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

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

- 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" \o "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



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 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 x – General Package Diagram

## 4.4. Class Detailed Description

### 4.4.1 DropIt.Models Package

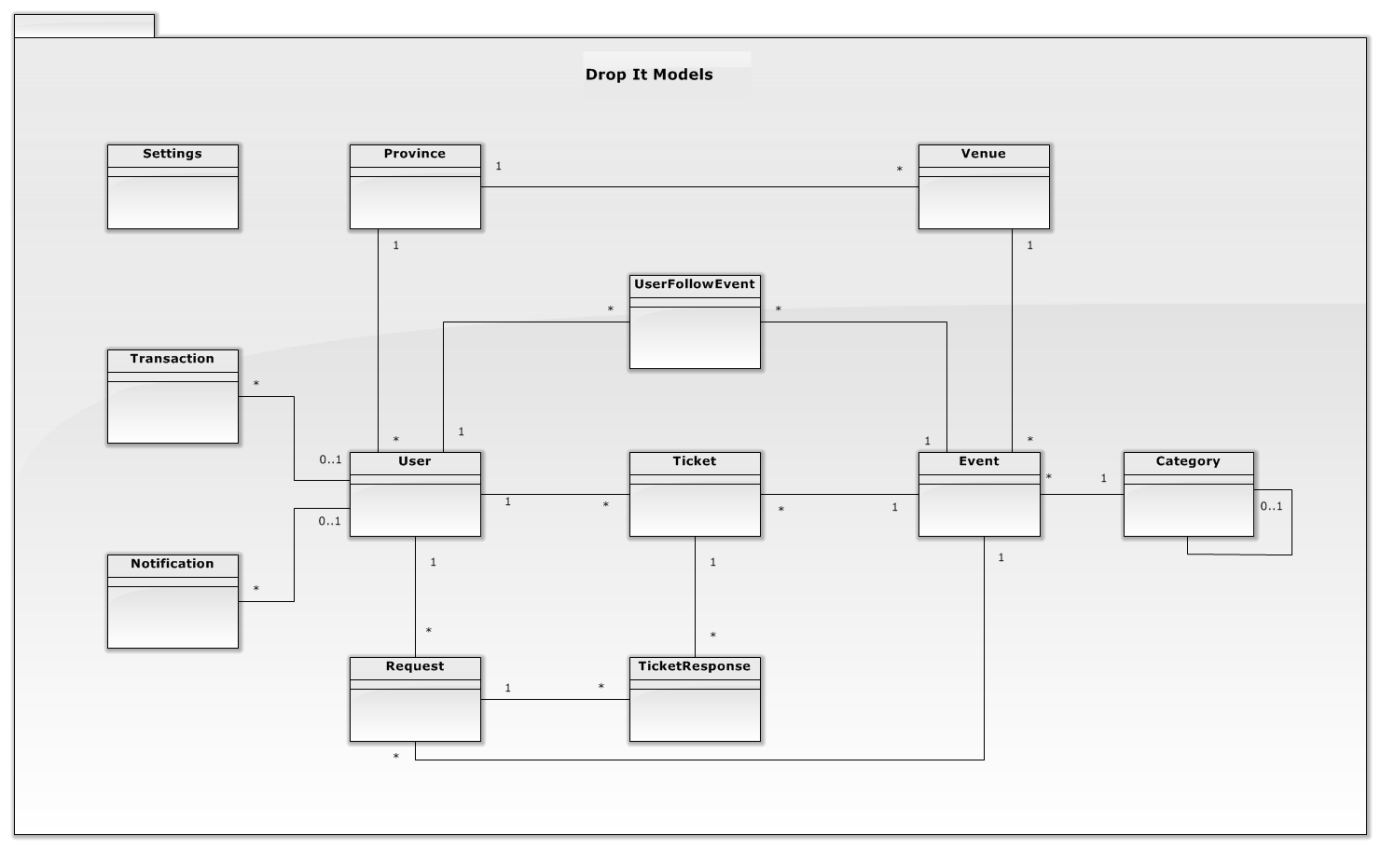


Figure x – DropIt.Models Package

*Province*

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

*Category*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*Event*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |
|  |  |  |

*Notification*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*Request*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*Setting*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*Ticket*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*TicketResponse*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*User*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*UserFollowEvent*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

