|  |  |
| --- | --- |
| **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Drop It |
|  |
| |  |  | | --- | --- | | **Team 08** | | | **Group Members** | Lê Khôi Phong 1 60473  Đào Như Tùng 2 60408  Công Minh Hiếu 3  Nguyễn Thanh Tùng 4 60513  Đặng Quốc Duy 4 | | **Supervisor** | Lâm Hữu Khánh Phương | | **Ext Supervisor** | None | | **Capstone Project code** | DropIt | |
|  |

-Ho Chi Minh City, 08/2013-

Table of Contents

Table of Contents 2

1.1 Introduction 7

1.1.1 Project Information 7

1.1.2 Purpose 7

1.1.3 The People 7

1.1.4 Background 7

1.1.5 Problems 8

1.1.6 Our Proposal 8

1.1.7 Benefits 8

2.1 Problem Definition 9

2.1.1 Name of this Capstone Project 9

2.1.2 Problem Abstract 9

2.1.3 Project Overview 9

2.1.3.1 The Current System 9

2.1.3.2 The Proposed System 9

2.1.3.3 Boundaries of the System 10

2.1.3.4 Development Environment 10

2.2 Project Organization 11

2.2.1 Software Process Model 11

2.2.2 Roles and Responsibilities 11

2.2.3 Tools and Techniques 12

2.3 Project management Plan 13

2.3.1 Tasks 13

2.3.2 Task Sheet: Assignment and Timetable 14

2.4 Coding Convention 17

2.4.1 Naming Convention 17

2.4.2 Lengths 17

3.1 User Requirement Specification 18

3.1.1 Common Features 18

3.1.2 Authorized User 18

3.1.3 Admin 18

3.1.4 Guest 18

3.2 System Requirement Specification 19

3.2.1 External Interface Requirements 19

3.2.1.2 User Interfaces 19

3.2.1.2 Hardware Interfaces 19

3.2.1.3 Software Interfaces 19

3.2.1.4 Communications Protocol 19

3.2.2 System Features 20

3.2.2.1 Admin 21

3.2.2.2 Customer 56

3.2.2.3 Authorized User 75

3.2.2.4 Guest 81

3.2.2.5 Unauthorized User 85

3.2.2.6 Auto System 87

3.2.3 Software System Attributes 92

3.2.1.2 Reliability 92

3.2.3.2 Availability 92

3.2.3.3 Security 92

3.2.3.6 Maintainability 92

3.2.3.6 Portability 92

3.2.3.6 Performance 93

3.3 Entity Relationship Diagram 93

## Introduction

### Project Information

* Project name: **Drop-It**
* Project code: **DIC2C**
* Product type: **Website**
* Timeline: **From 14 May 2013 To 14 Aug 2013**

### Purpose

This project is developed and registered as the capstone project for group number 8. The purpose is fulfilling the requirement from FPT University training program.

### The People

**Supervisor:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Full name** | **Phone** | **E-Mail** | **Title** |
| Lâm Hữu Khánh Phương | 0915353001 | phuonglhk@fpt.edu.vn | Teacher |

**Team members:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Student** | **Full name** | **Student code** | **Phone** | **E-Mail** | **Role in Group** |
| 1 | Lê Khôi Phong | 60473 | 0903045572 | phonglk60473@fpt.edu.vn | Leader |
| 2 | Đào Như Tùng | 60408 | 0909364008 | tungdn60408@fpt.edu.vn | Member |
| 3 | Công Minh Hiếu | 60535 | 0918125726 | hieucm60535@fpt.edu.vn | Member |
| 4 | Nguyễn Thanh Tùng | 60513 | 0934779004 | tungnt60513@fpt.edu.vn | Member |
| 5 | Đặng Quốc Duy | 00276 | 0943578179 | duydq00276@fpt.edu.vn | Member |

### Background

In addition to its economic growth, HCMC's cultural and artistic activities have been on the rise to form the habit of enjoying arts every night in the local residents. Together with technological demands and developments, a large number of online banking services for business transactions and ticket selling sprung up, making it more convenient for the buyers. However, whether the transactions are online or offline, there are always situations in which the ticket buyers cannot use the tickets and would have to find a way to resell the tickets to other people, often at a lower price. They can either advertise their tickets online or sell the tickets on the spot of the events.

### Problems

Most of online ticket selling businesses have these two common characteristics:

* Generality - Different types of tickets are sold and those tickets are not inclined towards any particular artistic activities.
* Most of them are B2C businesses, direct selling and are the distributors of programs to the end users.

Most of the Classified advertising websites are also general in nature and low in efficiency

Moreover, reliability and online security should be taken note of because those transactions are likely to have high risks.

### Our Proposal

DropIt is born with an aim to become a safe and convenient platform for users to resell their tickets. Although the market for second-hand tickets are smaller than that of first-hand tickets, second-hand tickets hold many potentials if DropIt can tap on to this and create a safe and anti-phish platforms.

Main Features:

* Allowing users to publish and advertise the tickets they want to resell
* Allowing users to buy second-hand tickets
* Holding money until buyer confirm that ticket is ok
* Keeping statistics and transaction records

### Benefits

* Sellers can have their tickets resold to the desired customers and are guaranteed to receive the money from the transaction.
* Buyers are able to buy their desired tickets, and only lose the money upon receiving the desired tickets.

## Problem Definition

### Name of this Capstone Project

Project Full name: **Drop It**

Project Code: **DIC2C**

### Problem Abstract

In addition to its economic growth, HCMC's cultural and artistic activities have been on the rise to form the habit of enjoying arts every night in the local residents. Together with technological demands and developments, a large number of online banking services for business transactions and ticket selling sprung up, making it more convenient for the buyers. However, whether the transactions are online or offline, there are always situations in which the ticket buyers cannot use the tickets and would have to find a way to resell the tickets to other people, often at a lower price. They can either advertise their tickets online or sell the tickets on the spot of the events.

### Project Overview

#### The Current System

Most of online ticket selling businesses have these two common characteristics:

* Generality - Different types of tickets are sold and those tickets are not inclined towards any particular artistic activities.
* Most of them are B2C businesses, direct selling and are the distributors of programs to the end users.
* Most of the Classified advertising websites are also general in nature and low in efficiency
* Moreover, reliability and online security should be taken note of because those transactions are likely to have high risks.

#### The Proposed System

DropIt is born with an aim to become a safe and convenient platform for users to resell their tickets. Although the market for second-hand tickets are smaller than that of first-hand tickets, second-hand tickets hold many potentials if DropIt can tap on to this and create a safe and anti-phish platforms.

Main Features:

* Allowing users to publish and advertise the tickets they want to resell
* Allowing users to buy second-hand tickets
* Holding money until buyer confirm that ticket is ok
* Keeping statistics and transaction records

#### Boundaries of the System

The system under development of this Capstone Project will include:

* Based on web application
* Real payment is not included in this version
* Real event also is not included in this version

#### Development Environment

Below is the list of hardware and software requirements needed for development environments:

**Hardware requirements:**

* Personal computers for developing with the minimum configuration: CPU Core 2 Duo 2.0GHz, 2GB of RAM, 30GB of hard disk, and internet.

**Software requirements:**

* Operating system: Windows 7
* Web Server: IIS
  + Microsoft Windows 7: operating system and platform for development
  + Microsoft SQL Server 2012 Express: used to create and manage the database for system
  + Assembla: used to control source code ,documents and task management project
  + IIS: web server
  + Idea Software Modeler: used to create models and diagrams
  + Microsoft Project 2010: used to manage process and work schedules.
  + Skype: used for communication and meeting
  + DBMS: Microsoft SQL Server 2012
* Source Control: Tortoise Subversion (SVN) and Assembla code server

## Project Organization

### Software Process Model



**Figure 2.2.1.1 - Agile Development Model**

All of the phases in the implemented modal are included in the scope of this project and are the responsibilities of the team.

### Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Lam Huu Khanh Phuong | Supervisor | * Tracking & managing progress * Advising Idea & solutions * Suggesting & supporting in technologies |
| 2 | Le Khoi Phong | Team Leader | * Tracking & managing progress * Designing database * Creating coding framework * Analyzing requirements * Planning & scheduling * Coding * GUI Design * Testing * Writing documents & reports |
| 3 | Dao Nhu Tung | Team member | * Analyzing requirements * Coding * Writing documents * Testing |
| 4 | Cong Minh Hieu | Team member | * Analyzing requirements * Coding * Writing documents * Testing |
| 5 | Nguyen Thanh Tung | Team member | * Analyzing requirements * Coding * Writing documents * Testing |
| 6 | Dang Quoc Duy | Team member | * Analyzing requirements * Coding * Writing documents * Testing |

Table 1: Roles and Responsibilities

### Tools and Techniques

**Tools:**

* *Microsoft Visual Studio 2012*: Used to implement software modules.
* *Microsoft SQL server 2008 R2 Express*: Used as the database of the system.
* *Microsoft Excel*: For the team leader to manage tasks of the members and the progress of the project.
* *Microsoft Project 2010*: for team tracking
* *TortoiseSVN*: Control Source code of the whole project.
* *VisualSVN*: extension for using subversion (SVN) inside Visual Studio.
* *Assembla:* SVN Repository
* *Idea Software Modeler:* Draw Diagram and Use case
* *moqups.com:* for prototyping
* *Crystal Report 13:*for reporting function
* *Google Chrome, Firefox*: Used to test the system

**Technologies:**

* ASP.NET MVC 4
* LINQ
* HTML 5, CSS 3, AJAX, jQuery, Knockout, Lesscss, coffescript, Bootstrap

## Project management Plan

### Tasks

|  |  |
| --- | --- |
| **Scope Study and Technology** | |
| Description | General requirements analysis, technology & business process study |
| Deliverables | The feasibility report and decisions for the project |
| Resources Needed | 25 man-days |
| Dependencies and Constraints | N/A |
| Risks | * The project or the chosen technology is not feasible. * Team members don’t |
| **Documentation and review** | |
| Description | Create all the necessary documents for research and delivery |
| Deliverables | 1. Project Management Plan (PMP) 2. Software Requirements Specification (SRS) 3. Create Software Architecture Design (SAD) 4. Software Design Description (SDD) 5. Software Test Documentation (STD) 6. Software User’s Manual (SUM) |
| Resources Needed | FPT templates,  75 man-day |
| Dependencies and Constraints | Follow FPT templates |
| Risks | * Not follow FPT Templates * Team members lack experience in creating documents, * Bootle-Neck in review because lack of resources that have experience in review * Requirements are changing so quickly, CR appears |
| **GUI design and implementation** | |
| Description | Design user interface |
| Deliverables | Prototype (in HTML & PNG format) |
| Resources Needed | 25 man-days |
| Dependencies and Constraints | Web Application |
| Risks | * CR appears frequently * Lack of Desinger and HCI experience * Not all team members can design front-end (2/4 Members can design) |
| **Implementation** | |
| Description | Create the executable files |
| Deliverables | Solution (SLN) with all needed project files. |
| Resources Needed | Visual Studio 2012, .NET framework 4.5, Web browsers  200 man-days |
| Dependencies and Constraints | * All Teammembers should understand their module precisely * All Teammember have to experience the MVC4.0, and EF5 |
| Risks | * Team members don’t have experience with Technologies * Some function and modules are underestimate so can keep the deadline * Coding not follow conventions. |
| **Release and deployment** | |
| Description | Release the complete application and deploy it on the server |
| Deliverables | DropIt installation file and all the related documents (SUM) |
| Resources Needed | Installation package  10 man-days |
| Dependencies and Constraints | Meet the user requirements |
| Risks | * Can’t baseline and release on time |
| **Quality control** | |
| Description | Testing application’s performance and usability |
| Deliverables | STD |
| Resources Needed | QA, testers (team member),members), FPT template test case  75 man-days |
| Dependencies and Constraints | Follow FPT template test case |
| Risks | * Bug rate is very high * Some bugs can detect from Implementation by Perform UT not by IT * Test cases doesn’t meet the LOCs |
| **Human resource management** | |
| Description | Manage human resource, task assignments and member’s performances |
| Deliverables | Project Task List – Assignment Table sheet |
| Resources Needed | Project Task List |
| Dependencies and Constraints | N/A |
| Risks | * First time teamwork together * Communication Risk |

### Task Sheet: Assignment and Timetable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task Name | Duration | Start | Finish | Resource Names |
| **Initiating** | **0.5 days** | **Fri 5/3/13** | **Fri 5/3/13** |  |
| **Planning** | **2 days** | **Sat 5/4/13** | **Tue 5/7/13** |  |
| Prepare project introduction | 1 day | Sat 5/4/13 | Sat 5/4/13 |  |
| Develop Software Management Plan | 2 days | Sun 5/5/13 | Mon 5/6/13 |  |
| Review Project Plan | 1 day | Tue 5/7/13 | Tue 5/7/13 |  |
| **Analyzing** | **7 days** | **Wed 5/8/13** | **Thu 5/16/13** |  |
| Define Problem | 2 days | Wed 5/8/13 | Thu 5/9/13 |  |
| Analyzing Ticket.st | 5 days | Fri 5/10/13 | Thu 5/16/13 |  |
| **Mockup & HTML** | **10 days** | **Fri 5/17/13** | **Thu 5/30/13** |  |
| Seller's functions mockup | 3 days | Fri 5/17/13 | Tue 5/21/13 | N.Tùng |
| Buyer's functions mockup | 3 days | Fri 5/17/13 | Tue 5/21/13 | Hiếu |
| User's function mockup | 3 days | Fri 5/17/13 | Tue 5/21/13 | T.Tùng |
| Seller's functions HTML | 7 days | Wed 5/22/13 | Thu 5/30/13 | N.Tùng,Phong |
| Buyer's functions HTML | 7 days | Wed 5/22/13 | Thu 5/30/13 | Hiếu,Phong |
| User's function HTML | 7 days | Wed 5/22/13 | Thu 5/30/13 | T.Tùng |
| **Design** | **11 days** | **Wed 5/22/13** | **Thu 6/6/13** |  |
| ERD Design | 0 days | Wed 5/22/13 | Wed 5/22/13 | N.Tùng |
| Database design | 7 days | Thu 5/23/13 | Fri 5/31/13 | Team |
| Graphic design | 11 days | Thu 5/23/13 | Thu 6/6/13 | Phong |
| **Coding** | **20 days** | **Mon 6/3/13** | **Fri 6/28/13** |  |
| **Buyer's functions** | **20 days** | Mon 6/3/13 | **Fri 6/28/13** | **Hiếu,Phong** |
| Create Basic Design (Architecture Design) | 6 days | Mon 6/3/13 | Mon 6/10/13 |  |
| Review Basic Design | 2 days | Mon 6/10/13 | Tue 6/11/13 |  |
| Create Detail Design | 4 days | Wed 6/12/13 | Mon 6/17/13 |  |
| Review Detail Design | 1 day | Tue 6/18/13 | Tue 6/18/13 |  |
| Coding | 5 days | Wed 6/19/13 | Tue 6/25/13 |  |
| Coding Review | 1 day | Wed 6/26/13 | Wed 6/26/13 |  |
| Unit Testing | 2 days | Thu 6/27/13 | Fri 6/28/13 |  |
| **Seller's functions** | **20 days** | **Mon 6/3/13** | **Fri 6/28/13** | **N.Tùng,Phong** |
| Create Basic Design (Architecture Design) | 6 days | Mon 6/3/13 | Mon 6/10/13 |  |
| Review Basic Design | 2 days | Mon 6/10/13 | Tue 6/11/13 |  |
| Create Detail Design | 4 days | Wed 6/12/13 | Mon 6/17/13 |  |
| Review Detail Design | 1 day | Tue 6/18/13 | Tue 6/18/13 |  |
| Coding | 5 days | Wed 6/19/13 | Tue 6/25/13 |  |
| Coding Review | 1 day | Wed 6/26/13 | Wed 6/26/13 |  |
| Unit Testing | 2 days | Thu 6/27/13 | Fri 6/28/13 |  |
| **User's functions** | **20 days** | **Mon 6/3/13** | **Fri 6/28/13** | **T.Tùng,Duy,Phong** |
| Create Basic Design (Architecture Design) | 6 days | Mon 6/3/13 | Mon 6/10/13 |  |
| Review Basic Design | 2 days | Mon 6/10/13 | Tue 6/11/13 |  |
| Create Detail Design | 4 days | Wed 6/12/13 | Mon 6/17/13 |  |
| Review Detail Design | 1 day | Tue 6/18/13 | Tue 6/18/13 |  |
| Coding | 5 days | Wed 6/19/13 | Tue 6/25/13 |  |
| Coding Review | 1 day | Wed 6/26/13 | Wed 6/26/13 |  |
| Unit Testing | 2 days | Thu 6/27/13 | Fri 6/28/13 |  |
| **Testing** | **30 days** | **Mon 7/1/13** | **Fri 8/9/13** | **Team** |
| Project Scope Studying | 5 days | Mon 7/1/13 | Fri 7/5/13 |  |
| Create Test View Point Document | 5 days | Sun 7/14/13 | Thu 7/18/13 |  |
| Review Test Viewpoint Document | 1 day | Fri 7/19/13 | Fri 7/19/13 |  |
| Create Integration Test Case Document | 7 days | Sat 7/20/13 | Mon 7/29/13 |  |
| Review Integration Test Case Document | 1 day | Tue 7/30/13 | Tue 7/30/13 |  |
| Perform Test | 7 days | Wed 7/31/13 | Thu 8/8/13 |  |
| Review Test Report | 1 day | Thu 8/8/13 | Thu 8/8/13 |  |
| Deliverable: Test Report | 0 days | Fri 8/9/13 | Fri 8/9/13 |  |
| Monitoring and Controlling | 57.04 days | Wed 5/22/13 | Fri 8/9/13 | Phong |

