

Contents:

A-1 : Getting Started

1) Hardware Requirements
   Processor : Intel Pentium IV Processor
   Memory : 225 MB RAM
   OS : Microsoft XP Profesional

2) Software Requirements
   Web Server : Internet Information Services 5.1
   Database : Microsoft Access XP
   Browser : Internet Explorer 6.0

3) Installation Instructions
   a. Create a new sub-directory in your default web server directory (common default web server directory is C:/Inetpub/wwwroot).
   b. Copy the *research* folder into the default web server directory.
   c. Run the application through browser (http://localhost/research).

4) Troubleshooting

   a. Page cannot be displayed
      i. Check the IIS configuration and ensure that the default web directory has been set correctly.

   b. Unable to run script
      i. Check the IIS configuration. Your server should be permitted to run and execute scripts.

   c. Unable to update database.
      i. Check the database file properties (right-click on the database file icon). Ensure that *everyone* has *Full Access*.

A-2 How to Use the System

1) Home Page

Figure A-1 shows the home page of Hotel Reservation System. On the left-hand side, there is a list of menu for both non-member and registered member. On the right-hand side, the relevant information about the hotel is displayed.

Figure A-1: Home page of Hotel Reservation System

2) New User

User needs to click the Non-Member menu. New page as shown in Figure A-2 will be displayed. The user has to fill in the form in order to search for room availability or to reserve room. The user must provide all the information required (ID, password, name, IC Number, phone Number, address, e-mail, credit card number, credit card expiry date and type of credit card).

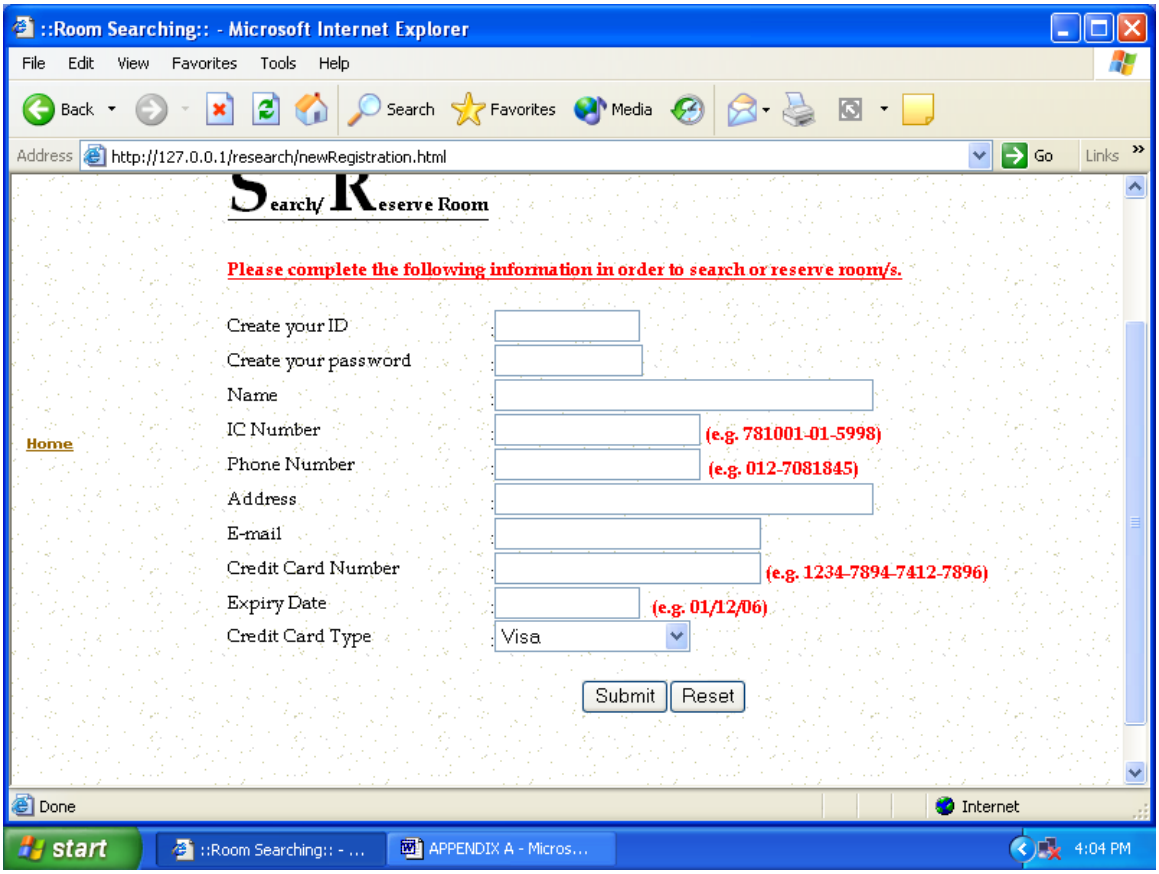


Figure A-2: Non-Member Registration.

A failure message will be displayed if:

i. The length of chosen ID is less than 5 characters (refer to Figure A-3).
ii. The length of chosen password is less than 5 characters (refer to Figure A-4).
iii. The format of IC Number was invalid (refer to Figure A-5).
iv. The format of Phone Number was invalid (refer to Figure A-6).
v. The format of e-mail was invalid (refer to Figure A-7).
vi. The format of Expiry Date was invalid (refer to Figure A-8).
vii. The format of Credit card Number was invalid (refer to Figure A-9).
viii. One or more fields are empty (refer to Figure A-10).

ix. The chosen ID has been used by other users (refer to Figure A-11).
x. The IC Number has been recorded in database (refer to Figure A-12).