*Venue*

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
|  |  |  |
|  |  |  |

##### 4.4.1.4 <ADMIN> MANAGE EVENT

###### 4.4.1.4.1 <ADMIN> CREATE EVENT

###### 4.4.1.4.2 <ADMIN> UPDATE EVENT

###### 4.4.1.4.3 <ADMIN> DELETE EVENT

##### 4.4.1.5 <ADMIN> MANAGE TRANSACTIONS

###### 4.4.1.5.1 <ADMIN> UPDATE STATUS TRANSACTION

##### 4.4.1.6 <ADMIN> MANAGE TICKET

###### 4.4.1.6.1 <ADMIN> APPROVE TICKET

###### 4.4.1.6.2 <ADMIN> UPDATE TICKET

###### 4.4.1.6.3 <ADMIN> DISAPPROVE TICKET

##### 4.4.1.7 <ADMIN> MANAGE VENUE

###### 4.4.1.7.1 <ADMIN> CREATE VENUE

###### 4.4.1.7.2 <ADMIN> UPDATE VENUE

###### 4.4.1.7.3 <ADMIN> DELETE VENUE

##### 4.4.1.8 <ADMIN> MANAGE CONFIG

###### 4.4.1.8.1 <ADMIN> REQUEST AUTO DELETE

###### 4.4.1.8.2 <ADMIN> TICKET AUTO DETLETE

##### 4.4.1.9 <ADMIN> MANAGE PROVINCE

###### 4.4.1.9.1 <ADMIN> CREATE PROVINCE

###### 4.4.1.9.2 <ADMIN> UPDATE PROVINCE

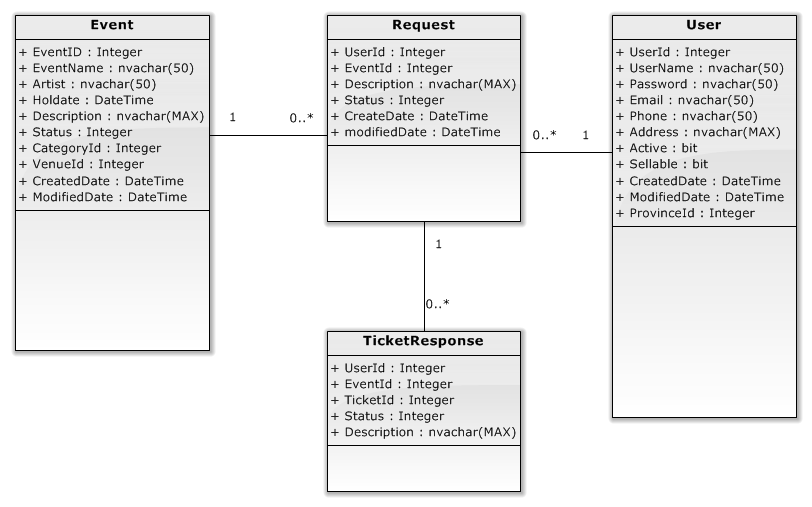
###### 4.4.1.9.3 <ADMIN> DELETE PROVINCE

##### 4.4.2 <user>detail description of components

###### 4.4.2.2 <USER> REQUEST

4.4.2.2.1 <USER> CREATE REQUEST

*Class Diagram*



*Class Diagram Explanation*

Request:

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| UserId | Integer | ID of user |
| EventId | Integer | The identifier of order |
| Description | Nvachar(MAX) | The identifier of customer |
| Status | Integer | Name of customer |
| CreateDate | DateTime | Name of order |
| ModifiedDate | DateTime | District of customer’s address |

**Methods**

Event:

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| EventID | Integer | ID |
| EventName | Nvachar(50) | The identifier of order |
| Artist | Nvachar(50) | The identifier of customer |
| Holdate | DateTime | The date, this event … |
| Description | Nvachar(MAX) | Name of order |
| Status | Integer | Phone number of customer |
| CategoryId | Integer | House number of order |
| VenueId | Integer | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |

**Methods:**

User:

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| UserId | Integer | ID |
| UserName | Nvachar(50) | The identifier of order |
| Password | Nvachar(50) | The identifier of customer |
| Email | Nvachar(50) | Name of customer |
| Phone | Nvachar(50) | Name of order |
| Address | Nvachar(MAX) | Phone number of customer |
| Active | bit | House number of order |
| Sellable | Bit | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |
| ProvinceId | Integer |  |

**Methods:**

ResponseTicket:

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| UserId | Integer | ID |
| EventId | Integer | The identifier of order |
| TicketId | Integer | The identifier of customer |
| Status | Integer | Name of customer |
| Description | Nvachar(MAX) | Name of order |