## Coding Convention

### Naming Convention

* Private Fields: underscore followed by lowerCamelCase. ( ex: int \_privateField)
* Non-private Fields and properties: UpperCamelCase. (ex: int PublicField)
* Local variables: lowerCamelCase. (ex: int localVariable)
* Do not use consecutive underscores in name.
* Do not use Hungarian style.

### Lengths

* Public name should not be longer than 32 characters or 7 words.
* Methods should contain no more than 70 lines of code (if it is, it must be divided into methods).
* Methods should contain no more than 5 levels of indentation (if it is, it must be divided into methods).
* A line of code should contain no more than 80 characters.

## User Requirement Specification

### Common Features

* Only authenticated users can access the system. Users can log in and log out using their own accounts
* User can change password
* All guests or users can view event. And they can search for the event or artist, and view ticket or request ticket.
* Only authorized users can use specific functions of the system.

### Authorized User

* Users can follow/unfollow event
* User can view list of follow event
* Users can view transaction(buy and sell) history
* User can request ticket on event, or they can response request if they have corresponding ticket
* User can claim for transaction when received ticket got problem(in 5 days since received ticket)
* If user wants to sell a ticket, they have to provide their bank account, and must be approved by admin
* User can post 1 ticket one time
* For each sold ticket, website got 7% on ticket price for website transaction fee
* User prestige is rated by other users

### Admin

* Admin is responsible for managing category, event, user, transaction, venue, request, ticket, setting

### Guest

* They can only view event and search event but they cannot use authorized user functions

## System Requirement Specification

### External Interface Requirements

#### User Interfaces

* The design should be simple and user-friendly. White and dark will be 2 main colors of the website while charts may have more color to visually express data more effectively.
* The menu bar should be on the left and not take too much space of the screen. It consists of the list of main functions of the system that users can access.
* The design should be responsive. It means that the web components should be scaled according to a range of resolutions and devices to provide a consistent experience, no matter what.

#### Hardware Interfaces

* To access to the system, users only need any type of computer, tablet, or mobile phone with a fair internet connection.

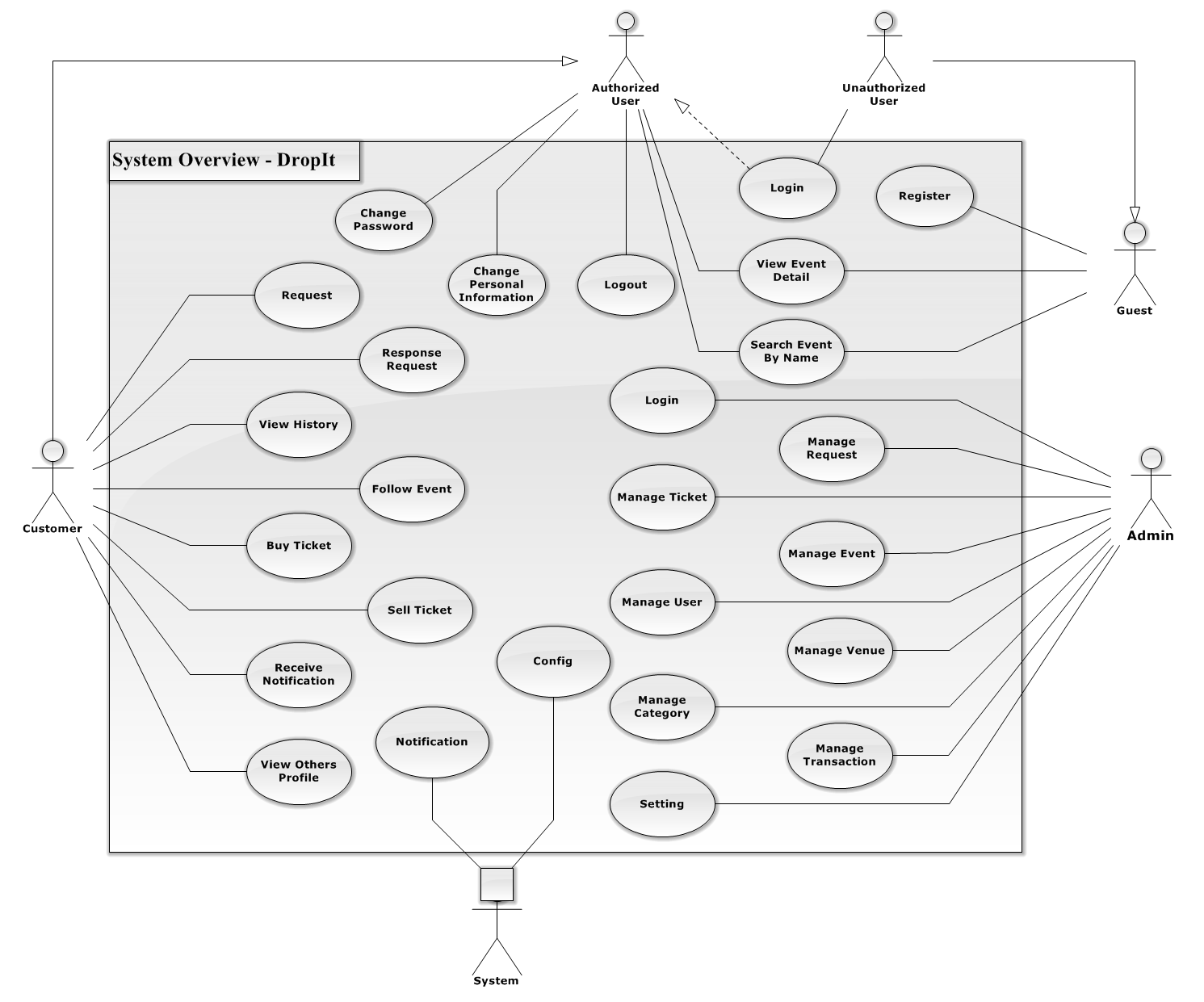
#### Software Interfaces

* At the server side, the system should run on top of Windows 7, Windows Server 2012 or later versions of Windows Server. Besides, Microsoft .NET Framework 4.5 and MVC 3 should be installed on the server. The database management system use for SMDH is SQL Server 2012.
* At the client side, users can use any modern browser that supports JavaScript and HTML 5 to access to the system.

#### Communications Protocol

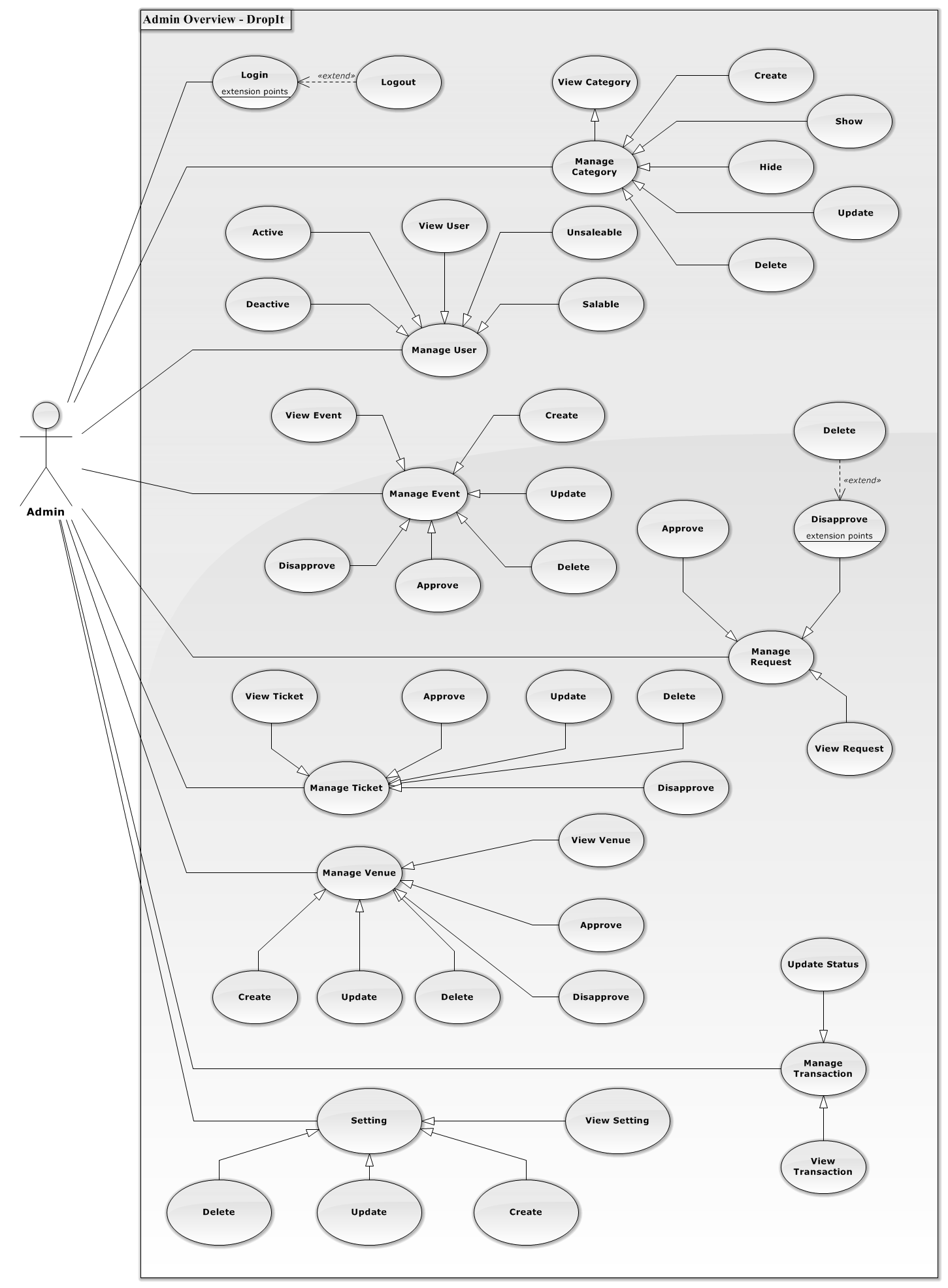
HTTP is the protocol used for loading the web site in browsers.