Figure A-3

Figure A-4

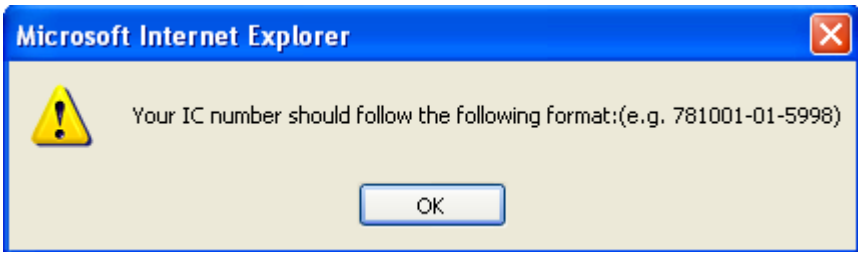


Figure A-5

Figure A-6

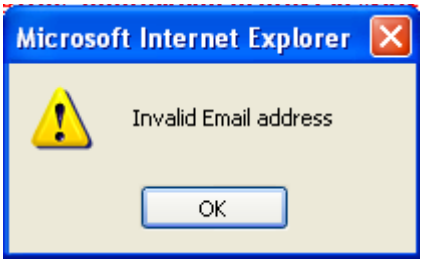


Figure A-7

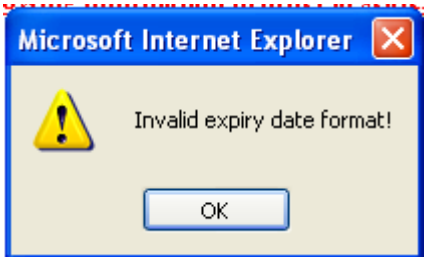


Figure A-8

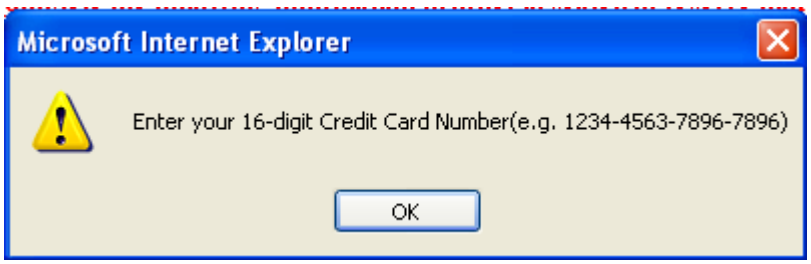


Figure A-9

Figure A-10

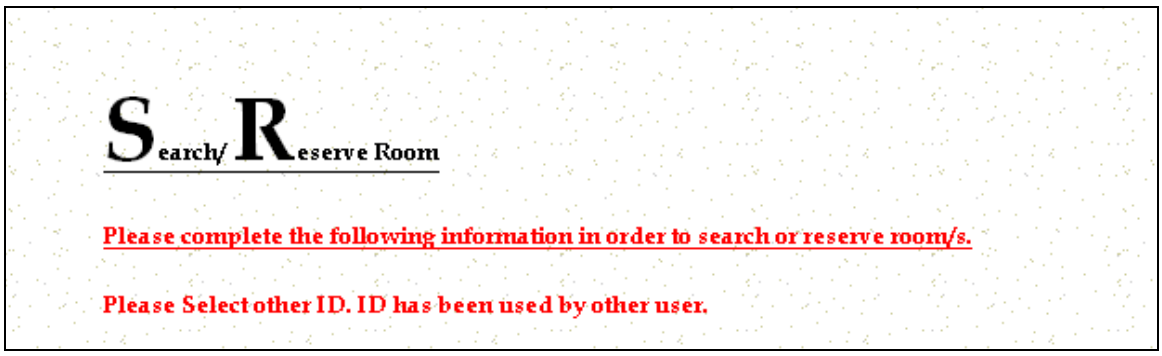


Figure A-11

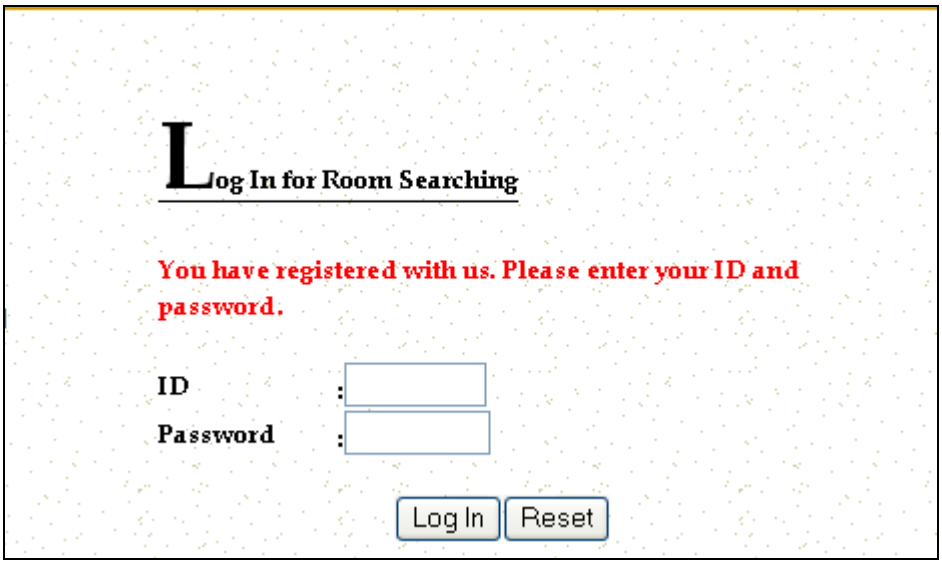


Figure A-12

User will successfully register with the system, if all information required is entered correctly. Figure A-13 shows the result.

