***WMA TEAM***

**MOVIE TICKET MANAGEMENT SYSTEM (MTMS)**

*Software Description Document*

Team members: Lâm Minh Phương – 51202846 – Leader

Huỳnh Tấn Phúc – 51202787

Nguyễn Xuân Thái – 51203374

Phạm Trần Minh Quang – 51202956

Nguyễn Thành Tâm – 51203264

Nguyễn Hữu Nhân – 51202524

Table of Contents

[**1.0.** **Purpose** 3](#_Toc404116161)

[1.1. Introduction 3](#_Toc404116162)

[1.2. Scope 3](#_Toc404116163)

[1.3. Glossary 3](#_Toc404116164)

[1.4. References 3](#_Toc404116165)

[1.5. Document overview 3](#_Toc404116166)

[**2.0. Class Diagram** 4](#_Toc404116167)

[**3.0. Sequence Diagrams** 30](#_Toc404116168)

[**4.0. Activity Diagrams** 35](#_Toc404116169)

[**5.0. User Interface Design** 40](#_Toc404116170)

[5.1. Main interface 40](#_Toc404116171)

[5.2. Booking movie interface 41](#_Toc404116172)

[5.3. Paying ticket interface 41](#_Toc404116173)

[5.4. Content creator interface 42](#_Toc404116174)

[5.5. Administrator interface 42](#_Toc404116175)

# **Purpose**

## 1.1. Introduction

This Software Requirements Specification provides a complete description of the Movie Ticket Management System (MTMS) developed by WMA Team for WMA Cinema Company.

The expected audience of this document is the staff of WMA Company, the Internet user who want to book ticket online of WMA Cinema and everyone who is interested in an online booking system.

## 1.2. Scope

The Movie Ticket Management System is designed to run on the WMA Company server and to allow Internet user to access for booking ticket online and the company staff to management booking information. The data will be held in a Hadoop database on the company server.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Internet user | Everyone who uses the Internet |
| Customer | People who want to book a ticket online using MTMS |
| Staff | Employee of WMA Cinema Company, who will use this system to management booking data |
| WMA Cinema Company | The owner of MTMS |
| WMA Team | The development team of MTMS |

## 1.4. References

The SDD sample from SE Course Module in Sakai website.

## 1.5. Document overview

The remaining chapters and their contents are listed below.

Section 2 is the Class Diagram that specifies the functional classes that collaborate to perform all the functions included in the system. Each of these entities has an Abstract description concerning the services that it provides to the rest of the system.

Section 3 concerns the Data Structure Design

Section 4 is the Sequence Diagrams that specify the sequence of works that the system will perform in order to provide functional features to the user

Section 5 is the Activity Diagrams that illustrate how the features of the MTMS work step-by-step

Section 6 discusses the User Interface Design

# **2.0. Class Diagram**

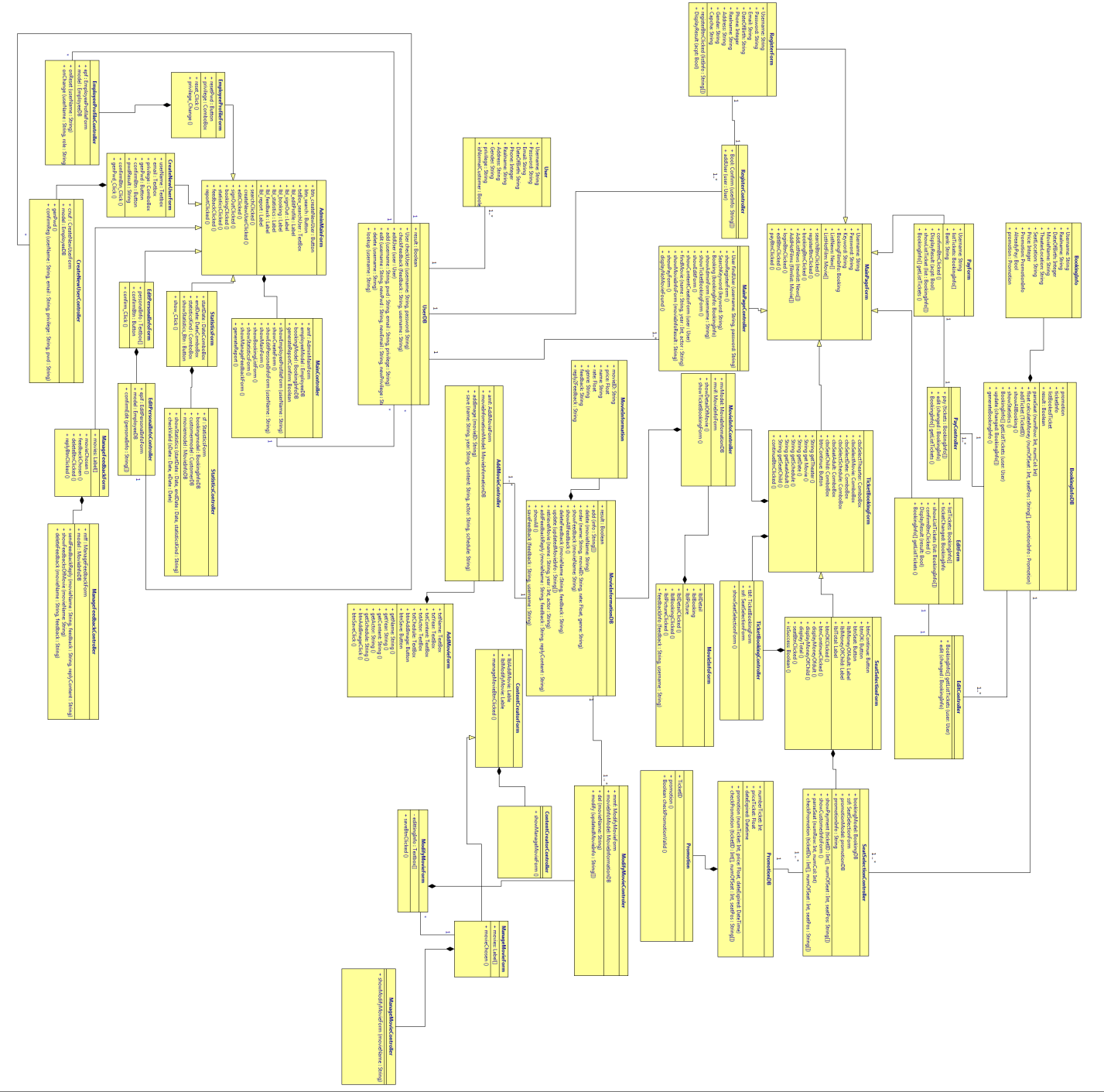


Figure 1 Class diagram for MTMS

**TicketBookingForm**

Name: TicketBookingForm

Type: Form

Description: After customer clicks button “Booking” on MainPageForm, this form is showed. And then, customer can select theater, movie, date, schedule, quantity seat.

Resources: None

Attributes:

cbxSelectTheater: ComboBox

cbxSelectMovie: ComboBox

cbxSelectDate: ComboBox

cbxSeatAdult: ComboBox

cbxSeatChild: Combobox

btnContinue: Button

Operations:

Name: continueBtnClicked ()

Arguments: None

Pre-condition: Connected to site

Post-condition: On another page

Exceptions: None

Flow of Events:

1. Customer is presented with the TicketBookingForm
2. Customer clicks a button
3. Customer is connected to another form

**TicketBookingController**

Name: TicketBookingController

Type: Controller

Description: this class will receive events form TicketBookingForm, then enforce.

Attributes:

tbf: TicketBookingForm

ssf: SeatSelectionForm

Operations:

Name: showSeatSelectionForm()

Argument: None

Return: No return value

Pre-condition: button continue is selected

Post-condition: On another page

Exceptions: None

Flow of Events:

1. Customer selects continue button in TicketBookingForm
2. Customer is moved to another form

**SeatSelectionForm**

Name: SeatSelectionForm

Type: Form

Resources: None

Description: with this form, customer can choose where to seat in theater and can see total money for booking ticket.

Attributes:

btnOK: Button

btnSeat: Button

lblMoneyOfAdult: Label

lblMoneyOfChild: Label

lblTotal: Label

Operations

Name: seatBtnClicked()

Argument: None

Return: No value return

Exceptions: None

Flow of Events:

1. Customer clicks button btnSeat in SeatSelectionForm
2. This event is moved to class SeatselctionController.

Name: displayMoneyOfDult(), displayMoneyofChild(), displayTotal()

Argument: None

Return: No value return

Exceptions: None

Flow of Events:

1. Values return from conculateMoney operation in BoookingInfoDB database
2. Those values are showed to customer on SeatSelectionForm.
3. Name: isSuccess ()

Arguments: None

Return: Boolean

Flow if events:

1. Value return from parseSeat operation in BookingInfoDB database
2. With this value, customer can know customer’s seat position is valid or invalid.

**SeatSelectionController**

Name: SeatSelectionController

Description: this class will receive events form SeatSelectionForm move to BookingInfoDB database and PromotionDB database.

Attribute:

bookingModel: BookingInfoDB

ssf: SeatSelectionForm

promotionModel: PromotionDB

promotionInfo: String

Operations:

Name: showPayment()

Arguments: ticketID[]: Int[], numOfSeat: Int, seatPos: String

Return: no value return.

Flow if events:

1. After customer clicks button OK
2. This event is moved to calculateMoney(seatAdult: Int, seatChild: Int) operation in BookingInfoDB

Name: parseSeat ()

Arguments: numRow: Int, numCol: Int

Return: Boolean

Flow if events:

1. After customer clicks choose where to seat
2. This event is moved to parseSeat(numRow: Int, numCol: Int) operation in BookingInfoDB

Name: checkPromotion ()

Arguments: ticketIDs: Int[], numOfSeat: Int, numPos: String

Return: Boolean

Flow if events:

1. If function parseSeat is true then call checkPromotion
2. Call promotion function in PromotionDB.

**MovieInfoForm**

Name: MovieInfoForm

Description: show movie’s information.

Attributes:

lbldetail: Label

lblBooking: Label

lblPicture: Label

Operations:

Name: lblDetailClick()

Argument: None

Return: None

Flow of events:

1. Customer clicks label detail
2. This event moves to MoveInfoController

Name: lblBookingClick ()

Argument: None

Return: None

Flow of events:

1. Customer clicks label booking
2. This event moves to MoveInfoController

Name: lblPictureClick ()

Argument: None

Return: None

Flow of events:

1. Customer clicks label picture
2. This event moves to MoveInfoController

**MovieInfoController**

Name: MovieInfoController

Description: this class receive events form MovieInfoForm move to MovieInfomationDB

Attributes:

mvModel: MovieInfomationDB

mvif: MovieInfoForm

Operations:

Name: showDetailOfMovie()

Arguments: None

Return: None

Flow of events:

1. After receive events from lblDetailClick, lblPictureClick functions
2. Call retrieveMovie(name: String, year: Int, actor: String) in MovieInformationDB

Name: show TicketBookingForm ()

Argument: none

Return: none

Flow of events:

1. After receive event from lblBookingClick function
2. Show TicketBookingForm()

**Content Creator Main**

Name: Content Creator Main Page

Type: Web page

Description: This is the page of content creators to manage, modify movies information. There are two links. One to add new movie information, one to modify.

On the Add Movie page, the MTMS present with a form to be completed by the content creator include: Movie name, Year, Content, Actor, Add Image… and then submit it by click Save button. When the form is submitted the server will check to ensure that the required fields are not blank. If the required fields are blank the server will return a message stating a required field is blank. The all required fields are not blank the server will create a new Movie Information record in the Movie Information Database.

On the Modify Movie page, the MTMS present search box for the creator can search what movie want to modify. After searching, the creator choose a movie with 2 buttons. One to modify it, one delete its information from the Movie Information Database.

Attributes: lblAddMovie: Label

lblModifyMovie: Label

Resource: None

Operations:

Name: lblAddClicked()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: On another page

Exceptions: None

Flow of events:

1. Content creator is presented Content Creator Main page
2. Content creator clicks Add Movie button
3. Function showAddMovie() in Content Creator Controller is called

Name: lblModifyClicked()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: On another page

Exceptions: None

Flow of events:

1. Content creator is presented Content Creator Main page
2. Content creator clicks Modify Movie button
3. Function showModifyMovie() in Content Creator Controller is called

**Content Creator Controller**

Name: Content Creator Controller

Type: Controller

Description: This class receives events from Content Creator Main page and enforces.

Attributes: None

Resource: None

Operations:

Name: showAddMovie()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: On another page

Exceptions: None

Flow of events:

1. Content creator is presented Content Creator Main page
2. Content creator clicks Add Movie button
3. MTMS show Add Movie Form

Name: showModifyMovie()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: On another page

Exceptions: None

Flow of events:

1. Content creator is presented Content Creator Main page
2. Content creator clicks Add Movie button
3. MTMS show Modify Movie Form

**Add Movie Form**

Name: Add Movie Form

Type: Form

Description: This form appears when the content creator clicks Add Movie label in Content Creator Main Page. Then the creator can edit, modify or delete movie information.

Attributes:

txtName: textbox

txtYear: textbox

txtContent: textbox

txtActor: textbox

btnAddImage: button

btnSave: button

Resource: None

Operations:

Name: btnAddImageClick()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: A function from Add Movie Controller is called

Exceptions: None

Flow of events:

1. Content creator is presented with Add Movie Form
2. Content creator clicks AddImage button
3. Function ‘addImage’ in Add Movie Controller is called

Name: btnSaveClick()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: Functions from Add Movie Controller is called

Exceptions: None

Flow of events:

1. Content creator is presented with Add Movie Form
2. Content creator clicks Save button
3. Function ‘getName’, ‘getYear’, ‘getContent’, ‘getActor’ in Add Movie Controller are called

**Add Movie Controller**

Name: Add Movie Controller

Type: Controller

Description: This class receives events from Add Movie Form and enforces.