**Methods:**

4.4.2.2.2 <USER> CLOSE REQUEST

Optional

###### 4.4.2.3 <USER> RESPONSE

4.4.2.3.1 <USER> CREATE RESPONSE

4.4.2.3.2 <USER> EDIT RESPONSE

###### 4.4.2.4 <USER> BUY TICKET

4.4.2.4.1 <USER> MADE PAYMENT

4.4.2.4.2 <USER> UPDATE TRANSACTION STATUS

###### 4.4.2.5 <USER> SELL TICKET

4.4.2.5.1 <USER> POST TICKET

*Class Diagram*



*Class Diagram Explanation*

Ticket

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| EventID | Integer | ID |
| EventName | Nvachar(50) | The identifier of order |
| Artist | Nvachar(50) | The identifier of customer |
| Holdate | DateTime | Name of customer |
| Description | Nvachar(MAX) | Name of order |
| Status | Integer | Phone number of customer |
| CategoryId | Integer | House number of order |
| VenueId | Integer | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |

**Methods:**

Event:

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| EventID | Integer | ID |
| EventName | Nvachar(50) | The identifier of order |
| Artist | Nvachar(50) | The identifier of customer |
| Holdate | DateTime | Name of customer |
| Description | Nvachar(MAX) | Name of order |
| Status | Integer | Phone number of customer |
| CategoryId | Integer | House number of order |
| VenueId | Integer | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |

**Methods:**

User

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| EventID | Integer | ID |
| EventName | Nvachar(50) | The identifier of order |
| Artist | Nvachar(50) | The identifier of customer |
| Holdate | DateTime | Name of customer |
| Description | Nvachar(MAX) | Name of order |
| Status | Integer | Phone number of customer |
| CategoryId | Integer | House number of order |
| VenueId | Integer | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |

**Methods:**

TicketResponse

**Attributes:**

|  |  |  |
| --- | --- | --- |
| **Parameter/Return** | **Type** | **Description** |
| EventID | Integer | ID |
| EventName | Nvachar(50) | The identifier of order |
| Artist | Nvachar(50) | The identifier of customer |
| Holdate | DateTime | Name of customer |
| Description | Nvachar(MAX) | Name of order |
| Status | Integer | Phone number of customer |
| CategoryId | Integer | House number of order |
| VenueId | Integer | House number of customer |
| CreatedDate | DateTime | Ward of customer’s address |
| ModifiedDate | DateTime | District of customer’s address |

**Methods:**

4.4.2.5.2 <USER> EDIT TICKET SELL

4.4.2.5.3 <USER> DELETE SELL TICKET

###### 4.4.2.6 <USER> CHANGE PERSONAL INFORMATION

Optional

##### 4.4.3 <Guest> detail description of components

###### 4.4.3.1 <GUEST> VIEW EVENT

4.4.3.1.1 <GUEST> VIEW EVENT DETAIL

Optional

###### 4.4.3.2 <GUEST> VIEW REQUEST

4.4.3.2.1 <GUEST> VIEW REQUEST DETAIL

Optional

###### 4.4.3.3 <GUEST> SEARCH EVENT

4.4.3.3.1 <GUEST> SEARCH EVENT BY TITLE

4.4.3.3.2 <GUEST> SEARCH EVENT BY ARTIST

##### 4.4.4 <Unauthorizeduser> detail description of component

###### 4.4.4.1 <UNAUTHORIZEDUSER> LOGIN

Optional.

###### 4.4.4.2 <UNAUTHORIZEDUSER> REGISTER

Optional.

