UniCin Report

Software Design and development | CFS2160

testing of software “unicin”

S.C.Edgar U1257802

2019

Contents

[**Modelling** 2](#_Toc4619914)

[**Login Page Testing** 6](#_Toc4619915)

[Login Button 6](#_Toc4619916)

[Continue as a Guest Button 9](#_Toc4619917)

[**Film Page Testing** 10](#_Toc4619918)

[Film Image Buttons 10](#_Toc4619919)

[**Film Detail Page Testing** 11](#_Toc4619920)

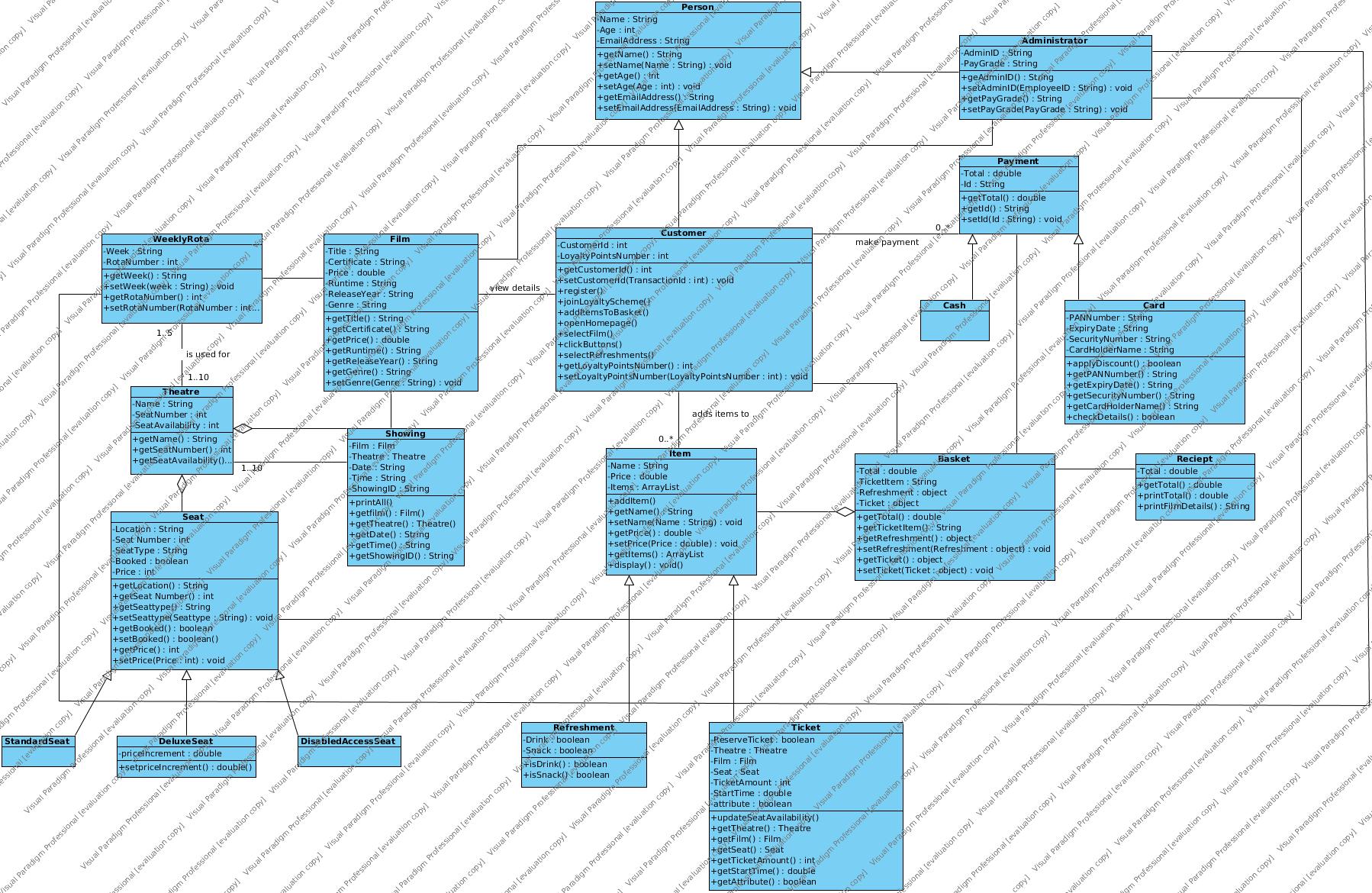
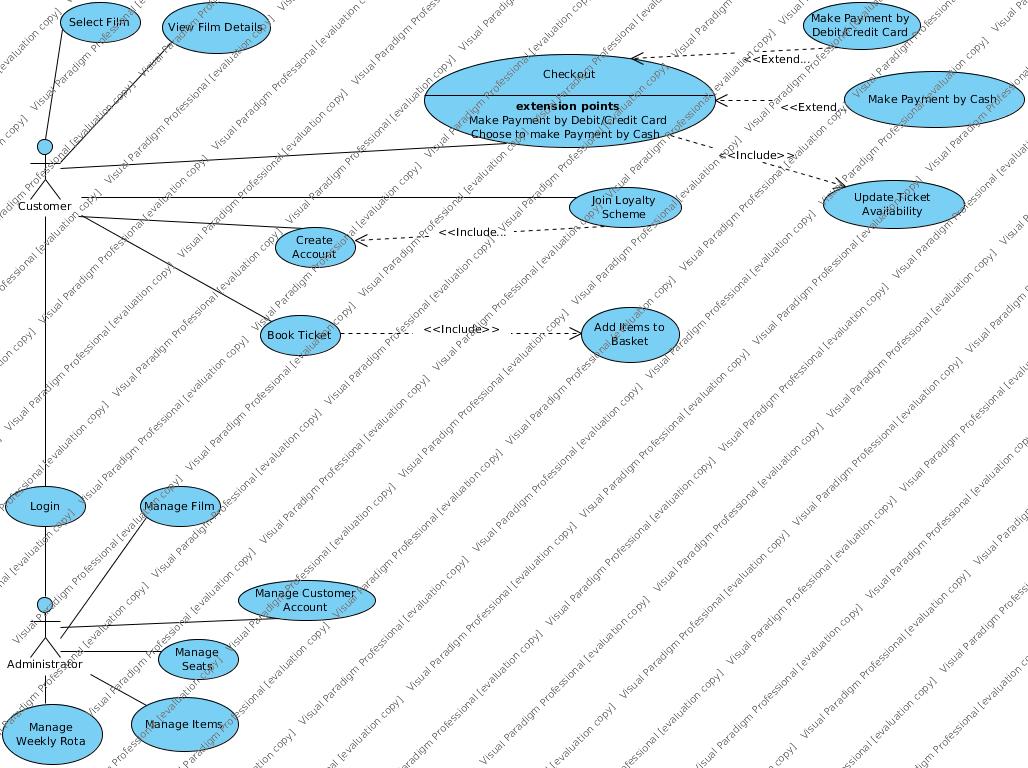
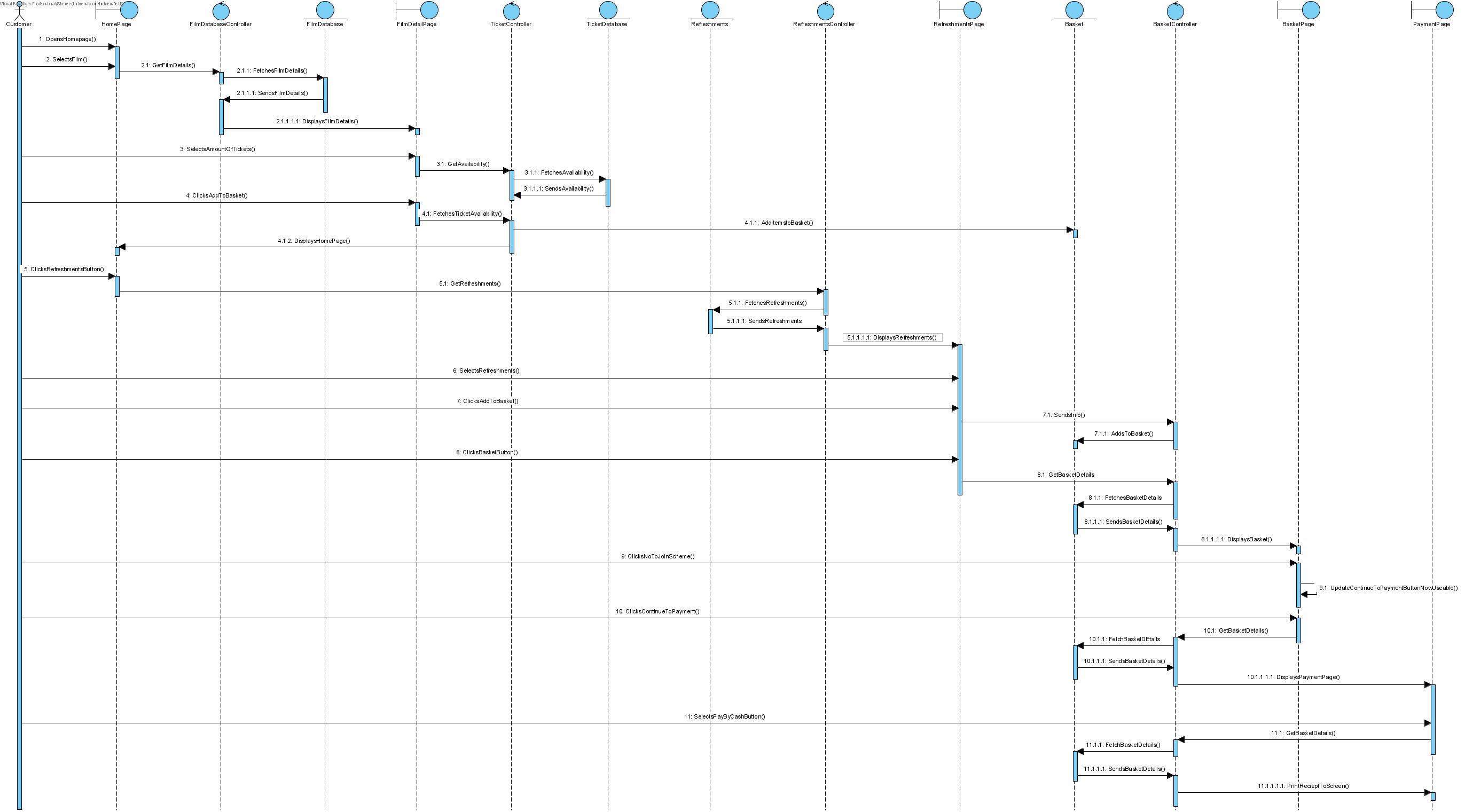
[Back to film options button 11](#_Toc4619921)