### System Features



**Figure 3.2.2.1 - General Use Case Diagram**

#### Admin



**Figure 3.2.2.1.1 - Admin Overview Diagram**

##### **View Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW CATEGORY | | | |
| Use-case No. | UC01.1 | Use-case Version | 1.0 |
| Use-case Name | View category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view category  **Goal:** View category successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Click “Hiển thị” tab | Display showed category list | | 3 | Click “Không hiển thị” tab | Display not showed category list |   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Create Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – CREATE CATEGORY | | | |
| Use-case No. | UC01.2 | Use-case Version | 1.0 |
| Use-case Name | Create category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to create category  **Goal:** Create category successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Category is created  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Click “Tạo danh mục” button | Show create new category page | | 3 | Fill mandatory fields and click “Tạo mới” button | - Validate, check conflict and save data  - Navigate to manage Category page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE CATEGORY | | | |
| Use-case No. | UC01.3 | Use-case Version | 1.0 |
| Use-case Name | Update category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to update category after created  **Goal:** Category is updated  **Triggers:** N/A  **Pre-conditions:** Category is already created  **Post-conditions:** Update category successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Click on “Hiển thị” tab | Navigate to list of shown Category | | 3 | Click on a category | Show update category form | | 4 | Fill mandatory fields and click “Cập nhật” button | Does validation, checks conflict and save data  Navigate back to manage Category page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Same as Main Success Scenario | | | 4 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Delete Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DELETE CATEGORY | | | |
| Use-case No. | UC01.4 | Use-case Version | 1.0 |
| Use-case Name | Delete category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to delete category after created  **Goal:** Category is deleted  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Delete category successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Choose Category and click “Xóa” button | Show confirm message | | 3 | Click “Yes” button | Selected category is deleted successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Hide Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – HIDE CATEGORY | | | |
| Use-case No. | UC01.5 | Use-case Version | 1.0 |
| Use-case Name | Hide category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to hide category after created  **Goal:** Category is hidden  **Triggers:** N/A  **Pre-conditions:** Category is shown  **Post-conditions:** Hide category successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Choose Category in shown category and click “Ẩn” button | Selected category is hidden successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Show Category**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – SHOW CATEGORY | | | |
| Use-case No. | UC01.6 | Use-case Version | 1.0 |
| Use-case Name | Show category | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to show category after hidden  **Goal:** Category is shown  **Triggers:** N/A  **Pre-conditions:** Category is hidden  **Post-conditions:** Show category successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Danh mục” tab | Navigate to Manage Category page | | 2 | Choose Category in hidden category and click “Hiện” button | Selected category is shown successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View User**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW USER | | | |
| Use-case No. | UC01.7 | Use-case Version | 1.0 |
| Use-case Name | View user | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about admin view user  **Goal:** View user successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Người dùng” tab | - Navigate to Manage User page  - Display list of users |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Active User**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – ACTIVE USER | | | |
| Use-case No. | UC01.8 | Use-case Version | 1.0 |
| Use-case Name | Active user | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to active a user  **Goal:** Active user successful  **Triggers:** N/A  **Pre-conditions:** User is deactivate  **Post-conditions:** User is activate  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Người dùng” tab | Navigate to Manage User page | | 2 | Choose user want to active and tick on “Kích hoạt” box | Active user successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Deactive User**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DEACTIVE USER | | | |
| Use-case No. | UC01.9 | Use-case Version | 1.0 |
| Use-case Name | Deactive user | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to deactive a user  **Goal:** Deactive user successful  **Triggers:** N/A  **Pre-conditions:** User is activate  **Post-conditions:** User is deactivate  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “User” button | Navigate to Manage User page | | 3 | Choose user want to deactive tick on “Kích hoạt” checkbox | Deactive user successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Allow User Selling**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – ALLOW USER SELLING | | | |
| Use-case No. | UC01.10 | Use-case Version | 1.0 |
| Use-case Name | Allow user selling | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to allow user selling  **Goal:** Allow user selling successful  **Triggers:** N/A  **Pre-conditions:** User is disallowed to sell  **Post-conditions:** User is allowed to sell  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “User” button | Navigate to Manage User page | | 3 | Choose user want to allow selling and tick on “Được bán” checkbox | Allow user selling successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Disallow User Selling**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DISALLOW USER SELLING | | | |
| Use-case No. | UC01.11 | Use-case Version | 1.0 |
| Use-case Name | Disallow user selling | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to disallow user selling  **Goal:** Disallow user selling successful  **Triggers:** N/A  **Pre-conditions:** User is allowed to sell  **Post-conditions:** User is disallowed to sell  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “User” button | Navigate to Manage User page | | 3 | Choose user want to allow selling tick on “Được bán” checkbox | Allow user selling successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW EVENT | | | |
| Use-case No. | UC01.12 | Use-case Version | 1.0 |
| Use-case Name | View event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view event  **Goal:** View event successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Event is created  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Sự kiện” tab | Navigate to Manage Event page | | 2 | Click “Đã duyệt” tab | Display approved event list | | 3 | Click “Chưa duyệt” tab | Display disapproved event list | | 4 | Click “Đang giao dịch” tab | Display on transaction event list | | 5 | Click “Quá ngày” tab | Display out of date event list |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Create Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – CREATE EVENT | | | |
| Use-case No. | UC01.13 | Use-case Version | 1.0 |
| Use-case Name | Create event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to create new event  **Goal:** Create event successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Event is created  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Sự kiện” tab | Navigate to Manage Event page | | 2 | Click “Tạo mới” buton | Navigate to Create new event page | | 3 | Fill mandatory fields and click “Tạo mới” button | - Does validation, check conflict and save data  - Navigate back to Event page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE EVENT | | | |
| Use-case No. | UC01.14 | Use-case Version | 1.0 |
| Use-case Name | Update event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to update event after created  **Goal:** Event is updated  **Triggers:** N/A  **Pre-conditions:** Event is already created  **Post-conditions:** Update event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Event” tab | Navigate to Manage Event page | | 2 | Click on an event | Show Update event page | | 3 | Fill mandatory fields and click “Cập nhật” button | - Does validation, check conflict and save data  - Navigate back to Event page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Delete Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DELETE EVENT | | | |
| Use-case No. | UC01.15 | Use-case Version | 1.0 |
| Use-case Name | Delete event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to delete an event after created  **Goal:** Event is deleted  **Triggers:** N/A  **Pre-conditions:** Event is already disapproved  **Post-conditions:** Delete event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Event” tab | Navigate to Manage Event page | | 2 | Click on “Chưa duyệt” tab | Show list of disapproved events | | 3 | Choose event and click “Xóa” button | Show confirm message | | 4 | Click “Yes” button | Selected event is deleted successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Approve Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – APPROVE EVENT | | | |
| Use-case No. | UC01.16 | Use-case Version | 1.0 |
| Use-case Name | Approve event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to approve an event after created  **Goal:** Event is approved  **Triggers:** N/A  **Pre-conditions:** Event is already disapproved  **Post-conditions:** Approved event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Event” tab | Navigate to Manage Event page | | 2 | Click on “Chưa duyệt” tab | Show list of disapproved events | | 3 | Choose event and click “Duyệt” button | Selected event is approved successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Disapprove Event**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DISAPPROVE EVENT | | | |
| Use-case No. | UC01.17 | Use-case Version | 1.0 |
| Use-case Name | Disapprove event | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to disapprove an event after created  **Goal:** Event is disapproved  **Triggers:** N/A  **Pre-conditions:** Event is already approved  **Post-conditions:** Disapproved event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Event” tab | Navigate to Manage Event page | | 2 | Click on “Đã duyệt” tab | Show list of approved events | | 3 | Choose event and click “Bỏ duyệt” button | Selected event is disapproved successfully |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Request**

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW REQUEST | | | |
| Use-case No. | UC01.18 | Use-case Version | 1.0 |
| Use-case Name | View request | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view request  **Goal:** Admin view request successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Yêu cầu” tab | Navigate to Manage Request page |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Close Request**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – CLOSE REQUEST | | | |
| Use-case No. | UC01.19 | Use-case Version | 1.0 |
| Use-case Name | Close request | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to close request after created  **Goal:** N/A  **Triggers:** N/A  **Pre-conditions:** Request is created  **Post-conditions:** Close request successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Yêu cầu” tab | Navigate to Manage Request page | | 2 | Choose request want to close and click on “Đóng” button | Display confirm message | | 3 | Click on “Yes” button | Selected request is closed successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Ticket**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW TICKET | | | |
| Use-case No. | UC01.20 | Use-case Version | 1.0 |
| Use-case Name | View Ticket | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view ticket  **Goal:** View ticket successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Vé” tab | Navigate to Manage ticket page | | 2 | Click on “Tạm hoãn” tab | Show list of disapproved ticket | | 3 | Click on “Sẵn Sàng” tab | Show list of approved ticket | | 4 | Click on “Người dùng duyệt” tab | Show list of waiting user-approve ticket | | 5 | Click on “Đang giao dịch” tab | Show list of on transaction ticket |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Approve Ticket**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – APPROVE TICKET | | | |
| Use-case No. | UC01.21 | Use-case Version | 1.0 |
| Use-case Name | Approve Ticket | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to approve ticket  **Goal:** Approve ticket successful  **Triggers:** N/A  **Pre-conditions:** Ticket is disapproved  **Post-conditions:** Ticket is approved  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Vé” tab | Navigate to Manage ticket page | | 2 | Click on “Tạm hoãn” tab | Show list of disapproved ticket | | 3 | Choose a ticket and click on “Duyệt” button | Approve ticket successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Ticket**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE TICKET | | | |
| Use-case No. | UC01.22 | Use-case Version | 1.0 |
| Use-case Name | Update ticket | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Date | 07/31/2013 |
| **Actor:** Admin  **Summary:** This use case is about how to update ticket  **Goal:** Update ticket successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Ticket is updated  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Vé” tab | Navigate to Manage Ticket page | | 2 | Click “Chi tiết” buton | Show update ticket page | | 3 | Fill mandatory fields and click “Cập nhật” button | Does validation, check conflict and save data  Navigate back to Manage Ticket page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Disapprove Ticket**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DISAPPROVE TICKET | | | |
| Use-case No. | UC01.23 | Use-case Version | 1.0 |
| Use-case Name | Disapprove ticket | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to disapprove ticket  **Goal:** Disapprove ticket successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Ticket is disapproved  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Vé” tab | Navigate to Manage ticket page | | 2 | Click on “Sẵn sàng” tab | Show list of approved ticket | | 3 | Choose a ticket and click on “Bỏ duyệt” button | Disapprove ticket successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Delete Ticket**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DELETE TICKET | | | |
| Use-case No. | UC01.24 | Use-case Version | 1.0 |
| Use-case Name | Delete ticket | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to delete ticket  **Goal:** Delete ticket successful  **Triggers:** N/A  **Pre-conditions:** N Ticket is disapproved  **Post-conditions:** Ticket is deleted  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Vé” tab | Navigate to Manage ticket page | | 2 | Click on “Tạm hoãn” tab | Show list of disapproved ticket | | 3 | Choose a ticket and click on “Xóa” button | Display confirm message | | 4 | Click “Yes” button | Delete ticket successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW VENUE | | | |
| Use-case No. | UC01.25 | Use-case Version | 1.0 |
| Use-case Name | View venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view venue  **Goal:** View venue successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Manage Venue page | | 2 | Click on “Chưa duyệt” tab | Display disapproved venue list | | 3 | Click on “Đã duyệt” tab | Display approved venue list |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Create Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – CREATE VENUE | | | |
| Use-case No. | UC01.26 | Use-case Version | 1.0 |
| Use-case Name | Create venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to create venue  **Goal:** Create venue successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Venue is created  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Manage Venue page | | 2 | Click on “Tạo địa điểm” button | Navigate to create new venue page | | 3 | Fill mandatory fields and click “Tạo mới” button | - Does validation, check conflict and save data  - Navigate back to Manage Venue page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE VENUE | | | |
| Use-case No. | UC01.27 | Use-case Version | 1.0 |
| Use-case Name | Update venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to update venue  **Goal:** Update venue successful  **Triggers:** N/A  **Pre-conditions:** Venue is not in transaction  **Post-conditions:** Venue is updated  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Manage Venue page | | 2 | Click on a venue | Navigate to Update Venue page | | 3 | Fill mandatory fields and click “OK” button | Does validation, check conflict and save data  Navigate back to Venue page |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Click on a venue is in transaction | Display warning message that venue is in transaction, cannot edit | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Delete Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DELETE VENUE | | | |
| Use-case No. | UC01.28 | Use-case Version | 1.0 |
| Use-case Name | Delete venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to delete venue after disapproved  **Goal:** Venue is deleted  **Triggers:** N/A  **Pre-conditions:** Venue is already disapproved  **Post-conditions:** Delete venue successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Venue page | | 2 | Click on “Chưa duyệt” tab | Show list of disapproved venues | | 3 | Choose venue want to delete and click on “Xóa” button | Show confirm message | | 4 | Click “Yes” button | Selected venue is deleted successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Approve Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – APPROVE VENUE | | | |
| Use-case No. | UC01.29 | Use-case Version | 1.0 |
| Use-case Name | Approve venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to approve venue  **Goal:** Venue is approved  **Triggers:** N/A  **Pre-conditions:** Venue is already disapproved  **Post-conditions:** Approve venue successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Venue page | | 2 | Click on “Chưa duyệt” tab | Show list of disapproved venues | | 3 | Choose venue want to approve and click on “Duyệt” button | Approve venue successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Disapprove Venue**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DISAPPROVE VENUE | | | |
| Use-case No. | UC01.30 | Use-case Version | 1.0 |
| Use-case Name | Disapprove venue | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to disapprove venue  **Goal:** Venue is approved  **Triggers:** N/A  **Pre-conditions:** Venue is already approved  **Post-conditions:** Disapprove venue successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Địa điểm” tab | Navigate to Venue page | | 2 | Click on “Đã duyệt” tab | Show list of approved venues | | 3 | Choose venue want to disapprove and click on “Bỏ duyệt” button | Disapprove venue successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Transaction**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW TRANSACTION | | | |
| Use-case No. | UC01.31 | Use-case Version | 1.0 |
| Use-case Name | View transaction | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view transaction  **Goal:** View transaction successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Giao dịch” tab | Navigate to Manage Venue page | | 2 | Click on a “Đã thanh toán” tab | Display paid transaction list | | 3 | Click on a “Giao hàng” tab | Display delivering transaction list | | 4 | Click on a “Đã nhận hàng” tab | Display received transaction list | | 5 | Click on a “Bị khiếu nại” tab | Display claimed transaction list | | 6 | Click on a “Đến hạn chuyển khoản” tab | Display need money transferred transaction list |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Transaction Status**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE TRANSACTION STATUS | | | |
| Use-case No. | UC01.32 | Use-case Version | 1.0 |
| Use-case Name | Update transaction status | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to update transaction  **Goal:** Update transaction successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** Transaction is updated  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Giao dịch” tab | Navigate to Manage Venue page | | 2 | Click on a “Đã thanh toán” tab | Display paid transaction list | | 3 | Choose a transaction and click “Giao hàng” button | - Transaction disappeared  - Transaction status changed to delivering | | 4 | Click on a “Giao hàng” tab | Show delivering transaction list | | 5 | Choose a transaction and click “Nhận hàng” button | - Transaction disappeared  - Transaction status changed to received | | 6 | Click on a “Đã nhận hàng” tab | Display received transaction list |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **View Setting**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – VIEW SETTING | | | |
| Use-case No. | UC01.33 | Use-case Version | 1.0 |
| Use-case Name | View Setting | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how admin view setting  **Goal:** View setting successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Tùy chỉnh” tab | - Navigate to Manage Setting page  - Display list of settings |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Create Setting**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – CREATE SETTING | | | |
| Use-case No. | UC01.34 | Use-case Version | 1.0 |
| Use-case Name | Create Setting | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to create setting  **Goal:** Create setting successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Tùy chỉnh” tab | - Navigate to Manage Setting page  - Display list of settings | | 2 | Input all mandatory field | Input information normally | | 3 | Click on “Thêm” button | Setting is created successful |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Rsponse | | 1 | Same as Main Success Scenario | | | 2 | Does not fill all required field | Input information normally | | 3 | Click on “Thêm” button | Display warning message to notify  the missing field |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Update Setting**



