HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

**FACULTY FOR HIGH QUALITY TRAINING**



**FINAL REPORT**

**SOFTWARE DESIGN PATTERNS**

**BUILDING A MOBILE APPLICATION**

**FOR BOOKING CINEMA TICKET**

**Lecturer: Nguyễn Trần Thi Văn,Msc**

**Class: DEPA330879E\_22\_2\_02CLC**

**Members:**

Phan Tấn Cường 20110356

Nguyễn Thanh Minh Triết 20110423

***Ho Chi Minh City,May 2023***

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

**FACULTY FOR HIGH QUALITY TRAINING**



**FINAL REPORT**

**SOFTWARE DESIGN PATTERNS**

**BUILDING A MOBILE APPLICATION**

**FOR BOOKING CINEMA TICKET**

**Lecturer: Nguyễn Trần Thi Văn,Msc**

**Class: DEPA330879E\_22\_2\_02CLC**

**Members:**

Phan Tấn Cường 20110356

Nguyễn Thanh Minh Triết 20110423

***Ho Chi Minh City,May 2023***

**COMMENTS**

**…………………………………………………………………………………….…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**…………………………………………………………………………………….**

**Score: …………………………**

*Sign*

**Nguyễn Trần Thi Văn**

**ACKNOWLEDGEMENT**

For the completion of this thesis, I would like to express my sincere thanks to:

The Board of Directors of Ho Chi Minh City University of Technology and Education for creating favorable conditions in terms of facilities with a modern library system, a variety of books and documents that are convenient for searching and researching information.

I would like to thank the subject lecturer – Mr. Nguyễn Trần Thi Văn for his dedicated and detailed teaching so that I have enough knowledge and apply them to this report.

Due to the lack of experience in making talent as well as the limitations of knowledge, in the report will certainly inevitably have shortcomings. We look forward to receiving comments, suggestions and criticism from the teacher to improve the report.

Finally, I would like to wish you good health, success and happiness.

**TABLE OF CONTENT**

[Chapter 1: SITUATION SURVEY AND DETERMINATION REQUEST 1](#_Toc136707731)

[3.1. Reasons for choosing the topic 1](#_Toc136707732)

[3.2. Research objectives 2](#_Toc136707733)

[3.3. Survey of the current sistuation 3](#_Toc136707734)

[Chapter 2: REQUIREMENTS ANALYSIS AND MODELING 5](#_Toc136707735)

[2.1. Functional requirement analyst 5](#_Toc136707736)

[2.2. Non-functional requirements analyst 6](#_Toc136707737)

[2.2.1 Security 6](#_Toc136707738)

[2.2.2 Safety 6](#_Toc136707739)

[2.2.3 Customer Support 6](#_Toc136707740)

[2.3. General Use Case 7](#_Toc136707741)

[2.4. Class diagram 8](#_Toc136707742)

[2.5. Detail use case for user 9](#_Toc136707743)

[2.5.1 Use Case Login 9](#_Toc136707744)

[2.5.2 Use Case Register 12](#_Toc136707745)

[2.5.3 Use Case Log Out 15](#_Toc136707746)

[2.5.4 Use Case Forget Password 18](#_Toc136707747)

[2.5.5 Use Case View Movie Information 21](#_Toc136707748)

[2.5.6 Use Case Change Password 24](#_Toc136707749)

[2.5.7 Use Case Booking Tickets 27](#_Toc136707750)

[2.5.8 Use Case Booking Food & Beverages 30](#_Toc136707751)

[2.5.9 Use Case Payment 32](#_Toc136707752)

[2.5.10 Use Case View Movie History 36](#_Toc136707753)

[2.6. Detail use case for admin 38](#_Toc136707754)

[2.6.1 Use Case Manage Movie Category 38](#_Toc136707755)

[2.6.2 Use Case Manage Food & Beverages 41](#_Toc136707756)

[2.6.3 Use Case Manage Movie 44](#_Toc136707757)

[2.6.4 Use Case View Revenue 47](#_Toc136707758)

[Chapter 3: IMPLEMENTED DESIGNS PATTERN 49](#_Toc136707759)

[3.1. Factory Method 49](#_Toc136707760)

[3.1.1. Definition 49](#_Toc136707761)

[3.1.2. Reason to use 49](#_Toc136707762)