Attributes:

amf: Add Movie Form

movieInfoModel: Movie Information Database

Resource: None

Operations:

Name: addImage

Arguments: movieID

Returns: None

Pre-condition: Function ‘btnAddImageClick()’ in Add Movie Form is called

Post-condition: An Image is added into movie record in Movie Information Database

Flow of events:

1. Function ‘btnAddImageClick()’ in Add Movie Form is called
2. addImage ‘lookup’ in Movie Information Database
3. Save an image into a record.

Name: save

Arguments: name: string, year: string, content: string, actor: string

Returns: None

Pre-condition: Function ‘btnSaveClick()’ in Add Movie Form is called

Post-condition: Infomation of movie is added into a new movie record in Movie Information Database

Flow of events:

1. Function ‘btnSaveClick()’ in Add Movie Form is called
2. Movie Information Database create a new movie record
3. Save movie information into a record.

**Modify Movie Form**

Name: Modify Movie Form

Type: Form

Description: This form appears when the content creator clicks Modify Movie label in Content Creator Main Page. Then the creator can edit, modify or delete movie information.

Attributes:

btnEdit

btnDelete

Searchbox

Resource: None

Operators:

Name: btnSaveClicked()

Arguments: None

Returns: None

Pre-condition: None

Post-condition: Form saved completely

Exceptions: None

Flow of events:

1. Content creator searched a movie
2. Content creator choose a movie with Edit button
3. Content creator edits movie information
4. Content creator clicks Save button

**Modify Movie Controller**

Name: Modify Movie Controller

Type: Controller

Description: This class receives events from ModifyMovieForm and enforces.

Attributes:

mmf: ModifyMovieForm

movieInfoModel: Movie Information Database

Resource: None

Operations:

Name: del

Arguments: movieID: string

Returns: None

Pre-condition: None

Post-condition: the movieID is gotten successful.

Exceptions: None

Flow of events:

1. Content creator choose a movie with Delete button
2. Modify Movie Controller get movieID send to Movie Information Database

Name: modify

Arguments:

movieID: string

name: string

year:

content: string

actor: string

Returns: None

Pre-condition: Connect to Movie Information Database

Post-condition: the movie information is updated successful.

Flow of events:

1. Content creator choose a movie with Edit button
2. Content creator edits movie information
3. Content creator clicks Save button
4. Modify Movie Controller get movie information send to Movie Information Database

**Statistics Form**

Name: Statistics Form

Type: Form

Description: This form appears when the administrator clicks Statistics label in Administrator Main Page. Then the admins can see statistics of a period of time they choose.

Attributes:

startDate: DateComboBox

endDate: DateComboBox

statisticsKind: ComboBox

showStatisticsBtn: Button

Resource: None

Operations:

Name: show\_Clicked()

Arguments: None

Returns: None

Pre-condition: has startDate, endDate and statisticsKind

Post-condition: parameters are sent to Statistics Controller

Exceptions: None

Flow of events:

1. Admin choose Statistics in Admin Main Page
2. Admin choose startDate, endDate and kind
3. Admin clicks show Statistics button
4. Parameters are sent to Statistics Controller

**Statistics Controller**

Name: Statistics Controller

Type: Controller

Description: This class receives events from Statistics Form and enforces.

Attributes:

sf: Statistics Form

bookingmodel: Booking Information Database

Resource: None

Operations:

Name: checkValid

Arguments: startDate: date, endDate: date

Returns: bool

Exceptions: None

Flow of events:

1. Statistics Controller receives info from Statistics Form
2. Checking if valid send startDate, endDate to Booking Information Database, if not require refilling.

Name: get

Arguments: startDate: date, endDate: date, statisticsKind: number

Returns: None

Pre-condition: checkValid returns true

Post-condition: the startDate, endDate, statisticsKind are sent to Booking Information Database

Exceptions: None

Flow of events:

1. Statistics Controller receives info from Statistics Form
2. ‘checkValid’ will check startDate, endDate. If valid return ‘true’ and ‘get’ will send to Booking Information Database.

**Manage Feedback Form**

Name: Manage Feedback Form

Type: Form

Description: This form appears when the administrator clicks Feedback label in Administrator Main Page. Then the admins can see, manage and reply feedback of every movie.

Attributes: btnDelete, btnReply

Resource: None

Operations:

Name: feedbackChosen()

Arguments: None

Returns: None

Pre-condition: Connected to site

Post-condition: A function from Manage Feedback Controller is called

Exceptions: None

Flow of events:

1. Admin choose a movie
2. Function ‘showFeedbackofMovie(movieID: string)’ in Manage Feedback Controller is called.

Name: deleteBtnClicked()

Arguments: None

Returns: None

Pre-condition: Feedback is exist in Movie Information Database

Post-condition: A function from Manage Feedback Controller is called

Exceptions: None

Flow of events:

1. Admin choose a movie
2. Function ‘deleteFeedback (movieID: string, feedback: string)’ in Manage Feedback Controller is called.

Name: replyBtnClicked()

Arguments: None

Returns: None

Pre-condition: Feedback is exist in Movie Information Database

Post-condition: A function from Manage Feedback Controller is called

Exceptions: None

Flow of events:

1. Admin choose a movie
2. Function ‘sendFeedbackReply (movieID: string, feedback: string, receiveUsername: string)’ in Manage Feedback Controller is called.

**Manage Feedback Controller**

Name: Manage Feedback Controller

Type: Controller

Description: This class receives events from Manage Feedback Form and enforces.

Attributes:

mff: Manage Feedback Form

model: Movie Information Database

Resource: None

Operations:

Name: showFeedbackofMovie

Arguments: movieID: string

Returns: None

Pre-condition: The action FeedbackChosen() of Manage Feedback Form is called.

Post-condition: A list of feedback of a movie appered.

Flow of events:

1. The action FeedbackChosen() of Manage Feedback Form is called.
2. The function ‘lookup’ in Movie Information Database and present a list of feedback

Name: deleteFeedbackofMovie

Arguments: movieID: string

Returns: None

Pre-condition: The action deleteBtnClicked() of Manage Feedback Form is called.

Post-condition: The feedback is not existed in the Movie Information Database

Flow of events:

1. The action deleteBtnClicked () of Manage Feedback Form is called.
2. The function ‘lookup’ in Movie Information Database and delete the feedback from the database.

Name: sendFeedbackReply

Arguments: movieID: string

Returns: None

Pre-condition: The action replyBtnClicked() of Manage Feedback Form is called.

Post-condition: The new feedback is store in the Movie Information Database

Flow of events:

1. The action replyBtnClicked() of Manage Feedback Form is called.
2. The function ‘lookup’ in Movie Information Database and show a textbox for user fill it
3. The new feedback is sent to Movie Information Database

**Admin Main Form**

Name: Admin Main Form

Type: Web page (View in MVC model)

Description: This is the initial administrator home page. There are two button, six label which can be clicked to go to another page. There is also a textbox to search for an employee.

Attributes: createNewUserBtn – Button, searchBtn – Button, searchUserTxtbox – Textbox, editProfilelbl – Label, signOutLbl – Label, bookingLbl – Label, statisticsLbl – Label, feedbackLbl – Label, reportLbl – Label, controller - AdminMainController

Resource: None

Operations:

Name: searchClicked()

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator type the name of the employee he wants to search for into the textbox “Search User”
3. Administrator clicks on the “Search” button
4. Function “showEmployeeProfileForm” from Admin Main Controller is called

Name: createNewUserClicked()

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Admistrator click on the “Create New User” button
3. Function “showCreateForm” from Admin Main Controller is called

Name: editClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Admistrator click on the “Edit Profile” button
3. Function “showEditPersonalInfoForm” from Admin Main Controller is called

Name: signOutClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator clicks on the “Sign Out” button
3. Function “showMainForm” from Admin Main Controller is called

Name: bookingClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator clicks on the “View Booking Info” button
3. Function “showBookingListForm” from Admin Main Controller is called

Name: statisticClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator clicks on the “View Statistic Info” button
3. Function “showStatisticsForm” from Admin Main Controller is called

Name: feedbackClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator clicks on the “Manage Feedback” button
3. Function “showManageFeedbackForm” from Admin Main Controller is called

Name: reportClicked()

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Admin Main Form
2. Administrator clicks on the “Generate Report” button
3. Function “generateReport” from Admin Main Controller is called

**Admin Main Controller**

Name: Admin Main Controller

Type: Controller (Controller in MVC model)

Description: This is the controller of the Admin Main Form. When user does an action, the action event of the form will be called. Then this event will call a responding action of controller.

Attributes: amf - Admin Main Form, employeeModel – EmployeeDB, bookingModel – BookingDB, generateReportConfirm - boolean

Resource: None

Operations:

Name: showEmployeeProfileForm

Arguments: userName - string

Return: No return value

Pre-condition: The action event “searchClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “searchClicked” calls this function
2. The function “lookup” in User DB is called and return the employee profile
3. Administrator is directed to the “Employee Profile Form” with the information of the employee they looked for.

Operations:

Name: showCreateForm

Arguments: None

Return: No return value

Pre-condition: The action event “createNewUserClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “createNewUserClicked” calls this function
2. Administrator is directed to the “Create New User” form.

Operations:

Name: showEditPersonalInfoForm

Arguments: None

Return: No return value

Pre-condition: The action event “editClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “editClicked” calls this function
2. The function “lookup” in User DB is called with name of current administrator passing as argument and return administrator profile
3. Administrator is directed to the “Edit Personal Profile” form with their information

Operations:

Name: showMainForm

Arguments: None

Return: No return value

Pre-condition: The action event “signOutClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “signOutClicked” calls this function
2. The account is logged out and user is directed to the main page.

Operations:

Name: showBookingListForm

Arguments: None

Return: No return value

Pre-condition: The action event “bookingClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “bookingClicked” calls this function
2. The function “showAllBooking” is called
3. Administrator is directed to “Booking List” form

Operations:

Name: showStatisticsForm

Arguments: None

Return: No return value

Pre-condition: The action event “statisticsClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “statisticsClicked” calls this function
2. Administrator is directed to “Statistics” form

Operations:

Name: showManageFeedbackForm

Arguments: None

Return: No return value

Pre-condition: The action event “feedbackClicked” of Admin Main Form is called

Post-condition: On another page

Exception: None

Flow of Events:

1. The action event “feedbackClicked” calls this function
2. Administrator is directed to “Manage Feedback” form

Operations:

Name: generateReport

Arguments: None

Return: No return value

Pre-condition: The action event “reportClicked” of Admin Main Form is called

Post-condition: The report is shown

Exception: None

Flow of Events:

1. The action event “reportClicked” calls this function
2. The function “get\_report” of Booking DB is called and returns some number
3. The number is placed in an report form
4. The form is show in Admin Main Form

**Employee Profile Form**

Name: Employee Profile Form

Type: Web Page (View in MVC model)

Description: This is the form for edit employee profile. The administrator can do two thing on an employee profile: reset the password and change their privilege. There are one button and one combo box to do this.

Attribute: resetPwd – Button, privilege – ComboBox, controller - EmployeeProfileController

Operations:

Name: resetClicked

Arguments: None

Return: No return value

Pre-condition: Connected to the site

Post-condition: The function “onReset” of Employee Profile Controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Employee Profile Form
2. Admistrator click on the “Reset Password” button
3. Function “onReset” from Employee Profile Controller is called

Operations:

Name: privilege\_Change

Arguments: None

Return: No return value

Pre-condition: Connected to the site

Post-condition: The function “onChange” of Employee Profile Controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Employee Profile Form
2. Admistrator choose an option from the “Privilege” combo box
3. Function “onChange” from Employee Profile Controller is called

**Employee Profile Controller**

Name: Employee Profile Controller

Type: Controller (Controller in MVC model)

Description: This is the controller of Employee Profile Form.

Attribute: epf – EmployeeProfileForm, model – UserDB

Resource: None

Operations:

Name: onReset

Arguments: userName - String

Return: No return value

Pre-condition: The action event “resetClicked” of Employee Profile Form is called

Post-condition: The employee profile is updated

Exception: None

Flow of Events:

1. The action event “resetClicked” calls this function
2. A random password is generated
3. The function “edit” of UserDB is called to change the account’s password
4. A message is display to announce changing result

Name: onChange

Arguments: userName – String, role - String

Return: No return value

Pre-condition: The action event “privilege\_Change” of Employee Profile Form is called

Post-condition: The employee profile is updated

Exception: None

Flow of Events:

1. The action event “privilege\_Change” calls this function
2. The function “edit” of UserDB is called to change the account’s privilege
3. A message is display to announce changing result

**Create New User Form**

Name: Create New User Form

Type: Web page (View in MVC model)

Description: This is form to create new user. There are some textboxes and combo box to fill in the new user’s information, there is one button to create a random password and one button to confirm.

Attributes: userName – Textbox, email – Textbox, privilege – Combo box, genPwd – Button, confirmBtn – Button, pwdResult – Label, controller – CreateNewUserController

Resource: None

Operations:

Name:genPwdClicked()

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Create New User Form
2. Administrator clicks on the “Generate Password” button
3. Function “genPwd” from Create New User Controller is called

Name: confirmBtnClicked()

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Create New User Form
2. Administrator fills the requested information of new user
3. Function “confirmReg” from Create New User Controller is called

**Create New User Controller**

Name: Create New User Controller

Type: Controller (Controller in MVC model)

Description: This is the controller of Create New User Form.

