

[Travel Services Website]

SOFTWARE DESIGN SPECIFICATION

**Project Code: TSW**

**Document Code: <TSW>-DD – v<0.1>**

**<Ho Chi Minh City, 22 January 2015>**

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\* M, D | Change Description | New Version |
| Jan 22 | Create new | A | Create this document | 0.1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

SIGNATURE PAGE

**ORIGINATOR:** Nguyen Chi Kha January 22, 2015

Member

**REVIEWERS:** Vu Quang Minh

Leader

Do Van Manh

Member

**APPROVAL:** Nguyen Trong Tai

Instructor

TABLE OF CONTENTS

1 Introduction 6

1.1 Purpose 6

1.2 Definitions, Acronyms and Abbreviations 6

1.3 References 7

1.4 Overview 7

2 System Architecture 8

2.1 Application Logical Architecture 8

2.2 Use case view 9

2.2.1 Use case diagram 9

2.2.2 Description 9

2.3 Architecture Presentation 10

2.3.1 Presentation Layer 11

2.3.2 Business Layer 11

2.3.3 Data Access Layer 11

2.4 Packages/Component view 12

2.4.1 UI Components 12

2.4.2 Business Entities 13

2.4.3 Data Access Object 13

2.4.4 Exceptions Lib 13

2.4.5 General Lib 13

2.4.6 Logging Lib 13

3 Screen Flow 14

4 Database Design 15

4.1 Entity Relationship Diagram 15

4.2 Detail 15

5 Detail Functions Desgin 16

5.1 Tour Management 16

5.1.1 Create new tour 16

5.1.2 Update tour 18

5.1.3 Close tour 20

5.2 Profile Management 22

5.2.1 View profile 22

5.2.2 Edit profile 23

5.2.3 Change password 25

5.3 Tour Reservation 27

5.3.1 Search tour 27

5.3.2 View tour 29

5.3.3 Add tour to Cart 30

5.3.4 Compare tour 31

5.3.5 Book tour 32

5.3.6 Pay tour 33

5.3.7 Comment tour 34

5.3.8 Rate tour 35

5.4 Registration 36

5.4.1 Register 36

5.4.2 Login 38

5.4.3 Logout 40

6 Interface Design 43

6.1 43

6.2 43

7 Configuration 44

7.1 Application Configuration 44

7.2 System Configuration 44

8 Other Considerations 45

9 Appendix 46

# Introduction

## Purpose

This document provides a technical approach for implementing Travel Services Website application, using a number of different architectural diagrams to view different aspects of the system. It also presents the decisions that will govern how the application will be built from Spring MVC framework with Java programming perspective.

This document’s aim is to describe the architecture at high level, in which the system will be described as a set of functional layers, with technologies suggested for each layers.

## Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| Abbreviations | Description | Comment |
| TSW | Travel Services Website |  |
| PL | Presentation Layer |  |
| BLL | Business Logic Layer |  |
| DAL | Data Access Layer |  |
| DAO | Data Access Object |  |
| UC | Use Case |  |
| JSTL | Java Standard Tag Library |  |
| JSP | Java Server Page |  |
| HTML | Hypertext Mark-up Language |  |
| HTTP | Hypertext Transfer Protocol |  |
| MVC | Model – View – Control |  |
| WYSYWYG | What You See Is What You Get |  |
| JVM | Java Virtual Machine |  |
| J2EE | Java 2 Enterprise Edition |  |

## References

List all the reference document such as: other document of the system, or the technical article,...

|  |  |
| --- | --- |
| Document Number | Title |
|  |  |
|  |  |

## Overview

General overview of the detail design such as what is the structure of the document

# System Architecture

## Application Logical Architecture



## Use case view

### Overview diagram



### General User use case diagram



### Administrator use case diagram



### Customer use case diagram



### Agent use case diagram



## Architecture Presentation

The following diagram shows the primary tiers in the proposed n-tier architecture. This diagram show the main layers in this architecture and the vision of how they fit together.



### Presentation Layer

This layer controls the display to the end user. The development framework is based on Java MVC Model architecture. The framework is responsible for:

* Managing requests/responses from/to the clients.
* Controlling display to the end user.
* Performing UI validation.
* Handling exceptions from other layers.

### Business Layer

This layer manages the business processing rules and logic.