### 4.4.2 DropIt.Controllers Package

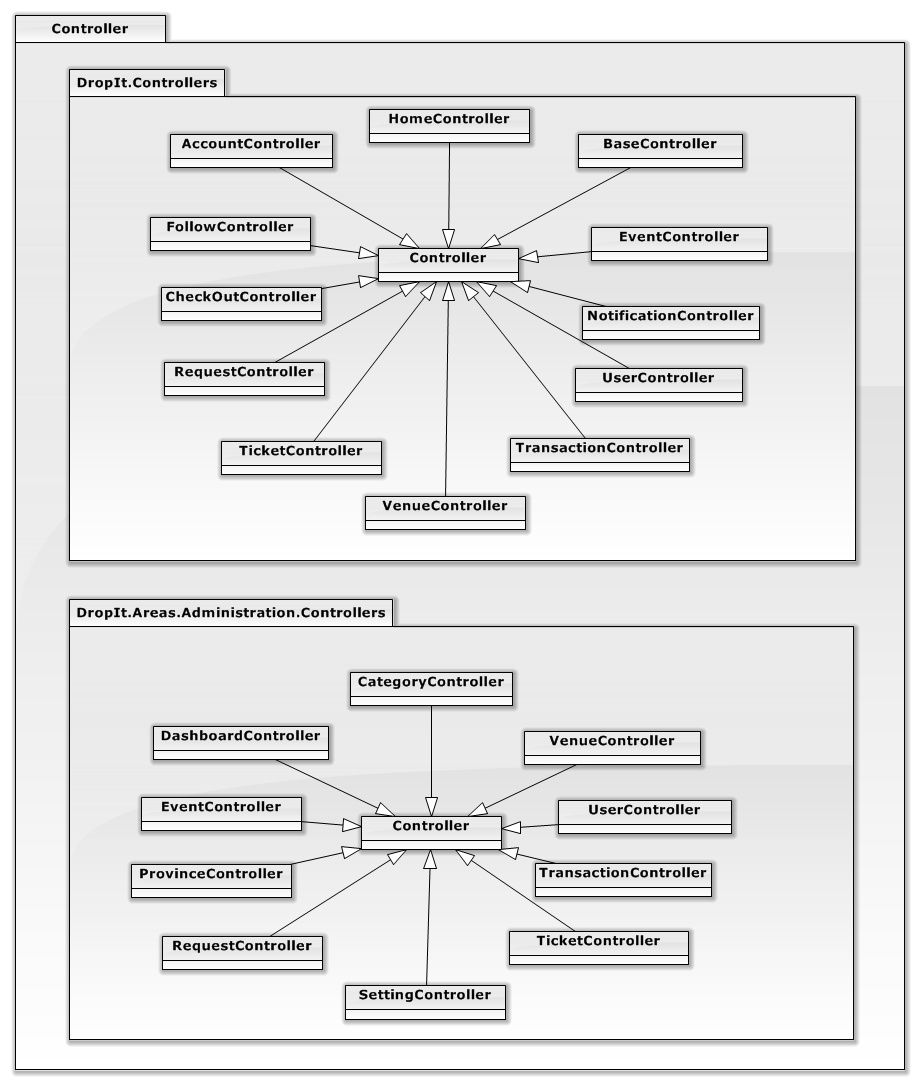


Figure x – Controller Package

DropIt.Controllers

*AccountController*

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

*BaseController*

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

*CategoryController*

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

*CheckOutController*

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

*EventController*

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

*FollowController*

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

*HomeController*

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

*NotificationController*

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

*ProvinceController*

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

*RequestController*

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

*TicketController*

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

*TicketResponseController*

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

*TransactionController*