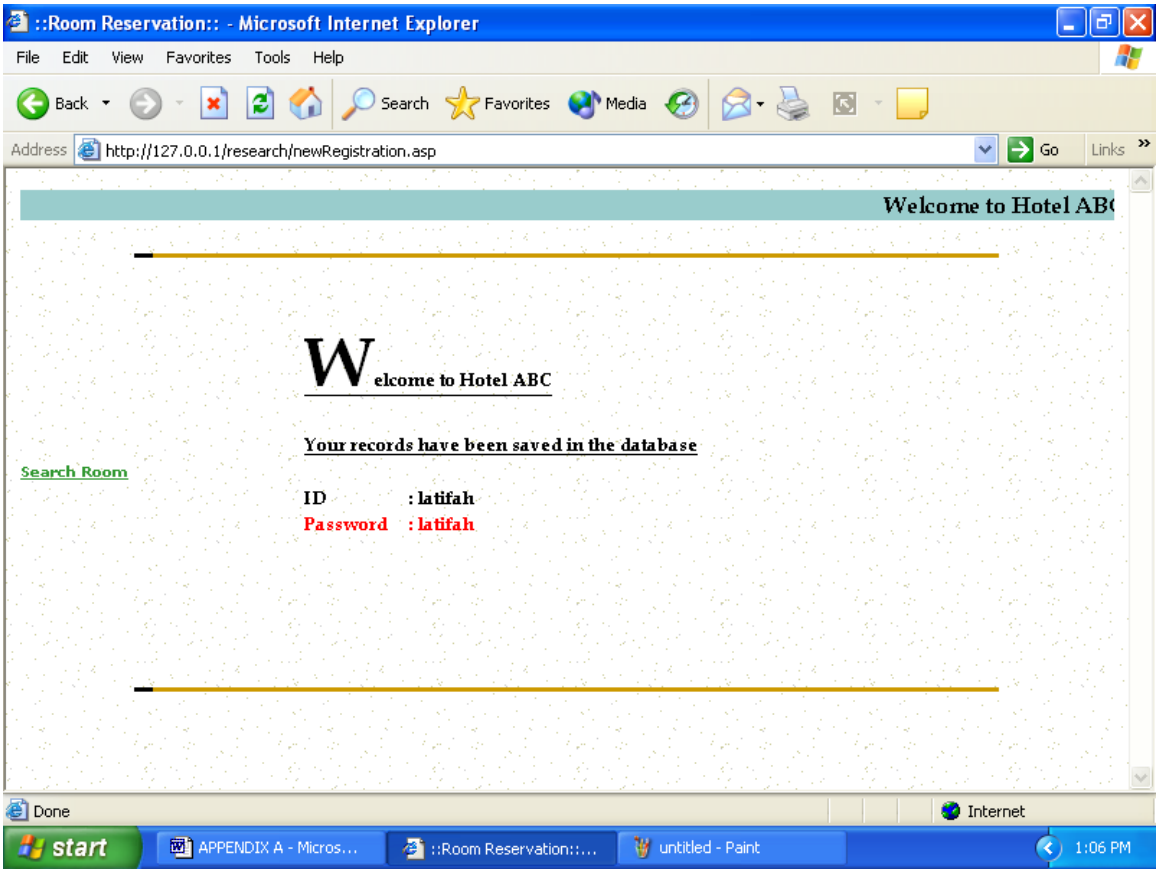


Figure A-13

User can start using the system by choosing the menu Search Room at the left-hand side of the page (refer to figure A-14). Then, user will be directed to the Room Searching page. This page displays 3 inputs; Check in Date, Check out Date and Number of Guest (refer to Figure A-15). User has to click the calendar icon beside the Check in Date input box. Then the window calendar will be displayed (refer to figure A-16).

User has to select the Check in Date. An error message will be displayed, if the Check in Date is less than current date (Refer to Figure A-17).