[3.1.3. Applying design pattern 50](#_Toc136707763)

[Chapter4: INSTALLATION AND EXPERIMEMTAL RESULT 54](#_Toc136707764)

[4.1. Results 54](#_Toc136707765)

[4.2. Limitations 54](#_Toc136707766)

[Chapter 5: CONCLUSION 55](#_Toc136707767)

[WORK ASSIGNMENT TABLE 56](#_Toc136707768)

[REFERENCES 57](#_Toc136707769)

**TABLE OF USECASE DESCRIPTION**

[Table 2.5. 1:Use Case Login 10](#_Toc136686164)

[Table 2.5. 2: Use case register 13](#_Toc136686165)

[Table 2.5. 3: Use case log out 16](#_Toc136686166)

[Table 2.5. 4: Use case forget password 19](#_Toc136686167)

[Table 2.5. 5: Use case view movie information 22](#_Toc136686168)

[Table 2.5. 6: Use case change password 25](#_Toc136686169)

[Table 2.5. 7: Use case booking ticket 28](#_Toc136686170)

[Table 2.5. 8: Use case booking food & beverages 30](#_Toc136686171)

[Table 2.5. 9: Use case payment 34](#_Toc136686172)

[Table 2.5. 10: Use case view movie history 36](#_Toc136686173)

[Table 2.6. 1: Use case manage movie category 39](#_Toc136686004)

[Table 2.6. 2: Use case manage food & beverages 42](#_Toc136686005)

[Table 2.6. 3: Use case manage movie 45](#_Toc136686006)

[Table 2.6. 4: Use case view revenue 47](#_Toc136686007)

**TABLE OF FIGURE**

[Figure 2.3. 1: General use case 7](#_Toc136707810)

[Figure 2.4. 1: Class diagram 8](#_Toc136687276)

[Figure 2.5. 1: Sequence diagram login 11](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687313)

[Figure 2.5. 2: Sequence diagram register user‘s account 14](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687314)

[Figure 2.5. 3: Sequence diagram log out 17](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687315)

[Figure 2.5. 4: Sequence diagram forget password 20](#_Toc136687316)

[Figure 2.5. 5: Sequence diagram view movie information 23](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687317)

[Figure 2.5. 6: Sequence diagram change user’s account password 26](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687318)

[Figure 2.5. 7: Sequence diagram booking ticket 29](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687319)

[Figure 2.5. 8: Sequence diagram booking food & beverages 31](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687320)

[Figure 2.5. 9: Sequence diagram making payment 35](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687321)

[Figure 2.6. 1: Sequence diagram manage movie categogy 40](file:///D:\Software%20Design%20Patern\Project\Final%20Project\Final_Report_Design_Pattern.docx#_Toc136687306)

[Figure 2.6. 2: Sequence diagram manage food & beverages 43](#_Toc136687307)

[Figure 2.6. 3: Sequence diagram maange movie 46](#_Toc136687308)

[Figure 2.6. 4: Sequence diagram view revenue 48](#_Toc136687309)

# Chapter 1: SITUATION SURVEY AND DETERMINATION REQUEST

1. **Reasons for choosing the topic**

The group's topic is to design an application on the Android platform, capable of helping users can book movie tickets online without going to competitors, our group has chosen this topic because of the fact that nowadays, when people want to see a movie,they have to go to the theater to buy tickets and have to wait until the movie starts showing. If we get close to the newly purchased showtime, there will be no more tickets or no more places we want. On the other hand, when buying tickets, they often have to wait in long lines on weekends to buy tickets. Therefore, when this online ticket booking application is completed, it will help a lot of people save time when they want to see a certain movie, and can watch it conveniently.

1. **Research objectives**

- Research on Google API, Facebook API.

- Understand the integration and utilization of Firebase as the backend database for the application.

- Investigate the development of an Android application for movie ticket booking.

- Learn the process of managing a chain of theater.

- Build a mobile application for booking movie ticket online with a friendly interface, easy to use.

1. **Survey of the current sistuation**

In this digital era, the usage of smartphones and mobile applications is rapidly growing. According to a survey conducted by QandMe in 2020, the average daily mobile app usage time in Vietnam is 5.1 hours. Therefore, having an app provides an opportunity to showcase your brand, leading to higher profits for businesses.