Attribute: cnuf – CreateNewUserForm, model – UserDB

Resource: None

Operations:

Name: genPwd

Arguments: None

Return: No return value

Pre-condition: The action event “genPwdClicked” of Employee Profile Form is called

Post-condition: A random password is created and shown on the form

Exception: None

Flow of Events:

1. The action event “genPwdClicked” calls this function
2. A random password is generated
3. The new password is displayed on the form

Name: confirmReg

Arguments: userName – String, email – String, privilege – String, pwd - String

Return: No return value

Pre-condition: The action event “confirmBtnClicked” of Employee Profile Form is called

Post-condition: A random password is created and shown on the form

Exception: None

Flow of Events:

1. The action event “confirmBtnClicked” calls this function
2. The function “add” from UserDB is called
3. A message is displayed to announce adding result

**Edit Personal Info Form**

Name: Edit Personal Info Form

Type: Web page (View in MVC model)

Description: This is form to edit the current user information. There are some textboxes for the administrator to edit his personal info, and one button to confirm the changes.

Attributes: personalInfo – Textbox[], confirmBtn – Button, controller – EditPersonalInfoForm

Resource: None

Operations:

Name: confirmClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Administrator is presented with the Edit Personal Info Form
2. Administrator types his changed information on suitable textboxes
3. Administrator clicks on the “Confirm Button”
4. The function “confirmEdit” from Edit Personal Info Controller is called

**Edit Personal Info Controller**

Name: Edit Personal Info Controller

Type: Controller (Controller in MVC model)

Description: This is the controller of Edit Personal Info Form.

Attribute: epif – CreateNewUserForm, model – UserDB

Resource: None

Operations:

Name: confirmEdit

Arguments: personalInfo – String[]

Return: No return value

Pre-condition: The action event “confirmClicked” of Edit Personal Info Form is called

Post-condition: The information of current user is updated

Exception: None

Flow of Events:

1. The action event “confirmClicked” calls this function
2. The function “edit” from UserDB is called to update changed information
3. A message is display to announce changing result

**Edit Form**

Name: EditForm

Type: Web page (View in MVC model)

Description: This is the form which allows customers to edit their tickets.Edit Form has a list of tickets, combobox of movie, combobox of time, map of seats and a button “Comfirm”. Customer can choose their ticket which they want to edit and edit it. Customer click on “Comfirm” button to finish editting.

Attributes:

listTickets: BookingInfo[]

ticketchanged: BookingInfo

customer: User

movie: String

time: Date

seat: String

Resource: None

Operations:

Name: showListTickets

Arguments:

- list: BookingInfo[]

Return: No return value

Pre-condition: Connected to site, Customer must login before.

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Edit” button
3. Customer is connected to EditForm
4. EditForm call function showListTickets to get lists of customer’s tickets and show them.

Name: comfirmBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: A ticket must be chosen.

Exception: None

Flow of Events:

1. After customer fill all field in EditForm.
2. Customer click on “Comfirm” button.
3. Function “comfirmBtnClicked” from EditForm is called. It will call function Edit in EditController Class.

Name: DisplayResult

Return: No return value

Arguments:

* result : bool

Pre-condition: Connected to site

Post-condition: A function from the EditForm is called

Exception: None

Flow of Events:

1. It will call function update in BookingInfoDB class, return result of this function

**Edit Controller**

Name: EditController

Type: Controller (Controller in MVC model)

Description: This class will receive request from EditForm class and call function in BookingInfoDB.

Attributes: None

Operations:

Name: getListTickets

Arguments:

- customer: User

Return: BookingInfo[]

Pre-condition: None

Post-condition: A function from the EditForm is called

Exception: None

Flow of Events:

1. It call function getListTickets from BookingInfoDB and return result of this function.

Name: Edit

Arguments:

* ticketchanged: BookingInfo

Return: Boolean.

Pre-condition: None.

Post-condition: None.

Exception: None

Flow of Events:

1. It is called by function comfirmBtnClicked from EditForm and return result of this function.

**Pay Form**

Name: PayForm

Type: Web Page (Model in MVC model)

Description: This is the form which allows customers to pay their tickets online.Pay Form has a list of tickets, combobox of list of banks and a button “Comfirm”. Customer can choose their tickets which they want to pay, select bank and click on “Comfirm” button to finish editting.

Attributes:

listTickets: BookingInfo[]

customer: User

bank: String

Operations:

Name: showListTickets

Arguments:

- list: BookingInfo[]

Return: No return value

Pre-condition: Connected to site, Customer must login before.

Post-condition: A function from the controller is called

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Edit” button
3. Customer is connected to EditForm
4. EditForm call function showListTickets to get lists of customer’s tickets and show them.

Name: comfirmBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site

Post-condition: Tickets must be chosen.

Exception: None

Flow of Events:

1. After customer fill all field in EditForm.

1. Customer click on “Comfirm” button.
2. Function “comfirmBtnClicked” from EditForm is called. It will call function Edit in EditController Class.

Name: DisplayResult

Return: No return value

Arguments:

* result : bool

Pre-condition: Connected to site

Post-condition: A function from the EditForm is called

Exception: None

Flow of Events:

1. It will call function update in BookingInfoDB class, return result of this function

**Pay Controller**

Name: PayController

Type: Controller (Controller in MVC model)

Description: This class will receive request from PayForm class and call function in BookingInfoDB.

Attributes: None

Operations:

Name: getListTickets

Arguments:

- customer: User

Return: BookingInfo[]

Pre-condition: None

Post-condition: A function from the EditForm is called

Exception: None

Flow of Events:

1. It call function getListTickets from BookingInfoDB and return result of this function.

Name: Pay

Arguments:

* listtickets: BookingInfo[]

Return: Boolean.

Pre-condition: None.

Post-condition: None.

Exception: None

Flow of Events:

1. It is called by function comfirmBtnClicked from PayForm
2. It call function update in BookingInfoDB class and return result of this function.

**Register Form**

Name: RegisterForm

Type: Web Page (Model in MVC model)

Description: This is the form which allows customers to create an account.Register Form has many field to obtain information about customer (username, password, date of birth, ...capcha code) and a button “Comfirm”. Customer fill up all field and click on “Comfirm” button to finish registering.

Attributes:

Username: String

Password: String

Re-password: String

Email: String

DateOfBirth: String

Realname: String

Phone: Integer

Address: String

Gender: String

Capcha: String

Operations:

Name: comfirmBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: All field is filled up. Information of fields is correct, capcha code must correct.

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Register” button
3. Customer is connected to RegisterForm
4. Customer fill up information and click “Comfirm” button.
5. Function “comfirmBtnClicked” from RegisterForm is called. It will call function addUser in RegisterController Class.

Name: DisplayResult

Return: No return value

Arguments:

* result : bool

Pre-condition: Connected to site

Post-condition: A function from the RegisterController is called.

Exception: None

Flow of Events:

1. When function addUser in UserDB is invoked, it will return resutl boolean.
2. RegisterController call DisplayResult function and show result about register.

**Register Controller**

Name: RegisterController

Type: Controller (Controller in MVC model)

Description: This class will receive request from RegisterForm class and call function in UserDB.

Attributes: None

Operations:

Name: addUser

Arguments:

- customer: User

Return: None

Pre-condition: None

Post-condition: A function from the RegisterForm is called

Exception: None

Flow of Events:

1. It call function addUser from UserDB and return result of this function.

**Main Form**

Name: MainForm

Type: Web Page (Model in MVC model)

Description: This is the main form which show all function customer can do. When customer access to our web, this form will be shown first, it includes a menu (Search, Edit, Pay, Register...) and list of news, list of hot film. It also has 2 textbox username, password and “Login” button. Customer can login in this form.

Attributes:

Username: String

Password: String

Keyword: String

ListNews: New[]

ListHotFilm: Movie[]

Operations:

Name: searchBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: None

Exception: None

Flow of Events:

1. This function call showSearchForm in MainController

Name: registerBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: None

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Register” button
3. This function call showRegisterForm in MainController
4. Customer is connected to RegisterForm

Name: editBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: Customer must login before.

Exception: None

Flow of Events:

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Edit” button
3. This function call showEditForm in MainController
4. Customer is connected to EditForm

Name: payBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: Customer must login before.

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Pay” button
3. This function call showPayForm in MainController
4. Customer is connected to PayForm

Name: loginBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: Username and passsword are filled up.

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Login” button
3. This function call findUser in MainController

Name: bookingBtnClicked

Arguments: None

Return: No return value

Pre-condition: Connected to site.

Post-condition: Customer must login before.

Exception: None

Flow of Events:

1. Customer is presented with the MainForm
2. Customer click on “Booking” button
3. This function call showTicketBookingForm in MainController
4. Customer is connected to TicketBookingForm

Name: AddListNews

Arguments: newlist: New[]

Return: No return value

Pre-condition: None.

Post-condition: None .

Exception: None

Flow of Events:

1. This function will show list of news.

Name: AddHotFilms

Arguments: filnlist: Moive[]

Return: No return value

Pre-condition: None.

Post-condition: None .

Exception: None

Flow of Events:

1. This function will show list of movies.

**Main Controller**

Name: MainController

Type: Controller (Controller in MVC model)

Description: This class will receive request from MainForm class and call function in UserDB, MovieDB.

Attributes: None

Operations:

Name: findUser

Arguments:

- customer: String

- password: String

Return: None

Pre-condition: None

Post-condition: None

Exception: None

Flow of Events:

1. It call function checkUser from UserDB and return result of this function.

Name: showRegisterForm

Arguments: None

Return: None

Pre-condition: None

Post-condition: None

Exception: None

Flow of Events:

1. It wil show RegisterForm.

Name: showAdminForm

Arguments: None

Return: None

Pre-condition: None

Post-condition: User must login as Admin

Exception: None

Flow of Events:

1. It wil show AdminForm.

Name: showTicketBookingForm

Arguments: None

Return: None

Pre-condition: None

Post-condition: User must login as customer.

Exception: None

Flow of Events:

1. It wil show TicketBookingForm.

Name: showEditForm

Arguments: None

Return: None

Pre-condition: None

Post-condition: User must login as customer

Exception: None

Flow of Events:

1. It wil show EditForm.

Name: showPayForm

Arguments: None

Return: None

Pre-condition: None

Post-condition: User must login as customer

Exception: None

Flow of Events:

1. It wil show PayForm.

Name: showContentCreatorForm

Arguments:

* User: User

Return: None

Pre-condition: None

Post-condition: User must login as Admin.

Exception: None

Flow of Events:

1. It wil show ContentCreatorForm.

Name: SearchKeyword

Arguments:

* keyword: String

Return: None

Pre-condition: None

Post-condition: keyword must not be empty.

Exception: None

Flow of Events:

1. It wil call retrieveMovie in MovieInformationDB.

Name: findMoive

Arguments:

* name: String
* year: Int
* actor: String

Return: None

Pre-condition: None

Post-condition: all params are not null.

Exception: None

Flow of Events:

1. It wil call retrieveMovie in MovieInformationDB.

# **3.0. Sequence Diagrams**

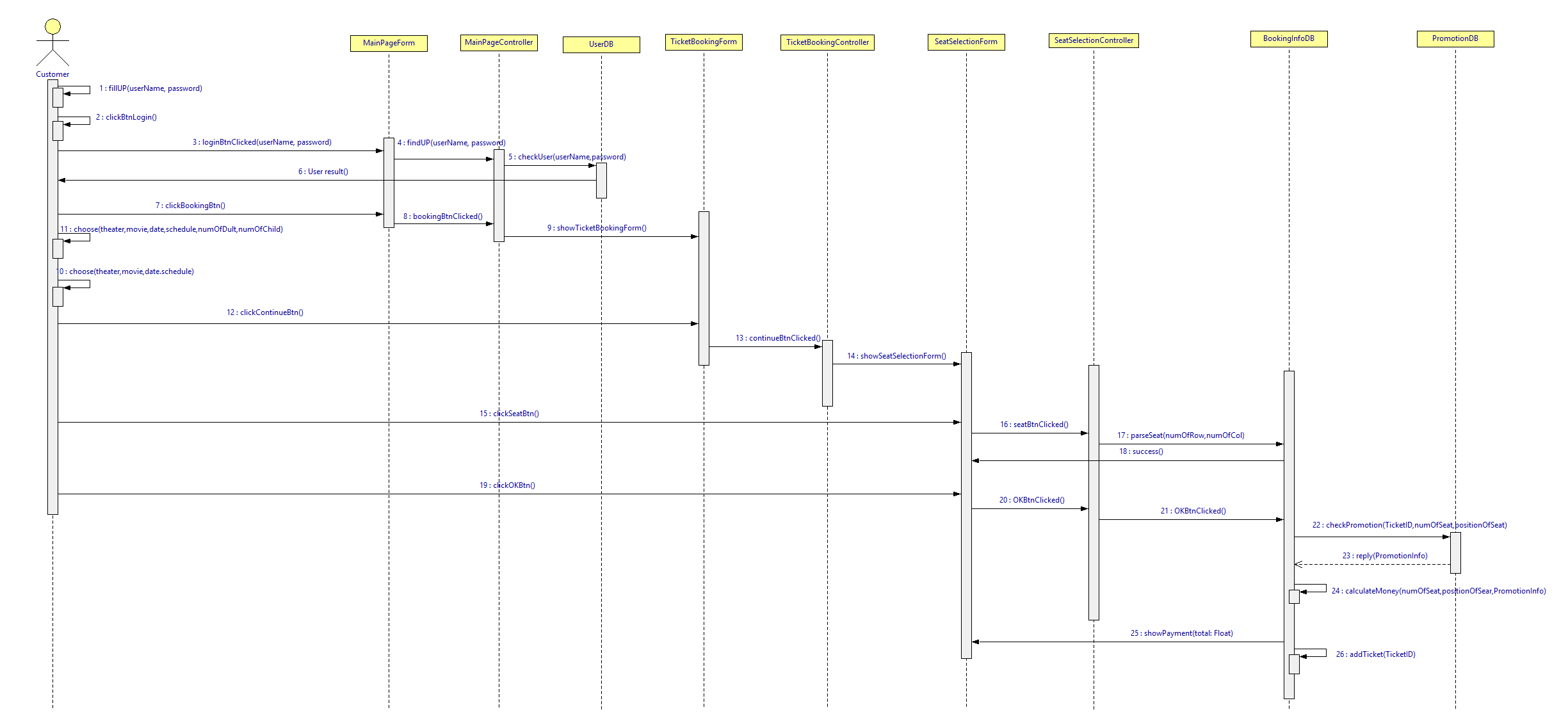


Figure 2 [Customer] ticket booking

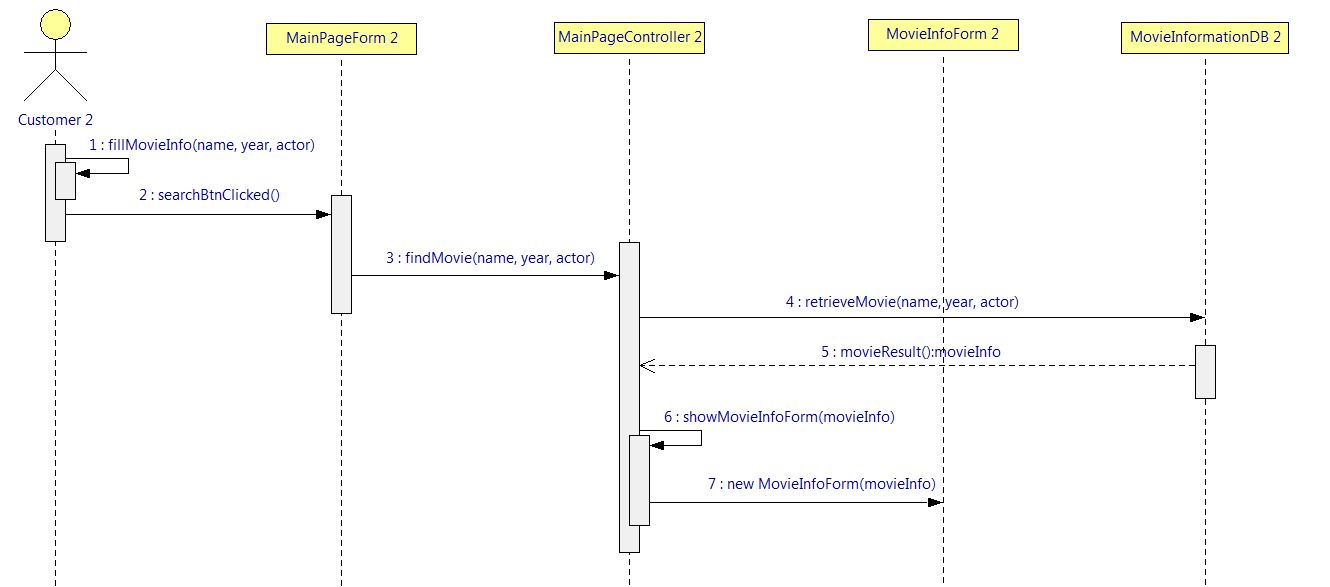


Figure 3 [Customer] search for movies

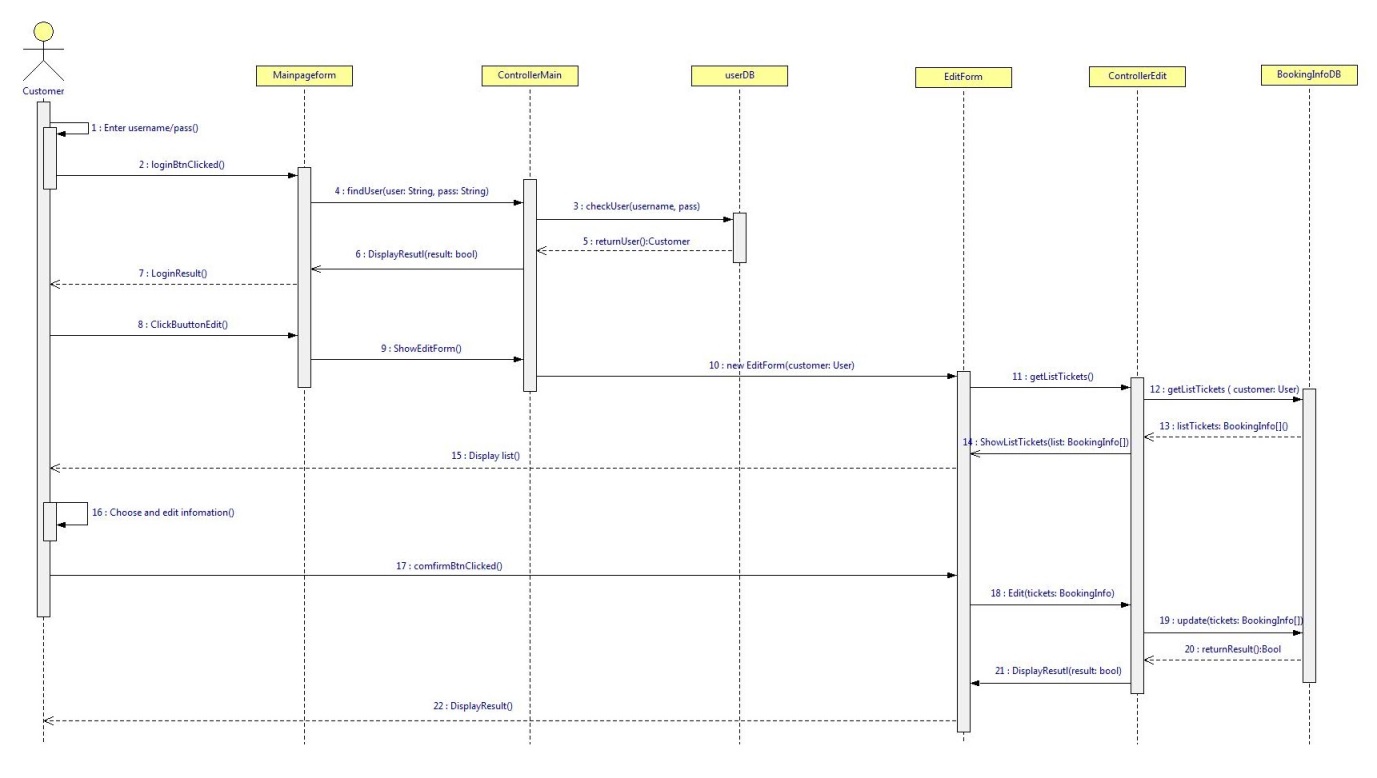


Figure 4 [Customer] edit one’s ticket booking

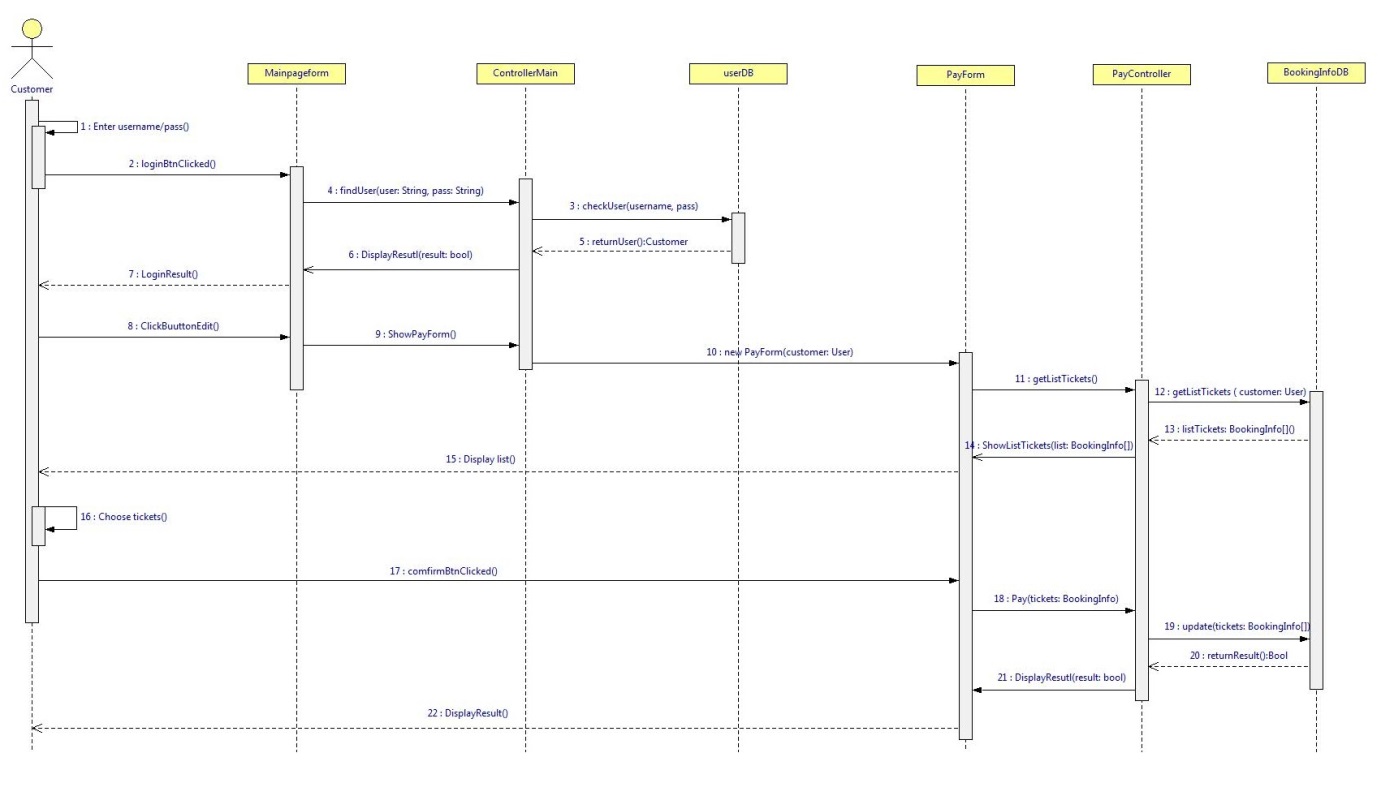


Figure 5 [Customer] ticket payment



Figure 6 [Customer] customer registry

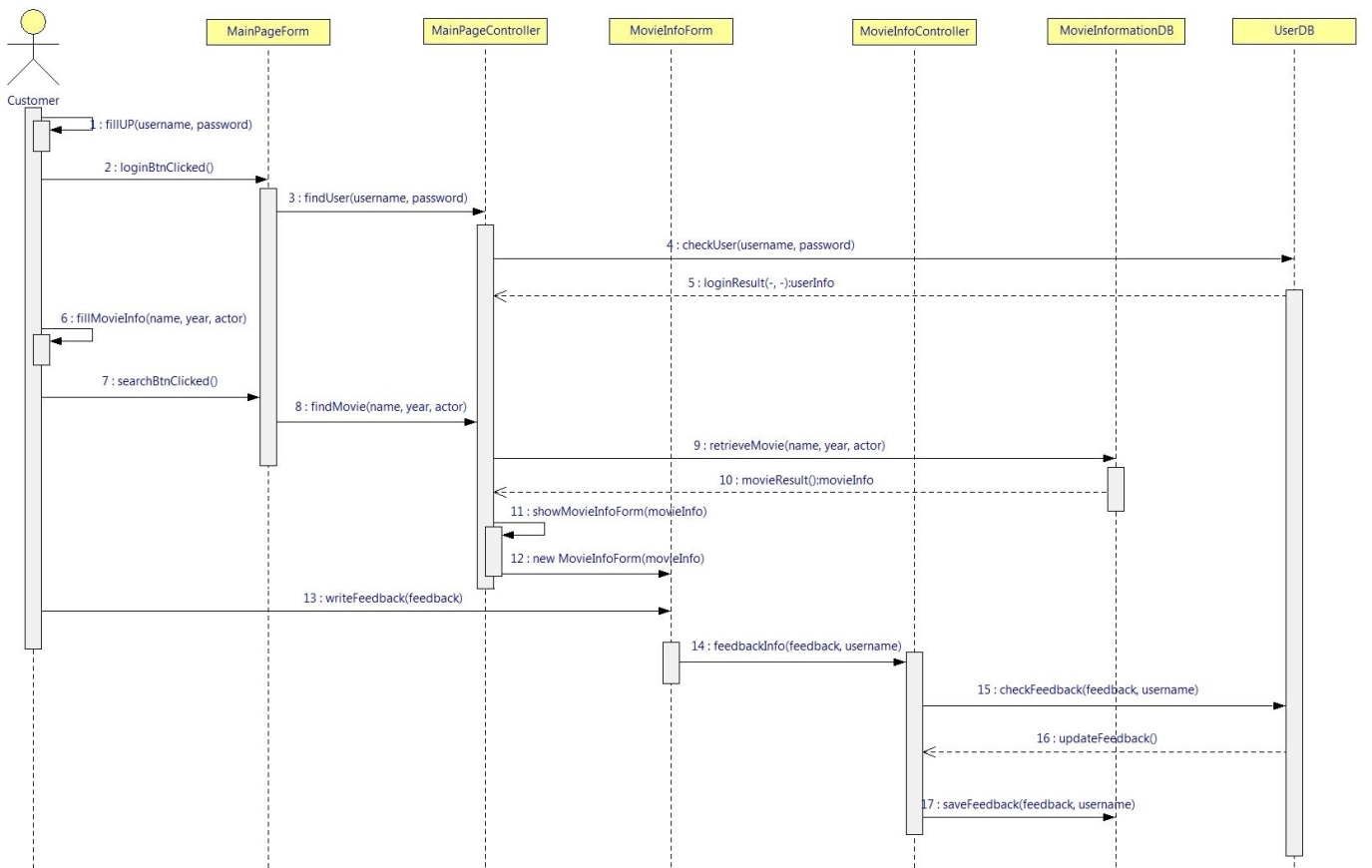


