|  |  |
| --- | --- |
| **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" \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



**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 Designs



**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.

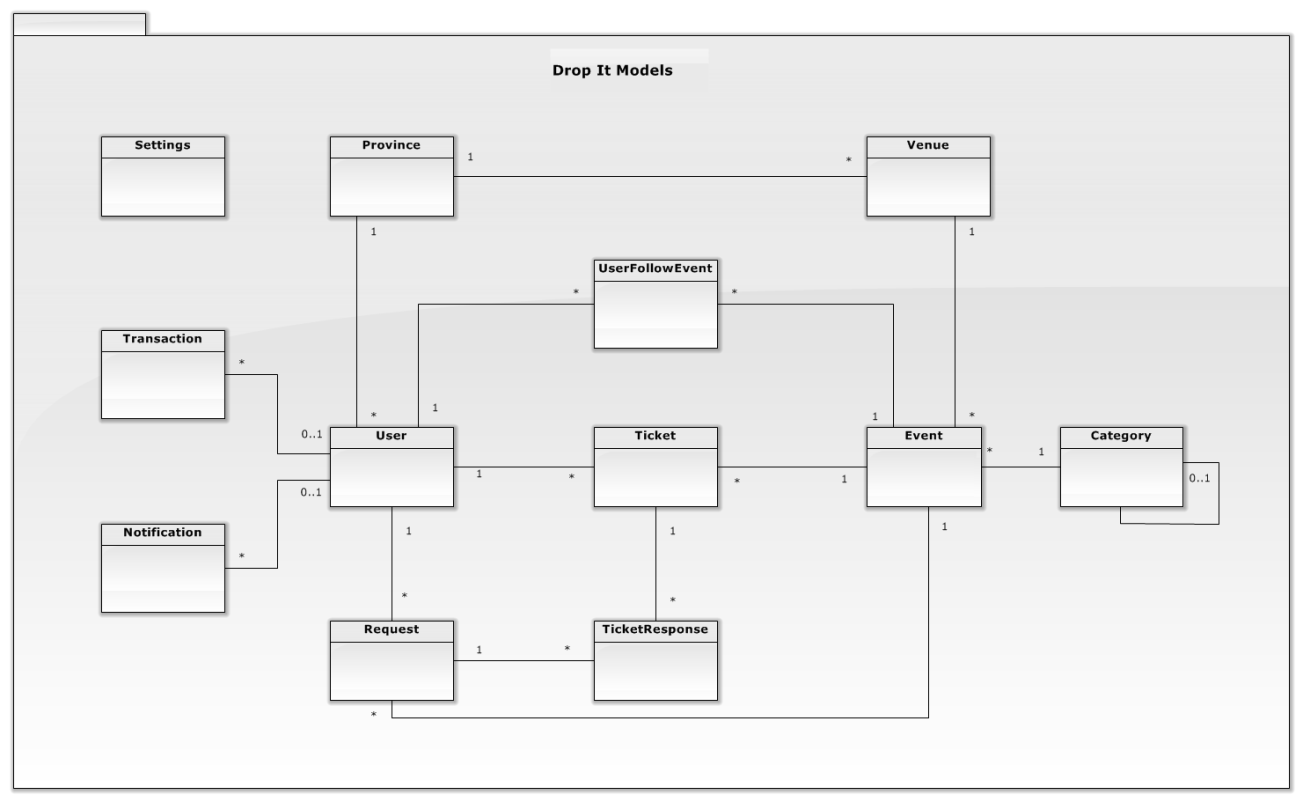
## 4.3. PACKAGE Diagram



**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 | Unique ID of each User |
| SenderId | int | Unique ID of each Sender |
| ActivityType | string | Type of each Notify Activity |
| ObjectType | string | Type of Notification |
| ObjectTitle | string | Title of each Object |
| 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 on 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 each User |
| 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 |