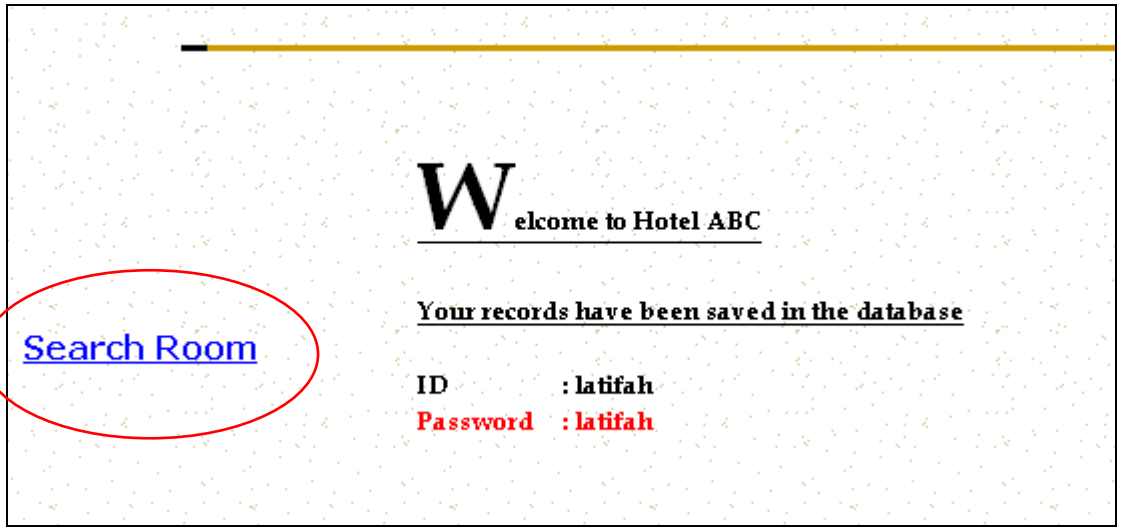


Figure A-14

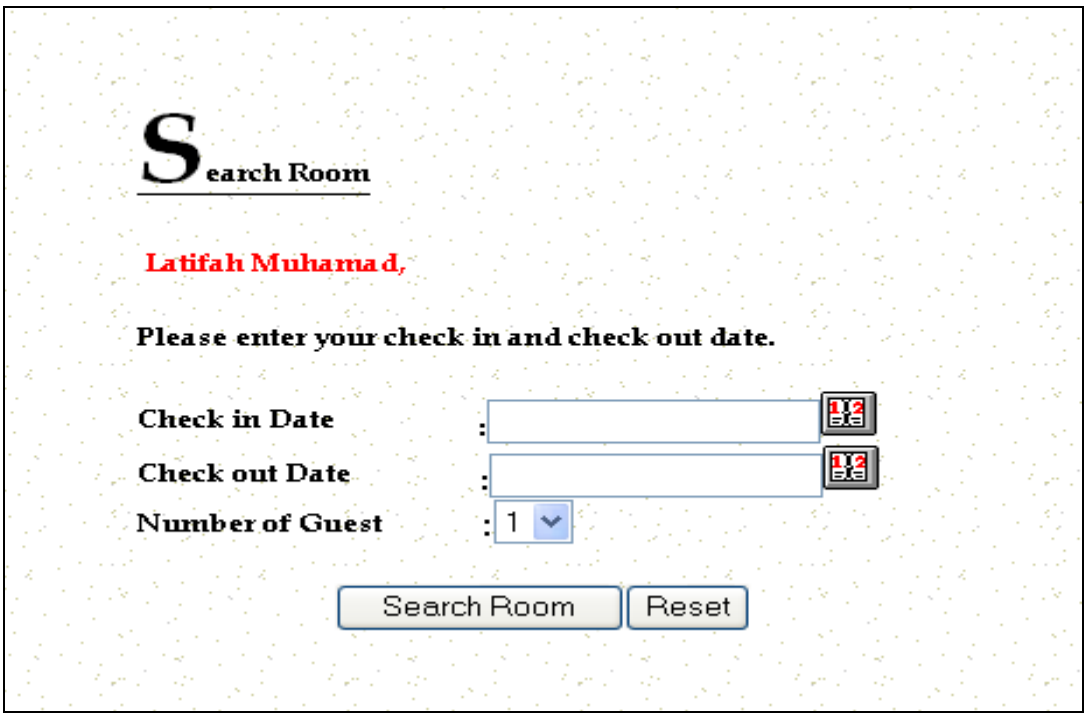


Figure A-15

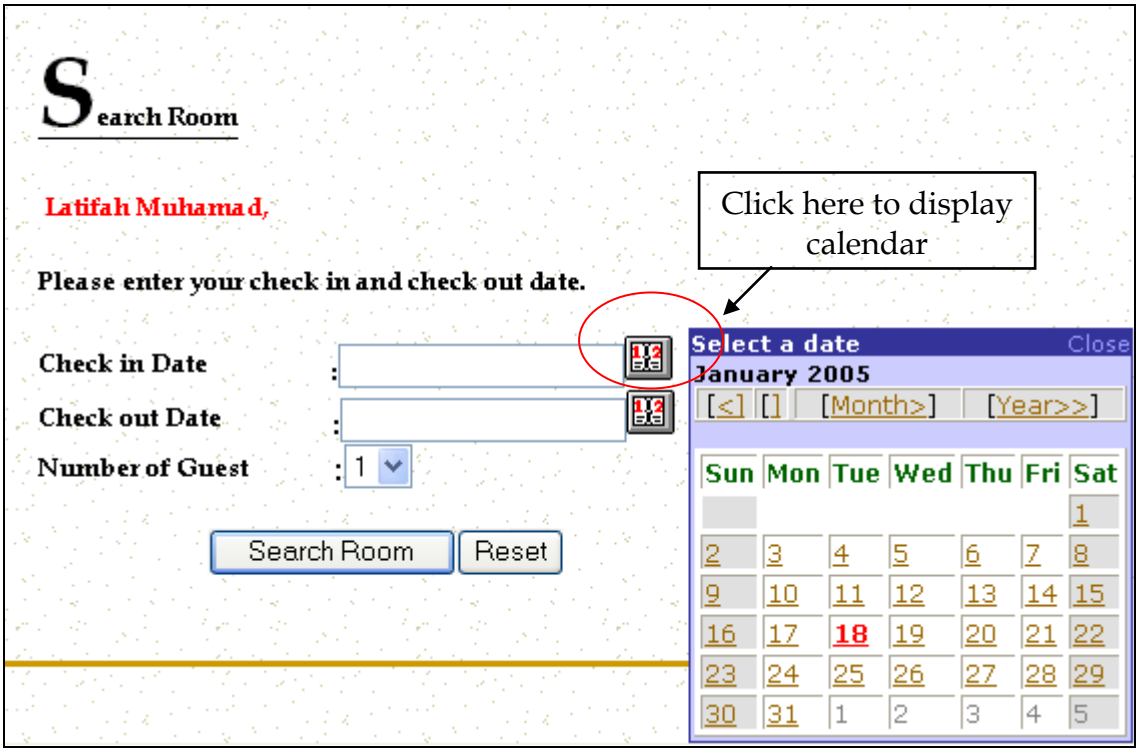


Figure A-16

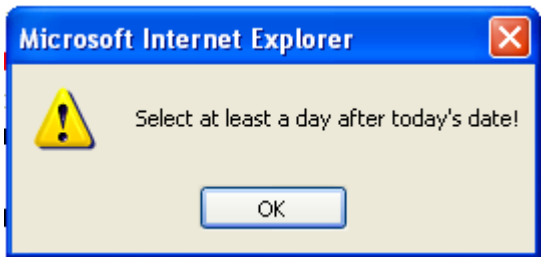


Figure A-17

Then, user has to select the Check out Date. The similar procedure as selecting Check in Date is applied. An error message will be displayed, if the Check out Date is less than Check in Date (Refer to Figure A-18).

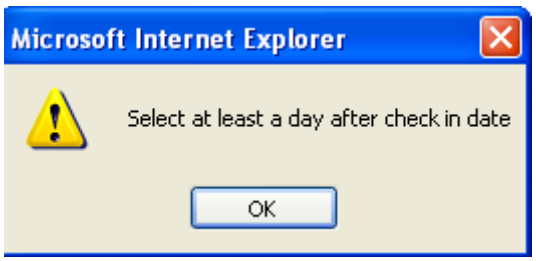


Figure A-18

User need to select the Number of Guest for final process. The default Number of Guest is 1 person. The process is completed when user click the Search Room button. The system will search for available room during the requested date (refer to Figure A-19).