* Handling application business logic and business validation.
* Managing transactions.
* Allowing interfaces for interaction with other layers.
* Managing dependencies between business level objects.
* Adding flexibility between the presentation and the persistence layer so they do not directly communicate with each other.
* Exposing a context to the business layer from the presentation layer to obtain business services.
* Managing implementations from the business logic to the persistence layer.

### Data Access Layer

This layer manages access to persistent storage. The primary reason to separate data access from the rest of the application is that it is easier to switch data sources and share Data Access Objects (DAOs) between applications.

This layer manages reading, writing, updating, and deleting stored data.

## Packages/Component view



### UI Components

This package includes the implementation for the Java architecture proposed to be used in the Presentation Layer to handle the display to the end user.

**Validation**: All validation of incoming requests parameters to the server should be validated using Java client side control.

### Business Entities

This package includes the implementation of business objects. **Business Entities** (BE) layer is used to perform the business operations. The Business Entities layer will access the DAO to access database. Transactions should be managed within this business layer.

### Data Access Object

This package includes the implementation of Data Access Object. Using Hibernate classes here to make the application more flexible to access database. Hibernate classes includes basic functions to work with database: *select, insert, update, delete*.

### Exceptions Lib

This package will include all general exceptions that will typically use by more than one package. The try-catch clauses should be kept to a minimum.

### General Lib

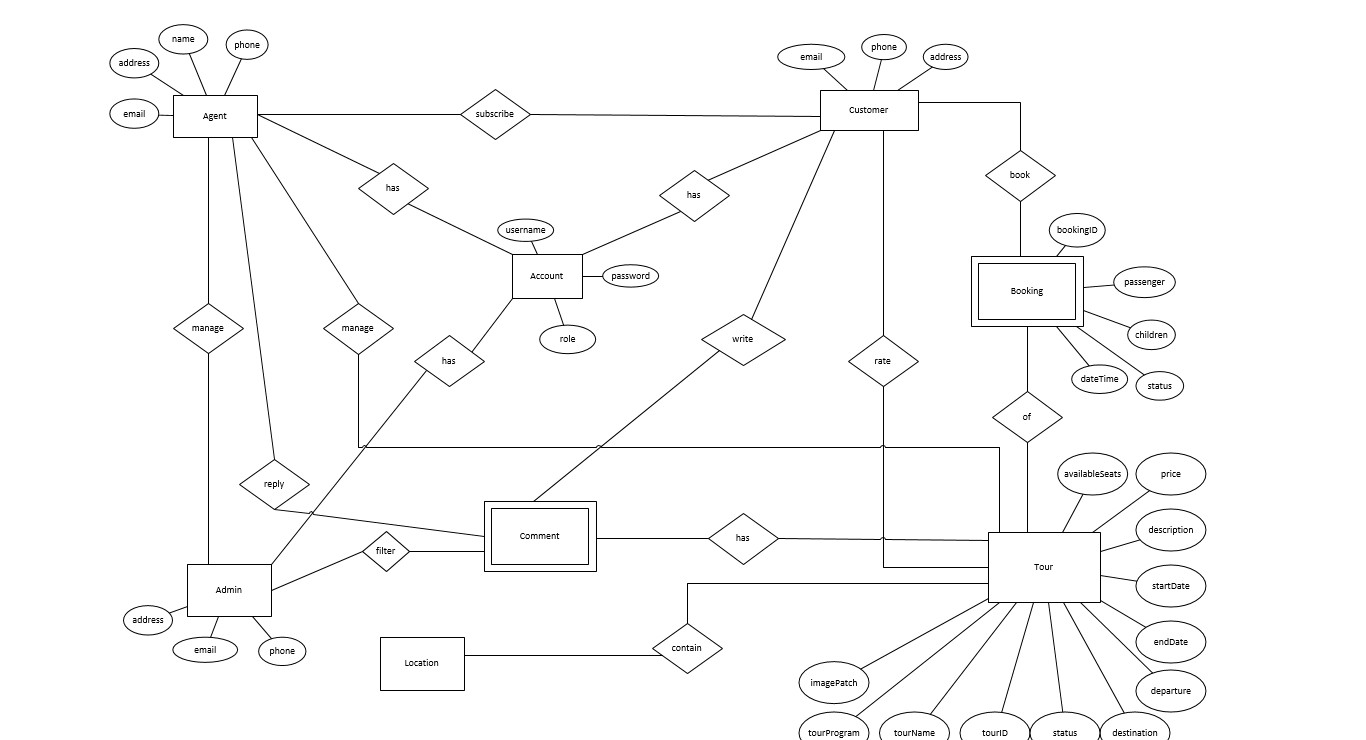
This package includes all utilities Java classes will be wisely used in the modules.