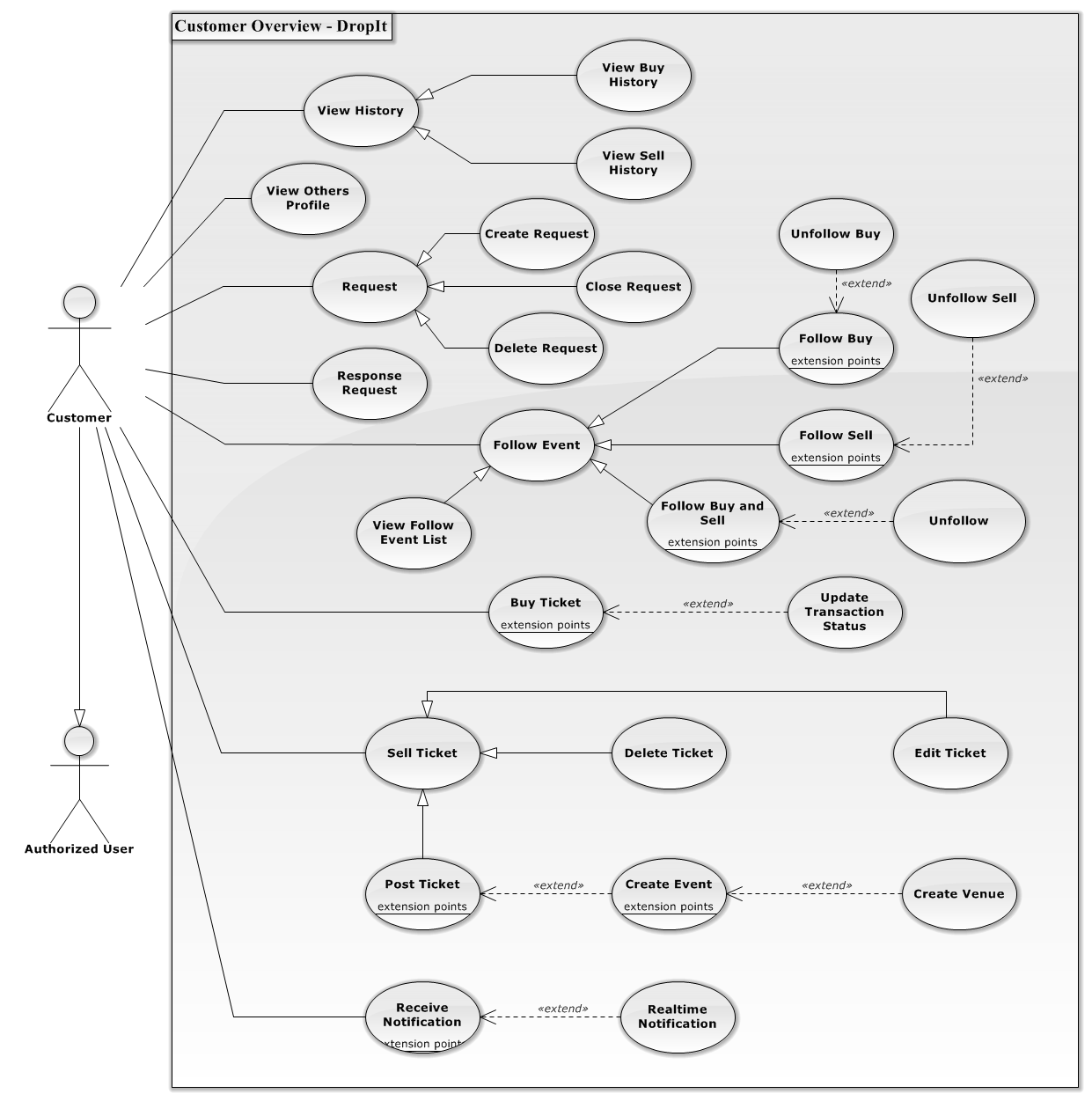
|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – UPDATE SETTING | | | |
| Use-case No. | UC01.35 | Use-case Version | 1.0 |
| Use-case Name | Update Setting | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to update setting  **Goal:** Update setting successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Tùy chỉnh” tab | - Navigate to Manage Setting page  - Display list of settings | | 2 | Choose a setting and click on “Sửa”  button | Display update field | | 3 | Input all mandatory field | Input information normally | | 4 | Click “Lưu” button | Update setting successful |   **Alternative Scenario:**  N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field | Input information normally | | 4 | Click on “Lưu” button | Display warning message to notify  the missing field |   **Relationships:**  N/A  **Business Rules:** N/A | | | |

##### **Delete Setting**



|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DELETE SETTING | | | |
| Use-case No. | UC01.36 | Use-case Version | 1.0 |
| Use-case Name | Delete Setting | | |
| Author | TungNT | | |
| Date | 06/01/2013 | Priority | Normal |
| **Actor:** Admin  **Summary:** This use case is about how to delete setting  **Goal:** Delete setting successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Tùy chỉnh” tab | - Navigate to Manage Setting page  - Display list of settings | | 2 | Choose a setting and click on “Xóa”  button | Display confirm message | | 3 | Click “Yes” button | Delete setting successful |   **Alternative Scenario:**  N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | |

#### Customer



**Figure 3.2.2.2.1 – Customer Overview Diagram**

##### **View Buy History**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW BUY HISTORY | | | | |
| Use-case No. | UC02.01 | Use-case Version | | 1.0 |
| Use-case Name | View Buy History | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user view buy history  **Goal:** User can view buy history successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** View buy history successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Mua vé” tab | Navigate to Buy ticket page | | 2 | Click on “Lịch sử giao dịch” link | Navigate to Buy history page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **View Sell History**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW SELL HISTORY | | | | |
| Use-case No. | UC02.02 | Use-case Version | | 1.0 |
| Use-case Name | View Sell History | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user view sell history  **Goal:** User can view sell history successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** View sell history successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” tab | Navigate to Sell ticket page | | 2 | Click on “Lịch sử giao dịch” link | Navigate to Sell history page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Create Request**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – CREATE REQUEST | | | | |
| Use-case No. | UC02.03 | Use-case Version | | 1.0 |
| Use-case Name | Create Request | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how to create request  **Goal:** User can create new request successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Create request successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Mua vé” tab | Navigate to Buy ticket page | | 2 | Click on “Đăng rao mua” link | Navigate to Create request page | | 3 | Fill mandatory fields and click “Đăng yêu cầu” button | Navigate to Home page |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 1 | Same as Main Success Scenario | 1 | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Close Request**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – CLOSE REQUEST | | | | |
| Use-case No. | UC02.04 | Use-case Version | | 1.0 |
| Use-case Name | Close Request | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how close request  **Goal:** User can close request successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Close request successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Mua vé” tab | Navigate to Buy ticket page | | 2 | Click on “Quản lí rao mua” link | Navigate to Request page | | 3 | Click on “Đóng” button | Display confirm message | | 4 | Click on “OK” button | Close request successful |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Response to Request**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE –RESPONSE TO REQUEST | | | | |
| Use-case No. | UC02.05 | Use-case Version | | 1.0 |
| Use-case Name | Response to request | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user response to request  **Goal:** User can response request successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Response request successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” tab | Navigate to Sell ticket page | | 2 | Click on “Danh sách rao mua” link | Navigate to Request page | | 3 | Click on “Phản hồi” button | Response to request successful |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Follow Buy**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – FOLLOW BUY | | | | |
| Use-case No. | UC02.06 | Use-case Version | | 1.0 |
| Use-case Name | Follow Buy | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user follow buy an event  **Goal:** User can follow buy event successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Follow buy event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event | Navigate to Event page | | 2 | Click on dropdown button next to “Theo dõi” button | Display follow dropdown list | | 3 | Click on “Theo dõi mua” button | “Theo dõi” button turns into “Bỏ theo dõi mua” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Unfollow Buy**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – UNFOLLOW BUY | | | | |
| Use-case No. | UC02.07 | Use-case Version | | 1.0 |
| Use-case Name | Unfollow Buy | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user unfollow buy an event  **Goal:** User can unfollow buy event successful  **Triggers:** N/A  **Pre-conditions:**  User has already follow-buy event  **Post-conditions:** Unfollow buy event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event has been buy-followed | Navigate to Event page | | 2 | Click on “Bỏ theo dõi mua” button | “Bỏ theo dõi mua” button turns into “Theo dõi” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Follow Sell**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – FOLLOW SELL | | | | |
| Use-case No. | UC02.08 | Use-case Version | | 1.0 |
| Use-case Name | Follow Sell | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user follow sell an event  **Goal:** User can follow sell event successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Follow sell event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event | Navigate to Event page | | 2 | Click on dropdown button next to “Theo dõi” button | Display follow dropdown list | | 3 | Click on “Theo dõi bán” button | “Theo dõi” button turns into “Bỏ theo dõi bán” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Unfollow Sell**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – UNFOLLOW SELL | | | | |
| Use-case No. | UC02.09 | Use-case Version | | 1.0 |
| Use-case Name | Unfollow Sell | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user unfollow sell an event  **Goal:** User can unfollow sell event successful  **Triggers:** N/A  **Pre-conditions:**  User has already follow-sell event  **Post-conditions:** Unfollow sell event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event has been buy-followed | Navigate to Event page | | 2 | Click on “Bỏ theo dõi bán” button | “Bỏ theo dõi bán” button turns into “Theo dõi” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Follow Buy and Sell**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – FOLLOW BUY AND SELL | | | | |
| Use-case No. | UC02.010 | Use-case Version | | 1.0 |
| Use-case Name | Follow Buy and Sell | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user follow buy and sell an event  **Goal:** User can follow buy and sell event successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Follow buy and sell event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event | Navigate to Event page | | 2 | Click on “Theo dõi” button | “Theo dõi” button turns into “Bỏ theo dõi” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Unfollow Buy and Sell**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – UNFOLLOW BUY AND SELL | | | | |
| Use-case No. | UC02.11 | Use-case Version | | 1.0 |
| Use-case Name | Unfollow Buy and Sell | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user unfollow buy and sell an event  **Goal:** User can unfollow buy and sell event successful  **Triggers:** N/A  **Pre-conditions:**  User has already follow-buy-and-sell event  **Post-conditions:** Unfollow buy and sell event successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event has been buy-followed | Navigate to Event page | | 2 | Click on “Bỏ theo dõi” button | “Bỏ theo dõi” button turns into “Theo dõi” button |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **View Follow List**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW FOLLOW EVENT LIST | | | | |
| Use-case No. | UC02.12 | Use-case Version | | 1.0 |
| Use-case Name | View Follow Event List | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user view list of follow events  **Goal:** User can view list of follow events successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** View list of follow events successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” or “Mua vé” tab | Navigate to corresponding page | | 2 | Click on “Danh sách theo dõi” link | Navigate to List follow event page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Update Transaction Status**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – UPDATE TRANSACTION STATUS | | | | |
| Use-case No. | UC02.13 | Use-case Version | | 1.0 |
| Use-case Name | Update Transaction Status | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user updates status of transaction  **Goal:** User can change status transaction successful  **Triggers:** click “lịch sử giao dịch” tab at the left side  **Pre-conditions:**  User login into system already.  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Select “lịch sử giao dịch” tab at the left side | System will display list of transaction of user in database. | | 2 | Click “chi tiết” button of a transaction | System will show detail transaction | | 3 | Choose status in text box | System will validate input. | | 4 | Click “Lưu” | System will validate, save to database, and change status of transaction. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Post Ticket**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – POST TICKET | | | | |
| Use-case No. | UC02.14 | Use-case Version | | 1.0 |
| Use-case Name | Post Ticket | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user posts a ticket  **Goal:** User can post ticket successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** Ticket is posted successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” tab | Navigate to Sell ticket page | | 2 | Click “Bán vé” link | Navigate to Create ticket page | | 3 | Fill mandatory fields and click “Đăng vé” button | - Does validation, check conflict and save data  - Navigate back to Home page |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Edit Posted Ticket**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – EDIT POSTED TICKET | | | | |
| Use-case No. | UC02.15 | Use-case Version | | 1.0 |
| Use-case Name | Edit Posted Ticket | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user edit ticket information  **Goal:** User can edit ticket information successful  **Triggers:** N/A  **Pre-conditions:**  Ticket want to edit is not in any transaction  **Post-conditions:** Ticket is updated successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” tab | Navigate to Sell ticket page | | 2 | Click on “Danh sách rao bán” link | Navigate to List posted ticket page | | 3 | Click on a posted ticket | Display ticket information | | 4 | Click on “Chỉnh sửa” button | Navigate to Edit ticket page | | 5 | Fill mandatory fields and click “Cập nhật” button | - Does validation, check conflict and save data  - Navigate back to Home page |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | ystem Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Same as Main Success Scenario | | | 4 | Same as Main Success Scenario | | | 5 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Delete Posted Ticket**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – DELETE POST TICKET | | | | |
| Use-case No. | UC02.16 | Use-case Version | | 1.0 |
| Use-case Name | Delete Posted Ticket | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user delete a posted ticket  **Goal:** User can delete a posted ticket successful  **Triggers:** N/A  **Pre-conditions:**  Ticket want to delete is not in any transaction  **Post-conditions:** Ticket is deleted successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” tab | Navigate to Sell ticket page | | 2 | Click on “Danh sách rao bán” link | Navigate to List posted ticket page | | 3 | Click on a posted ticket | Display ticket information | | 4 | Click on “Xóa” button | Show confirm message | | 5 | Click “Yes” button | Ticket is deleted successful |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Receive Notification**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – RECEIVE NOTIFICATION | | | | |
| Use-case No. | UC02.17 | Use-case Version | | 1.0 |
| Use-case Name | Receive Notification | | | |
| Author | PhongLK | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user receive notification  **Goal:** User can receive notification successful  **Triggers:** N/A  **Pre-conditions:**  User has followed an event  **Post-conditions:** N/A.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Login system with another user and post a ticket | Post ticket successful | | 2 | Login with user has followed event | - Navigate to home page with username on top right  - There’s notification about posted ticket in followed event | | 3 | Login system with another user and request ticket | Request ticket successful | | 4 | Login with user has followed event | - Navigate to home page with username on top right  - There’s notification about request ticket in followed event |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Receive Real-time Notification**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – RECEIVE REAL-TIME NOTIFICATION | | | | |
| Use-case No. | UC02.18 | Use-case Version | | 1.0 |
| Use-case Name | Receive Real-time Notification | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user receive real-time notification  **Goal:** User can receive real-time notification successful  **Triggers:** N/A  **Pre-conditions:**  User has followed an event  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Open two browser and login with two account, one has followed event | Navigate to home page with username on top right | | 2 | Post ticket in event that remain user has followed | - Posted ticket successful  - There’s real-time notification about posted ticket in followed event at bottom left | | 3 | Click on real-time notification | Navigate to posted ticket | | 4 | Request ticket in event that remain user has followed | - Requested ticket successful  - There’s real-time notification about posted ticket in followed event at bottom left | | 5 | Click on real-time notification | Navigate to request page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **View Others Profile**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW OTHERS PROFILE | | | | |
| Use-case No. | UC02.19 | Use-case Version | | 1.0 |
| Use-case Name | View Others Profile | | | |
| Author | HieuCM | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer  **Summary:** This use case is about how user views others profile  **Goal:** User can views others profile about history transaction  **Triggers:** N/A  **Pre-conditions:**  User has logged in to system  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on ticket “Details “ | Navigate to details page of ticket | | 2 | Click on username of customer want to know profile transaction | Navigate to customer profile page. List all transaction in history and truth rating |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