*4.4.1.4 <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*

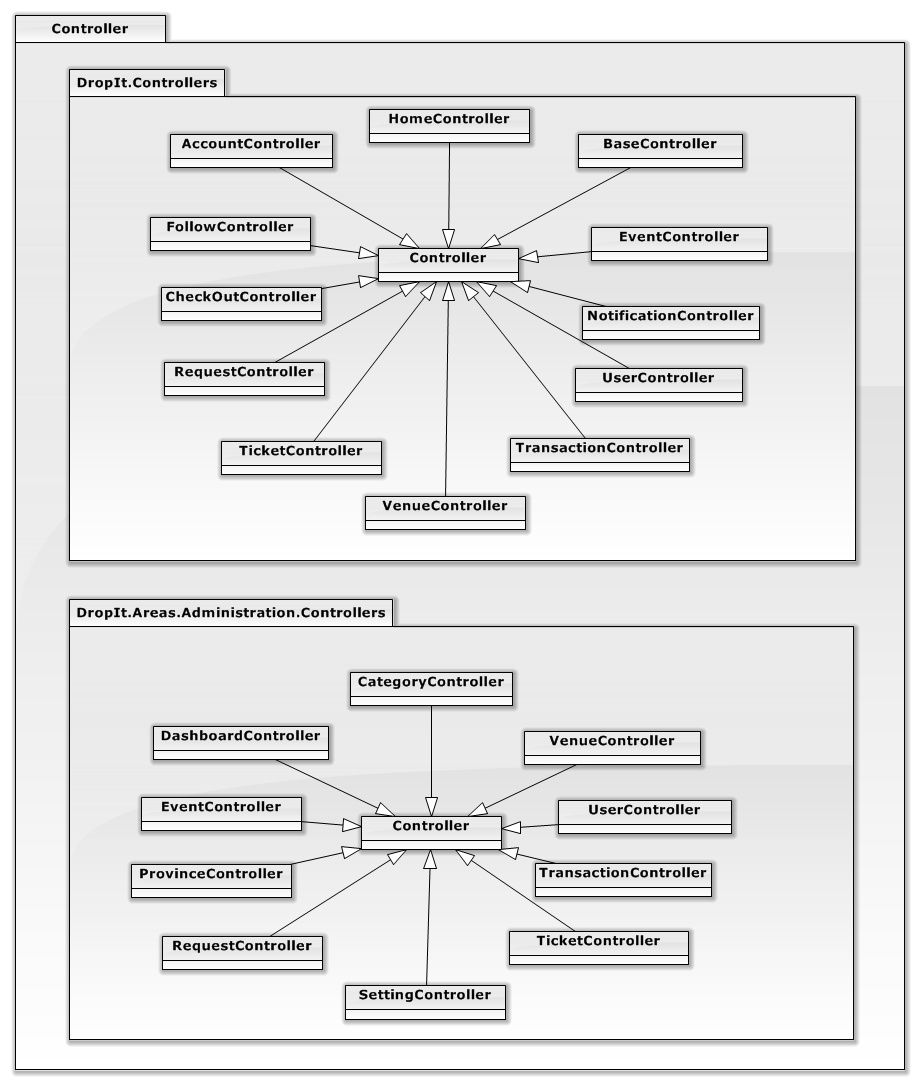
*4.4.1.9.3 <ADMIN> DELETE PROVINCE*

*4.4.2 <USER>DETAIL DESCRIPTION OF COMPONENTS*

*4.4.2.2 <USER> REQUEST*

*4.4.2.2.1 <USER> CREATE REQUEST*

### 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 |
| ExternalLoginCallback(string returnUrl) | ActionResult |  |
| ExternalLoginConfirmation(RegisterExternalLoginModel model, string returnUrl) | ActionResult |  |
| RemoveExternalLogins() | ActionResult |  |
| Disassociate(string provider, string providerUserId) | ActionResult |  |

*BaseController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| OnActionExecuted(ActionExecutedContext filterContext) |  |  |

