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

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Drop It |
|  |
| |  |  | | --- | --- | | **Group 08** | | | **Group Members** | Lê Khôi Phong  Đào Như Tùng  Công Minh Hiếu  Nguyễn Thanh Tùng  Đặng Quốc Duy | | **Supervisor** | Lâm Hữu Khánh Phương | |
|  |

- Ho Chi Minh City, 06/2013 -

# 4. SOFTWARE DESIGN DESCRIPTION

## 4.1. 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" \o "Database). Database design section shows and explains the physical model of system database which includes all tables and their relationship

## 4.2. System Architectural Design

### 4.2.1 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.

### 4.2.2 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.

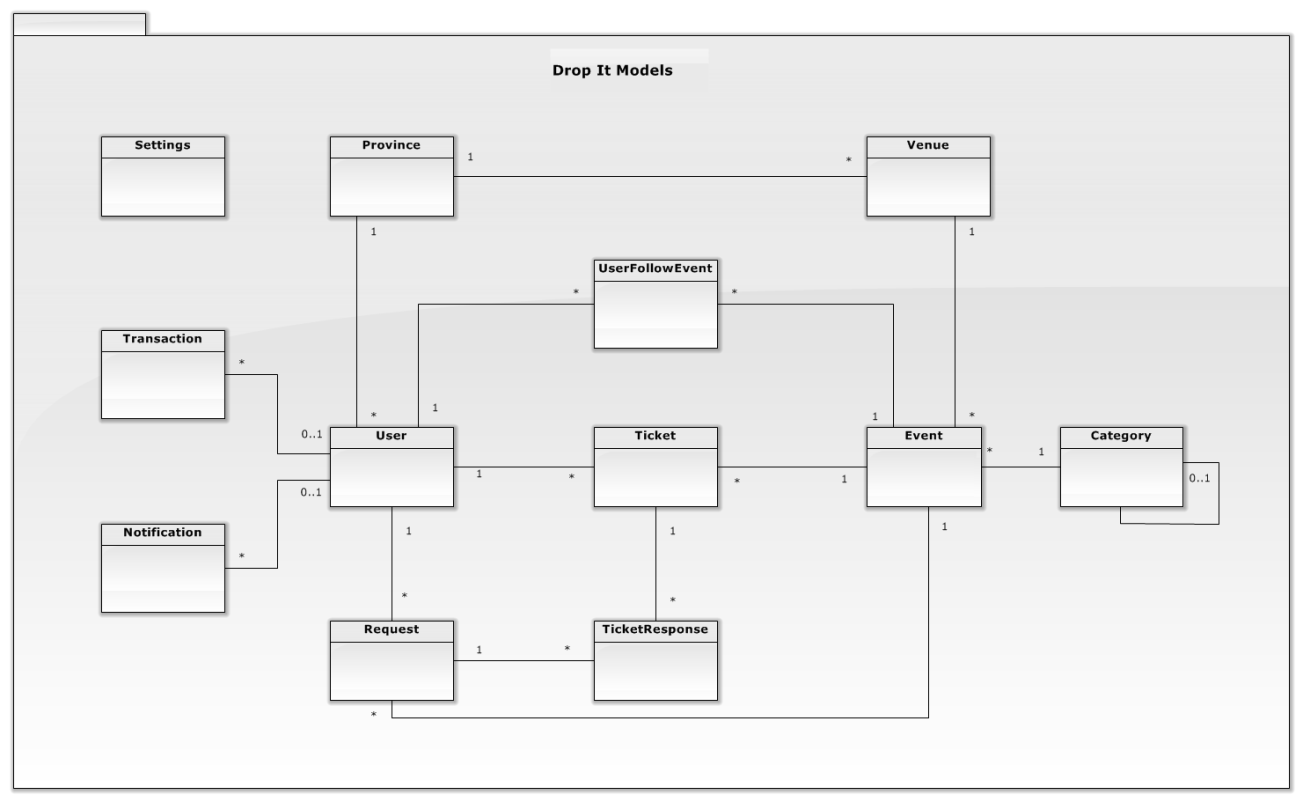
## 4.3. PACKAGE Diagram



**Figure 4.3.1 – General Package Diagram**

## 4.4. Class Detailed Description

### 4.4.1 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 |

### 4.4.2 DropIt.Controllers Package

**Figure 4.4.2.1 – 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 |

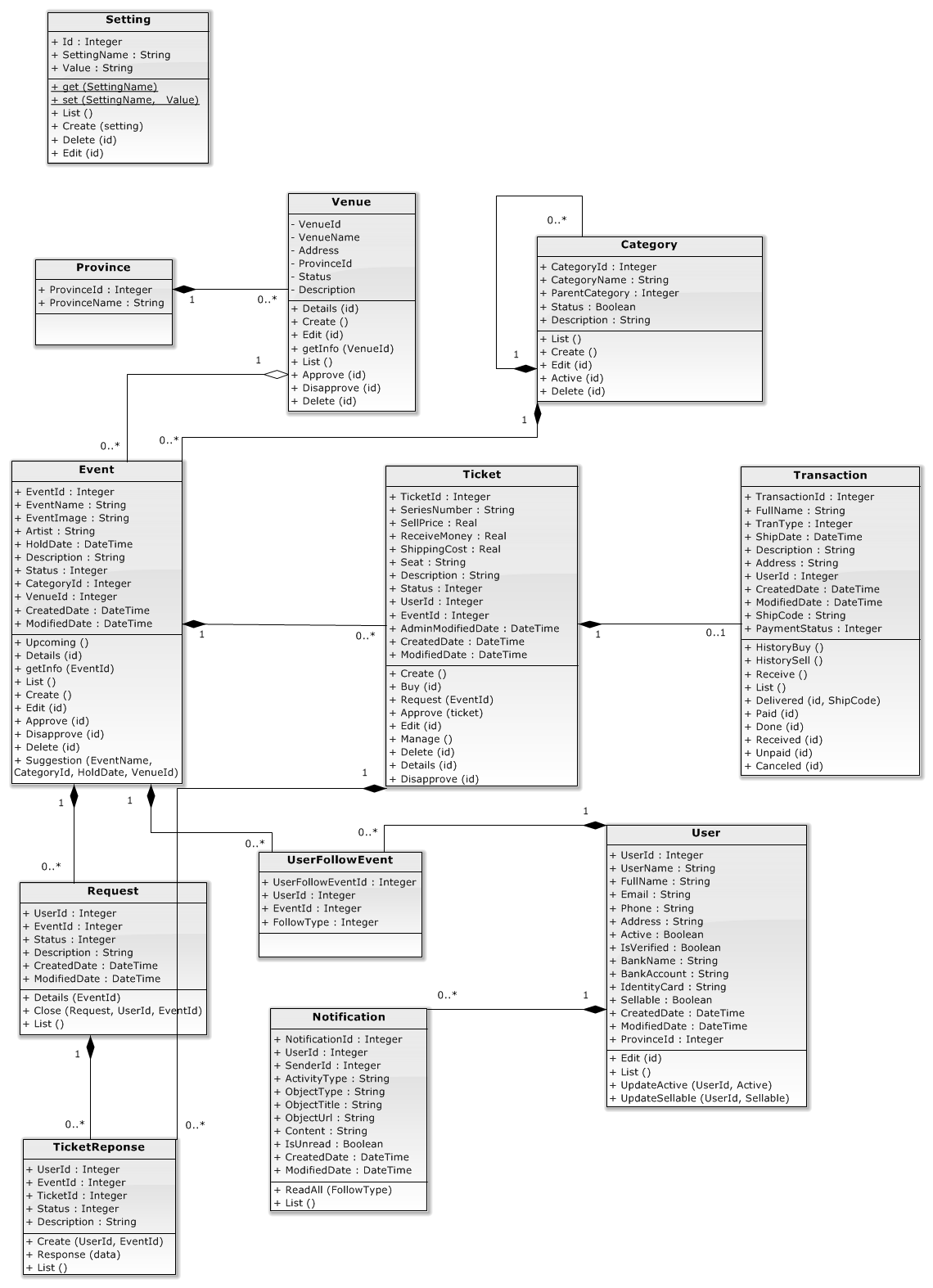
*UserController*

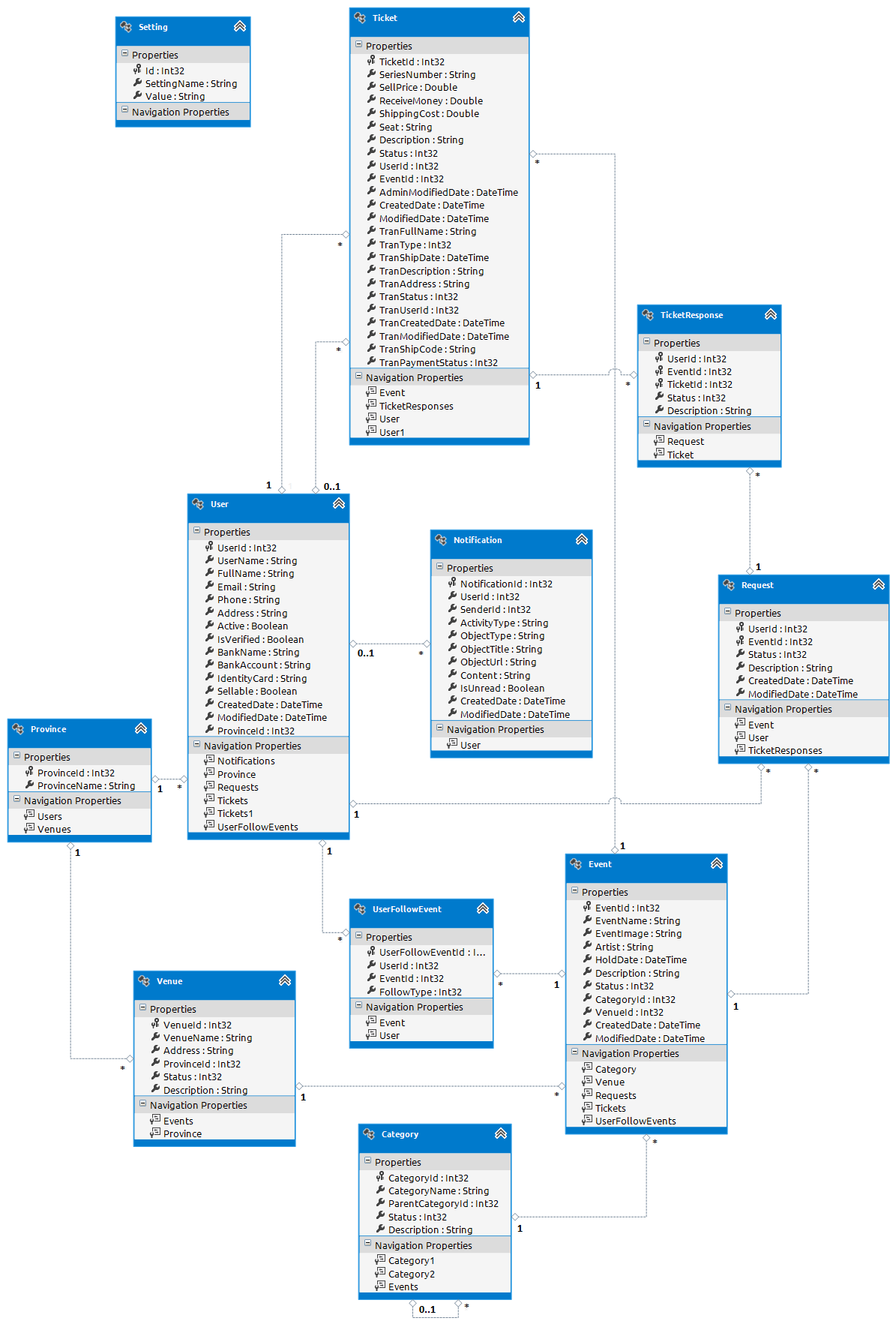
|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Display Manage User Index page |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null,int isActive = -1,int isSellable =-1,String UserName = "") | JsonResult | Paging for user |
| UpdateActive(int 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 |

### 4.4.3 Class Diagram





## 4.5. Sequence Diagram

### 4.5.1 Get list ticket



### 4.5.2 Manage ticket

### 4.5.3 Manage transaction



### 4.5.4 Manage User

### 4.5.5 Create Event



## 4.6 Activity diagram

### 4.6.1 Create Ticket



### 4.6.2 Request Ticket



## 4.7. State Diagram

### 4.7.1 Ticket State Diagram



**Figure 4.7.1.1 – 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.

### 4.7.2 Event State Diagram

**Figure 4.7.2.1 – Event State Diagram**

**States Description:**

Disapprove: The event does not present in user site.