#### Authorized User



**Figure 3.2.2.3.1 – Authorized User Overview Diagram**

##### **Change Personal Information**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – CHANGE PERSONAL INFORMATION | | | | |
| Use-case No. | UC03.2 | Use-case Version | | 1.0 |
| Use-case Name | Change Personal Information | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer, Authorized User  **Summary:** This use case is about how user change personal information  **Goal:** User can change personal information successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** User’s information is changed successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Bán vé” or “Mua vé” tab | Navigate to corresponding page | | 2 | Click on “Thông tin tài khoản” link | Navigate to User information page | | 3 | Fill mandatory fields and click “Xác nhận” button | - Does validation, check conflict and save data  - Navigate back to Home page |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | Systm Response | | 1 | Same as Main Success Scenario | | | 2 | Same as Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Change Password**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – CHANGE PASSWORD | | | | |
| Use-case No. | UC03.3 | Use-case Version | | 1.0 |
| Use-case Name | Change Password | | | |
| Author | DuyDQ | | | |
| Date | 06/01/2013 | Priority | Normal | |
| **Actor:** Customer, Authorized User  **Summary:** This use case is about how user change password  **Goal:** User can change password successful  **Triggers:** N/A  **Pre-conditions:**  User has logged into system  **Post-conditions:** User’s password is changed successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on user name at the top right | Navigate to change password page | | 2 | Fill mandatory fields and click “Xác nhận” button | - Does validation, check conflict and save data  - Navigate to change password page with success message at top of page |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Same a Main Success Scenario | | | 3 | Does not fill all required field or enters invalid data | Display warning message to notify the missing field or invalid data |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **View Event Detail** C:\Users\duydq00276\Desktop\usecase\GuestOverview\ViewEventDetail.png

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW EVENT DETAIL | | | | |
| Use-case No. | UC03.03 | Use-case Version | | 1.0 |
| Use-case Name | View Event Detail | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Guest, Admin, User, Authorized User, Unauthorized User  **Summary:** This use case is about how actor views event detail  **Goal:** Actor can view event detail successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event | Navigate to Event page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Search Event by Name**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – SEARCH EVENT BY NAME | | | | |
| Use-case No. | UC03.04 | Use-case Version | | 1.0 |
| Use-case Name | Search Event by Name | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Admin, User, Authorized User, Guest, Unauthorized User  **Summary:** This use case is about how actor search event by name  **Goal:** Actor can search event successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Input information want to find in text box | Make suggestion when actor input more than 1 character | | 2 | Click on search label or press Enter key | Navigate to search result page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Logout**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – LOGOUT | | | | |
| Use-case No. | UC03.05 | Use-case Version | | 1.0 |
| Use-case Name | Logout | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Admin, Customer, Authorized User  **Summary:** This use case is about how user change personal information  **Goal:** Logout from system successful  **Triggers:** N/A  **Pre-conditions:** User has logged into system  **Post-conditions:** Logout successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on the “Đăng Xuất” button t the top right | - Remove user’s session  - Navigate to Home page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

#### Guest

****

Figure 3.2.2.4.1 – Guest Overview Diagram

##### **Register**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – REGISTER | | | | |
| Use-case No. | UC03.01 | Use-case Version | | 1.0 |
| Use-case Name | Register | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Guest  **Summary:** This use case is about how guest register to website  **Goal:** Register successful  **Triggers:** N/A  **Pre-conditions:** Register with username not existed in database  **Post-conditions:** User is registered successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Đăng kí thành viên” button | Navigate to register page | | 2 | Fill mandatory fields and click “Đăng ký” button | Navigate to Home page with username at the top right |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Does not fill all required field | Display warning message to notify the missing field | | 2 | Enter invalid data (wrong confirm password or register with username existed in database) | Display warning message to notify in each field |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **View Event Detail** C:\Users\duydq00276\Desktop\usecase\GuestOverview\ViewEventDetail.png

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – VIEW EVENT DETAIL | | | | |
| Use-case No. | UC04.01 | Use-case Version | | 1.0 |
| Use-case Name | View Event Detail | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Guest, Admin, User, Authorized User, Unauthorized User  **Summary:** This use case is about how actor views event detail  **Goal:** Actor can view event detail successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on an event | Navigate to Event page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Search Event by Name**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – SEARCH EVENT BY NAME | | | | |
| Use-case No. | UC04.02 | Use-case Version | | 1.0 |
| Use-case Name | Search Event by Name | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Admin, User, Authorized User, Guest, Unauthorized User  **Summary:** This use case is about how actor search event by name  **Goal:** Actor can search event successful  **Triggers:** N/A  **Pre-conditions:** N/A  **Post-conditions:** N/A  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Input information want to find in text box | Make suggestion when actor input more than 1 character | | 2 | Click on search label or press Enter key | Navigate to search result page |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

#### Unauthorized User



Figure 3.2.2.5.1 – Unauthorized User Overview Diagram

##### **Login**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – LOGIN | | | | |
| Use-case No. | UC05.01 | Use-case Version | | 1.0 |
| Use-case Name | Login | | | |
| Author | DuyDQ | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** Admin, Customer, Authorized User, Unauthorized User  **Summary:** This use case is about how actor login to website  **Goal:** Login to system successful  **Triggers:** N/A  **Pre-conditions:** No user has logged into system when access to home page  **Post-conditions:** Actor is logged in successful  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Đăng nhập” button | Navigate to login page | | 2 | Fill mandatory fields and click “Đăng nhập” button | Navigate to Home page with username at the top right |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Same as Main Success Scenario | | | 2 | Does not fill all required field | Display warning message to notify the missing field | | 2 | Enter invalid data (wrong password or wrong username) | Display warning message “Tên đăng nhập hoặc mật khẩu không đúng” |   **Relationships:**  N/A  **Business Rules:** N/A | | | | |

#### Auto System



Figure 3.2.2.6.1 – Auto System Overview Diagram

##### **Notification**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – NOTIFICATION | | | | |
| Use-case No. | UC06.01 | Use-case Version | | 1.0 |
| Use-case Name | Notification | | | |
| Author | PhongLK | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** System  **Summary:** This use case is about how system notifies customer which their followed  **Goal:** Show event that customer followed has something new  **Triggers:** N/A  **Pre-conditions:** Customer follows event and another customer requests or creates ticket  **Post-conditions:**  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Theo dõi” button on event details. | Change text button to “Bỏ theo dõi” | | 2 | When someone creates or requests this event ticket. | Show the event in notification bar. |   **Alternative Scenario:** N/A  **Exceptions:**  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Real-Time Notification**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – REAL-TIME NOTIFICATION | | | | |
| Use-case No. | UC06.02 | Use-case Version | | 1.0 |
| Use-case Name | Notification | | | |
| Author | PhongLK | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** System  **Summary:** This use case is about how system push real – time notification to customer  **Goal:** Real-time notification works normally when customer online  **Triggers:** N/A  **Pre-conditions:** Customer has logged into system  **Post-conditions:**  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Theo dõi” button on event details. | Change text button to “Bỏ theo dõi” | | 2 | When someone creates or requests this event ticket. | Show the event notification popup in left bottom website. |   **Alternative Scenario:** N/A  **Exceptions:**  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Auto Update TranPayment Status**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – AUTO UPDATE TRANPAYMENT STATUS | | | | |
| Use-case No. | UC06.03 | Use-case Version | | 1.0 |
| Use-case Name | Notification | | | |
| Author | TungDN | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** System  **Summary:** This use case is about how the system auto change TranPaymentStatus when deadline comes.  **Goal:** The TranPaymentStatus change to “Transferred” after 5 days from TranShipDate was created  **Triggers:** N/A  **Pre-conditions:** TranShipDate was created  **Post-conditions:**  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Select TranShipDate not null | Return all records that have TranShipDate | | 2 | Check TranShipDate with today | Return all records that have TranShipDate more than 5 days old | | 3 | Auto change TranPaymentStatus | Change TranPaymentStatus to “Transferred” |   **Alternative Scenario:** N/A  **Exceptions:**  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

##### **Auto Update Event Status**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE – AUTO UPDATE EVENT STATUS | | | | |
| Use-case No. | UC06.04 | Use-case Version | | 1.0 |
| Use-case Name | Notification | | | |
| Author | PhongLK | | | |
| Date | 01/06/2013 | Priority | Normal | |
| **Actor:** System  **Summary:** This use case is about how to change status of event after over hold date  **Goal:** The status of event become “OutDate”  **Triggers:** N/A  **Pre-conditions:** The event was approved  **Post-conditions:**  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Select all event has status not equal “Delete”and “OutDate” and not over hold date | Return all records of event that suitable | | 2 | Change event status to “OutDate” | Event status becomes “OutDate” |   **Alternative Scenario:** N/A  **Exceptions:**  **Relationships:**  N/A  **Business Rules:** N/A | | | | |

### Software System Attributes



#### Reliability

* The database can be backed up easily and recovered if necessary.
* The system should never crash or hang, except for the cause from an operating system or network error.
* Mean Time Between Failures (MTBF): The acceptable failure is once a year. The failure should not be because of the database, or else the data may be lost and cannot be recovered.
* Mean Time To Repair (MTTR): When the failure occurs, it should take as little time as possible to repair. The acceptable mean time for a particular failure must be less than 8 hours.
* Maximum Bugs or Defect Rate: 05 bugs / KLOC

#### Availability

* The uptime percentage should be at least 99.95%.
* The acceptable time for maintenance or backup should not be more than 8 hours per month.
* When the system goes in under-maintenance, the website should display a message to inform that.

#### Security

* Only users with proper account can access certain information of the system. All the information of users must not be available for anyone or software that is not part of the system. User password is also encrypted and not available to the system administrators.

#### Maintainability

* The code must follow C#.NET coding and naming convention.
* There should be comments in code files that explain the functions of each code segment.
* All the errors should be logged, which supports bug fixing and maintenance.

#### Portability

System can run best with the requirements:

* Hardware: Pentium IV, Ram 512 or above
* Network: ADSL or above
* Environment: Window 7
* Browser: Chrome 27

#### Performance

* Load time

## Entity Relationship Diagram

****

**Figure 3.3.1 Entity Relationship Diagram**

## SOFTWARE DESIGN DESCRIPTION

### Design Overview

SDD document objective is providing for project’s stakeholders multi specific views of TWS website like:

* System Architectural design which describes the planned model of system, the initial design process of identifying sub-systems and establishing a framework for sub-system control and communication.
* Component design which describes all system’s component (class, interface…) together with relationships between them. It also explains clearly their purposes, methods within each component and detailed algorithms, pseudo code to implement them.
* Class diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.
* Sequence diagram design which shows how system processes operate and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
* User interface design which describes all screens’ interface; each screen includes which types of input, output element and used by what action. It makes the user's interaction as simple and efficient as possible, in terms of accomplishing user goals.
* Database design is the process of producing a detailed [data model](http://en.wikipedia.org/wiki/Data_model) of a [database](http://en.wikipedia.org/wiki/Database). Database design section shows and explains the physical model of system database which includes all tables and their relationship

### System Architectural Design

#### Choice of System Architecture

**Figure** **4.2.1.1 System Architecture**

ASP.NET MVC 4 is a framework for building scalable, standards-based web applications using well-established design patterns and the power of ASP.NET and the .NET Framework. MVC stands for model-view-controller. MVC is a pattern for developing applications that are well architected and easy to maintain. MVC-based applications contain:

Views are template files that your application uses for dynamically generating HTML responses.

Models are classes that represent the data of the application and that use validation logic to enforce business rules for that data.

Controllers are classes that handle incoming requests to the application, retrieve model data, and then specify view templates that return a response to the client.

#### Discussion of Alternative DesignsE:\Capstone Project\trunk\Document\Picture\MVP vs MVC.jpg

**Figure 4.2.2.1 – Alternative Designs**

MVP model also has three components like MVC mode, but Controller component is replaced by Presenter component which receives call from View component, interacts with Model component and bring changes back to the View.

In MVP model business logic section may be included directly in View or Presenter which can cause ambiguity and difficulty for implementation and testing.

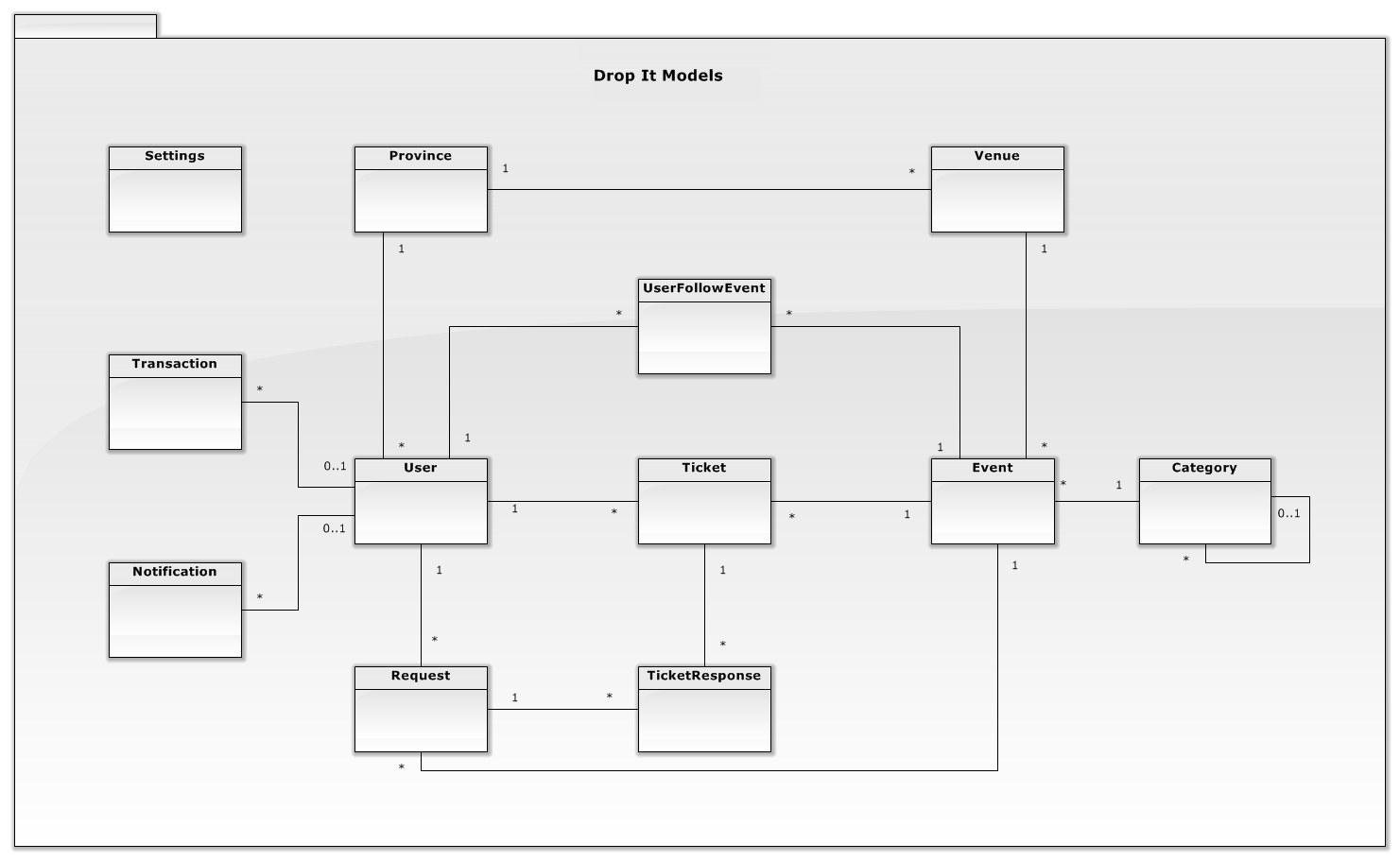
### PACKAGE Diagram



**Figure 4.3.1 – General Package Diagram**

### Class Detailed Description

#### DropIt.Models Package



**Figure 4.4.1.1 – DropIt.Models Package**

*Province*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| ProvinceId | int | Unique ID of each City |
| ProvinceName | string | Name of Province |