*CategoryController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Details(int id = 0) | ActionResult | Allow user to view category details |
| Create() | ActionResult |  |
| Create(Category category) | ActionResult |  |
| Edit(int id = 0) | ActionResult |  |
| Edit(Category category) | ActionResult |  |
| Delete(int id = 0) | ActionResult |  |
| DeleteConfirmed(int id) | ActionResult |  |
| getAll() | JsonResult |  |
| Dispose(bool disposing) |  |  |

*CheckOutController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Checkout(int Id) | ActionResult |  |
| Success() | ActionResult |  |

*EventController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int PageSize = 10, int StartIndex = 0,int CategoryId = -1) | ActionResult |  |
| Upcoming() | ActionResult |  |
| Details(int id = 0) | ActionResult |  |
| FollowEvent() | ActionResult |  |
| FollowEvent(int EventId = 0) | ActionResult |  |
| Create() | ActionResult |  |
| Create(Event evt) | ActionResult |  |
| Edit(int id = 0) | ActionResult |  |
| Edit(Event evt) | ActionResult |  |
| Delete(int id = 0) | ActionResult |  |
| getInfo(int EventId) | JsonResult |  |
| Image(int Id) | ActionResult |  |
| Count() | JsonResult |  |
| DeleteConfirmed(int id) | ActionResult |  |
| Dispose(bool disposing) |  |  |

*FollowController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Event(int Id,int FollowType) | JsonResult |  |
| Events() | ActionResult |  |

*HomeController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Allow user to view homepage |
| NearEvent() | ActionResult | Allow user to view upcoming event |
| BuyIndex() | ActionResult | Allow user to view buy index page |
| SellIndex() | ActionResult | Allow user to view sell index page |
| ReduceRedundancy(List<string> strings) |  |  |
| Search(string query, string sortBy = "relevant", int PageSize = 10, int StartIndex = 0) | ActionResult |  |
| SearchAjax(string searchText) | JsonResult |  |
| About() | ActionResult | Allow user to view about page |
| Contact() | ActionResult | Allow user to view contact us page |
| GuideForNew() | ActionResult | Allow user to view guide for new user page |

*NotificationController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| ReadAll(int FollowType = -1) | JsonResult | Allow user to |
| List(int FollowType = 2,String Object="\*",Nullable<Boolean> IsUnread = null,int jTStartIndex = 0,int FromId = 0,int jtPageSize = 10,String jtSorting = "CreatedDate DESC") | JsonResult |  |

*ProvinceController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| Details(int id = 0) | ActionResult |  |
| Create() | ActionResult |  |
| Create(Province province) | ActionResult |  |
| Edit(int id = 0) | ActionResult |  |
| Edit(Province province) | ActionResult |  |
| Delete(int id = 0) | ActionResult |  |
| DeleteConfirmed(int id) | ActionResult |  |
| Dispose(bool disposing) |  |  |

*RequestController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Allow user |
| Details(int EventId) | ActionResult | Allow user to view ticket request details |
| Close(Request Request, int UserId, int EventId) | JsonResult | Allow user to close ticket |
| CloseorOpen(int EventId) | ActionResult |  |

*TicketController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult | Allow user to view homepage |
| 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 |  |
| 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 |  |
| Delete(int Id) | ActionResult | Allow user to delete ticket |
| Details (int id=0) | ActionResult | Allow user to view ticket details |

*TicketResponseController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| Create() | ActionResult |  |
| List() | ActionResult |  |

*TransactionController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| HistoryBuy() | ActionResult |  |
| HistorySell() | ActionResult |  |
| Receive(int Id) | ActionResult |  |
| Count(int id) | JsonResult |  |

*UserController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Edit(int id = 0) | ActionResult |  |
| Edit(UserViewModel user) | ActionResult |  |

*VenueController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| Details(int id = 0) | ActionResult |  |
| Create() | ActionResult |  |
| Create(Venue venue) | ActionResult |  |
| Edit(int id = 0) | ActionResult |  |
| getInfo(int VenueId) | JsonResult |  |
| Edit(Venue venue) | ActionResult |  |
| Delete(int id = 0) | ActionResult |  |
| DeleteConfirmed(int id) | ActionResult |  |
| Dispose(bool disposing) |  |  |