[Ticket Amount Spinner / Add to basket button 14](#_Toc4619922)

[Add to Basket Button 19](#_Toc4619923)

[Bibliography 20](#_Toc4619924)

# **Modelling**



# **Login Page Testing**

## Login Button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 Login Button  (Testing without any input)  The program should first check if there is a string in the username field and if not it will display an error message explaining what the problem is |  |  | The error message appears successfully. |
| Test 2  (Testing with only a username)  The program will be happy that there is a string in username however it will now pop up another alert to now enter a password |  |  | The error message appears successfully. |
| Test 3  (Testing with only a password)  The program should show the error message for the username because that’s what the if statement checks for first |  |  | The User Name error message appears as expected. |
| Test 4  (Testing with two random strings)  The program should display a different error message explaining that the information does not match a record |  |  | The error message appears successfully. |
| Test 5  (Testing with pre hardcoded customer data)  The program should take the input from the user and reference that against the pre-set data in the CustomerData Class to see if it exists if it does they will be able to continue to the next page through the login button if not they will again see an error message explaining that the information does not match a record. |  |  |  |

## Continue as a Guest Button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1  (Testing confirmation message)  When you click on the okay button you should be taken to the film selection page. |  |  | When clicking on the OK button it successfully takes the user to the film selection page. |
| Test 2  (Testing confirmation message)  When you click the cancel button it should leave you on the current page. |  |  | When selecting the cancel option on the alert it successfully keeps the user on the Login Page. |

# **Film Page Testing**

## Film Image Buttons

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (clicking the film image button)  When an image of the film is selected it will take the user to the Film Detail page which will populate all of the information for the selected film |  |  | The film image takes you to the new page where it has populated the information from the selected film from the film selection page.  The user interface on this screen is very simple however some other features could be added in future |