*Category*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| CategoryId | int | Unique ID of each Category |
| CategoryName | string | Name of each Category |
| ParentCategoryId | int | Unique ID of each ParentCategory |
| Status | int | Status of Category |
| Description | string | Description of Category |

*Event*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| EventId | int | Unique ID of each Event |
| EventName | string | Name of Event |
| EventImage | string | Image of Event |
| Artist | string | Name of Artist |
| HoldDate | datetime | Date event is hold |
| Description | string | Description of Event |
| Status | int | Status of Event |
| CategoryId | int | Unique ID of each Category |
| VenueId | int | Unique ID of each Venue |
| CreatedDate | datetime | Date event is created |
| ModifiedDate | datetime | Date event is modified |

*Notification*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| NotificationId | int | Unique ID of each Notification |
| UserId | int | ID of received User |
| SenderId | int | ID of object occurs notification |
| ActivityType | string | Type of each Notify Activity |
| ObjectType | string | Type of object occurs notification |
| ObjectTitle | string | Title of each object occurs notification |
| ObjectUrl | string | Url of Notification |
| Content | string | Content of Notification |
| IsUnread | bool | Notification is read or not |
| CreatedDate | datetime | Time Notification is created |
| ModifiedDate | datetime | Time Notification is modified |

*Request*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| UserId | int | Unique ID of each User |
| EventId | int | Unique ID of each Event |
| Status | int | Status of Request |
| Description | string | Description of Request |
| CreatedDate | datetime | Time Request is created |
| ModifiedDate | datetime | Time Request is modified |

*Setting*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id | int | Unique ID of each Setting |
| SettingName | string | Name of each Setting |
| Value | string | Value of each Setting |

*Ticket*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| TicketId | int | Unique ID of each Ticket |
| SeriesNumber | string | Series Number of each Ticket |
| SellPrice | float | Price of each Ticket |
| ReceiveMoney | float | Money user receives when Ticket is sold |
| ShippingCost | float | Cost of shipping Ticket |
| Seat | string | Seat location of Ticket |
| Description | string | Description of Ticket |
| Status | int | Status of Ticket |
| UserId | int | Unique ID of each User |
| EventId | int | Unique ID of each Event |
| AdminModifiedDate | datetime | Time Admin modified Ticket |
| CreatedDate | datetime | Time Ticket is created |
| ModifiedDate | datetime | Time User modified Ticket |
| TranFullName | string | Name of Ticket Buyer |
| TranDescription | string | Description of Transaction |
| TranAddress | string | Address of Ticket Buyer |
| TranStatus | int | Status of Transaction |
| TranUserId | int | Unique ID of Buyer |
| TranCreatedDate | datetime | Time Transaction is created |
| TranModifiedDate | datetime | Time Transaction is modified |
| TranShipCode | string | Code of shipping Transaction |
| TranPaymentStatus | int | Status of Transaction payment |

*TicketResponse*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| UserId | int | Unique ID of each User |
| EventId | int | Unique ID of each Event |
| TicketId | int | Unique ID of each Ticket |
| Description | string | Description of Ticket Response |
| Status | int | Status of Category |

*User*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| UserId | int | Unique ID of each User |
| UserName | string | Name of User |
| FullName | string | Full name of User |
| Email | string | Email of User |
| Phone | string | Phone of User |
| Address | string | Address of User |
| Active | bool | Active status of User |
| IsVerified | bool | Verified status of User |
| BankName | string | Bank name of User |
| BankAccount | string | Bank account of User |
| IdentityCard | string | Identity card of User |
| Sellable | bool | Sellable status of User |
| CreatedDate | datetime | Time user registered |
| ModifiedDate | datetime | Time User modified information |
| ProvinceId | int | Unique ID of each Province |

*UserFollowEvent*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| UserFollowEventId | int | Unique ID of each follow Event |
| UserId | int | Unique ID of each User |
| EventId | int | Unique ID of each Event |
| FollowType | int | Follow type of follow Event |

*Venue*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| VenueId | int | Unique ID of each |
| VenueName | string | Name of Venue |
| Address | string | Address of Venue |
| ProvinceId | int | Unique ID of each Province |
| Status | int | Status of Venue |
| Description | string | Description of Venue |

#### DropIt.Controllers Package

**Figure 4.3.4.2.1 – Controller Package**

##### **DropIt.Controllers**

*AccountController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Login(LoginModel model, string returnUrl) | ActionResult | Allow user to login to system |
| ActionResult LogOff() | ActionResult | Allow user to logout from system |
| Register() | ActionResult | Display page for register |
| Register(RegisterModel model) | ActionResult | Allow user to create new account |
| Manage(ManageMessageId? message) | ActionResult | Display page for change password |
| Manage(LocalPasswordModel model) | ActionResult | Allow user to change password |

*CategoryController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Details(int id = 0) | ActionResult | Display category details base on CategoryId |
| Create() | ActionResult | Display page for create category |
| getAll() | JsonResult | Get all category |

*CheckOutController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Checkout(int Id) | ActionResult | Checkout ticket |
| Success() | ActionResult | Display checkout success page |

*EventController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int PageSize = 10, int StartIndex = 0,int CategoryId = -1) | ActionResult | Paging for event |
| Details(int id = 0) | ActionResult | Display event detail base on EventId |
| getInfo(int EventId) | JsonResult | Get event information |
| Image(int Id) | ActionResult | Return event image |
| Count() | JsonResult | Return all records of event |

*FollowController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Event(int Id,int FollowType) | JsonResult | Allow user follow event |
| Events() | ActionResult | List all follow events |

*HomeController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Homepage |
| BuyIndex() | ActionResult | Display Buy Index page |
| SellIndex() | ActionResult | Display Sell Index page |
| Search(string query, string sortBy = "relevant", int PageSize = 10, int StartIndex = 0) | ActionResult | Allow user to search event by name |
| SearchAjax(string searchText) | JsonResult | Suggestion on searching |
| About() | ActionResult | Display About page |
| Contact() | ActionResult | Display Contact Us page |
| GuideForNew() | ActionResult | Display Guide for New User page |

*NotificationController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| ReadAll(int FollowType = -1) | JsonResult | To mark all notifications are read |
| List(int FollowType = 2,String Object="\*",Nullable<Boolean> IsUnread = null,int jTStartIndex = 0,int FromId = 0,int jtPageSize = 10,String jtSorting = "CreatedDate DESC") | JsonResult | List notification |

*RequestController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Request Ticket page |
| Details(int EventId) | ActionResult | Display ticket request details base on RequestId |
| Close(Request Request, int UserId, int EventId) | JsonResult | Allow user to close ticket request |

*TicketController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Create() | ActionResult | Display page for create ticket |
| Create(PostTicket ticket) | ActionResult | Allow user to create new ticket |
| Buy(int Id) | ActionResult | Display page for buy ticket |
| Buy(BuyTicket buyTicket) | ActionResult | Allow user to buy ticket |
| RequestTicket(int EventId = 0) | ActionResult | Display page for request ticket |
| RequestTicket(RequestTicket ticket) | ActionResult | Allow user to create request ticket |
| Count() | JsonResult | Return all records of ticket |
| Approve(int id = 0) | ActionResult | Display page for approve ticket |
| Approve(Ticket ticket) | ActionResult | Allow user to approve ticket |
| Edit(int id=0) | ActionResult | Display page for edit ticket |
| Edit(Ticket ticket) | ActionResult | Allow user to edit ticket |
| Manage() | ActionResult | Display Manage Ticket page |
| Delete(int Id) | ActionResult | Allow user to delete ticket |
| Details (int id=0) | ActionResult | Allow user to view ticket details base on TicketId |

*TicketResponseController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Request Ticket page |
| Create() | ActionResult | Display Response Ticket page |
| Response(CreateResponses data) | ActionResult | Allow user to respond ticket request |
| List() | ActionResult | Display list of responded ticket |

*TransactionController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| HistoryBuy() | ActionResult | Display Buy History page |
| HistorySell() | ActionResult | Display Sell History page |
| Receive(int Id) | ActionResult | Display Receive History page |
| Reported(int Id) | ActionResult | Display Claimed History page |
| Count(int id) | JsonResult | Return all records of transaction |

*UserController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Edit(int id = 0) | ActionResult | Display Change Personal Information page |
| Edit(UserViewModel user) | ActionResult | Allow user to change personal information |

*VenueController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| getInfo(int VenueId) | JsonResult | Get venue information |

##### **DropIt.Areas.Administration.Controllers**

*CategoryController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int CategoryStatus = 0) | ActionResult | Display Manage Category Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = "CategoryName ASC", int CategoryStatus = -1) | JsonResult | Paging for category |
| Create() | ActionResult | Display Create Category page |
| Create(Category Category) | JsonResult | Allow admin to create new category |
| Edit(int Id) | ActionResult | Display Edit Category page |
| Edit(Category Category) | JsonResult | Allow admin to edit category |
| Active(int Id) | JsonResult | Active category base on CategoryId |
| Deactive(int Id) | JsonResult | Deactive category base on CategoryId |
| Delete(int Id) | JsonResult | Delete category base on CategoryId |

*DashboardController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
|  |  |  |
|  |  |  |

*EventController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int EventStatus = 0) | ActionResult | Display Manage Event Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = "HoldDate ASC", int EventStatus = -1) | JsonResult | Paging for event |
| Create() | ActionResult | Display Create Event page |
| Create(Event Event, HttpPostedFileBase EventImage) | JsonResult | Allow admin to create new event |
| Edit(int Id) | ActionResult | Display Edit Event page |
| Edit(Event Event, HttpPostedFileBase EventImage) | JsonResult | Allow admin to edit event |
| getInfo(int EventId) | JsonResult | Get event information |
| Approve(int Id) | JsonResult | Approve event base on EventId |
| Disapprove(int Id) | JsonResult | Disapprove event base on EventId |
| Delete(int Id) | JsonResult | Delete event base on EventId |

*ProvinceController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Manage Province Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null) | JsonResult | Paging for province |
| Create(Province province) | JsonResult | Allow admin to create new province |
| Update(Province province) | JsonResult | Allow admin to update province |
| Delete(int ProvinceId) | JsonResult | Allow admin to delete province |

*RequestController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int RequestStatus = 0) | ActionResult | Display Manage Request Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int RequestStatus = -1) | JsonResult | Paging for request |
| Close(int UserId, int EventId) | JsonResult | Allow admin to close request |

*SettingController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Manage Setting Index page |
| List() | ActionResult | Return list of settings |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null) | JsonResult | Paging for setting |
| Create(SettingViewModel setting) | JsonResult | Allow admin create new setting |
| Delete(int Id) | JsonResult | Allow admin delete setting |
| Edit(Setting setting) | JsonResult | Allow admin update setting |

*TicketController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int TicketStatus = 0) | ActionResult | Display Manage Ticket Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int TicketStatus = -1) | JsonResult | Paging for ticket |
| Approve(int Id) | JsonResult | Allow admin to approve ticket |
| Disapprove(int Id) | JsonResult | Allow admin to disapprove ticket |
| Delete(int Id) | JsonResult | Allow admin to delete ticket |
| Edit(int Id) | ActionResult | Display Update Ticket page |
| Edit(Ticket Ticket) | JsonResult | Allow admin to update ticket |

*TransactionController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int TranStatus = (int)Statuses.Transaction.Paid, int TranPaymentStatus = 0) | ActionResult | Display Manage Transaction Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int TranStatus = -1, int TranPaymentStatus = -1) | JsonResult | Paging for transaction |
| Delivered(int Id, string TranShipCode) | JsonResult | Allow admin to update transaction status to delivered |
| Paid(int Id) | JsonResult | Allow admin to update transaction status to paid |
| Done(int Id) | JsonResult | Allow admin to update transaction status to done |
| Received(int Id) | JsonResult | Allow admin to update transaction status to received |
| Canceled(int Id) | JsonResult | Allow admin to update transaction status to received canceled |