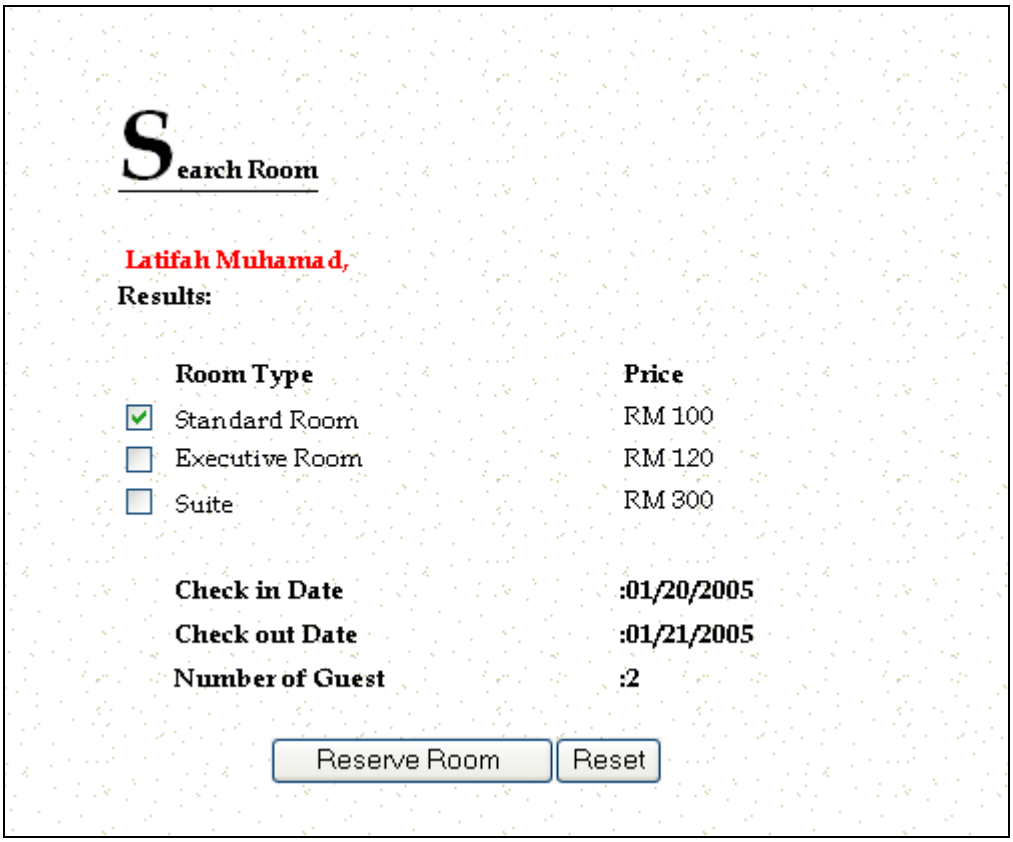


Figure A-19

An error message will be displayed if one or more input box is left unfilled (refer to Figure A-20).

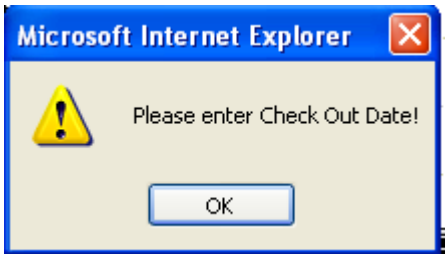


Figure A-20

User can select the available room type and click the Reserve Room button to reserve a room. User will be given a Booking Code for the purpose of Check in process (refer to Figure A-21). User has to present this booking code during check in.

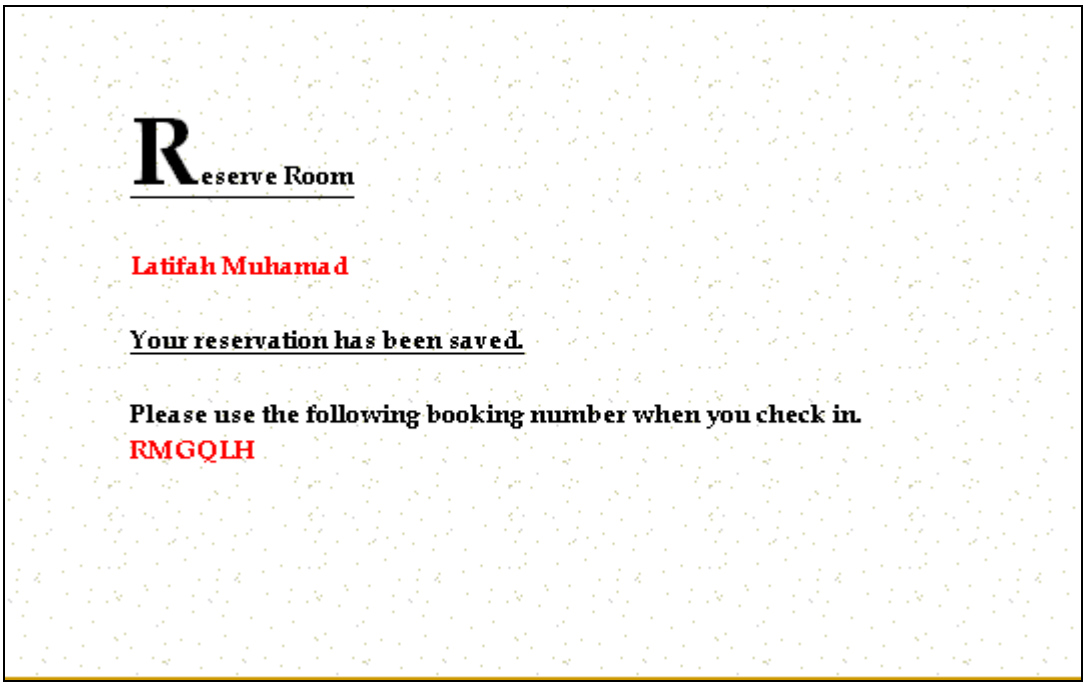


Figure A-21

From this point, user is also allowed to cancel the reservation that he/she has made. User has to click the menu Cancel Reservation at the left-hand side of the page (refer to Figure A-22).