### Logging Lib

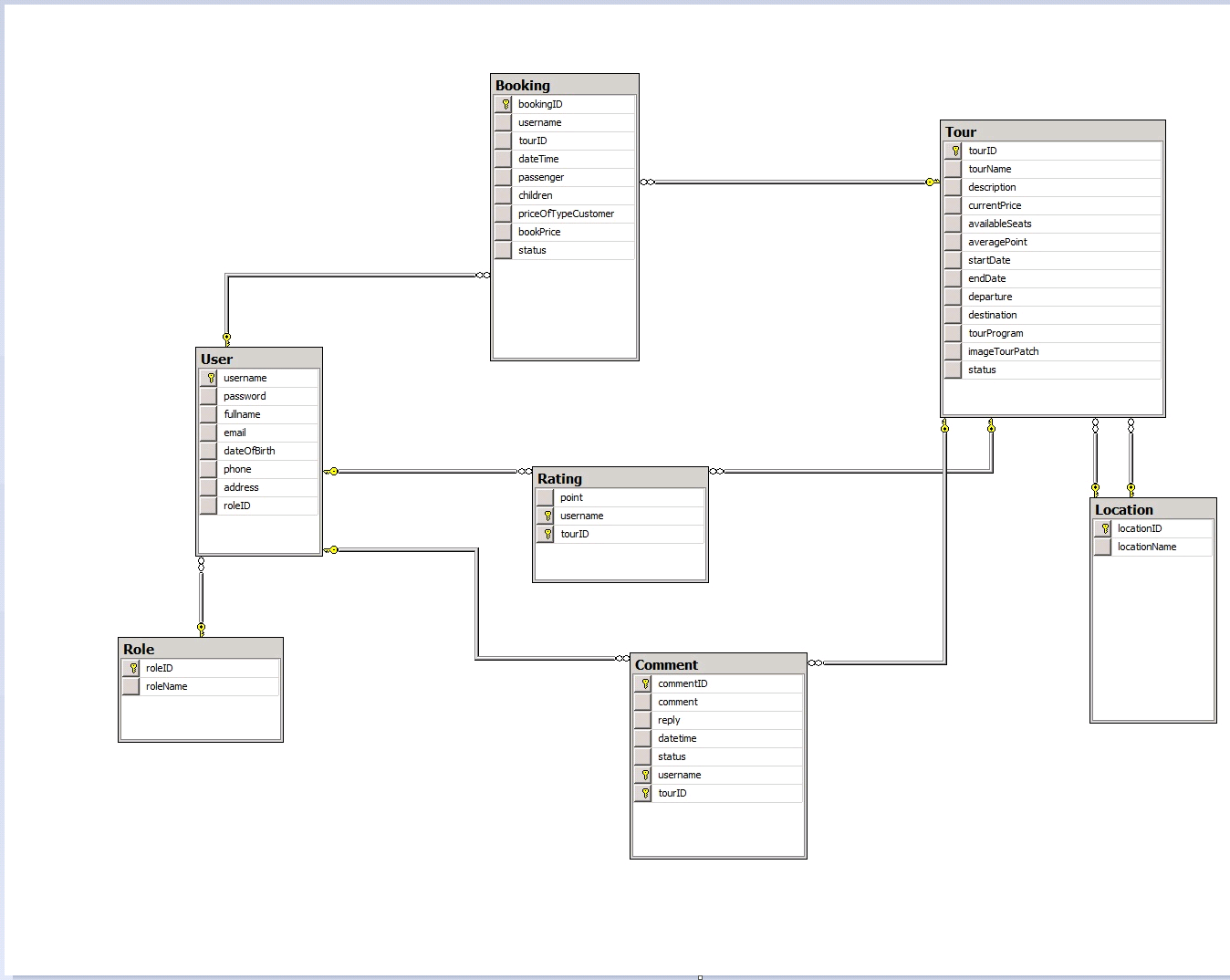
This package includes implemented logging classes.

# Database Design

## Entity Relationship Diagram



## Relational Database Diagram



# Detail Functions Desgin

Describe diagrams in system such as collaboration diagram, sequence diagram, activities diagram and state chart for some functionalities of the system.