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

*CategoryController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int CategoryStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = "CategoryName ASC", int CategoryStatus = -1) | JsonResult |  |
| Create() | ActionResult |  |
| Create(Category Category) | JsonResult |  |
| Edit(int Id) | ActionResult |  |
| Edit(Category Category) | JsonResult |  |
| Active(int Id) | JsonResult |  |
| Deactive(int Id) | JsonResult |  |
| Delete(int Id) | JsonResult |  |
| GetOptions() | JsonResult |  |
| Dispose(bool disposing) |  |  |

*DashboardController*

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

*EventController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int EventStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = "HoldDate ASC", int EventStatus = -1) | JsonResult |  |
| Create() | ActionResult |  |
| Create(Event Event, HttpPostedFileBase EventImage) | JsonResult |  |
| Edit(int Id) | ActionResult |  |
| Edit(Event Event, HttpPostedFileBase EventImage) | JsonResult |  |
| getInfo(int EventId) | JsonResult |  |
| getEventOption() | JsonResult |  |
| Approve(int Id) | JsonResult |  |
| Disapprove(int Id) | JsonResult |  |
| Delete(int Id) | JsonResult |  |
| Dispose(bool disposing) |  |  |

*ProvinceController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null) | JsonResult |  |
| Create(Province province) | JsonResult |  |
| Update(Province province) | JsonResult |  |
| Delete(int ProvinceId) | JsonResult |  |
| GetProvinceOptions() | JsonResult |  |
| Dispose(bool disposing) |  |  |

*RequestController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int RequestStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int RequestStatus = -1) | JsonResult |  |
| Close(int UserId, int EventId) | JsonResult |  |
| Dispose(bool disposing) |  |  |

*SettingController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| List() | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null) | JsonResult |  |
| Create(SettingViewModel setting) | JsonResult |  |
| Delete(int Id) | JsonResult |  |
| Edit(Setting setting) | JsonResult |  |

*TicketController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int TicketStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int TicketStatus = -1) | JsonResult |  |
| Approve(int Id) | JsonResult |  |
| Disapprove(int Id) | JsonResult |  |
| Delete(int Id) | JsonResult |  |
| Edit(int Id) | ActionResult |  |
| Edit(Ticket Ticket) | JsonResult |  |
| Dispose(bool disposing) |  |  |

*TransactionController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int TranStatus = (int)Statuses.Transaction.Paid, int TranPaymentStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int TranStatus = -1, int TranPaymentStatus = -1) | JsonResult |  |
| Delivered(int Id, string TranShipCode) | JsonResult |  |
| Paid(int Id) | JsonResult |  |
| Done(int Id) | JsonResult |  |
| Received(int Id) | JsonResult |  |
| Unpaid(int Id) | JsonResult |  |
| Canceled(int Id) | JsonResult |  |
| Edit(int Id) | JsonResult |  |
| Edit(Ticket Ticket) |  |  |
| Dispose(bool disposing) |  |  |

*UserController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index() | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null,int isActive = -1,int isSellable =-1,String UserName = "") | JsonResult |  |
| UpdateActive(int UserId, bool Active) | JsonResult |  |
| UpdateSellable(int UserId, bool Sellable) | JsonResult |  |
| Create(User user) | JsonResult |  |
| Update(User user) | JsonResult |  |
| GetUserOptions() | JsonResult |  |
| Dispose(bool disposing) |  |  |