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

*UserController*

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

*VenueController*

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

DropIt.Areas.Administration.Controllers

*CategoryController*

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

*DashboardController*

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

*EventController*

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

*ProvinceController*

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

*RequestController*

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

*SettingController*

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

*TicketController*

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

*TransactionController*

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

*UserController*

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

*VenueController*

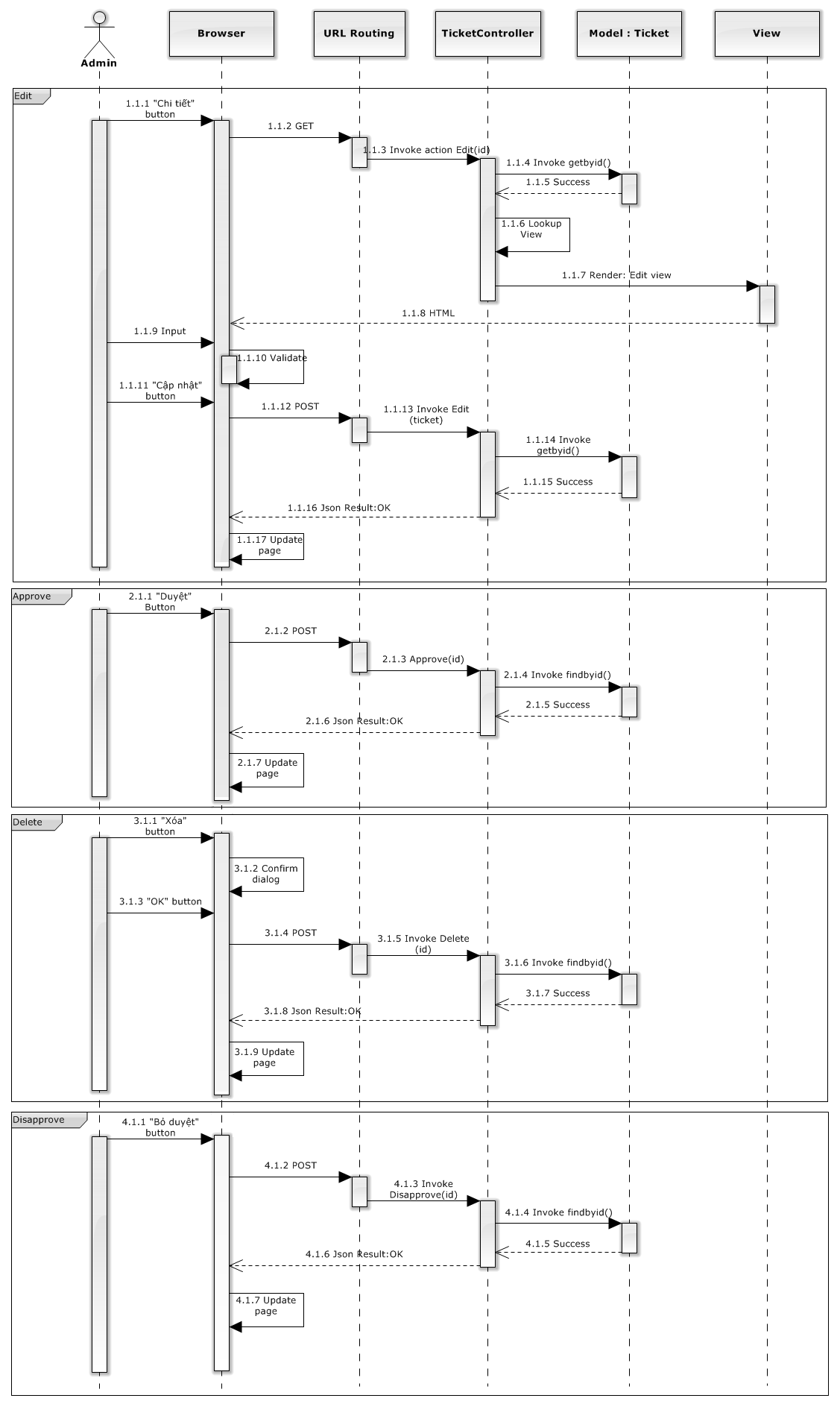
|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
|  |  |  |
|  |  |  |