In the Vietnamese cinema market, going to the movies is a popular entertainment activity, especially among the youth. However, when purchasing tickets directly at the cinema, customers often have to endure long queues and the inconvenience of traveling.

To address this issue, many cinema businesses have chosen to design an app as a solution. This allows customers to easily access information about movie tickets, schedules, and available seating options.

Furthermore, cinema apps are tools that make the ticket booking and purchasing process more comfortable than ever before. When users have a seamless shopping experience on the app, businesses attract a significant number of loyal customers who regularly utilize their services.

By leveraging the benefits of a movie ticket booking app, businesses can streamline the ticketing process, enhance customer satisfaction, and ultimately increase their box office revenue.

In Viet Nam, the most well-known cinema business is CGV cinemas. CGV cinemas application has partially addressed the challenges in today's business environment by providing essential features and functionalities that small and medium-sized businesses need. They have successfully built necessary platforms to automate management processes for customers with similar business models.

General addvantages of CGV cinemas application:

* Supports quick and easy online ticket booking and purchase.
* Continuously updates users with the latest information about upcoming action movies and blockbusters.
* Supports user login and allows access to personal information and membership benefits.
* Helps users find the nearest movie theaters.
* Attractive user interface

Some limitations:

* Instability and slow performance
* Connectivity and payment errors
* Lack of function such as movie ratings, ticket history storage, and social interaction.
* Have not applied AI technology yet

# Chapter 2: REQUIREMENTS ANALYSIS AND MODELING

1. **Functional requirement analyst**

The system consists of the following main components:

1. User section

* User can see posters of new cinema movies displayed on the main screen.
* Users can click on the movie poster to view information related to the movie (actors, movie intro, movie genre, etc.).
* On the movie information display, User can click on the movie trailer to view the trailer.
* Users can see the critics' and audience's ratings below shown on the info screen (under the movie trailer video).
* User can choose the registration method to create their app account.
* User is entitled to User features and can book tickets.
* User can recover password if forgot password.
* User can change personal information as well as password (for password, a confirmation code will be sent to User's contact method).
* In the ticket booking section, users can choose movie information (cinema, slottime, seats, type of seats).
* User can order more food and drink of the theater.
* Users can use vouchers/points cards before paying (if any)
* Users can pay using multiple payment methods.
* After payment, User will have payment invoice shown on screen.
* User can view transaction history in the last 3 months.

1. Admin section

* Admin interacts on a separate interface that is different from that of User and User.
* Admin can add delete edit movie / movie information / fast food information.
* Admin can see the theater's sales through the theater's monthly revenue statistics table

## Non-functional requirements analyst

1. Security

Decentralize users in the system, encrypt passwords, use tokens to authenticate users

1. Safety

Apply information security techniques to prevent hackers from entering the database and taking over user accounts.

High reliability: The mobile application should always be up and running and meet the quality standards that guarantee its reliability.

1. Customer Support

The mobile application have not provided customer support channels such as live chat, email and phone to resolve customer issues yet.

## General Use Case

Figure 2.3. : General use case

## Class diagram

Figure 2.4. : Class diagram

## Detail use case for user

### Use Case Login

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.1 |
| **Use Case Name** | Login |
| **Description** | This use case allows the actor to login application by account. |
| **Actor(s)** | User/Admin |
| **Priority** | Must Have |
| **Precondition(s):** | Users need to have a BoxCine app account |
| **Post-Condition(s):** | User successfully login to the application. |
| **Main Flow** | 1. Actor enters account information with user gmail and password. 2. Actor selects “User” role[A1] 3. Actor presses the “Login” button. 4. System checks login information[E1] 5. The system authenticates the login information successfully. 6. The system returns the actor main form[A2] |
| **Alternative Flow** | [A1] Actor selects "Administrrator” role  [A2] The system returns the admin main form |
| **Exception Flow** | [E1] User enters incorrect login information, user repeats step (1). |