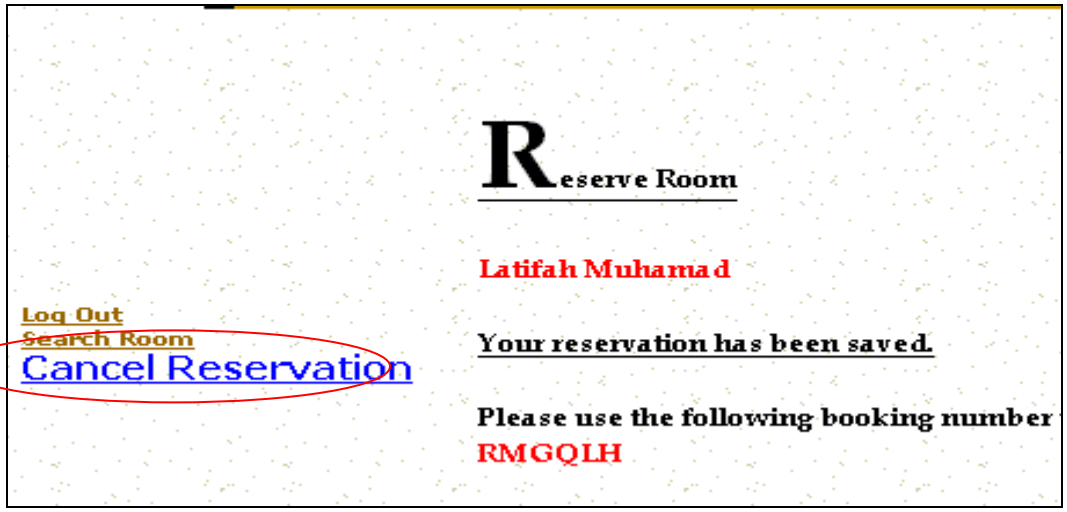


Figure A-22

User will be directed to a page containing his/her own reservation record (refer to Figure A-23).

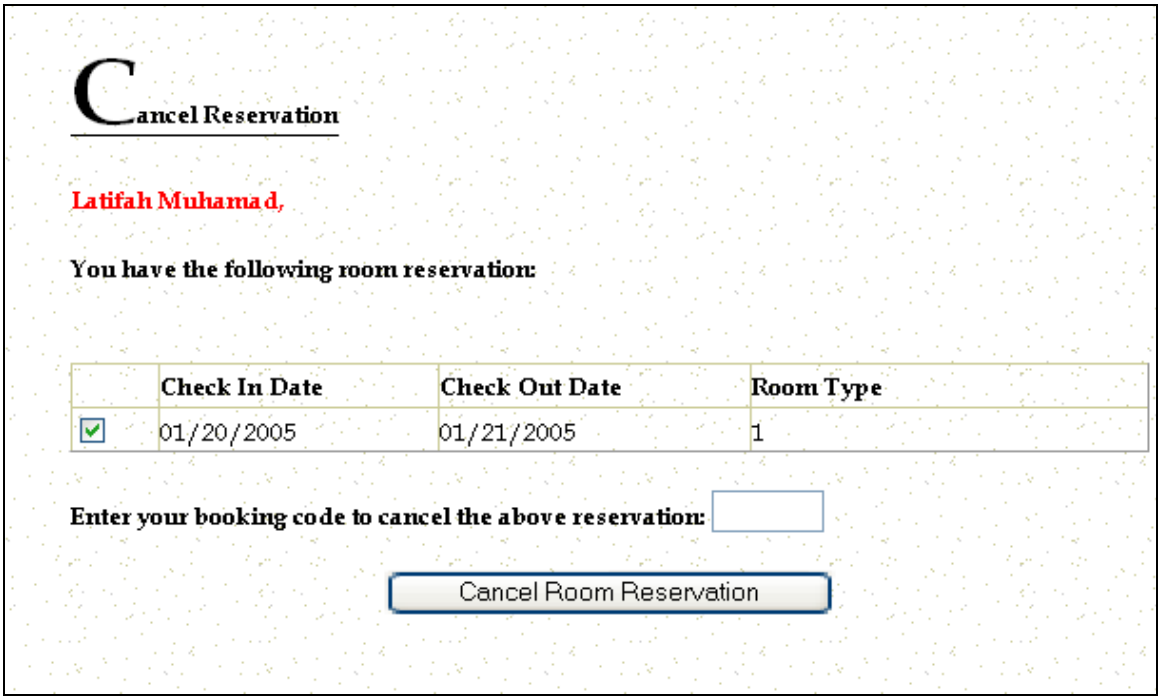


Figure A-23

In order to cancel the reservation, user need to provide the booking code for the corresponding room reservation. An error message will be displayed; if the booking code was not in 6 characters long (refer to Figure A-24).

Figure A-24

An error message will also be displayed, if the booking code input box was empty (refer to Figure A-25).

Figure A-25

An error message will also be displayed, if user entered the invalid booking code (refer to Figure A-26).

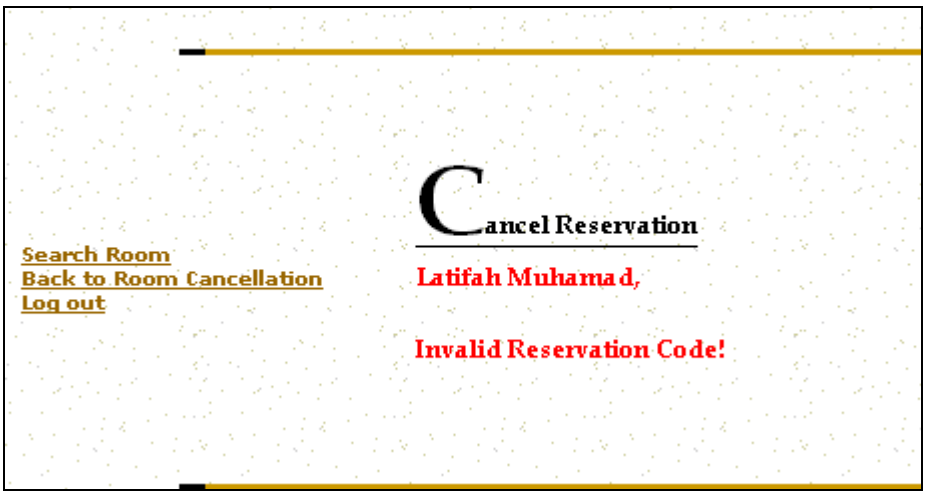


Figure A-26

User can switch back to the Room Cancellation page by choosing the menu Back to Room Cancellation at the left-hand side of the page (refer to Figure A-27).