## Registration & Login

### Register

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Register | |
| **Actor** | Guest | |
| **Description** | Allow guest to register an account | |
| **Goal** | Register an account | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | Guest clicks on “Register” button on the home page | |
| **Post-condition** | An account is registered  Redirect to home page and login new account | |
| **Normal flow** | **Actor request**  [Guest register account successfully]   1. Guest clicks on “Register” button on the home page 2. Guest fill out their information  * Username * Password & Confirm password * First name & Last name * Email * Phone number * Address * Etc…  1. User clicks on “Register” button | **System response**  [System register account successfully]   1. Display registration form. 2. Verify registration information 3. Register new account with filled information 4. Display registration successfully message and redirect to home page with new account. |
| **Alternative flow** |  |  |
| **Exception flow** | [When entered username is existed]  [When confirm password does not match entered password]  [When entered email does not exist]   1. Repeat steps in normal flow | [When entered username is existed]  [When confirm password does not match entered password]  [When entered email does not exist]   1. Notify error message to guest. |
| **Business rules** |  | |

### Login

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Login | |
| **Actor** | Guest | |
| **Description** | Allow guest to login to the system | |
| **Goal** | Login to the system | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | Guest clicks on “Login” button on the home page | |
| **Post-condition** | Account login successfully  Redirect to home page | |
| **Normal flow** | **Actor request**  [Guest login account successfully]   1. Guest clicks on “Login” button on the home page 2. Guest fill out their account  * Username * Password  1. User clicks on “Login” button | **System response**  [System login account successfully]   1. Display login form. 2. Verify login account 3. Display login successfully message and redirect to home page with new account. |
| **Alternative flow** | [When guest click “Remember username”] | [When guest click “Remember username”]   1. Create cookie to remember username |
| **Exception flow** | [When entered username is existed]  [When entered password is wrong]   1. Repeat steps in normal flow | [When entered username is existed]  [When entered password is wrong]   1. Notify error message to guest. |
| **Business rules** | Session’s timeout is 1 hour.  Cookie has timeout is 1 day. | |

### Logout

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Logout | |
| **Actor** | General user | |
| **Description** | Allow guest to logout the system | |
| **Goal** | Logout the system successfully | |
| **Pre-condition** | User’s account login to the system successfully | |
| **Trigger** | User clicks on “Logout” button | |
| **Post-condition** | Account logout successfully  Redirect to home page | |
| **Normal flow** | **Actor request**  [User logout account successfully]   1. User clicks on “Logout” button | **System response**  [System logout account successfully]   1. Logout account from system 2. Display logout successfully message and redirect to home page |
| **Alternative flow** |  |  |
| **Exception flow** |  |  |
| **Business rules** |  | |

### Retrieve password

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Retrieve password | |
| **Actor** | Guest | |
| **Description** | Allow guest to retrieve their password | |
| **Goal** | Retrieve user’s password | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | User clicks on “Forgotten your password?” link | |
| **Post-condition** | Redirect to login page | |
| **Normal flow** | **Actor request**  [Guest retrieve password successfully]   1. Guest clicks on “Forgotten your password?” link 2. Guest input their register email 3. Guest click “Retrieve password” button | **System response**  [System send email with password successfully]   1. Display retrieve password page. 2. Check input email. 3. Send email with reset password link. 4. Redirect to login page |
| **Alternative flow** |  |  |
| **Exception flow** | [Email is not existed]   1. Repeat normal steps. | [Email is not existed]   1. Notify error message for guest. |
| **Business rules** |  | |

## Profile Management

### View profile

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | View profile | |
| **Actor** | Customer | |
| **Description** | Allow a customer to view their profile | |
| **Goal** | View customer’s profile | |
| **Pre-condition** | Logged in to the system as a customer | |
| **Trigger** | Customer clicks on “Manage profile” button dropdown from customer’s username in the homepage | |
| **Post-condition** | Customer’s profile is displayed. | |
| **Normal flow** | **Actor request**  [Customer view profile successfully]   1. Customer clicks on “Manage profile” button dropdown from customer’s username in the homepage | **System response**  [System display customer’s profile successfully]   1. Show customer’s profile management page. |
| **Alternative flow** |  |  |
| **Exception flow** |  |  |
| **Business rules** | Display all customer’s information | |