Table 2.5. :Use Case Login

**Sequence diagram**

Figure 2.5. : Sequence diagram login

### Use Case Register

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.2 |
| **Use Case Name** | Register |
| **Description** | This use case allows actor to create an application account and allows the actor to sign in application after having an account. |
| **Actor(s)** | User/Admin. |
| **Priority** | Must Have |
| **Precondition(s):** |  |
| **Post-Condition(s):** | The system returns actor main form after registering actor account. |
| **Basic Flow** | 1. Actor presses the “Register” button. 2. The system returns the BoxCine application account registration form. 3. Actor fills in email and password field and selects “User” role[A1]. 4. Actor presses the “Register” button. 5. The system checks invalid account information[E1]. 6. The system returns user main form[A2]. |
| **Alternative Flow** | [A1]Actor fills in gmail and password field and selects “Administrator” role.  [A2] The system returns admin main form |
| **Exception Flow** | [E1] User enters the invalid information, the system reports an error, the user performs step (3) |

Table 2.5. : Use case register

****

Figure 2.5. : Sequence diagram register user‘s account

**Sequence diagram**

### Use Case Log Out

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.3 |
| **Use Case Name** | Log out |
| **Description** | This Use Case allows users to log out of their application account. |
| **Actor(s)** | User |
| **Priority** | Must Have |
| **Precondition(s):** | The user account has been successfully logged into the application.  The user's device is already connected to the internet. |
| **Post-Condition(s):** | User successfully logs out of the application. The system records the successful logout from BoxCine application. |
| **Basic Flow** | 1. User accesses BoxCine application. 2. The user has logged into the BoxCine application. 3. The user chooses the method of logging out of the system. 4. The system validates the login information successfully and allows the user to access the application with the User account. If the user does not confirm the logout from the system, the system will keep the status quo. 5. The system records successful logout from BoxCine application. |
| **Alternative Flow** |  |
| **Exception Flow** |  |

Table 2.5. : Use case log out

**Sequence diagram**

Figure 2.5. : Sequence diagram log out

### Use Case Forget Password

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.4 |
| **Use Case Name** | Forget Password |
| **Description** | This Use Case allows users to recover passwords when they forget or lose their current passwords. |
| **Actor(s)** | User |
| **Priority** | Must Have |
| **Precondition(s):** | The user account has been created and exists on the system. |
| **Post-Condition(s):** | The user receives the message "BoxCine application account password reset successful" from the Gmail communication method. |
| **Basic Flow** | 1. User accesses BoxCine application. 2. User chooses to forget BoxCine account password 3. User fills in the account name via BoxCine application 4. The system will send sms containing OTP code to the user's phone number. 5. After the user authenticates the OTP code on the system, the system authenticates and sends a password reset application to the user via the email registered by the user. 6. The user enters gmail and clicks on the website link to reset the password on Gmail. 7. After entering the new password and re-authenticating the new password, the user selects the “Agree” button and the system will re-enter the user password. 8. After validating the new password, the system will take the user back to the login page |
| **Alternative Flow** |  |
| **Exception Flow** | [E1] The system validates that the account information is incorrect or does not exist.  The system displays the wrong account name or the account name does not exist. |

Table 2.5. : Use case forget password

**Sequence diagram**

****

Figure 2.5. : Sequence diagram forget password

### Use Case View Movie Information

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.5 |
| **Use Case Name** | View Movie Information |
| **Description** | Allows user to view movie descriptions (genre, actors, directors, etc.) |
| **Actor(s)** | User |
| **Priority** | Must have |
| **Precondition(s):** | User successfully login application |
| **Post-Condition(s):** | The user will enter a describing information form which related to the movie. |
| **Basic Flow** | 1. The system asks to display a list of movie posters. 2. The list of posters will be displayed on the main screen of the application. 3. User clicks on the movie poster image[A1]. 4. The application displays a form describing the movie information. |
| **Alternative Flow** | [A1] User clicks on the category movie image.   1. The list of posters will be displayed on the category form of the application. 2. User clicks on the movie poster image. 3. The application displays a form describing the movie information |
| **Exception Flow** |  |

Table 2.5. : Use case view movie information

**Sequence diagram**

Figure 2.5. : Sequence diagram view movie information

### Use Case Change Password

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.8 |
| **Use Case Name** | Change Password |
| **Description** | This use case allows the user to change the user's account password, the system must update for the next logins. |
| **Actor(s)** | User |
| **Priority** | Optional |
| **Precondition(s):** | User must be awake UC-1.6 |
| **Post-Condition(s):** | The user receives the message “Password changed successfully from the communication method. |
| **Basic Flow** | 1. After entering the "personal information" form successfully, the user clicks on the password item. 2. in the password section, the user password will be displayed. 3. The user selects the change button in the password section 4. The system will switch to the password change form. 5. User enters a new password. 6. User presses the button and confirms the new password. 7. The system sends a confirmation code to the user gmail. 8. User waits and receives confirmation code 9. The user enters the confirmation code in the password change form and clicks the “Confirm” button. 10. The system notifies the successful password update and the user confirms it. |
| **Alternative Flow** |  |
| **Exception Flow** |  |