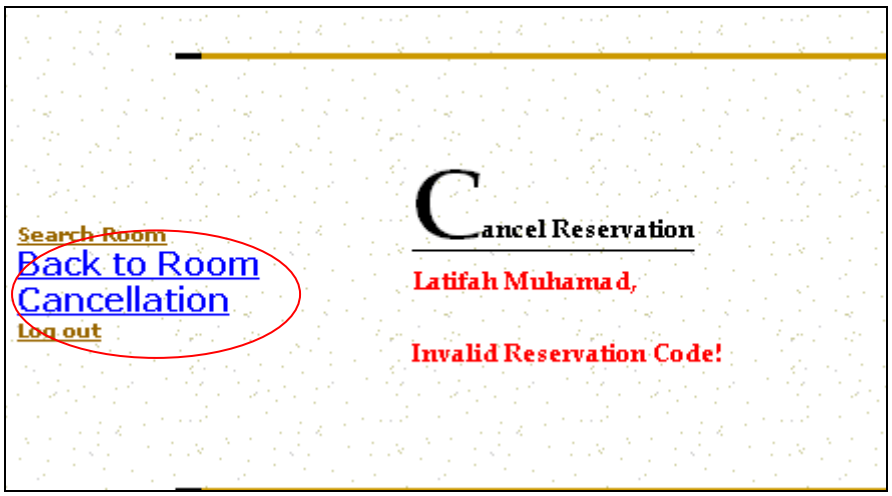


Figure A-27

When user entered the valid booking code, the reservation record will be deleted from the database (refer to Figure A-28).

Figure A-28

From this point, user can continue searching for another room or log out from the system. If user wishes to search for another room, use the menu Search Room. The searching procedure has been discussed (refer to Figure A-15). If user wishes to log out from the system, select the menu Log Out and user will be immediately directed to Home page (refer to Figure A-29).

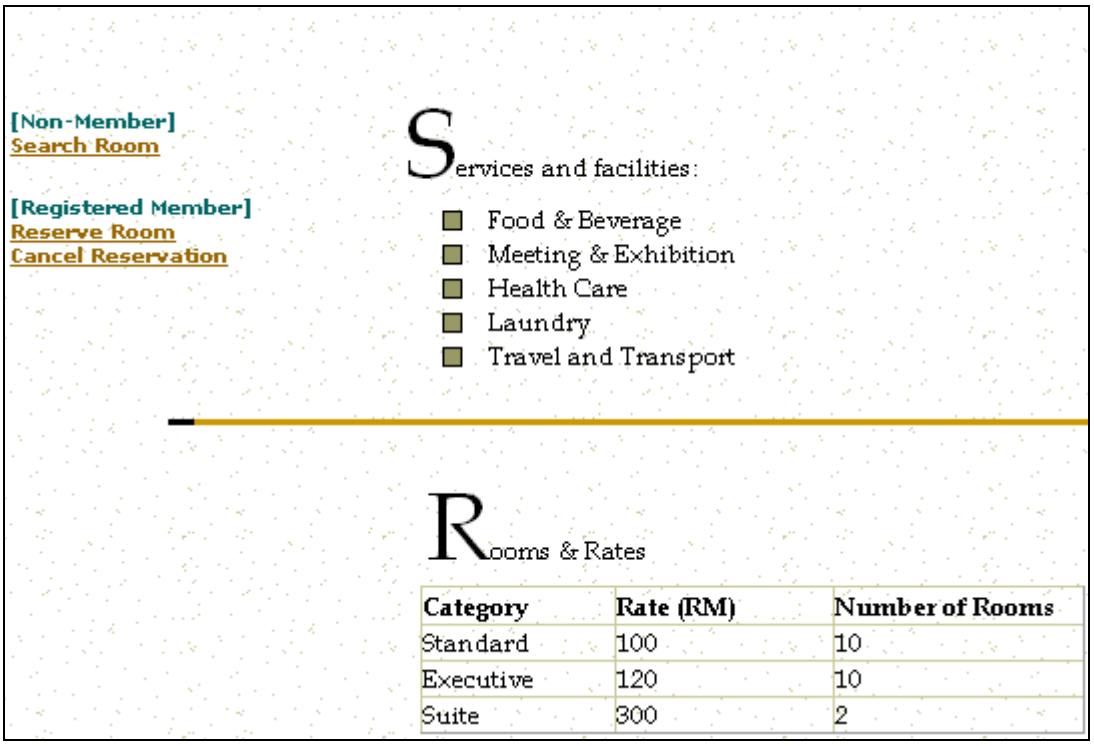


Figure A-29

3) Registered User

a. Reserve Room

A registered user can search or reserve room using the menu Reserve Room at the left-hand side of the Home page (refer to figure A-30).

Figure A-30

User will be directed to a log in page (refer to Figure A-31). User need to provide user ID and password.

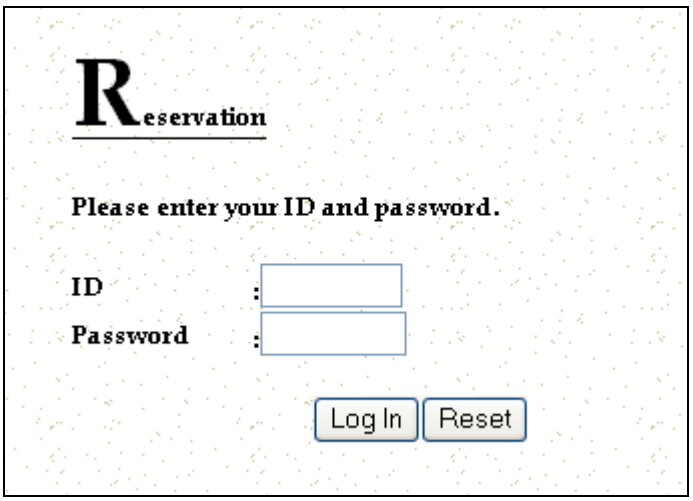


Figure A-31

Error message will be displayed if:

i. ID length is less than 5 characters (Figure A-32).
ii. Password length is less than 5 characters (Figure A-33).
iii. ID or password input box is empty (Figure A-34).
iv. Invalid ID or password (Figure A-35).