# **Film Detail Page Testing**

## Back to film options button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (clicking the options in the confirmation alert)  At this point in the program the user is warned that any tickets that have been selected will not be added to their basket |  |  | The confirmation alert opens as expected I have used the confirmation alert to allow the option to continue as expected or the user can cancel and stay on that page and continue. |
| Test 2 (using the cancel button)  When the user clicks the cancel button, they should remain on the film detail page. |  |  | Currently when clicking cancel it takes the user back to the film page which is incorrect, extra edge casing needs adding to the button handler to ensure the user will stay on the film detail page.  See code listings below for the solution on page 13. |

**Original event handler code for the ‘Back to Film Options’ button**

backBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert backConfirmation = new Alert(Alert.AlertType.CONFIRMATION);

backConfirmation.setHeaderText("Tickets will not be added to the basket unless the 'Add To Basket'" +

" button is pressed");

backConfirmation.showAndWait();

close();

FilmPage filmPage = new FilmPage();

fp.show();

}

});

**Amended event handler code for the ‘Back to Film Options’ button**

backBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert backConfirmation = new Alert(Alert.AlertType.CONFIRMATION);

backConfirmation.setHeaderText("Tickets will not be added to the basket unless the 'Add To Basket'" +

" button is pressed");

backConfirmation.showAndWait();

if (backConfirmation.getResult() == ButtonType.OK) {

close();

FilmPage filmPage = new FilmPage();

filmPage.show();

} else {

backConfirmation.close();

}

}

});

## Ticket Amount Spinner / Add to basket button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (using the spinner)  When the user uses the ticket amount spinner, they will be able to add up to 20 tickets to the basket, it will default to a value of zero. |  |  | When the user first visits the film detail page the spinner selector shows the value of zero which they can successfully increment to 20 at a maximum and they cannot go below zero.  When I created the spinner, I have set min, max and initial values to ensure the rules are adhered to.  I have also added a little label under the spinner to give the user the information on the limit of tickets available for one transaction. |
| Test 2 (adding zero ticket to basket)  If the user tries to add zero amount of tickets to basket it should pop up an error and keep the user on that page and not let them progress. |  |  | Currently the user can click the OK button on the error message and it lets the user carry on as normal the confirmation message comes up where they can continue to the next page.  Like an earlier problem in the program the event handler needs some more edge casing to ensure the user cannot continue to the basket without adding any tickets.  See below for code listing of solution on pages 16, 17 & 18. |

**Original event handler code for the ‘Add To Bakset’ button**

addToBasketBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert zeroAlert = new Alert(Alert.AlertType.ERROR);

if ((int)ticketSpinner.getValue() == 0) {

zeroAlert.setHeaderText("Please select an amount higher than 0 to add to basket");

zeroAlert.showAndWait();

}else{

int numberOfTickets = ((int)ticketSpinner.getValue());

MainPage.setNumberOfTickets(numberOfTickets);

}

Alert basketOptions = new Alert(Alert.AlertType.CONFIRMATION);

basketOptions.setTitle("Add to Basket");

basketOptions.setContentText("You will now be directed to the Refreshment page");

basketOptions.showAndWait();

close();

RefreshmentsPage rp = new RefreshmentsPage(item);

rp.show();

}

});

After rectifying the issue highlightest in Test 2 I realised that the event handler needed more work that I initially thought. Assessing the code I could see that the structure was completely wrong and I had code outside of my if statement. If the first statement was true it was then was jumping out of the loop and completing tasks that I didn’t want it to do at that point. I have completely reformatted the event handler code and it is now working as expected.

**Amended event handler code for the ‘Add To Bakset’ button**

addToBasketBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert zeroAlert = new Alert(Alert.AlertType.ERROR);

if ((int)ticketSpinner.getValue() == 0) {

zeroAlert.setHeaderText("Please select an amount higher than 0 to add to basket");

zeroAlert.showAndWait();

zeroAlert.close();

}else{

int numberOfTickets = ((int)ticketSpinner.getValue());

MainPage.setNumberOfTickets(numberOfTickets);

Alert basketOptions = new Alert(Alert.AlertType.CONFIRMATION);

basketOptions.setTitle("Add to Basket");

basketOptions.setContentText("You will now be directed to the Refreshment page" + "\n" +

"If you select cancel your tickets will not be added to the basket");

basketOptions.showAndWait();

if (basketOptions.getResult() == ButtonType.OK){

close();

RefreshmentsPage rp = new RefreshmentsPage(item);

rp.show();

} else {

basketOptions.close();

System.out.println(MainPage.getItems());

}

}

}

});