## 4.5. Sequence Diagram

### 4.5.1 Get list ticket



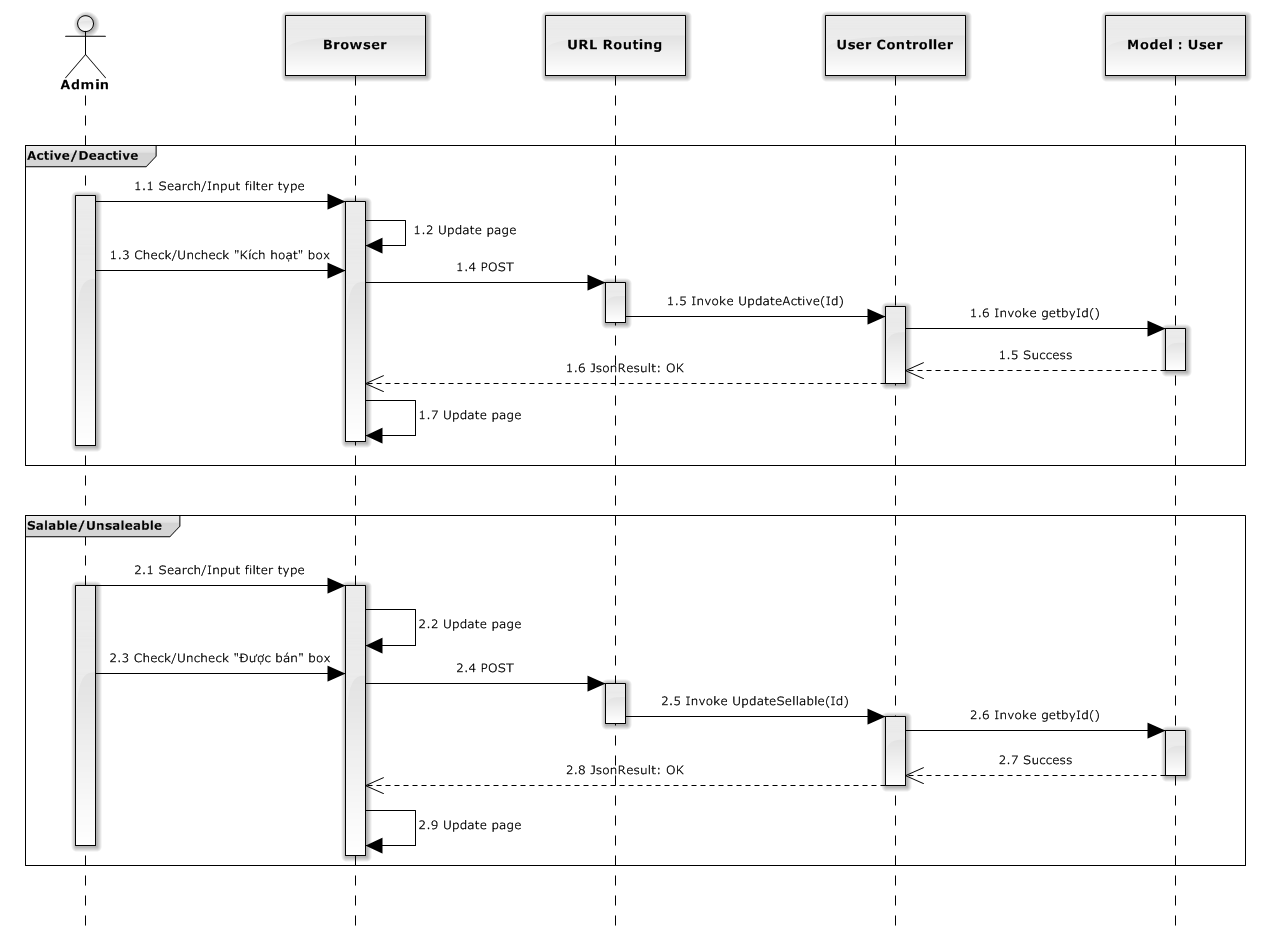
### 4.5.2 Manage ticket



### 4.5.3 Manage transaction



### 4.5.4 Manage User

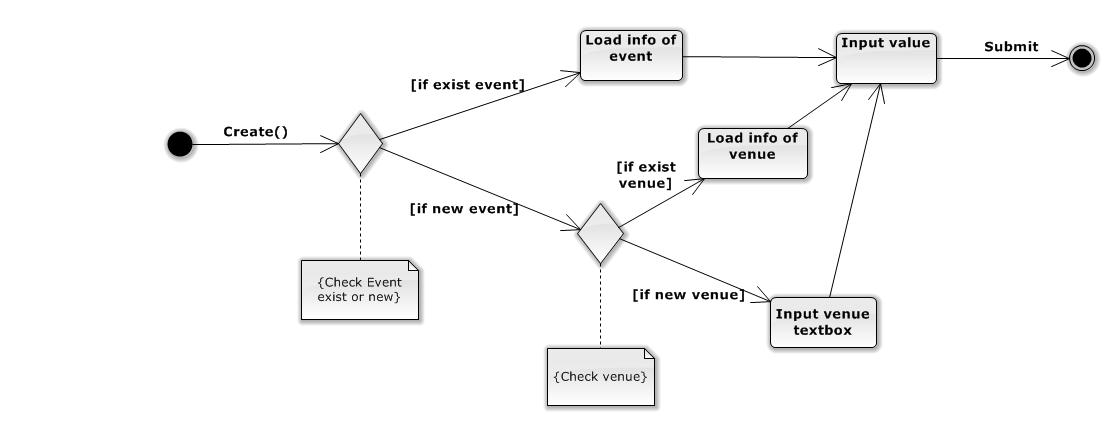


### 4.5.5 Create Event



## 4.6 Activity diagram

### 4.6.1 Create Ticket



## 4.7. State Diagram

### 4.7.1 Ticket State Diagram

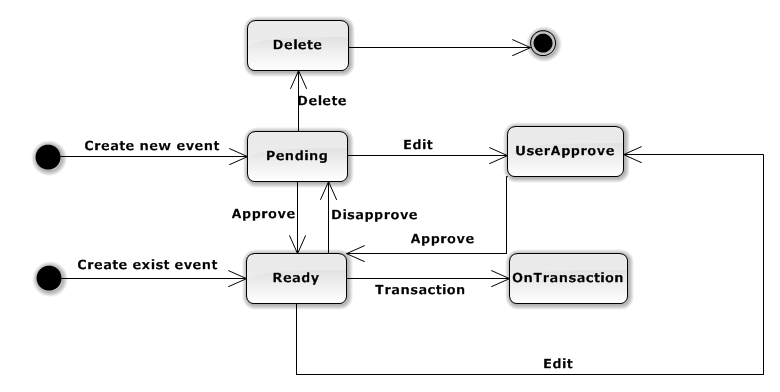