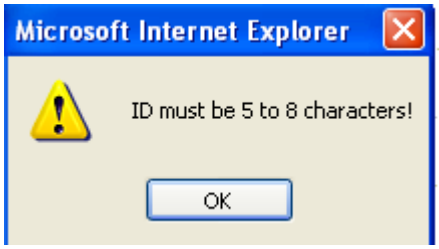


Figure A-32

Figure A-33

Figure A-34

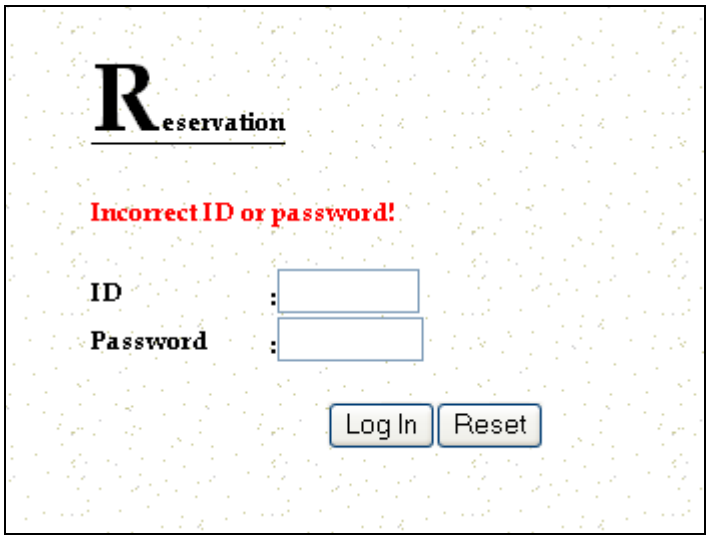


Figure A-35

User will be directed to the Room Searching page if the system provided with valid ID and password. User will get similar page as in Figure A-15.

b. Cancel Room Reservation

For the purpose of Room Cancellation, user needs to log in to the system. The similar procedure as Room Searching is applied to this process.

APPENDIX B:

Test Cases for W-HReS

Table 7.1: Test Cases for `Make Registration`

Table 7.2: Test Cases for `Log in`

No	Test Case Parameters				Associated Class	Coverage Element	
	ID	password	Submit	Reset	logInProcess	Status	Criterion
1	I_1_ ID	I_1_password	click	-	F	NA	AEM: (0,0): `User (0) –Login (0)` CA: `ID, password, Submit`
2	I_1_ ID	V_password	click	-	F	NA	AEM: (0,0): `User (0) –Login (0)` CA: `ID, password, Submit`
3	V_ ID	I_1_password	click	-	F	NA	AEM: (0,0): `User (0) –Login (0)` CA: `ID, password, Submit`
4	V_ ID	V_password	click	-	T	S	AEM: (1,1): `User (1) –Login (1)` CA: `ID, password, Submit`
Table matching							
5	I_2_ ID	I_1_password	click	-	T	US	AEM: (0,0):`User (0) –Login (0)` CA: `ID, password, Submit`
6	I_2_ ID	V_password	click	-	T	US	AEM: (0,0):`User (0) –Login (0)` CA: `ID, password, Submit`
7	V_ ID	I_1_password	click	-	T	US	AEM: (0,0):`User (0) –Login (0)` CA: `ID, password, Submit`
Click Reset Button							
8	V_ ID	V_password	-	click	F	NA	AEM: (0,0):`User (0) –Login (0)` CA: `ID, password, Reset`

Table 7.3: Test Cases for `Make Reservation`

No	Test Case Parameters			Associated Class	Coverage Element	
	roomType	**Submit (Reserve Room)**	**Reset**	**reserveProcess**	**Status**	**Criterion**
1	V_ roomType	click	-	T	S	AEM: (1,1): `User (1) –Reserve (1)` CA: `roomType, Submit`
2	V_2_ roomType	click	-	T	S	AEM: (1,2): `User (1) –Reserve(2)` CA: `roomType, Submit`
Click Reset Button						
3	V_ roomType	-	click	F	NA	AEM: (1,0): `User (1) –Reserve(0)` CA: `roomType, Reset`

Table 7.4: Test Cases for `Search for Room Availability`

No	Test Case Parameters					Associated Class	Coverage Element	
	checkInDate	**checkOutDate**	**numberOfGuest**	**Submit (Search Room)**	**Reset**	**searchResult**	**Status**	**Criterion**
1	I_ checkInDate	I_ checkOutDate	V_ numberOfGuest	click	-	F	NA	AEM: (1,0): `User (1) –Search Room(0)` CA: `checkInDate, checkOutDate, numberOfGuest, Submit`
2	I_ checkInDate	V_ checkOutDate	V_ numberOfGuest	click	-	F	NA	AEM: (1,0): `User (1) –Search Room(0)` CA: `checkInDate, checkOutDate, numberOfGuest, Submit`
3	V_ checkInDate	I_ checkOutDate	V_ numberOfGuest	click	-	F	NA	AEM: (1,0): `User (1) –Search Room(0)` CA: `checkInDate, checkOutDate, numberOfGuest, Submit`
4	V_ checkInDate	V_ checkOutDate	V_ numberOfGuest	click	-	T	S	AEM: (1,1): `User (1) –Search Room(1)` CA: `checkInDate, checkOutDate, numberOfGuest, Submit`
5	V_2_ checkInDate	V_2_ checkOutDate	V_2_ numberOfGuest	click	-	T	S	AEM: (1,2): `User (1) –Search Room(2)` CA: `checkInDate, checkOutDate, numberOfGuest, Submit`
Click Reset Button								
6	V_ checkInDate	V_ checkOutDate	V_ numberOfGuest	-	click	F	F	AEM: (1,0): `User (1) –Search Room(0)` CA: `checkInDate, checkOutDate, numberOfGuest, Reset`

Table 7.5: Test Cases for `Cancel Room Reservation`

No	Test Case Parameters			Associated Class	Coverage Element	
	bookingCode	**reservation**	**Submit (Cancel Room Reservation)**	**ConfirmCancellation**	**Status**	**Criterion**
1	I_ bookingCode	V_reservation	click	F	NA	AEM: (1,0): `User (1) – Cancellation(0)` CA: `bookingCode, reservation, Submit`
2	V_ bookingCode	V_reservation	click	T	S	AEM: (1,1): `User (1) – Cancellation(1)` CA: `bookingCode, reservation, Submit`
3	V_2_ bookingCode	V_2_reservation	click	T	S	AEM: (1,2): `User (1) – Cancellation(2)` CA: `bookingCode, reservation, Submit`
Table matching						
4	V_3_ bookingCode	V_reservation	click	F	US	AEM: (1,0): `User (1) – Cancellation(0)` CA: `bookingCode, reservation, Submit`