#### **Activities Flow**



### Edit profile

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Edit profile | |
| **Actor** | Customer | |
| **Description** | Allow a customer to edit their profile | |
| **Goal** | Edit customer’s profile | |
| **Pre-condition** | Logged in to the system as a customer | |
| **Trigger** | Customer clicks on “Edit profile” button on navigation pane of customer’s profile management page | |
| **Post-condition** | Customer’s profile is edited.  Redirect to view customer’s profile page. | |
| **Normal flow** | **Actor request**  [Customer edit profile successfully]   1. Customer clicks on “Edit profile” button on navigation pane of customer’s profile management page 2. Customer edit their profile (Example: change avatar, email, address or phone number etc...) 3. Customer save their changes. | **System response**  [System update customer’s profile successfully]   1. Show edit customer’s profile page. 2. Verify new information 3. Update customer’s profile |
| **Alternative flow** | [Customer cancel edit profile process]   1. Customer click “Cancel” button. | [Customer cancel edit profile process]   1. Redirect to view profile page |
| **Exception flow** |  |  |
| **Business rules** |  | |

#### **Activities Flow**



### Change password

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Change password | |
| **Actor** | Customer | |
| **Description** | Allow a customer to change their password | |
| **Goal** | Change customer’s password | |
| **Pre-condition** | Logged in to the system as a customer | |
| **Trigger** | Customer clicks on “Change password” button on navigation pane of customer’s profile management page | |
| **Post-condition** | Customer’s password is changed. | |
| **Normal flow** | **Actor request**  [Customer change password successfully]   1. Customer clicks on “Change password” button on navigation pane of customer’s profile management page 2. Customer enter their current password, new password and confirm new password. | **System response**  [System update customer’s password successfully]   1. Show change customer’s password page. 2. Update customer’s password |
| **Alternative flow** |  |  |
| **Exception flow** | [Customer entered wrong current password]   1. Repeat steps in normal flow | [System failed to change password]   1. Clear all forms and notify error to the customer. |
| **Business rules** |  | |

#### **Activities Flow**



#### **Detail Processing**

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (4) | BR01 | **Check change password**   * Get old password of customer and compare with new password he/she have just entered * IF old password not match with new password THEN   + DISPLAY ‘Wrong password! Please enter again’ * Show change PIN screen again.   ELSE IF confirm password not match with new password THEN   * + DISPLAY ‘Confirm password does not match new password! Please enter again’ * Show change PIN screen again. |

## Tour Management

### Create tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Create tour | |
| **Actor** | Agent | |
| **Description** | Allow an agent to create a new tour and post it to the website. | |
| **Goal** | Create a tour | |
| **Pre-condition** | Logged in to the system as an agent  Tour’s Management page is displayed successfully | |
| **Trigger** | Agent clicks on “Create new tour” button on the Tour’s Management page. | |
| **Post-condition** | A tour is created.  This tour has been posted to the website where customer can view.  Redirect to new tour’s detail page | |
| **Normal flow** | **Actor request**  [Agent create tour successfully]   1. Agent click to “Create new tour” button on the Tour’s Management page. 2. Agent fill out the form. 3. Agent confirm | **System response**  [System receive new tour’s information and create tour successfully]   1. Show create new tour form. 2. Verify new tour’s information 3. Add new tour’s information to database and post it to the website. 4. Redirect to new tour’s detail page. |
| **Alternative flow** | [Agent click “Cancel” button]   1. Agent click “Cancel” | [System cancel process]   1. Cancel all process 2. Redirect to Tour’s Management page. |
| **Exception flow** | [This tour is existed]   1. Refill tour’s information | [This tour is existed]   1. Show error message to the agent. |
| **Business rules** | New tour has status is “Active” and post it to the home page where user can view. | |

#### **Activities Flow**



#### **Sequence Diagram**

### Update tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Update tour | |
| **Actor** | Agent | |
| **Description** | Allow an agent to update a tour’s information | |
| **Goal** | Update a tour | |
| **Pre-condition** | Logged in to the system as an agent  Tour’s Management page is displayed successfully | |
| **Trigger** | Agent clicks on “Update tour” button in-line of updating tour on the Tour’s Management page. | |
| **Post-condition** | A tour’s information is updated.  Redirect to updated tour’s detail page. | |
| **Normal flow** | **Actor request**  [Agent update tour successfully]   1. Agent click to “Update tour” button in-line of updating tour on the Tour’s Management page. 2. Agent update new tour’s information. 3. Agent confirm | **System response**  [System receive tour’s information and update tour successfully]   1. Show update tour’s information form. 2. Verify tour’s information 3. Update tour’s information to database. 4. Redirect to updated tour’s detail page. |
| **Alternative flow** | [Agent click “Cancel” button]   1. Agent click “Cancel” | [System cancel process]   1. Cancel all process 2. Redirect to Tour’s Management page. |
| **Exception flow** |  |  |
| **Business rules** | Updating tour has status is “Block” and change to “Active” when complete update process. | |