Figure 7 [Customer] send feedbacks to the managers

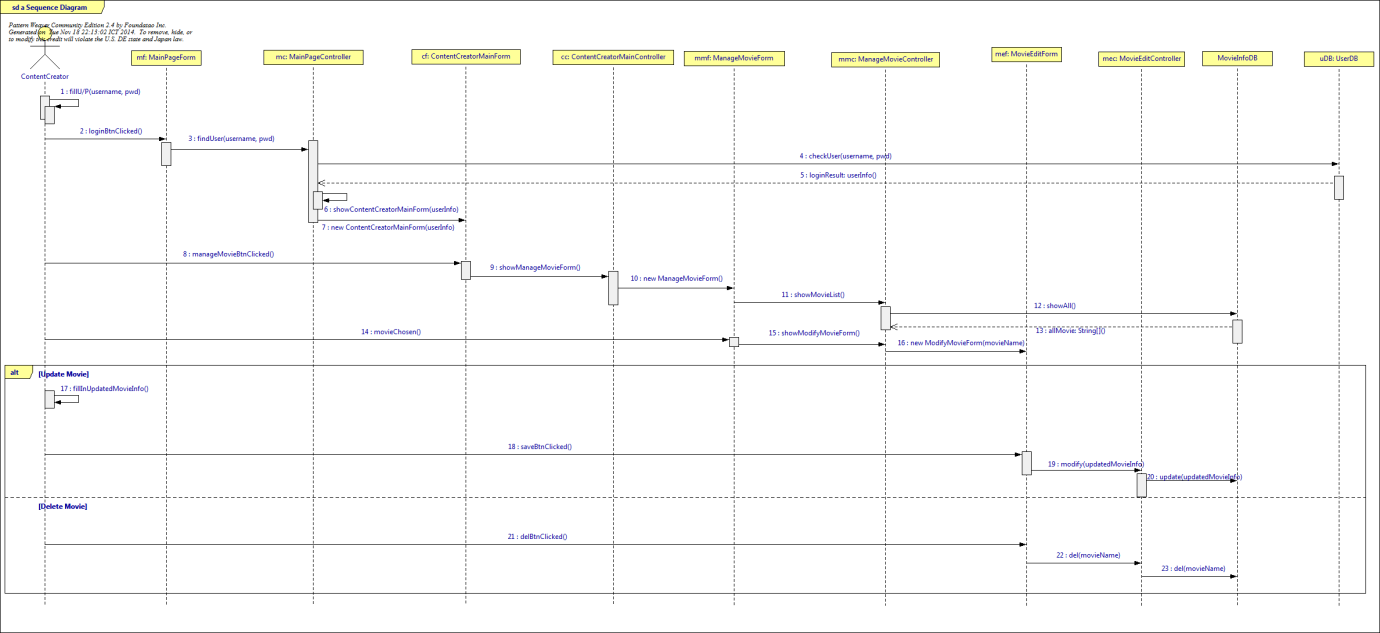


Figure 8 [Content creator] edit/delete movie’s information



Figure 9 Manage customers’ feedbacks

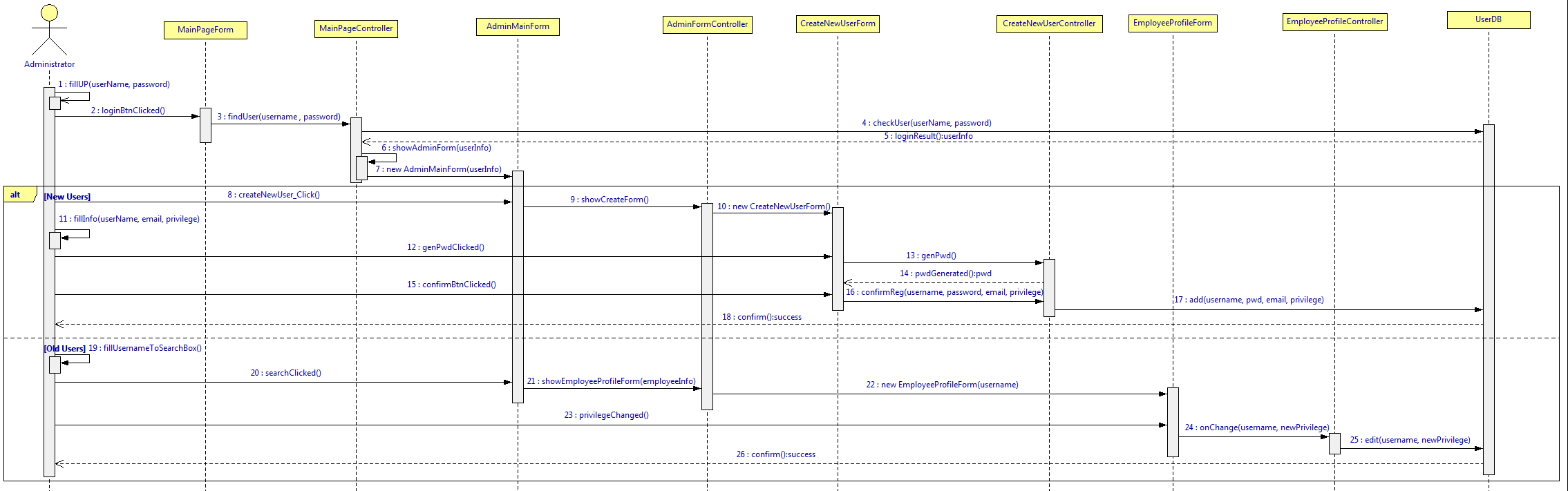


Figure 10 [Administrator] grant privilege to a specified user

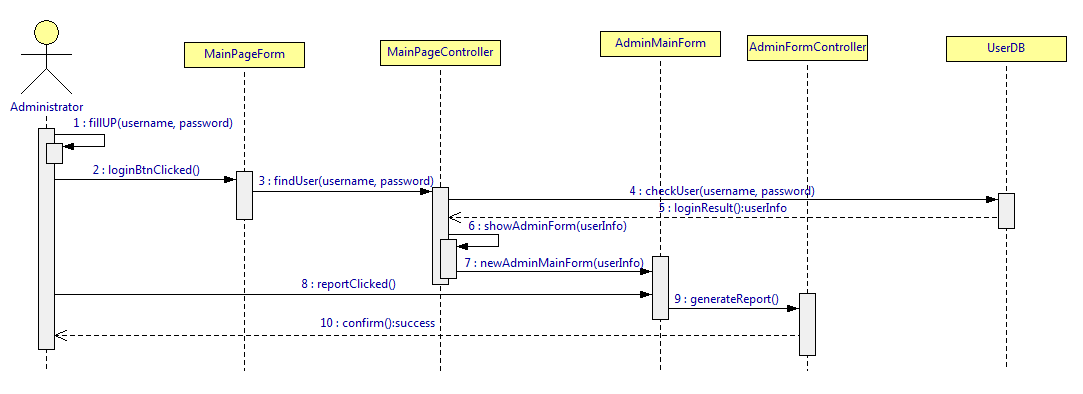


Figure 11 [Administrator] generate report

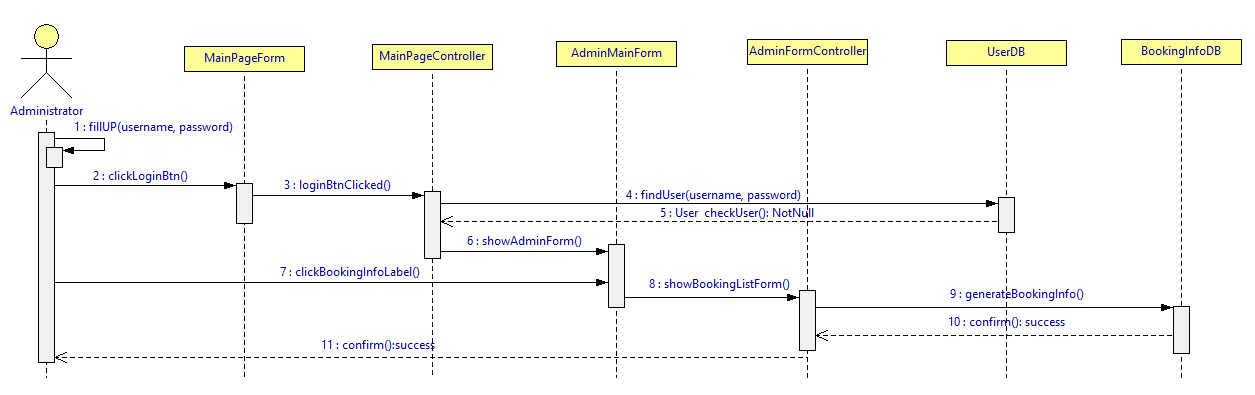


Figure 12 [Administrator] Review booking information

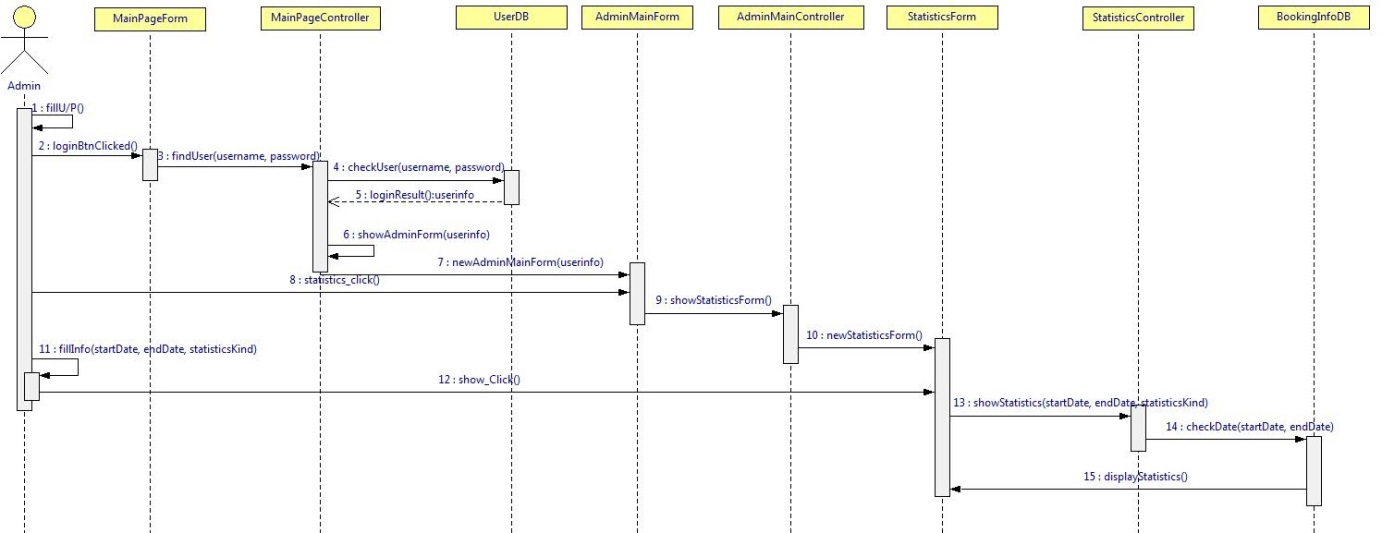