*VenueController*

|  |  |  |
| --- | --- | --- |
| Method | Return type | Description |
| Index(int VenueStatus = 0) | ActionResult |  |
| List(int jtStartIndex = -1, int jtPageSize = 0, string jtSorting = null, int VenueStatus = -1) | JsonResult |  |
| Create() | ActionResult |  |
| Create(Venue Venue) | JsonResult |  |
| Approve(int Id) | JsonResult |  |
| Disapprove(int Id) | JsonResult |  |
| Edit(int Id) | ActionResult |  |
| Edit(Venue Venue) | JsonResult |  |
| Delete(int Id) | JsonResult |  |
| GetProvinceOptions() | JsonResult |  |
| GetOptions() | JsonResult |  |
| Dispose(bool disposing) |  |  |

## 4.5. Sequence Diagram

### 4.5.1 Get list ticket



### 4.5.2 Manage ticket

### 4.5.3 Manage transaction

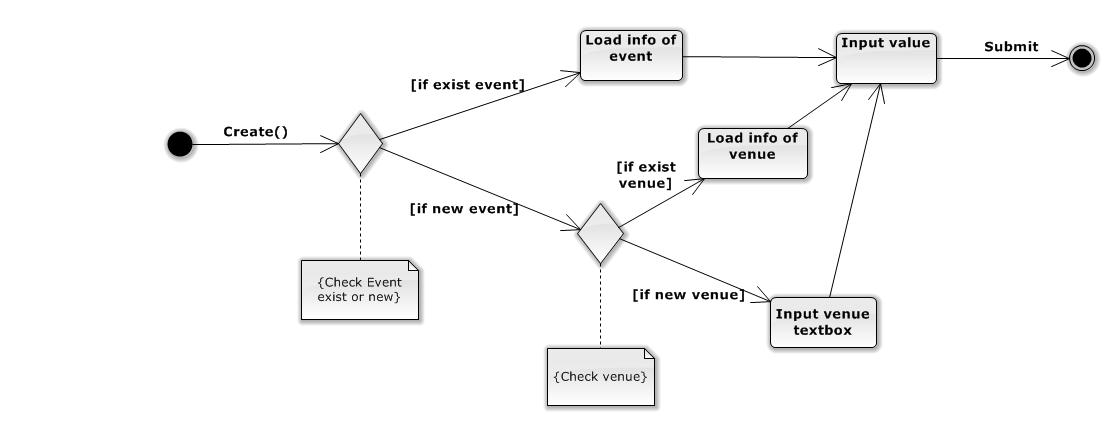


### 4.5.4 Manage User

### 4.5.5 Create Event

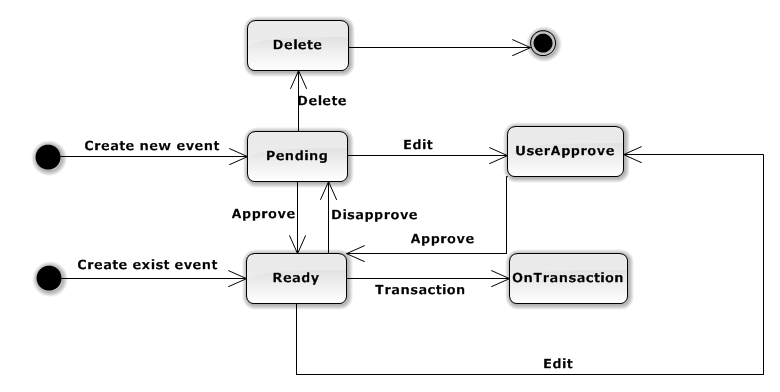
## 4.6 Activity diagram

### 4.6.1 Create Ticket



## 4.7. State Diagram

### 4.7.1 Ticket State Diagram



**Figure 4.7.1.1 – Ticket State Diagram**

States Description:

Delete: …

Pending: …

UserApprove: …

OnTransaction: …

Ready: …

### 4.7.2 Event State Diagram

**Figure 4.7.2.1 – Event State Diagram**

States Description:

Disapprove: ……

Delete: …..

Approve: …

Trading: ….

OutDate: …

## 4.8. Database Design or Data Structures

### 4.6.1 Database diagramE:\Capstone\trunk\Document\Database\DB_DropIt.png

**Figure 4.6.1.1 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 |  |