Table 2.5. : Use case change password

**Sequence diagram**

Figure 2.5. : Sequence diagram change user’s account password

### Use Case Booking Tickets

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.9 |
| **Use Case Name** | Booking Tickets |
| **Description** | This Use Case allows users to book tickets directly on this movie ticket booking application and can issue that ticket to the cashier for confirmation. |
| **Actor(s)** | User |
| **Priority** | Must Have |
| **Precondition(s):** | User successfully login and in the “View Information Movie” form |
| **Post-Condition(s):** | Display the ticket information screen that you booked. |
| **Basic Flow** | 1. User presses “book ticket” button 2. The system returns to booking ticket. 3. User selects cinema 4. User selects screening time 5. User selects the seat position and the number of viewing seats. 6. User selects payment method. 7. User presses “Accept” button 8. The system checks the information[E1]. 9. The system returns detail information ticket screen |
| **Alternative Flow** |  |
| **Exception Flow** | [E1] User have not selected fields (cinema/screening time/number of seat) , the system notifies error message and UC-1.9 end. |

Table 2.5. : Use case booking ticket

**Sequence diagram**

****

Figure 2.5. : Sequence diagram booking ticket

### Use Case Booking Food & Beverages

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-1.10 |
| **Use Case Name** | Booking Food & Beverages |
| **Description** | This use case allows the user to order more food or drink while watching the movie, and the information about the food or drink will display the ticket information form. |
| **Actor(s)** | User |
| **Priority** | Optional |
| **Precondition(s):** | User have to be in booking ticket form. |
| **Post-Condition(s):** | The food/drink information selected by the user is displayed on the booking ticket form. |
| **Basic Flow** | 1. The system returns a form containing a list of food and food items. 2. User chooses the food or drink which user want to select 3. User presses “Accept” button. |
| **Alternative Flow** |  |
| **Exception Flow** |  |

Table 2.5. : Use case booking food & beverages

**Sequence diagram**

Figure 2.5. : Sequence diagram booking food & beverages

### Use Case Payment

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | Usecase-1.11 |
| **Use Case Name** | Payment |
| **Description** | This use case allows users to pay their own costs. |
| **Actor(s)** | User |
| **Priority** | Must Have |
| **Precondition(s):** | User has logged in account  User has made the booking function  Users must have an e-wallet, a bank account |
| **Post-Condition(s):** | The user successfully pays the cost of using the service.  The system records the successful payment operation. |
| **Basic Flow** | 1. In the ticket information form, the user presses the "Payment" button. 2. The user selects the payment method and confirms the payment of his expenses. 3. The system will record it after the user has successfully paid and send the invoice to the user. 4. The system records successful payment operations on the system. |
| **Alternative Flow** | 2a.User chooses payment method by bank account.  2a(1) The system switches the screen to the bank selection screen and confirms the bank information.  3a.After the user selects the bank and confirms the bank information and selects the payment button.The user needs to enter the security code sent by the bank.  4a.User selects the cost payment confirmation button.  5a.After successful transfer, the system confirms and takes the user back to the app  2b.User chooses payment method by e-wallet.  2b(1) The system switches the screen to the screen to select the e-wallet the user uses and confirm the wallet information.  3b. After the user selects the e-wallet, confirm the e-wallet information in use and select the payment button. The user needs to enter the security code of the e-wallet sent back.  4b.User selects the payment confirmation button  5b.After successful transfer, the system confirms and takes the user back to the application. |
| **Exception Flow** | [E1] User chooses to cancel login.  Use Case stops.  [E2]The user cancels the payment or closes the application.  Use Case stops. |