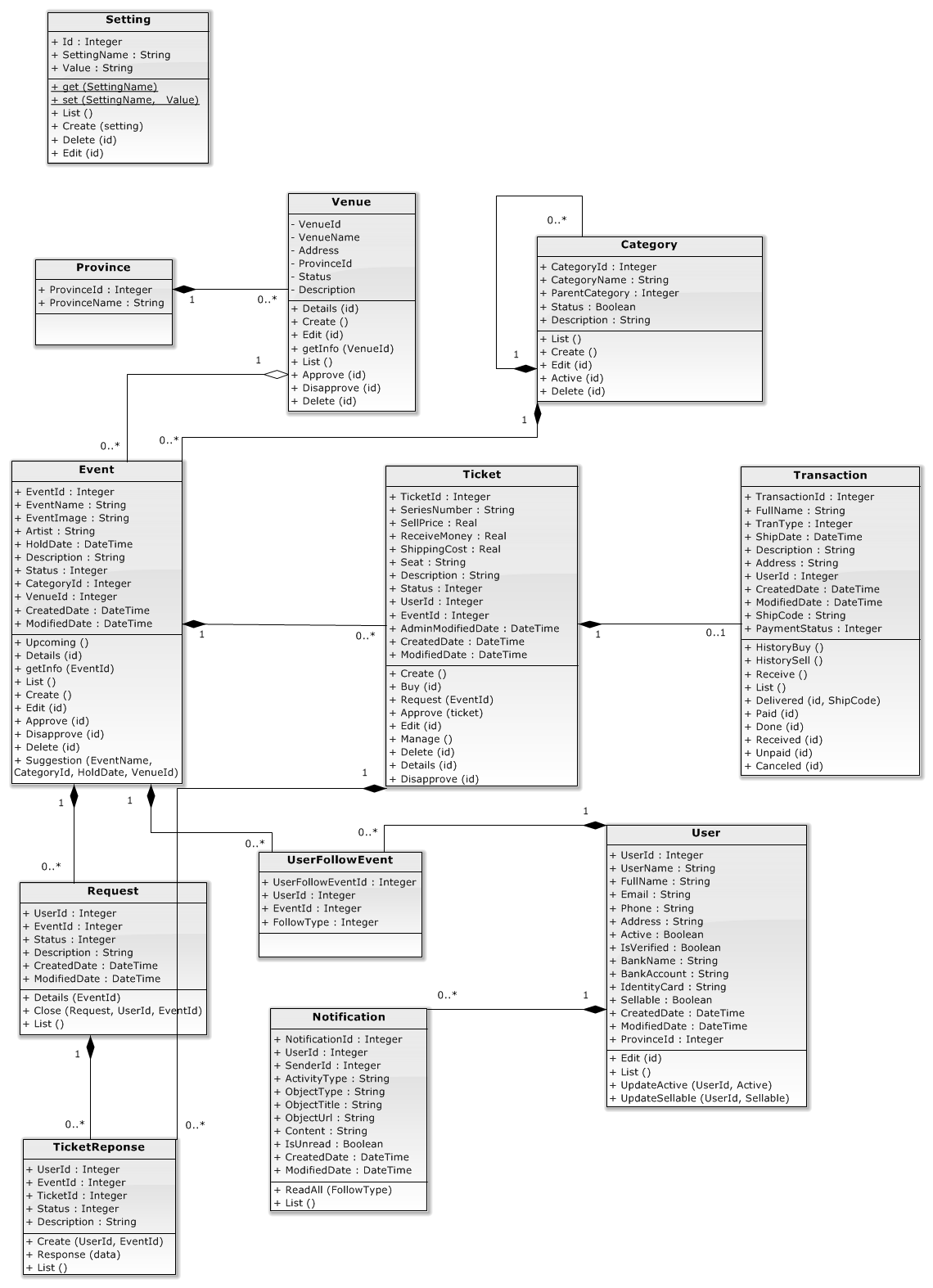
*UserController*

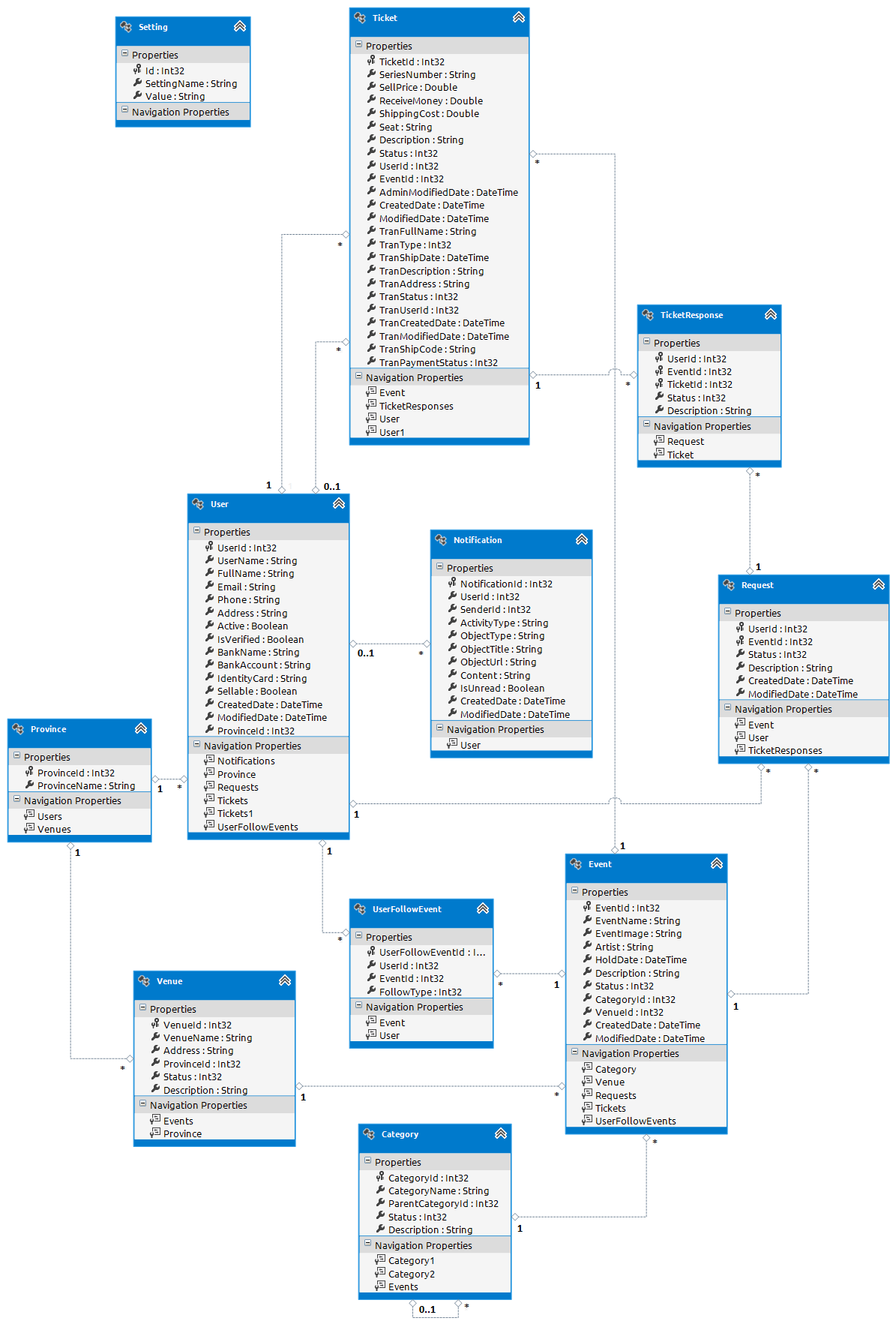
|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Manage User Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null,int isActive = -1,int isSellable =-1,String UserName = "") | JsonResult | Paging for user |
| UpdateActive(int UserId, bool Active) | JsonResult | Active user base on UserId |
| UpdateSellable(int UserId, bool Sellable) | JsonResult | Allow user selling base on UserId |

*VenueController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int VenueStatus = 0) | ActionResult | Display Manage Venue Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int VenueStatus = -1) | JsonResult | Paging for venue |
| Create() | ActionResult | Display Create Venue page |
| Create(Venue Venue) | JsonResult | Allow admin to create new venue |
| Approve(int Id) | JsonResult | Allow admin approve venue |
| Disapprove(int Id) | JsonResult | Allow admin disapprove venue |
| Edit(int Id) | ActionResult | Display Update Venue page |
| Edit(Venue Venue) | JsonResult | Allow admin to update venue |
| Delete(int Id) | JsonResult | Allow admin to delete venue |

#### Class Diagram



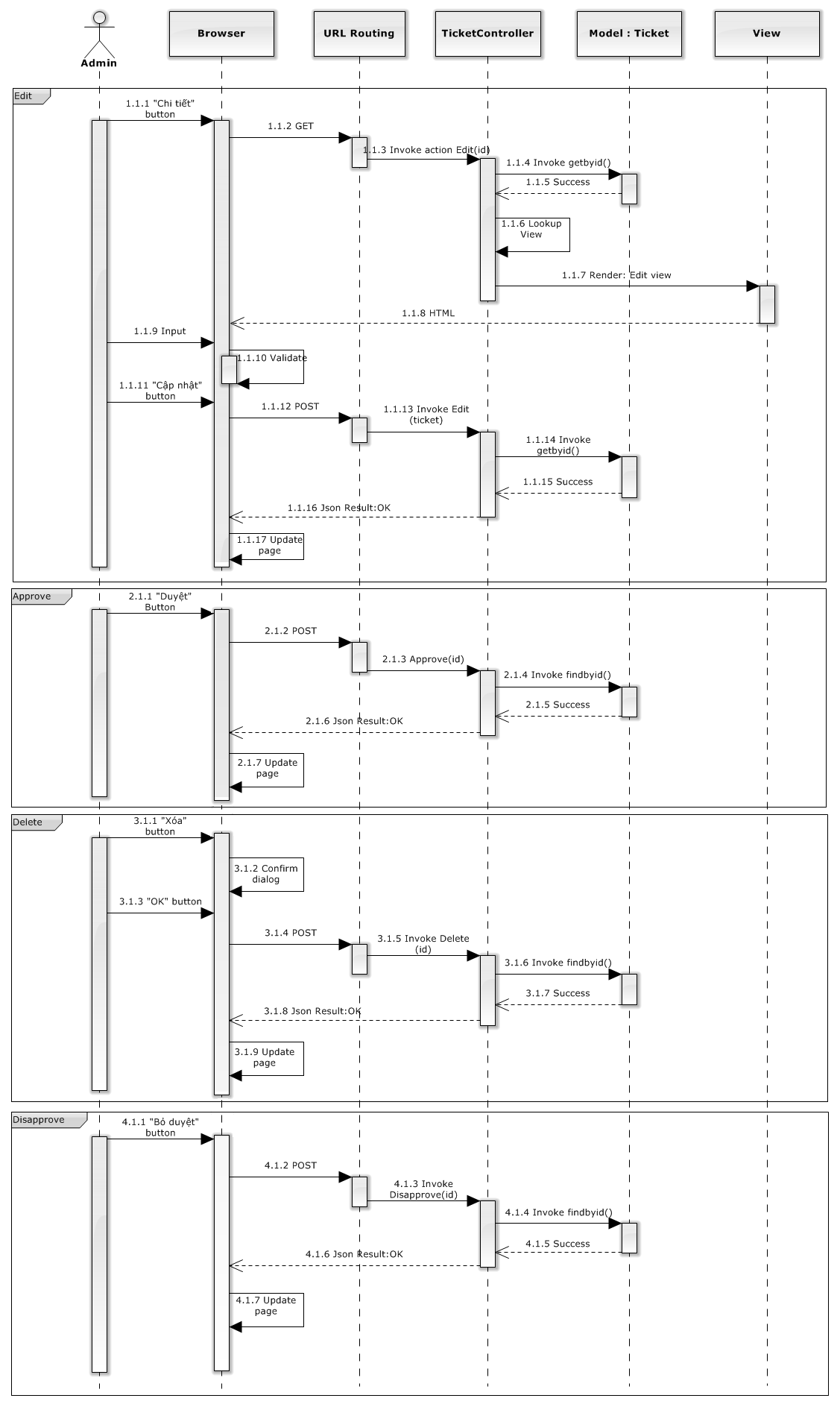


### Sequence Diagram

#### Get list ticket



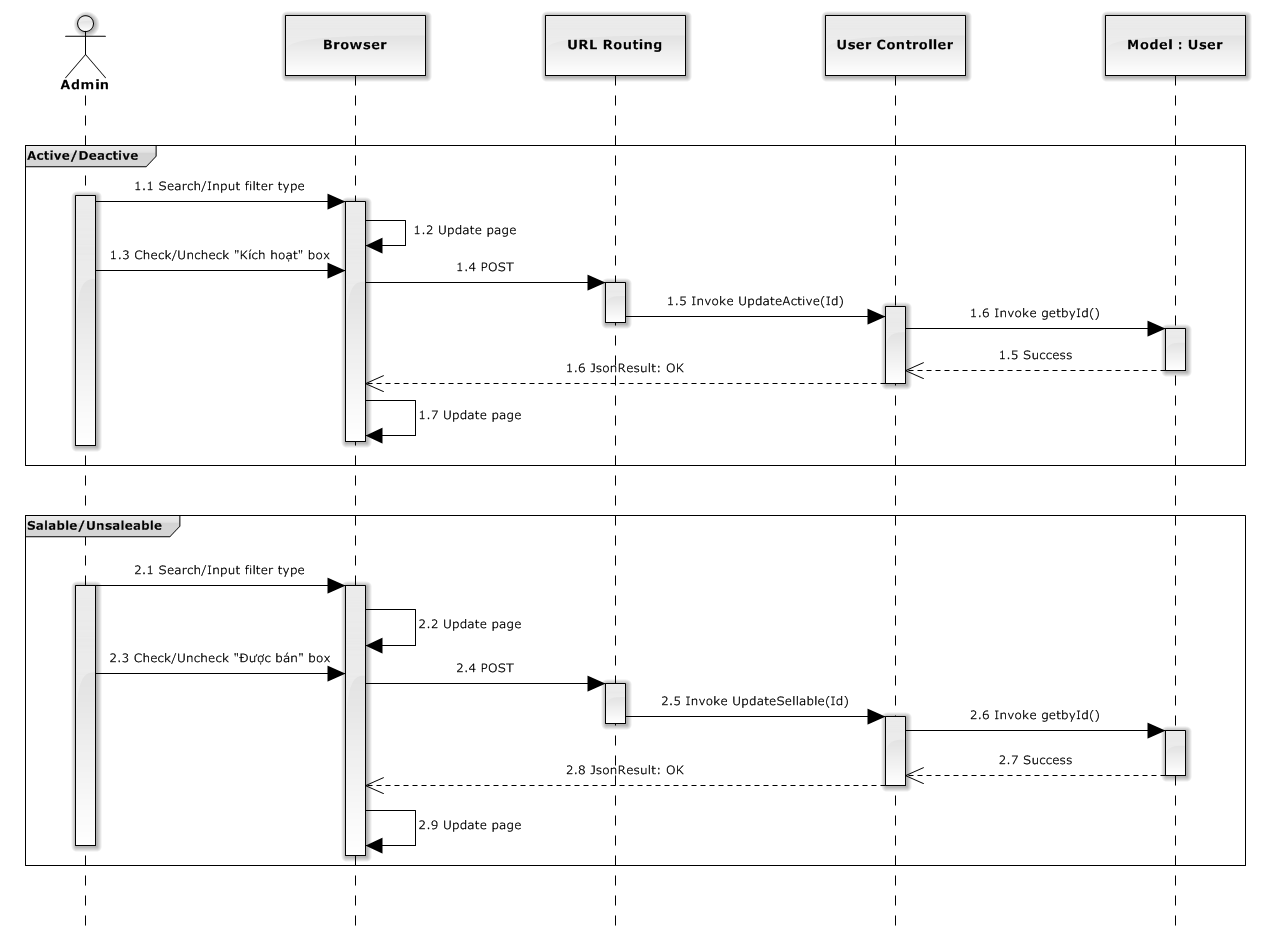
#### Manage ticket



#### Manage transaction



#### Manage user



#### Create Event



### Activity diagram

#### Create Ticket



#### Request Ticket



### State Diagram

#### Ticket State Diagram



**Figure 4.3.7.1.1 – Ticket State Diagram**

**States Description:**

Delete: The ticket is not presented to user and admin.

Pending: The ticket can’t transact and present in user site.

UserApprove: The ticket was waiting for user approve when Admin edited information of ticket event.

OnTransaction: The ticket is transacting, user can’t buy or edit it.

Ready: The ticket is available for buying now.

Invalid: The ticket has some error on delivery to buyer.

#### Event State Diagram



**Figure 4.3.7.2.1 – Event State Diagram**

**States Description:**

Disapprove: The event does not present in user site.

Delete: The event is not presented to user and admin.