#### **Activities Flow**



#### **Sequence Diagram**

### Change status

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Change status | |
| **Actor** | Agent | |
| **Description** | Allow an agent to change a tour’s status | |
| **Goal** | Change a tour’s status | |
| **Pre-condition** | Logged in to the system as an agent  Tour’s Management page is displayed successfully | |
| **Trigger** | Agent click to “Status” drop-down list in-line of tour on the Tour’s Management page. | |
| **Post-condition** | A tour’s status is updated.  Redirect to tour’s detail page. | |
| **Normal flow** | **Actor request**  [Agent change tour’s status successfully]   1. Agent click to “Status” drop-down list in-line of tour on the Tour’s Management page. 2. Agent select new tour’s status. 3. Agent save their changes. | **System response**  [System change tour’s status successfully]   1. Show tour’s status selection. 2. Ask for saving changes. 3. Update tour’s status to database. |
| **Alternative flow** | [Agent click “Cancel” button]   1. Agent click “Cancel” | [System cancel process]   1. Cancel all process |
| **Exception flow** |  |  |
| **Business rules** |  | |

### Delete tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Delete tour | |
| **Actor** | Agent | |
| **Description** | Allow an agent to delete a tour | |
| **Goal** | Delete a tour | |
| **Pre-condition** | Logged in to the system as an agent  Tour’s Management page is displayed successfully | |
| **Trigger** | **Option 1**. Agent clicks on “Delete tour” button in-line of updating tour on the Tour’s Management page.  **Option 2**. Agent check in box at first line of closing tour, then select “Delete selected tours” on the Tour’s Management page. | |
| **Post-condition** | A tour is deleted. | |
| **Normal flow** | **Actor request**  [Agent close tour successfully]   1. Agent click to “Close tour” button in-line of updating tour or check in box at first line of closing tour, then select “Close selected tours” on the Tour’s Management page. 2. Agent confirm | **System response**  [System close tour successfully]   1. Show message to notice about closing tour activities. 2. Update tour’s information to database. |
| **Alternative flow** | [Agent click “Cancel” button]   1. Agent click “Cancel” | [System cancel process]   1. Cancel all process |
| **Exception flow** |  |  |
| **Business rules** |  | |

#### **Activities Flow**



## Tour Reservation

### Search

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Search | |
| **Actor** | General user | |
| **Description** | Allow user to search with keyword | |
| **Goal** | Search with keyword | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | User clicks on “Search” button on the home page | |
| **Post-condition** | Search result is displayed successfully. | |
| **Normal flow** | **Actor request**  [User search successfully]   1. User input their keyword to search 2. User clicks on “Search” button | **System response**  [System display search result successfully]   1. Search with specified keyword. 2. Display search result. |
| **Alternative flow** |  |  |
| **Exception flow** | [When no result is available for this keyword] | [When no result is available for this keyword]   1. Notify error for user. |
| **Business rules** | Search in index first.  Search result sorted from newest to oldest.  Display 10 result each page.  Error message is “Sorry! There’re no result for “keyword”. Please try another one” | |

#### **Activities Flow**



#### **Sequence Diagram**



### Search tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Search tour | |
| **Actor** | General user | |
| **Description** | Allow user to search for tours | |
| **Goal** | Search result will be shown to user | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | User clicks on “Search tour” button on the home page | |
| **Post-condition** | Search result is displayed successfully. | |
| **Normal flow** | **Actor request**  [User search tour successfully]   1. User specifies their trip’s information (more detailed information filling, more accurate search result)  * Start location and destination * Expecting price * Time and duration * Passengers (with or without children, how many?)  1. User clicks on “Search tour” button | **System response**  [System display search result successfully]   1. Search tour with specified information and display search result. |
| **Alternative flow** | [When no tour is available on the selected date]   1. Repeat steps in normal flow | [When no tour is available on the selected date]   1. Show option to select another day |
| **Exception flow** | [When no tour is available on the selected location] | [When no tour is available on the selected location]   1. Display error message. |
| **Business rules** | Search in index first.  Search result sorted tour from newest to oldest.  Display 10 tour result each page. | |