Table 2.5. : Use case payment

****Sequence diagram**

Figure 2.5. : Sequence diagram making payment

### Use Case View Movie History

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | Usecase-1.12 |
| **Use Case Name** | Movie History |
| **Description** | This usecase allows users to view transaction history with the store for tickets, food and drinks |
| **Actor(s)** | User |
| **Priority** | Must Have |
| **Precondition(s):** | User has logged in account  The user has successfully paid for the in-app transaction. |
| **Post-Condition(s):** | Users can review their transaction history.  Users can use transaction history such as invoices to get movie tickets, food and drinks. |
| **Basic Flow** | 1. User accesses BoxCine application. 2. User go to the list and click on “my ticket” to review the transaction history. 3. The system will display the transactions that the user's account has transacted online within the last 3 months. |
| **Alternative Flow** |  |
| **Exception Flow** | [E1] Database system error Use Case stopped |

Table 2.5. : Use case view movie history

**Sequence diagram**

Figure 2.5. : Sequence diagram view movie history

**Collaboration diagram**

## Detail use case for admin

1. Use Case Manage Movie Category

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-2.1 |
| **Use Case Name** | Manage Movie Category |
| **Description** | This use case allows admin to add, edit, delete movie categories. For users to interact and use the service |
| **Actor(s)** | Admin |
| **Priority** | Must Have |
| **Precondition(s):** | Admin needs to have the information of the movie category to be managed (category name)  Admin needs to login application with admin account |
| **Post-Condition(s):** | Admin receives a success message from the notification window |
| **Basic Flow** | 1. Admin accesses BoxCine application with role admin. 2. Admin clicks the 'Type' button. 3. The system returns the movie category management form 4. Admin clicks 'Add' button to add a new movie category [A1] [A2]. 5. Admin enters information in the movie category management form including information (category name, image link) and press the ‘Add’ button to complete the task [E1] [A3]. 6. The system returns the category management form. |
| **Alternative Flow** | [A1] Admin clicks the ‘Edit’ button to edit a movie's category.  [A2] Admin clicks the ‘Delete’ button to delete a movie. Skip step 5, continue to step 6  [A3] Admin edits information on the movie category management form including information (category name, image link) and press the 'Chỉnh Sửa' button to complete the work. [E1] |
| **Exception Flow** | [E1] Admin not enter full information. The system displays the message ‘Please enter data' and lets the admin re-enter the data. |

Table 2.6. : Use case manage movie category

**Sequence diagram**

Figure 2.6. : Sequence diagram manage movie categogy

1. Use Case Manage Food & Beverages

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-2.2 |
| **Use Case Name** | Manage Food & Beverages |
| **Description** | This use case allows admin to add, remove, edit food & drinks. For users to interact and use the service. |
| **Actor(s)** | Admin |
| **Priority** | Must Have |
| **Precondition(s):** | Admin needs to have information about food & drinks to be managed (Name, price, quantity)  Admin needs to login application with admin account |
| **Post-Condition(s):** | Admin receives success message from notification window |
| **Basic Flow** | 1. Admin accesses BoxCine application with role admin. 2. Admin clicks the button ‘Food/Beverage’. 3. The system returns the food & beverage management form. 4. Admin clicks ‘Add’ button to add a new food & drink [A1] [A2]. 5. Admin enters information into the food & drink form including information (Name, price, quantity) and presses the ‘Add’ button to complete the job [E1] [A3]. 6. The system returns the admin to the food & beverage management form. |
| **Alternative Flow** | [A1] Admin clicks the ‘Edit’ button to edit a food & drink.  [A2] Admin clicks ‘Delete’ button to delete a food & drink. Skip step 5, continue to step 6  [A3] Admin edit the information on the food & beverage management form including the information (Name, price, quantity) and press the ‘Edit’ button to complete the job. [E1] |
| **Exception Flow** | [E1] Admin entered incorrect input information. The system displays the message 'Incorrect input information' and lets the admin re-enter the data. |