Figure 13 [Administrator] see statistics information

# **4.0. Activity Diagrams**

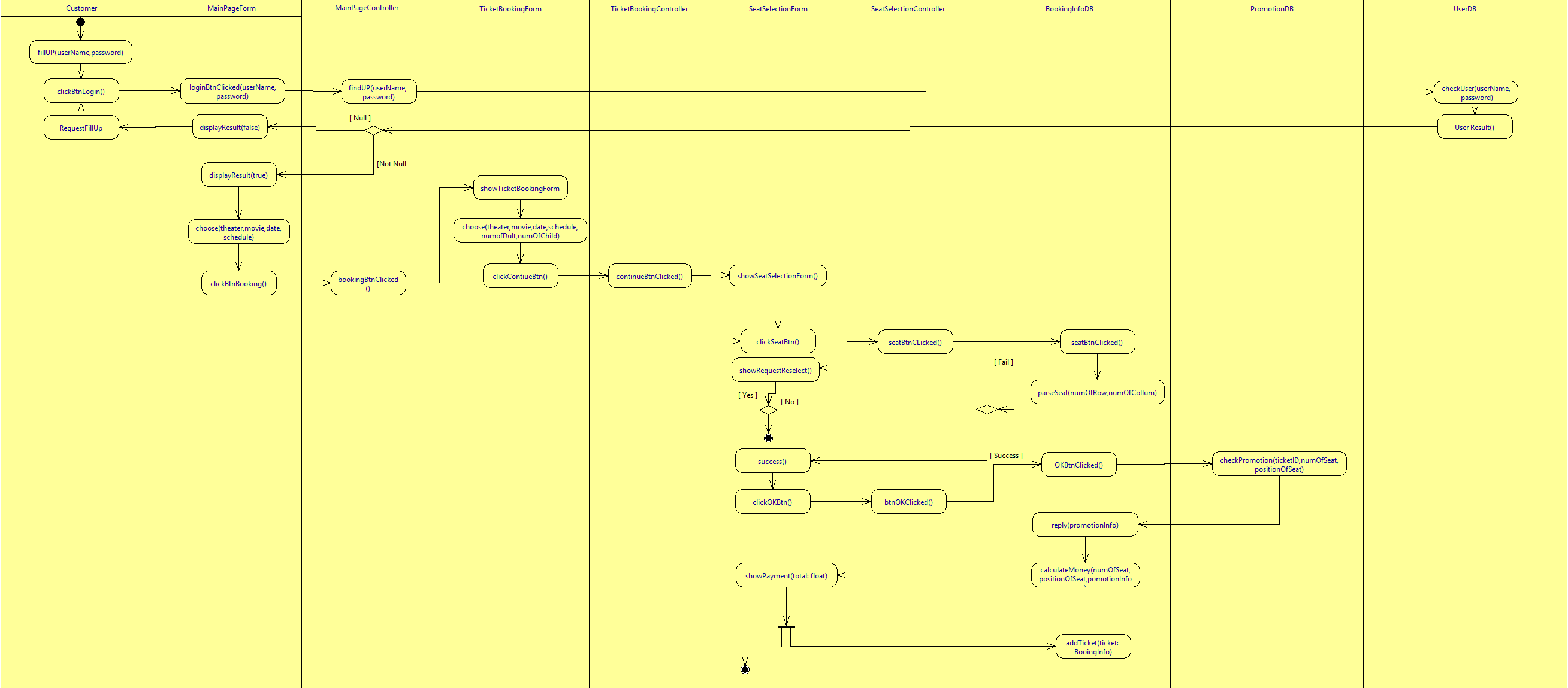


Figure 14 [Customer] ticket booking

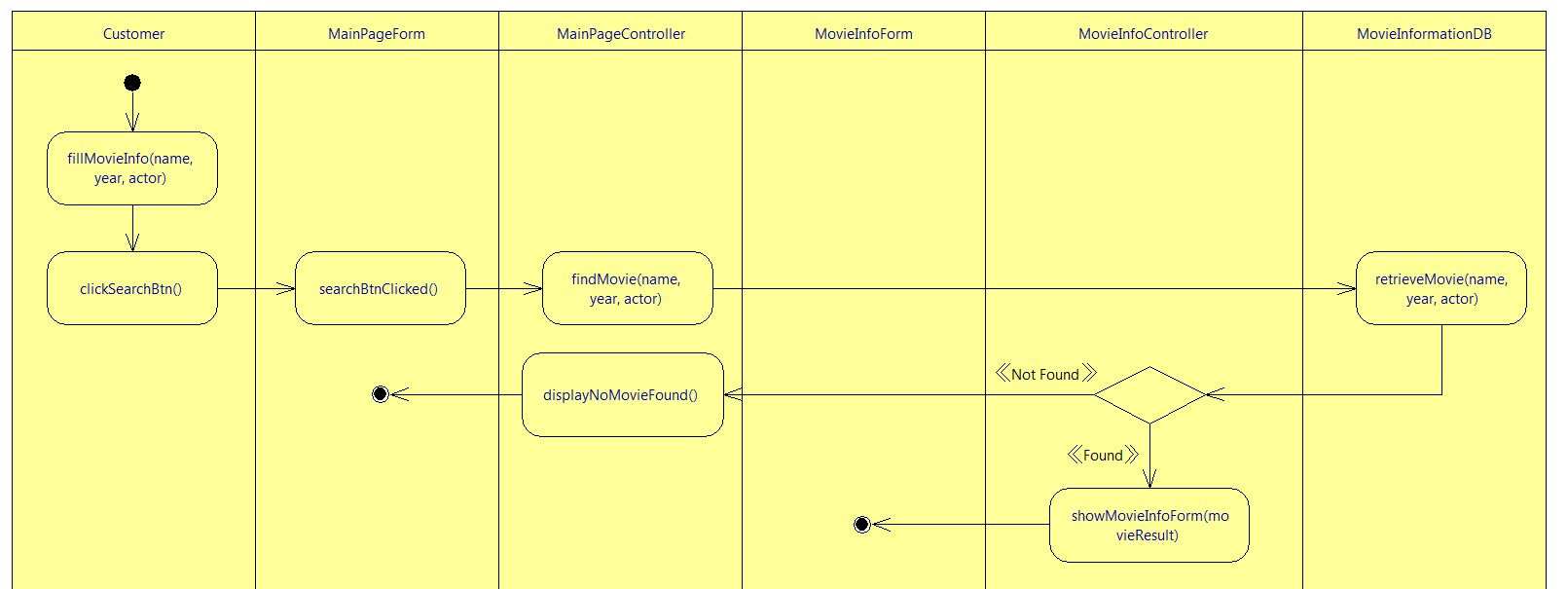


Figure 15 [Customer] search for movies

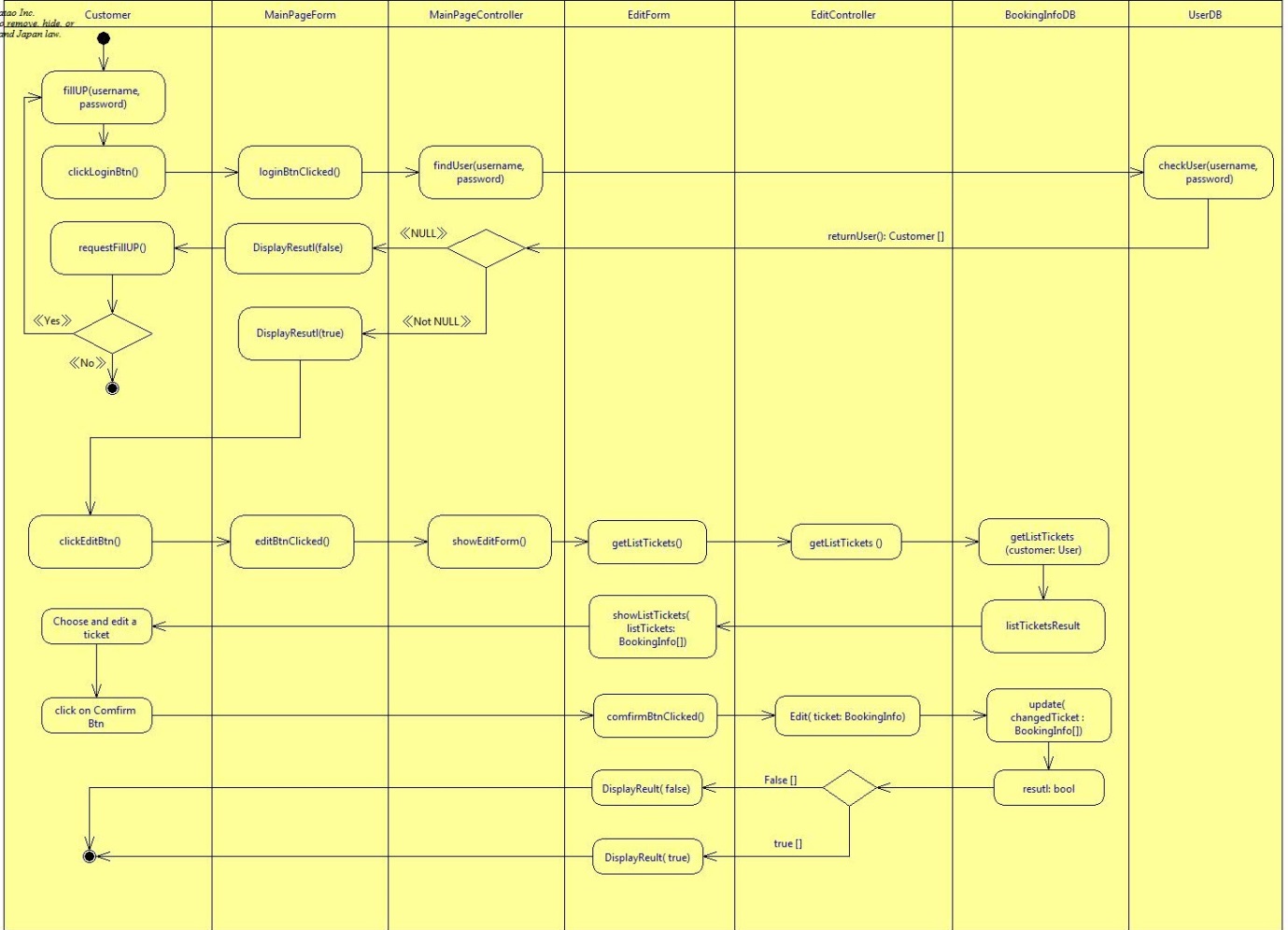


Figure 16 [Customer] edit one’s ticket booking

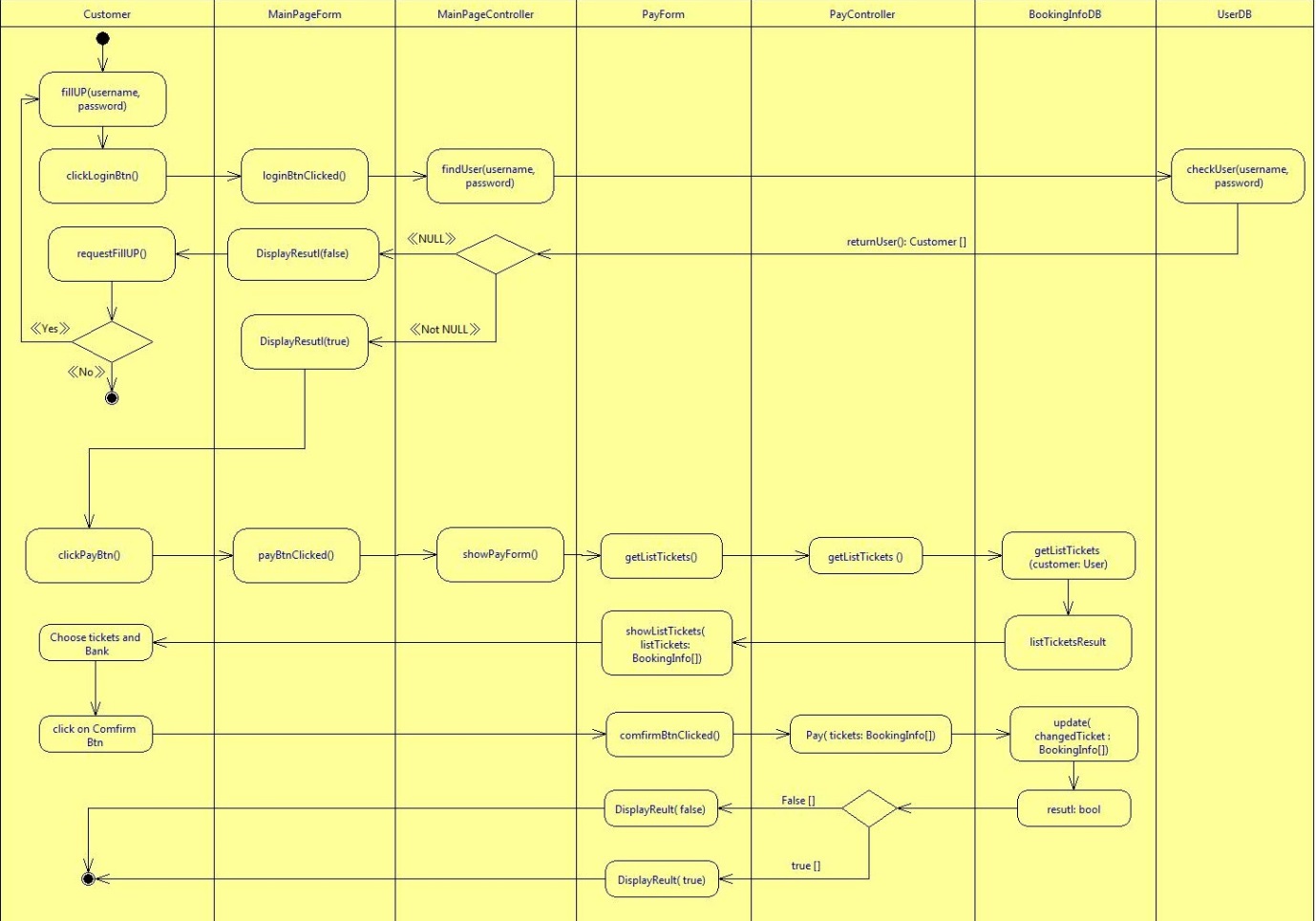


Figure 17 [Customer] ticket payment



Figure 18 [Customer] customer registry

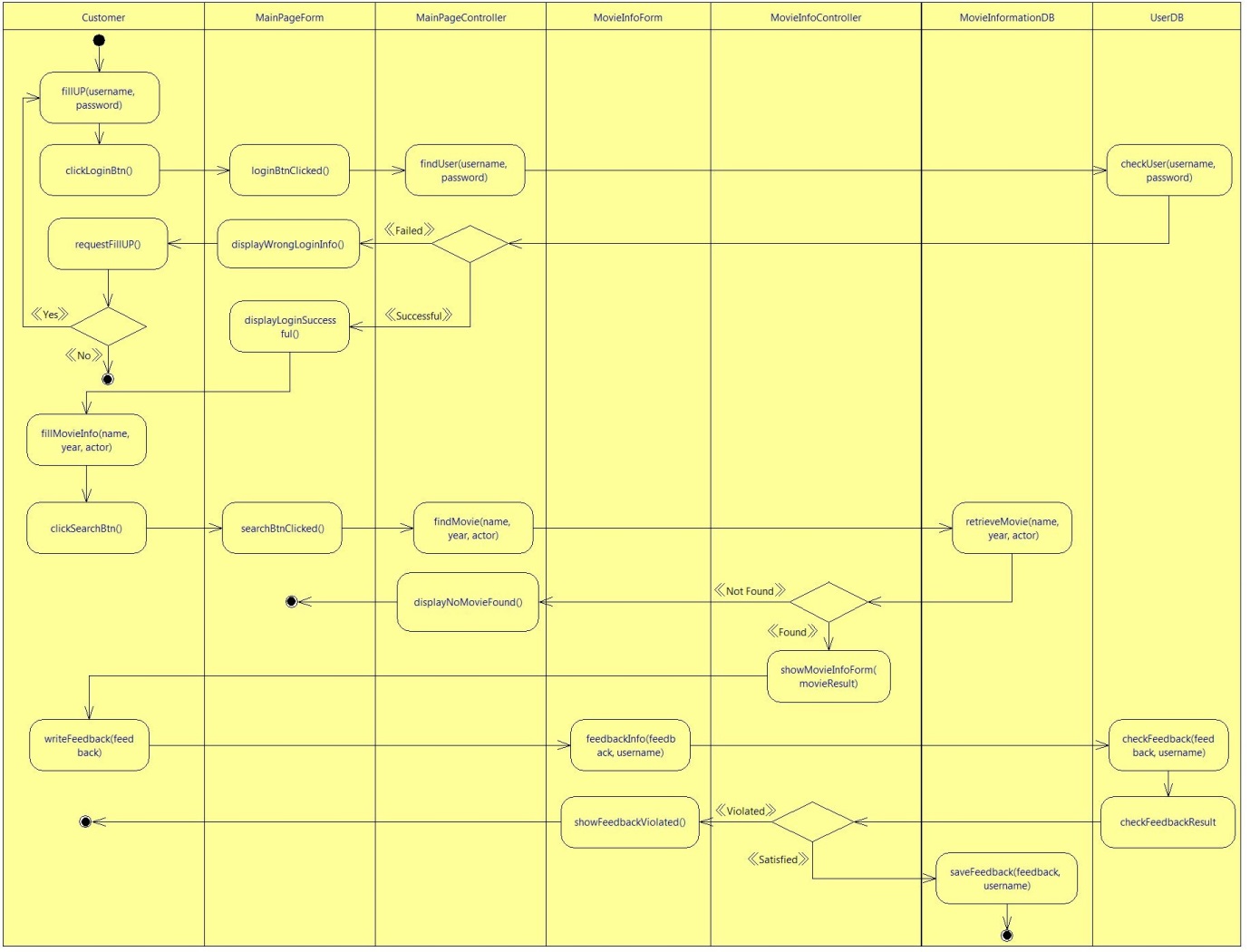