Figure x – Ticket State Diagram

States Description:

Delete: …

Pending: …

UserApprove: …

OnTransaction: …

Ready: …

### 4.7.2 Event State Diagram



Figure x – Event State Diagram

States Description:

Disapprove: ……

Delete: …..

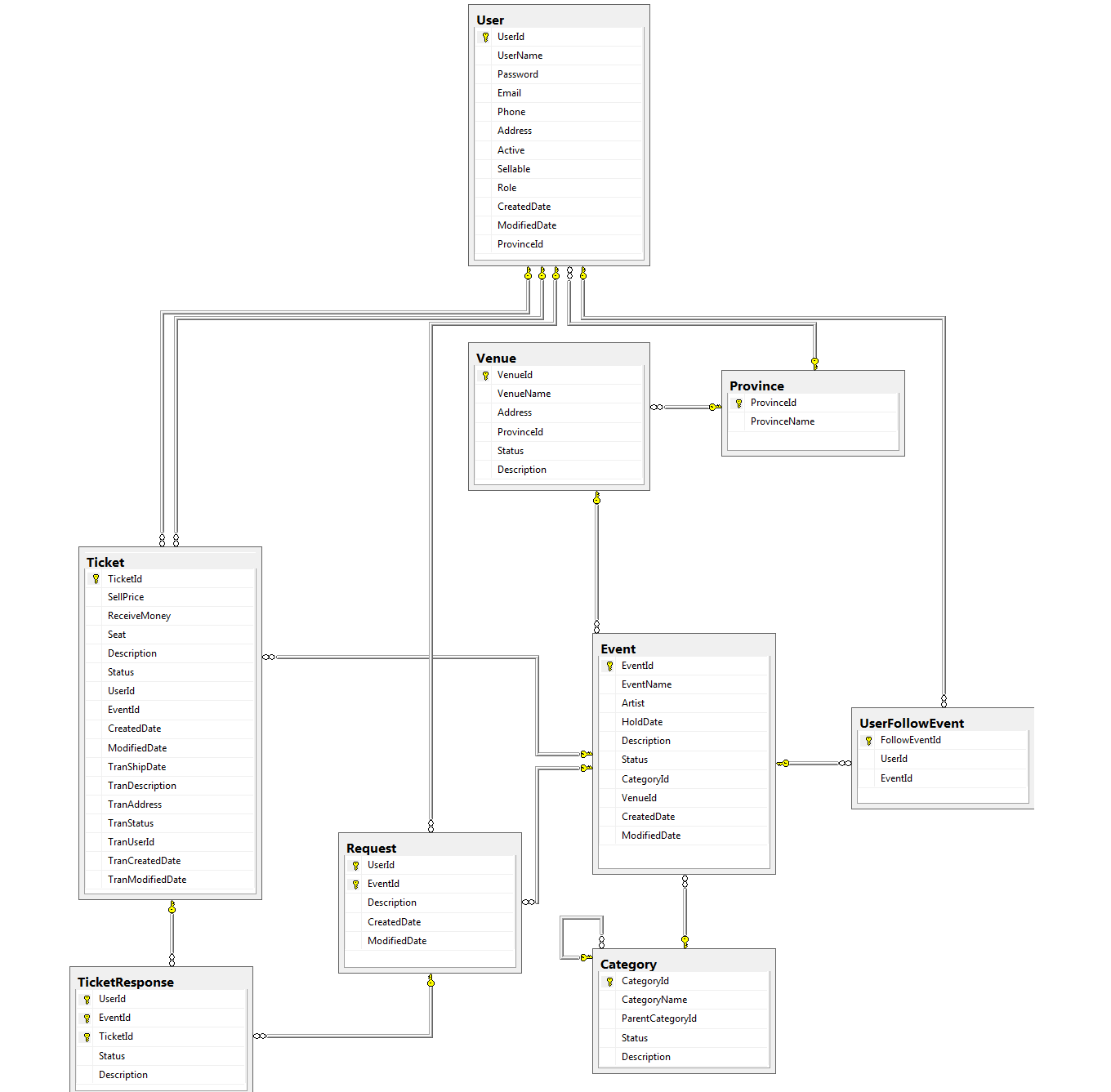
Approve: …

Trading: ….