Table 2.6. : Use case manage food & beverages

**Sequence diagram**

****

Figure 2.6. : Sequence diagram manage food & beverages

1. Use Case Manage Movie

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-2.3 |
| **Use Case Name** | Manage Movie |
| **Description** | This use case allows admin to add, edit, delete movies or movie information. For users to interact and use the service |
| **Actor(s)** | Admin |
| **Priority** | Must Have |
| **Precondition(s):** | Admin needs to have the information of the movie to be managed (movie category, movie name, description, premiere date, image link, banner image link, trailer video link)  Admin needs to login application with admin account |
| **Post-Condition(s):** | Admin receives a success message from the notification window |
| **Basic Flow** | 1. Admin accesses BoxCineAdmin application. 2. Admin clicks the 'Movie' button. 3. The system returns the movie management form 4. Admin clicks 'Add' button to add a new movie [A1] [A2]. 5. Admin enters information in the movie management form including information (movie category, movie name, description, premiere date, image link, banner image link, trailer video link) and press the 'Thêm' button to complete the job [E1] [A3]. 6. The system returns the admin to the home page. |
| **Alternative Flow** | [A1] Admin clicks the ‘Edit’ button to edit a movie's information.  [A2] Admin clicks the ‘Delete’ button to delete a movie. Skip step 5, continue to step 6  [A3] Admin edits information on the movie management form including information (movie category, movie name, description, premiere date, image link, banner image link, trailer video link) and press the ‘Edit’ button to complete the work. [E1] |
| **Exception Flow** | [E1] Admin entered incorrect input information. The system displays the message 'Incorrect input information' and lets the admin re-enter the data. |

Table 2.6. : Use case manage movie

**Sequence diagram**

****

Figure 2.6. : Sequence diagram maange movie

1. Use Case View Revenue

**Use case description**

|  |  |
| --- | --- |
| **Use Case ID** | UC-2.4 |
| **Use Case Name** | View Revenue |
| **Description** | This use case allows the admin to calculate the theater's revenue in a certain period of time |
| **Actor(s)** | Admin |
| **Priority** | Must Have |
| **Precondition(s):** | Admin needs to login application with admin account |
| **Post-Condition(s):** | The system will return a page of results that the admin searched for |
| **Basic Flow** | 1. Admin accesses BoxCineAdmin application. 2. Admin selects 'Manage' button. 3. The system will display a fragment of ‘Manage’. 4. Admin clicks “Revenue” button to view revenue 5. The system will return the solution of revenue each movie. |
| **Alternative Flow** |  |
| **Exception Flow** |  |

Table 2.6. : Use case view revenue

**Sequence diagram**

****

Figure 2.6. : Sequence diagram view revenue

# Chapter 3: IMPLEMENTED DESIGNS PATTERN

1. **Factory Method**
2. Definition

**Factory Method** is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.



1. Reason to use

* Simplify the object creation process on the client side.
* Desire to create an object where the details of its creation process can be intricate and complex.
* The usage of the created object does not need to differentiate between objects of the same type.