# **Refreshment Page Testing**

## Add Item button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (adding without selecting an item)  If the user tries to add an item without selecting one first, they will receive an error message. |  |  | The error is displayed as expected and clicking the OK button will close the alert window and allow the user to carry on and add items to basket. |
| Test 2 (adding a selected item)  When the user selects an item and then clicks the ‘Add Item’ button it should then appear in the Your Refreshments List. |  |  | The item is successfully added to the ‘Your Refreshments list. There is no limit on the number of items that you can add to the list, in future development that may want to be set. |

## Remove Item button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (removing without selecting an item)  If the user tries to remove an item without selecting one first, they will receive an error message. |  |  | The error is displayed as expected and clicking the OK button will close the alert window and allow the user to carry on and remove items from their Refreshment list. |
| Test 2 (removing a selected item)  When the user selects an item and then clicks the ‘Remove Item’ button it should then disappear from the ‘Your Refreshments’ List. |  |  | The item is successfully removed from the ‘Your Refreshments’ list. |

## Add to Basket button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (clicking the button with adding any items)  It is not compulsory for a user to add refreshments to their basket so the button will allow a user to got to the next page |  |  | The user is taken to the basket page as expected. |
| Test 2 (adding multiple items)  The user can add numerous items to their Refreshment list and when they click add to basket it, it will take them to the next screen and display the information of the items they added. |  |  | As you can see when the next page has loaded it has passed the information through from the previous page and displayed it underneath the order summary. |

# **Basket Page Testing**

## Back to Film Options button

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Result | Comments |
| Test 1 (clicking bac to film options button)  When the user clicks on the ‘Back to Film Options’ button they should see a confirmation alert that gives them some information about the action they are about to take. |  |  | The alert is displayed as intended giving the user some important information. |
| Test 2 (clicking OK on the alert)  When the user clicks the OK button on the alert it will taken them back to the film options page and forget what they have previously added to the basket. |  |  | The user is taken to the basket page as expected.  If I now choose a different film add that and different refreshments to the basket it will be the new information that has been supplied in the basket page. |
| Test 3 (clicking the Cancel button on the alert)  When the user clicks cancel they should be left on the same page with their basket still in tact. |  |  | The cancel button doesn’t do what is should do but instead takes the customer back to the film options page.  I need to look at the code and change it so that this doesn’t happen.  See below for code listing of solution on pages 28 & 29. |

**Original event handler code for the ‘Back to Film Options’ button**

backBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert backConfirmation = new Alert(Alert.AlertType.CONFIRMATION);

backConfirmation.setHeaderText("You will lose your current items in your basket");

backConfirmation.showAndWait();

close();

FilmPage filmPage = new FilmPage();

fp.show();

}

});

**Amended event handler code for the ‘Back to Film Options’ button**

backBtn.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Alert backConfirmation = new Alert(Alert.AlertType.CONFIRMATION);

backConfirmation.setHeaderText("You will lose your current items in your basket");

backConfirmation.showAndWait();

if (backConfirmation.getResult() == ButtonType.OK){

close();

FilmPage filmPage = new FilmPage();

filmPage.show();

} else {

backConfirmation.close();

}

}

});

# Bibliography

<https://www.youtube.com/watch?v=7LxWQIDOzyE> switching between scenes – this guy’s git hub <https://github.com/buckyroberts/Source-Code-from-Tutorials/blob/master/JavaFX/004_switchingScenes/Main.java>

<https://github.com/TonyJenkins/cfs2160-2018-java-public/blob/master/src/cfs2160/lectures/week15/social/network/v6/Post.java> tony polymorhism

<https://docs.oracle.com/javafx/2/ui_controls/button.htm> button set image to button

<https://www.youtube.com/watch?v=0aZy81etURE> tutorial helped with Spinner

<https://www.lynda.com/Java-tutorials/Lay-out-user-interface/466182/498141-4.html?autoplay=true> different pane uses

<https://github.com/TonyJenkins/cfs2160-2018-java-public/blob/master/src/cfs2160/lectures/week11/fund/raising/FundRaiser.java> formatting gb

<https://examples.javacodegeeks.com/desktop-java/javafx/listview-javafx/javafx-listview-example/> listview

<https://code.makery.ch/blog/javafx-dialogs-official/?fbclid=IwAR1F89XzcUhHShDieeWzkJeKepbrTC0n8QF9vnW47xVBiXwBVhRxMZAQ03U> alert type examples