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

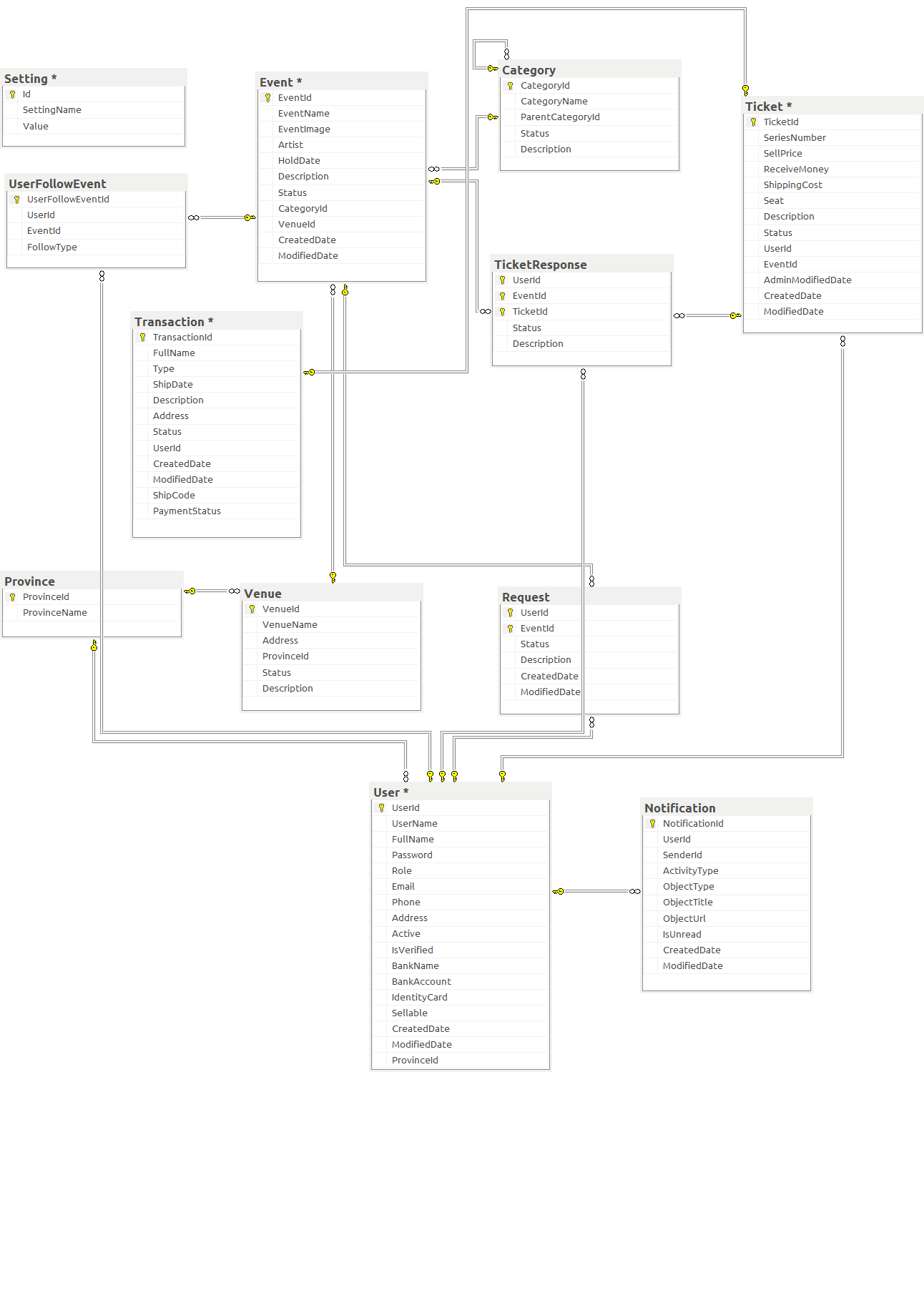
Approve: The event presents in user site.

Trading: The event has ticket is transacted.

OutDate: The event is out of date.

## 4.8. Database Design or Data Structures

### 4.8.1 CONCEPT Database diagram



**Figure 4.8.1.2 – Concept database diagram**

### 4.8.2 Database diagram

**Figure 4.8.1.2 Database Diagram**

### 4.8.3 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 |

#### 4.8.3.1 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 |

#### 4.8.3.2 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 |  |

#### 4.8.3.3 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 |  |

#### 4.8.3.4 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 |  |

#### 4.8.3.5 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 |  |

#### 4.8.3.6 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 |  |

#### 4.8.3.7 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 |  |

#### 4.8.3.8 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 |  |

#### 4.8.3.9 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 |  |

#### 4.8.3.10 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 |  |

#### 4.8.3.11 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 |  |