1. Applying design pattern

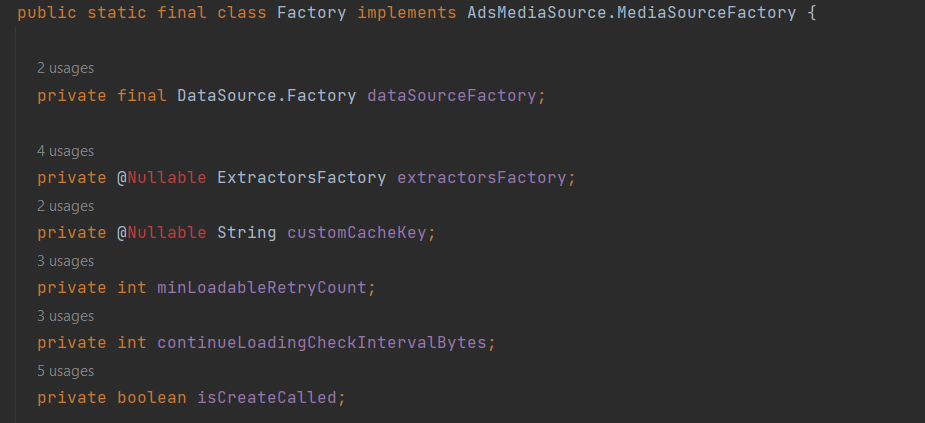


Figure 3.1. : Define the factory class for creating MediaSource objects

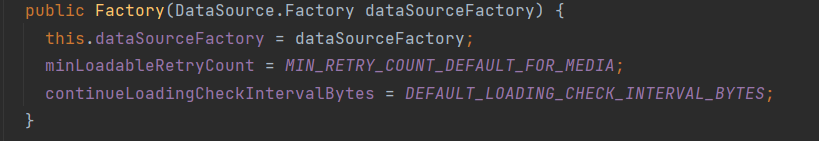
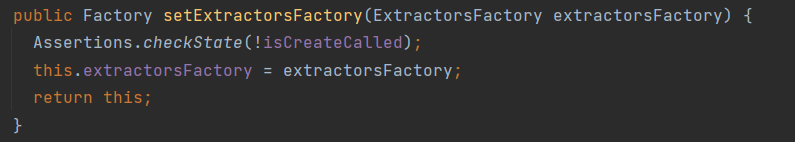
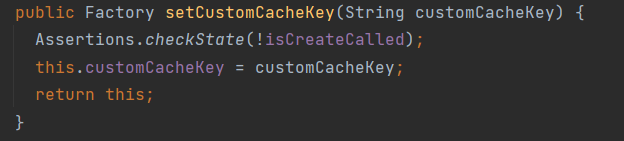


Figure 3.1. 4: setCustomCacheKey() method

Figure 3.1. 3: The setExtractorsFactory() method

Figure 3.1. 2: Constructor of factory

Figure 3.1. 5: The setContinueLoadingCheckIntervalBytes() method

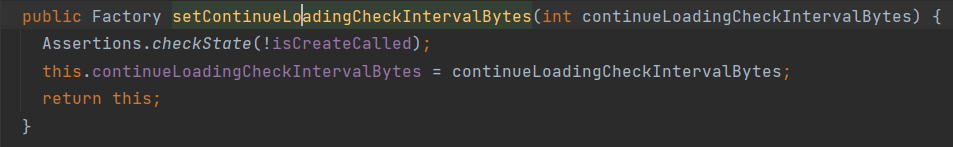


Figure 3.1. 6: creates a new instance of ExtractorMediaSource using its Factory class.

# Chapter4: INSTALLATION AND EXPERIMEMTAL RESULT

1. **Results**

After a period of project implementation, the program has been completed and achieved the following outcomes:

* Understanding the management and bussiness process of the booking cinema ticket mobile application.
* Building a mobile application for managing a cinema business.
* Researching and understanding the technologies used:Android Studio and Firebase.
* The program's interface is user-friendly and easy to use.

1. **Limitations**

* Some features are not yet completed.
* We not finish applying design pattern because of lack of knowledge and human resource.

# Chapter 5: CONCLUSION

Design patterns help to address common issues in software design and provide proven solutions for building effective software systems. In this project, design patterns were applied to improve the flexibility, reusability, and scalability of the website. Through this, we have gained a better understanding of design patterns and how to apply them in projects.

However, during the design process, applying design patterns also encountered some difficulties. Understanding and applying the right design patterns can be challenging and time-consuming. Furthermore, merging design patterns can lead to complexities in source code.

With the emergence of ChatGPT, artificial intelligence (AI) has become an essential part of the technology industry. Therefore, we have plans to integrate AI into our mobile application in the future, for instance implementing chatbots to support users in many useful featers such as movie search, movie recommendations, information provision, scheduling and ticket booking, technical support, and mood-based movie suggestions. This helps enhance user interaction and convenience in using the movie streaming application.

In conclusion, applying design patterns has helped improve the flexibility, reusability, and maintainability of the booking online ticket mobile application. However, applying design patterns can encounter some difficulties during the design process. Applying design patterns must be carefully and correctly implemented to ensure the consistency and effectiveness of the system.

# WORK ASSIGNMENT TABLE

|  |  |  |
| --- | --- | --- |
| **Student’s name** | **Completion Rate** | **Taskwork** |
| Phan Tấn Cường | 100% | * Log in * Register * View Information Movie * Book Ticket * Order Food & Beverage |
| Nguyễn Thanh Minh Triết | 100% | * Log Out * Recover password * Change password * Payment * View Booked Movie History |
| Phan Tấn Cường  Nguyễn Thanh Minh Triết | 100% | * Manage Movie * Manage Movie Category * View Booking History * Manage Food & Beverage * View Revenue |

# REFERENCES

[Documentation for android studio](https://developer.android.com/studio)

[Video instruction Firebase for Android Studio](https://youtu.be/HYzw8LFvmw4)

[Featers for design pattern](https://refactoring.guru/design-patterns)