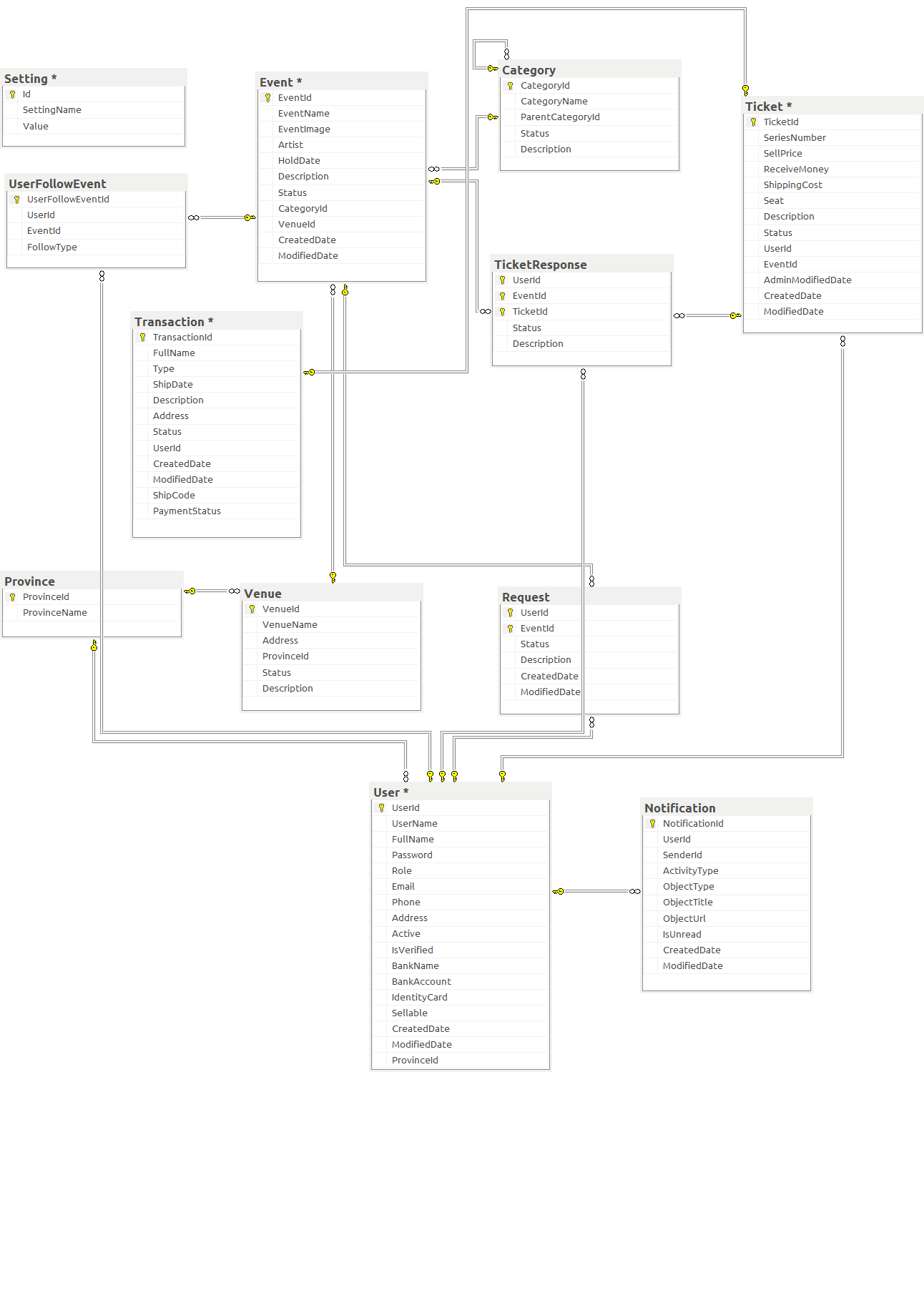
Approve: The event presents in user site.

Trading: The event has ticket is transacted.

OutDate: The event is out of date.

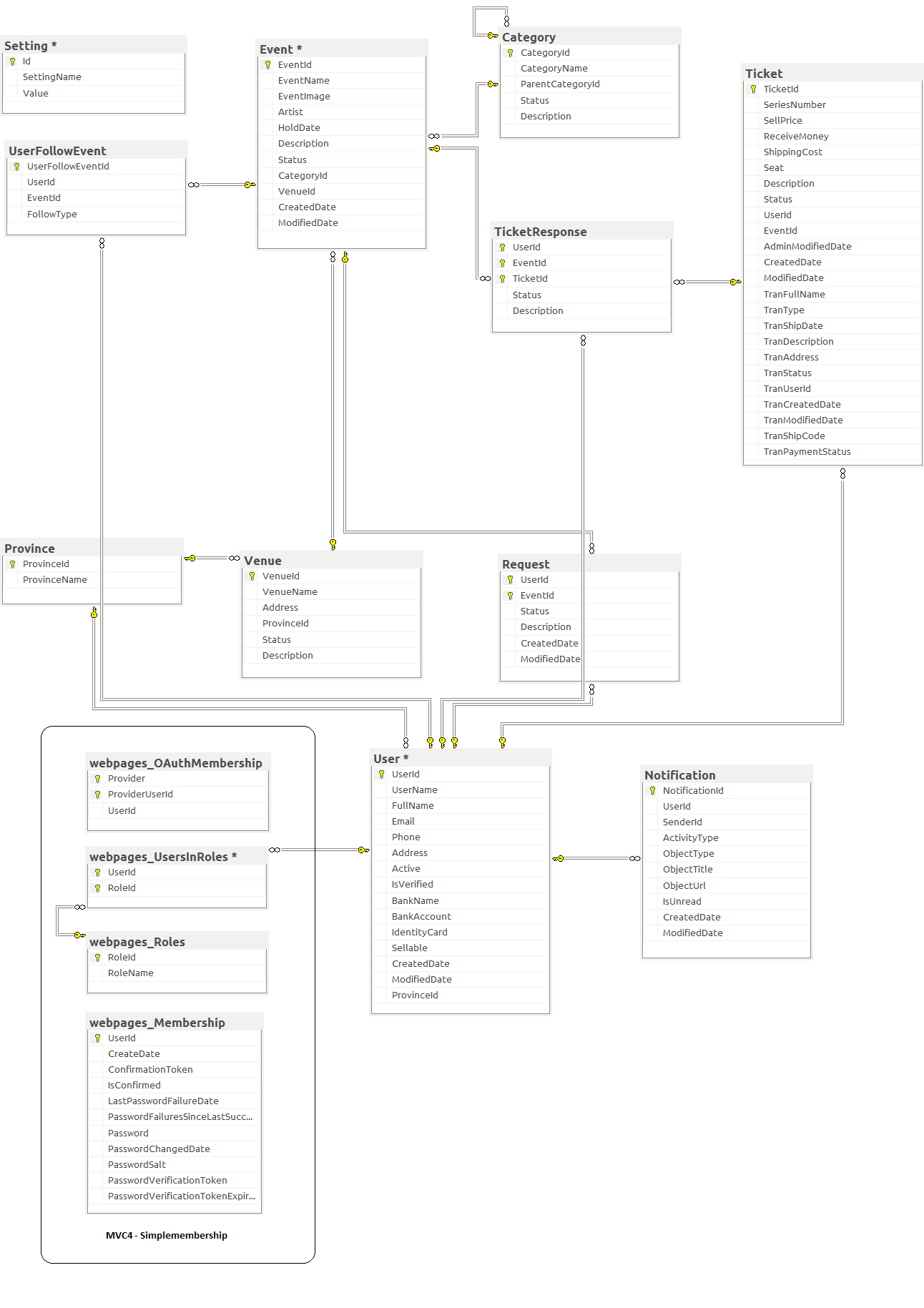
### Database Design or Data Structures

#### CONCEPT Database diagram



**Figure 4.3.8.1.1 – Concept database diagram**

#### Database diagram



**Figure 4.3.8.2.1 Database Diagram**

#### Table

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| User | Store detail of user |
| Ticket | Store feedback response content |
| Venue | Store detail of venue |
| Event | Store detail of event |
| UserFollowEvent | Store detail of follow |
| Request | Store detail of request |
| TicketResponse | Store detail of response |
| Category | Store detail of category |
| Province | Store detail of province |
| Setting | Store detail of setting |
| Notification | Store detail of notification |

##### **4.3.8.3.1 user table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| UserId | This is unique id for user | int | Yes | Yes | PK |
| UserName | This is name for login | nvarchar(50) | Yes | Yes |  |
| FullName | This is name for payment | nvarchar(50) | No | No |  |
| Email | This is email of user | nvarchar(50) | Yes | No |  |
| Phone | This is phone number of user | nvarchar(50) | Yes | No |  |
| Address | This is address of user | nvarchar(MAX) | Yes | No |  |
| Active | This is status when user registers | bit | Yes | No |  |
| IsVerified | User is verified by admin | bit | Yes | No |  |
| BankName | This is name of bank for transaction | nvarchar(50) | No | No |  |
| BankAccount | This is user account name | nvarchar(50) | No | No |  |
| IdentityCard | This is user identity card | Nvarchar(10) | No | No |  |
| Sellable | This is status for user can sell or not | bit | Yes | No |  |
| CreatedDate | This is date when user creates an account | datetime | No | No |  |
| ModifiedDate | This is date when user changes personal information | datetime | No | No |  |
| ProvinceId | This is province when user registers | int | Yes | No | FK |

##### **4.3.8.3.2 ticket table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| TicketId | This is unique id for ticket | int | Yes | Yes | PK |
| SeriesNumber | This is series number on ticket | nvarchar(50) | No | No |  |
| SellPrice | This is money that user want to sell a ticket | float | Yes | No |  |
| ReceiveMoney | This is money user receives after tax | float | Yes | No |  |
| ShippingCost | This is ticket shipping cost to buyer | float | No | No |  |
| Seat | This is number of seat in ticket | nvarchar(MAX) | No | No |  |
| Description | This is ticket description details | nvarchar(MAX) | No | No |  |
| Status | This is status of ticket: Pending, Ready, Delete, UserApprove, OnTransaction, Invalid | int | Yes | No |  |
| UserId | This is id of user posted ticket | int | Yes | No | FK |
| EventId | This is id of event in this ticket | int | Yes | No | FK |
| AdminModifiedDate | This is date ticket is modified by admin | datetime | No | No |  |
| CreatedDate | This is date ticket is created | datetime | No | No |  |
| ModifiedDate | This is date ticket is modified by user | datetime | No | No |  |
| TranFullName | This is name of buyer | nvarchar(50) | No | No |  |
| TranType | This is transaction type user choose for payment | Int | No | No |  |
| TranShipDate | This is date buyer receive ticket | datetime | No | No |  |
| TranDescription | This is transaction description details | nvarchar(MAX) | No | No |  |
| TranAddress | This is address of buyer | nvarchar(MAX) | No | No |  |
| TranStatus | This is status for transaction: Unpaid, Paid, Delivered, Received, Reported, Canceled | int | No | No |  |
| TranUserId | This is id of buyer | int | No | No | FK |
| TranCreatedDate | This is date transaction is created | datetime | No | No |  |
| TranModifiedDate | This is date transaction is modified | datetime | No | No |  |
| TranShipCode | This is code of shipping | nvarchar(50) | No | No |  |
| TranPaymentStatus | This is status of transaction to determine admin to pay for user or not | int | No | No |  |

##### **4.3.8.3.3 venue table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| VenueId | This is unique id of the venue | int | Yes | Yes | PK |
| VenueName | This is name of venue | nvarchar(50) | Yes | No |  |
| Address | This is address of venue | nvarchar(MAX) | Yes | No |  |
| ProvinceId | This is id of province | int | Yes | No | FK |
| Status | This is status of venue: Approve, disapprove, delete | int | Yes | No |  |
| Description | This is venue description details | nvarchar(MAX) | No | No |  |

##### **4.3.8.3.4 province table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| ProvinceId | This is unique id for each province | int | Yes | Yes | PK |
| ProvinceName | This is province’s name | nvarchar(50) | Yes | No |  |

##### **4.3.8.3.5 event table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| EventId | This is unique id for each event | int | Yes | Yes | PK |
| EventName | This is name of event | nvarchar(50) | Yes | No |  |
| EventImage | This is image for each event | nvarchar(MAX) | No | No |  |
| Artist | The event usually have artist, this is name of artist | nvarchar(50) | No | No |  |
| HoldDate | This is date when event is hold | datetime | Yes | No |  |
| Description | This is description for event | nvarchar(MAX) | No | No |  |
| Status | Status of event: Approve, disapprove, delete, trading, outdate | int | Yes | No |  |
| CategoryId | This is id for category | int | Yes | No | FK |
| VenueId | This is id for venue | int | Yes | No | FK |
| CreatedDate | The date event is created | datetime | No | No |  |
| ModifiedDate | The date event is modified | datetime | No | No |  |

##### **4.3.8.3.6 userfollowevent table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| FollowEventId | This is unique id for each follow event | int | Yes | Yes | PK |
| UserId | This is unique id for each user, the user to follow event | int | Yes | No | FK |
| EventId | This is unique id for each event, the event was follow by user | int | Yes | No | FK |
| FollowType | This is unique id for each follow event | int | Yes | No |  |

##### **4.3.8.3.7 request table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| UserId | This is user’s id, the user create this request | int | Yes | No | PK, FK |
| EventId | This is id for each event, the event was request by user | int | Yes | No | PK, FK |
| Status | This is status of request: Open, close | int | Yes | No |  |
| Description | This is description for request | nvarchar(MAX) | No | No |  |
| CreatedDate | The date create this request | datetime | No | No |  |
| ModifiedDate | The date modified request. | datetime | No | No |  |

##### **4.3.8.3.8 ticketresponse table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| UserId | This is user’s id, the user response for ticket | int | Yes | No | PK, FK |
| EventId | This is event’s id | int | Yes | No | PK, FK |
| TicketId | This is ticket’s id, the ticket have response. | int | Yes | No | PK,FK |
| Status | This is status of ticket response: Appear, delete | int | Yes | No |  |
| Description | This is description for response. | nvarchar(MAX) | No | No |  |

##### **4.3.8.3.9 categories table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| CategoryId | This is unique id for each category. | int | Yes | Yes | PK |
| CategoryName | This is name of category. | nvarchar(50) | Yes | No |  |
| ParentCategoryId | This is id for each parent category | int | No | No | FK |
| Status | This is status of category: Active, deactive, delete | int | Yes | No |  |
| Description | This is description for category | nvarchar(MAX) | No | No |  |

##### **4.3.8.3.10 setting**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary/ Foreign key |
| Id | This is unique id for each setting | int | Yes | Yes | PK |
| SettingName | This is name of setting | nvarchar(50) | Yes | No |  |
| Value | This is value of setting | nvarchar(50) | Yes | No |  |

##### **4.3.8.3.11 notification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| NotificationId | This is unique id for each notification | int | Yes | Yes | PK |
| UserId | This is id for each user | int | Yes | No | FK |
| SenderId | This is id of object occurs notification | int | Yes | No |  |
| ActivityType | This is type of notification | nvarchar(50) | Yes | No |  |
| ObjectType | This is type of object occurs notification | nvarchar(50) | No | No |  |
| ObjectTitle | This is title of object occurs notification | nvarchar(250) | No | No |  |
| ObjectUrl | This is Url of notification | nvarchar(MAX) | No | No |  |
| Content | This is content of notification | nvarchar(MAX) | No | No |  |
| IsUnread | To know notification is read or not | bit | Yes | No |  |
| CreatedDate | This is date notification is created | datetime | No | No |  |
| ModifiedDate | This is date notification is modified | datetime | No | No |  |

## 