Figure 19 [Customer] send feedbacks to the managers

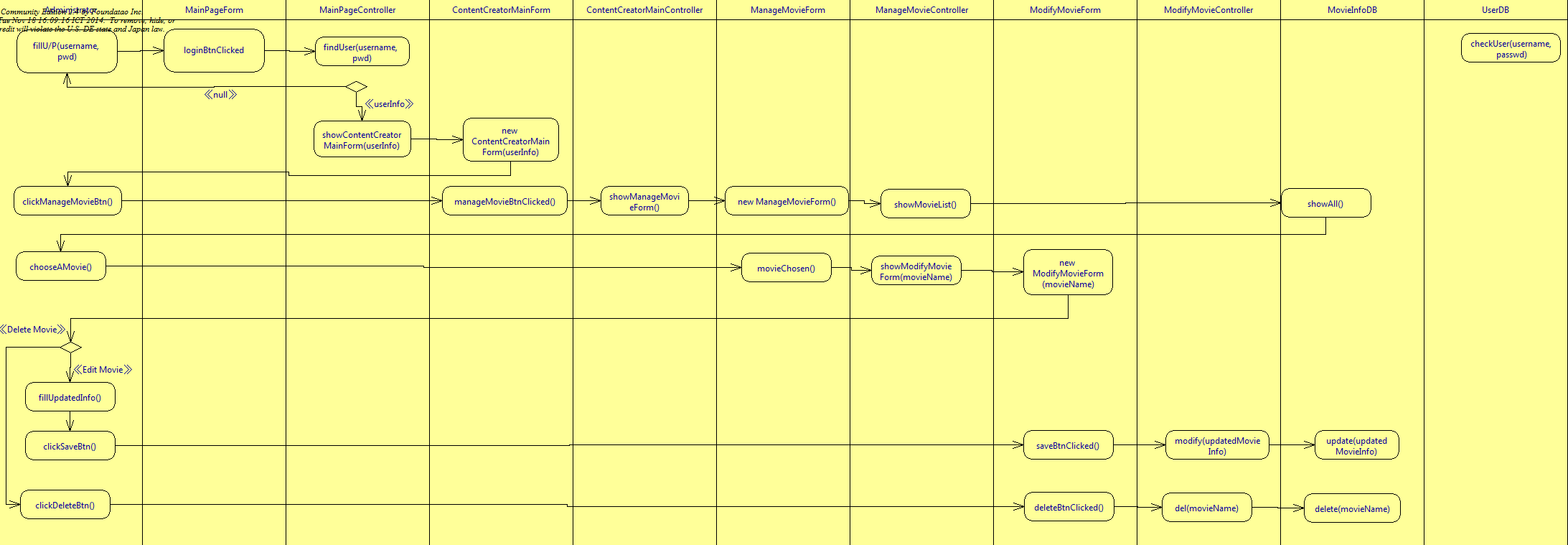


Figure 20 [Content creator] edit/delete movies’ information

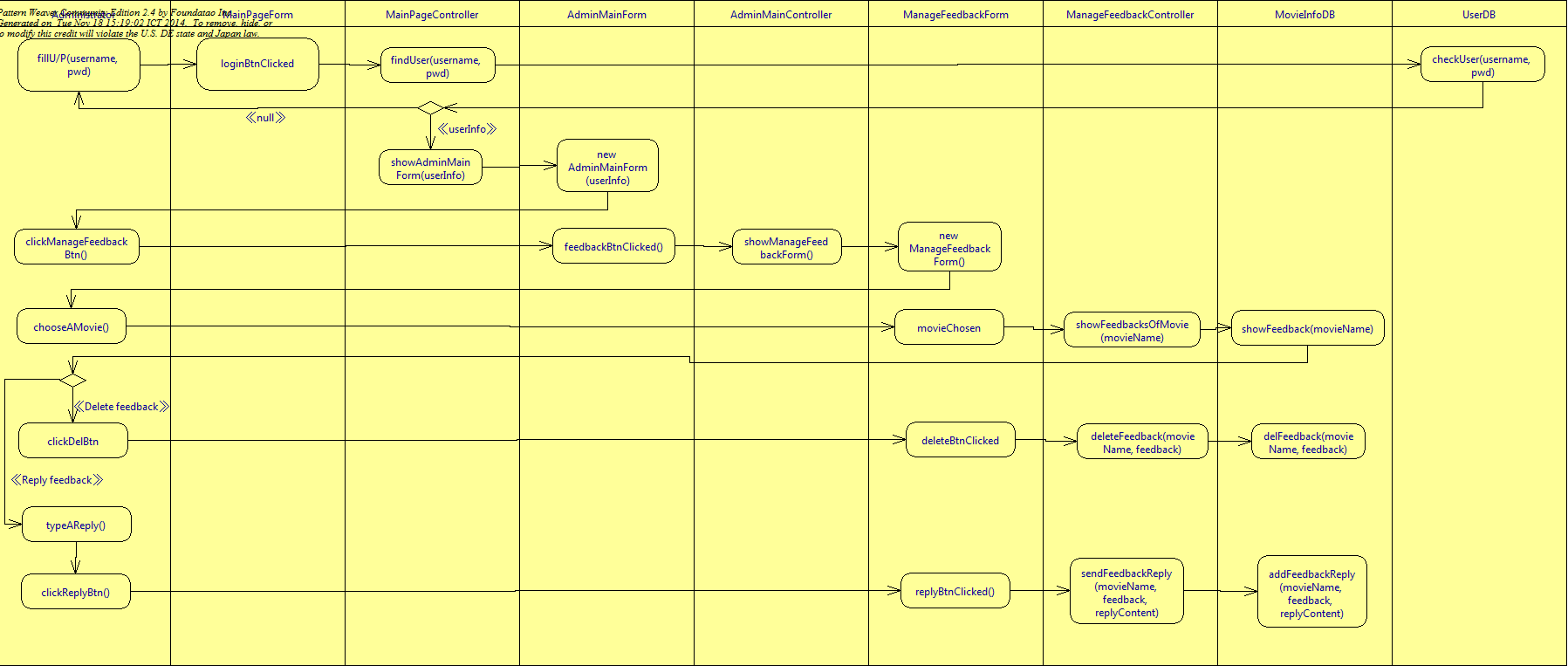


Figure 21 [Administrator] manage customers’ feedbacks

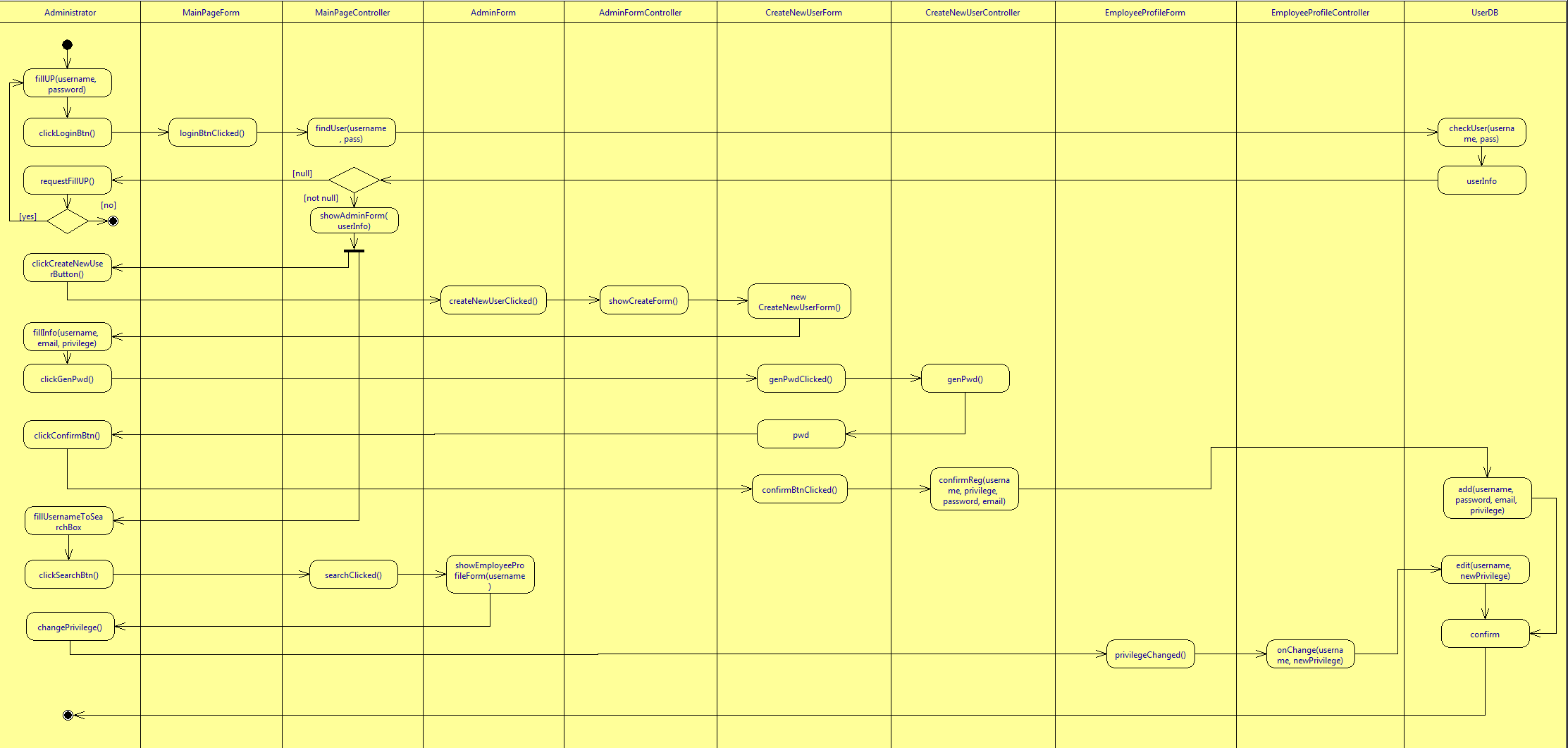


Figure 22 [Administrator] grant privilege to a specified user

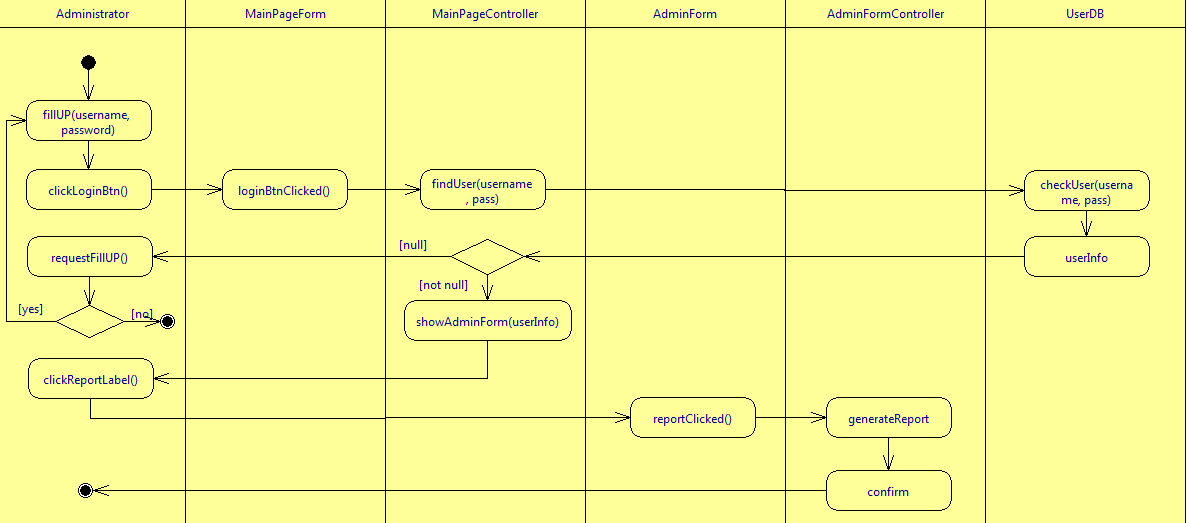


Figure 23 [Administrator] generate report

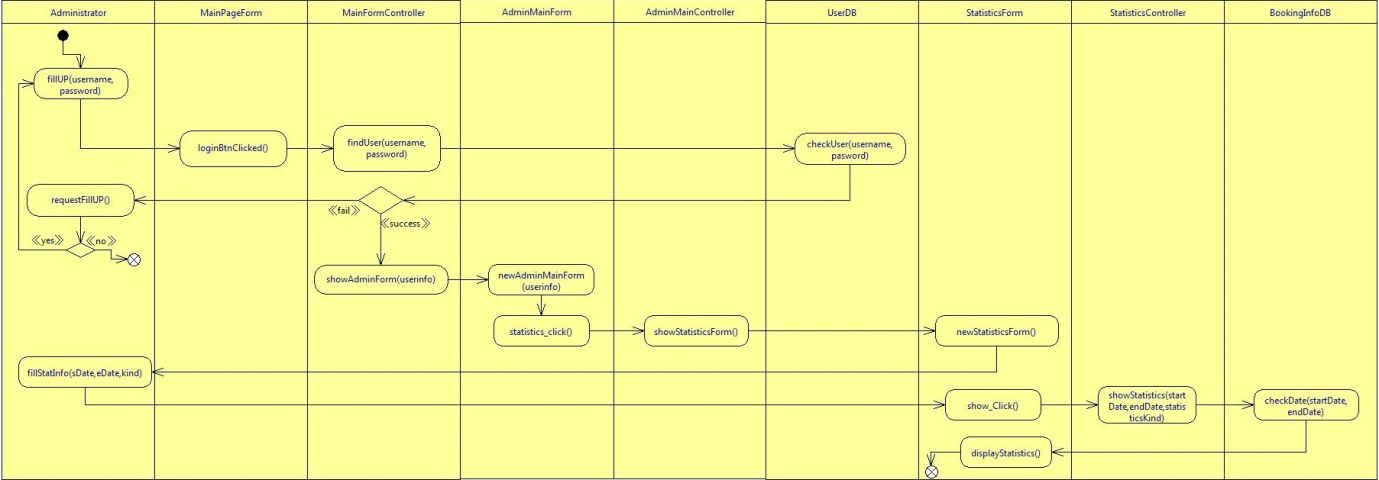