OutDate: …

## 4.8. Database Design or Data Structures

### 4.6.1 Database diagram



### 4.6.2 Table

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| Users | Store detail of user |
| Ticket | Store feedback response content |
| Venues | Store detail of venue |
| Events | Store detail of event |
| UserFollowEvent | Store detail of follow |
| Requests | Store detail of request |
| TicketResponse | Store detail of response |
| Categories | Store detail of category |

#### 4.6.2.1 users 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 | No |  |
| 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 | Boolean | No | No |  |
| BankName | This is name of bank for transaction | nvarchar(50) | No | No |  |
| BankAccount | This is user bank account name | nvarchar(50) | No | No |  |
| CMND | This is user identity card | Nvarchar(50) | 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 | Yes | No |  |
| ModifiedDate | This is date when user changes personal information | datetime | Yes | No |  |
| ProvinceId | This is province when user registers | int | Yes | No | FK |

#### 2.6.2.2 ticket table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| TicketId | This is unique id for ticket | int | Yes | Yes | PK |
| 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 |  |
| 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 for the ticket was approve. | int | Yes | No |  |
| UserId | This is id of user, the user have this ticket. | int | Yes | No | FK |
| EventId | This is id of event, event have this ticket. | int | Yes | No | FK |
| AdminModifiedDate | This is edit information date of ticket by admin | datetime | No | No |  |
| CreatedDate | This is date to create ticket on website. | datetime | No | No |  |
| ModifiedDate | This is date to edit information of ticket. | datetime | No | No |  |
| TranType | This is transaction type user choose for payment | Int | No | No |  |
| TranDescription | This is transaction description details | nvarchar(MAX) | No | No |  |
| TranAddress | This is address of user who buy ticket | nvarchar(MAX) | No | No |  |
| TranStatus | This is status for transaction. | int | No | No |  |
| TranUserId | This is unique id of user who buy ticket | int | No | No | FK |
| TranCreatedDate | This is create transaction date | datetime | No | No |  |
| TranModifiedDate | This is modified transaction date | datetime | No | No |  |

#### 2.6.2.3 venues 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 unique id of province, the province have venue. | int | Yes | No | FK |
| Status | This is status of venue, the venue was approved, or delete. | int | Yes | No |  |
| Description | This is venue description details | nvarchar(MAX) | No | No |  |

#### 2.6.2.4 provinces 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 |  |

#### 2.6.2.5 events 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 | . | datetime | Yes | No |  |
| Description | This is description for event | nvarchar(MAX) | No | No |  |
| Status | Status for event was approve or disapprove | int | Yes | No |  |
| CategoryId | This is unique id for each category. | int | Yes | No | FK |
| VenueId | The address for event. | int | Yes | No | FK |
| CreatedDate | .the date to create event. | datetime | No | No |  |
| ModifiedDate | The date to edit event. | datetime | No | No |  |

#### 2.6.2.6 userfollowevent table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| FollowEventId | This is unique id for each followevent. | int(10) | Yes | Yes | PK |
| UserId | This is unique id for each user, the user to follow event. | varchar(100) | Yes | No | FK |
| EventId | This is unique id for each event, the event was follow by user. | varchar(2000) | Yes | No | FK |

#### 2.6.2.7 requests table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Member Name | Description | Type | Mandatory | Unique? | Primary / Foreign key |
| UserId | This is user’s id, the user create this request. | int | Yes | Yes | PK, FK |
| EventId | This is unique id for event, the event was request by user. | int | Yes | Yes | PK, FK |
| Status | This is status of request. | 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 |  |

#### 2.6.2.8 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 |
| 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 |  | int | Yes | No |  |
| Description | This is description for response. | nvarchar(MAX) | No | No |  |

#### 2.6.2.9 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 unique id for each parent category. | int | No | No | FK |
| Status | This is status of category. | int | Yes | No |  |
| Description | This is description for category. | nvarchar(MAX) | No | No |  |