#### **Activities Flow**



#### **Detail Processing**

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (4) | BR01 | **Verify search result**   * IF there are no tour in user’s specified date THEN   + DISPLAY ‘Sorry! There are no tour in this date, please select a different time’ * Show select date option. |

### View tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | View tour | |
| **Actor** | General user | |
| **Description** | Allow user to view tour’s detail | |
| **Goal** | View tour’s detail information | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | User clicks on “Detail” link on tour’s model | |
| **Post-condition** | Tour’s detail page is displayed successfully. | |
| **Normal flow** | **Actor request**  [User view tour’s detail successfully]   1. User clicks on “Detail” link on tour’s model | **System response**  [System display tour’s detail successfully]   1. Display tour’s detail information |
| **Alternative flow** |  |  |
| **Exception flow** |  |  |
| **Business rules** |  | |

### View agent

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | View agent | |
| **Actor** | General user | |
| **Description** | Allow user to view agent’s detail | |
| **Goal** | View agent’s detail information | |
| **Pre-condition** | Home page is displayed successfully | |
| **Trigger** | User clicks on agent’s name. | |
| **Post-condition** | Agent’s detail page is displayed successfully. | |
| **Normal flow** | **Actor request**  [User view agent’s detail successfully]   1. User clicks on agent’s name. | **System response**  [System display agent’s detail successfully]   1. Display agent’s detail information |
| **Alternative flow** |  |  |
| **Exception flow** |  |  |
| **Business rules** |  | |

### Compare tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Compare tour | |
| **Actor** | General user | |
| **Description** | Allow user to compare tours | |
| **Goal** | Compare result will be shown to user | |
| **Pre-condition** | Home page is displayed successfully  There are at least 2 tours in the cart | |
| **Trigger** | User clicks on “Cart” icon on the home page | |
| **Post-condition** | Cart is displayed successfully. | |
| **Normal flow** | **Actor request**  [User compare tour successfully]   1. User clicks on “Cart” icon on the home page | **System response**  [System display compare result successfully]   1. Compare tours in cart by each attribute |
| **Alternative flow** | [When only 1 tour is available in the cart] | [When only 1 tour is available in the cart]   1. Display user’s cart by normal |
| **Exception flow** | [When no tour is available in the cart] | [When no tour is available in the cart]   1. Notify to the user about their empty cart |

### Book tour

#### **Use Case Specification**

|  |  |  |
| --- | --- | --- |
| **Name** | Book tour | |
| **Actor** | Customer | |
| **Description** | Allow customer to book tour | |
| **Goal** | Book a tour | |
| **Pre-condition** | Logged in to the system as a customer | |
| **Trigger** | User clicks on “Book” button | |
| **Post-condition** | A tour is booked.  Number of available passengers is reduced | |
| **Normal flow** | **Actor request**  [User book tour successfully]   1. Customer specifies passenger’s details 2. Customer click “Book” button 3. Customer confirm transaction | **System response**  [System display compare result successfully]   1. Check for available passengers in this tour. 2. Display tour’s price (included taxes, insurance fee etc…) 3. Reduce number of available passengers for this tour. 4. Send booking information to agent managing this tour. |
| **Alternative flow** | [When customer cancel transaction]   1. Customer click “Cancel” button | [When customer cancel transaction]   1. Abort transaction. 2. Redirect to home page. |
| **Exception flow** | [When available passengers is not enough] | [When available passengers is not enough]   1. Notify to the customer |
| **Business rules** | Confirmation time is 15 minutes.  To solve conflict, customer who book earlier will have the priority. | |

#### **Activities Flow**



#### **Sequence Diagram**



# Interface Design

## 

## 

# Configuration

## Application Configuration

|  |  |  |
| --- | --- | --- |
| Field | Value | Remark |
|  |  |  |
|  |  |  |
|  |  |  |

## System Configuration

|  |  |  |
| --- | --- | --- |
| Field | Value | Remark |
| Date | dd/MM/yyyy |  |
| Time | hh:mm:ss |  |
| Money | ##,###,### |  |

# Other Considerations

<[This section provides a description of other design elements that were considered as alternatives in selection process for the above class design, i.e. a brief explanation of advantages and disadvantages of the selected package relationships and/or class implementation in comparison with others. It should be a clear answer to the question why the above class design is selected for this system, not the others.>

# Appendix