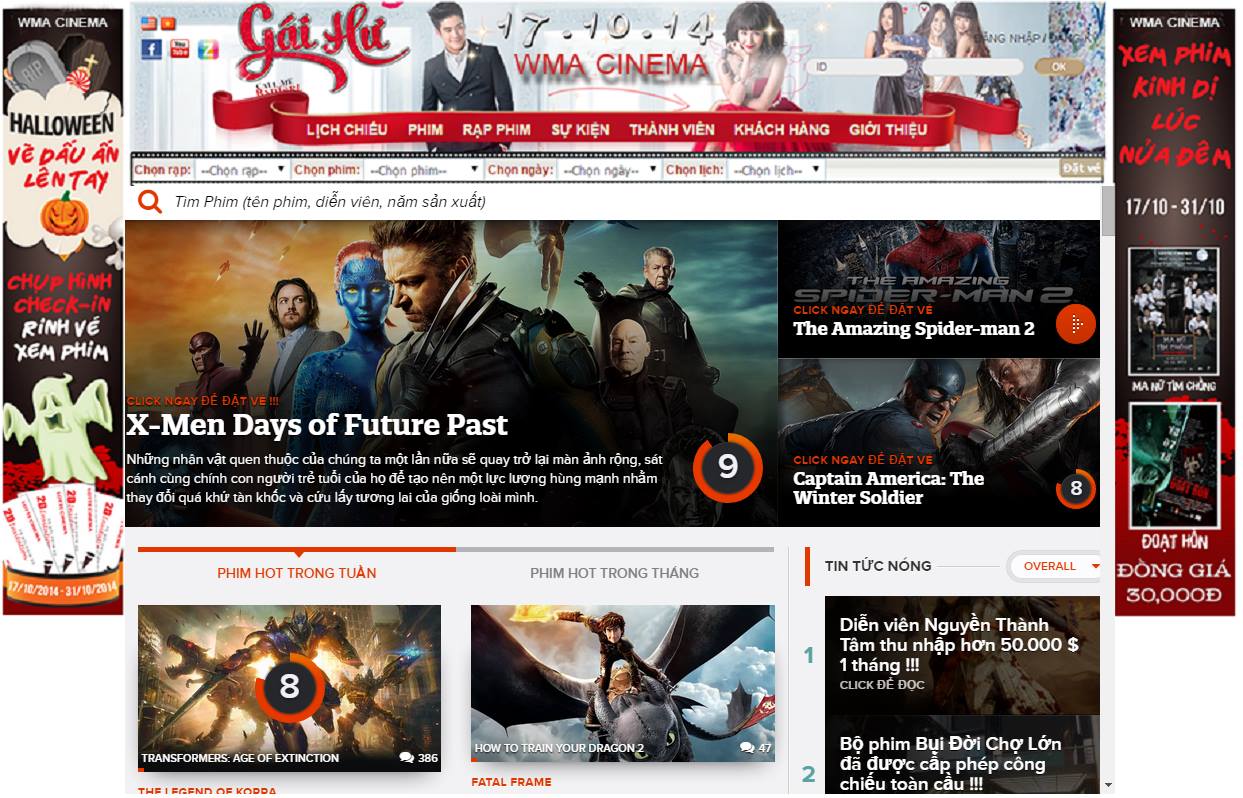


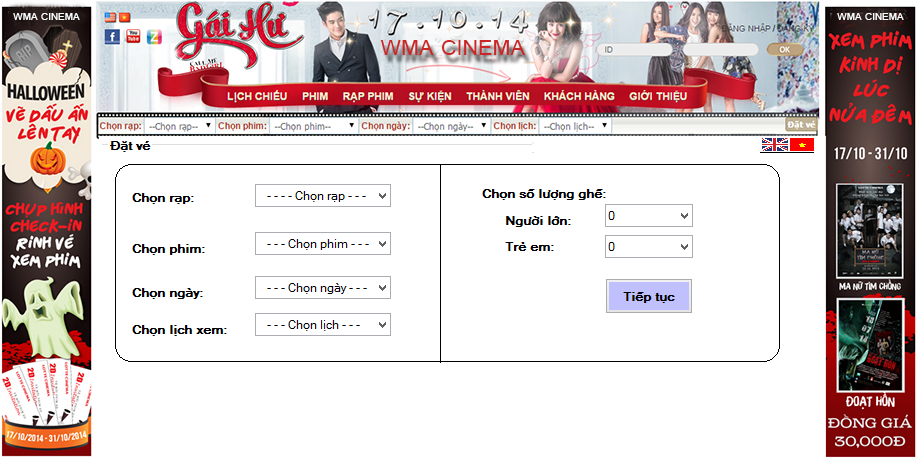
Figure 24 [Administrator] see statistics information

# **5.0. User Interface Design**

## 5.1. Main interface

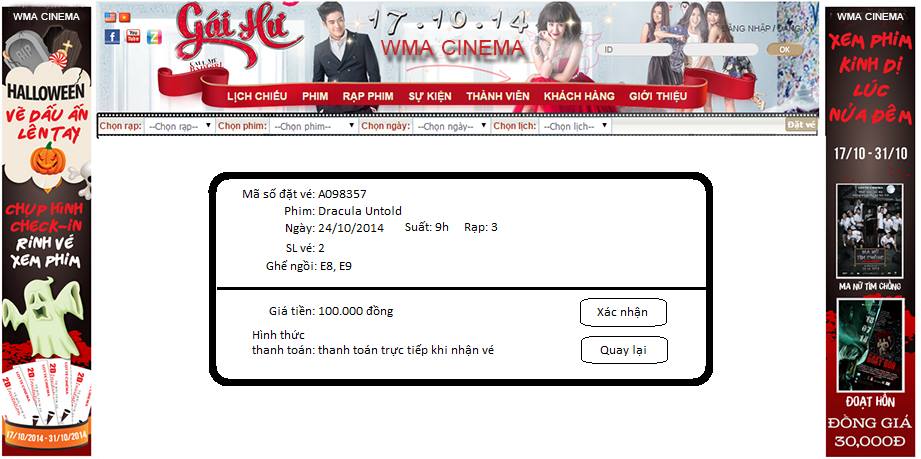


## 5.2. Booking movie interface

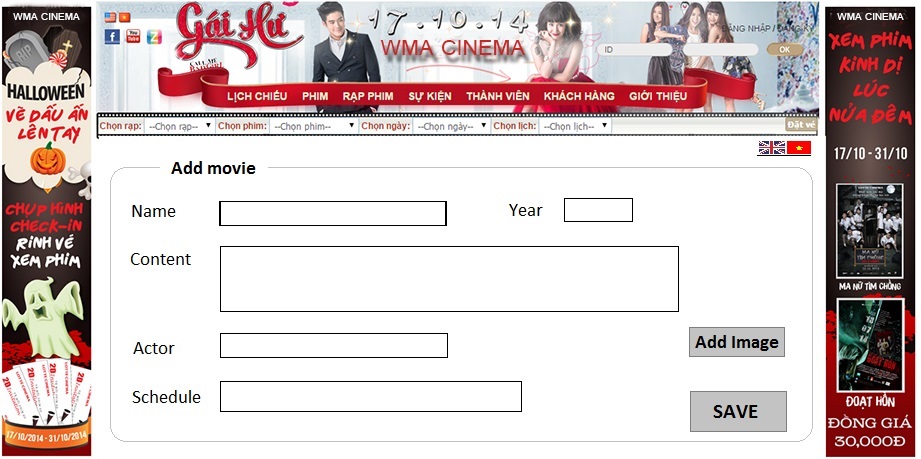


## 2

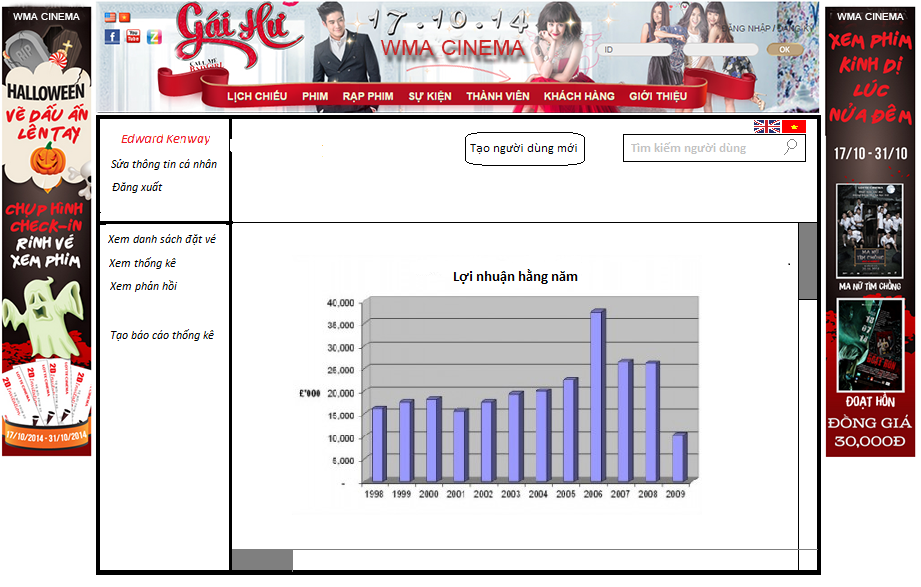
## 5.3. Paying ticket interface



## 5.4. Content creator interface

******

## 5.5. Administrator interface



*5.6 Register Interface*

