

TRƯỜNG ĐẠI HỌC THỦY LỢI  
KHOA CÔNG NGHỆ THÔNG TIN



**BÁO CÁO BÀI TẬP LỚN MÔN HỌC  
PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ  
DI ĐỘNG**

**Đề tài: Xây dựng ứng dụng quản lý đặt vé xem  
phim**

Mã Sinh Viên	Họ và Tên	Ngày Sinh	Lớp
2151170549	Đặng Quốc Hiệp	26/11/2003	63KTPM1
2151173766	Phí Văn Đức	18/11/2003	63KTPM1
2151173757	Lê Đình Dũng	30/10/2003	63KTPM1
2151173825	Cù Tiến Thịnh	06/09/2003	63KTPM1

**Giảng viên phụ trách môn học: ThS. Kiều Tuấn Dũng**

*Hà Nội, ngày 2 tháng 11 năm 2024*

## PHIẾU CHẤM ĐIỂM

Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
			Bảng số	Bảng chữ
2151170549	Đặng Quốc Hiệp	26/11/2003		
2151173766	Phí Văn Đức	18/11/2003		
2151173757	Lê Đình Dũng	30/10/2003		
2151173825	Cù Tiến Thịnh	06/09/2003		

**CÁN BỘ CHẤM THI 1**

**CÁN BỘ CHẤM THI 2**

## MỤC LỤC

LỜI CAM KẾT .....	5
LỜI CẢM ƠN .....	6
MỞ ĐẦU.....	7
PHÂN CÔNG CÔNG VIỆC .....	8
CHƯƠNG 1: GIỚI THIỆU – MÔ TẢ BÀI TOÁN .....	10
1.1. Giới thiệu .....	10
1.2. Mô tả chức năng .....	10
1.2.1. Ứng dụng FastCinema dành cho quản trị viên: .....	10
1.2.2. Ứng dụng FastCinema dành cho người dùng: .....	10
1.2.3. Yêu cầu phi chức năng: .....	11
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG .....	12
2.1. Phân tích yêu cầu .....	12
2.1.1. Xác Định Bên Liên Quan .....	12
2.1.2. Phương Pháp Thu Thập Yêu Cầu .....	12
2.1.3. Phân Tích Yêu Cầu .....	13
2.2. Thiết kế hệ thống .....	18
2.3. Triển khai.....	22
2.4. Vận hành và bảo trì.....	23
CHƯƠNG 3. KẾT QUẢ THỰC HIỆN .....	24
3.1. Công nghệ đã sử dụng .....	24
3.2. Tiến độ thực hiện.....	24
3.3. Hình ảnh sản phẩm .....	27
3.3.1. Đặng Quốc Hiệp .....	27
3.3.2. Phí Văn Đức.....	38
3.3.3. Lê Đình Dũng .....	45
3.3.4. Cù Tiến Thịnh .....	51
KẾT LUẬN .....	58
1. Kết quả đạt được .....	58
2. Hướng phát triển .....	58
DANH MỤC TÀI LIỆU THAM KHẢO .....	59
PHỤ LỤC.....	60

## MỤC LỤC HÌNH ẢNH

Hình 1: Biểu đồ lớp .....	21
Hình 2: Giao diện đăng nhập .....	27
Hình 3: Giao diện đăng ký tài khoản .....	28
Hình 4: Giao diện quên mật khẩu .....	29
Hình 5: Giao diện hiển thị danh sách mã giảm giá .....	30
Hình 6: Giao diện thêm mã giảm giá .....	31
Hình 7: Giao diện sửa mã giảm giá.....	32
Hình 8: Giao diện xác nhận xóa mã giảm giá.....	33
Hình 9: Giao diện hiển thị phim đang chiếu và sắp chiếu .....	34
Hình 10: Giao diện chi tiết phim đang chiếu .....	35
Hình 11: Giao diện chi tiết phim sắp chiếu.....	36
Hình 12: Giao diện hiển thị ngày chiếu và xuất chiếu .....	37
Hình 13: Giao diện thêm vé .....	38
Hình 14: Giao diện danh sách các vé.....	39
Hình 15: Giao diện hiển thị chi tiết vé và xóa vé.....	40
Hình 16: Giao diện danh sách các ghế ngồi.....	41
Hình 17: Giao diện chọn ghế ngồi .....	42
Hình 18: Giao diện danh sách ghế bao gồm ghế đã đặt.....	43
Hình 19: Giao diện danh sách các vé đã mua .....	44
Hình 20: Giao diện Showtime List .....	45
Hình 21: Giao diện Add Showtime.....	46
Hình 22: Giao diện Edit Showtime .....	47
Hình 23: Giao diện Payment.....	48
Hình 24: Giao diện Voucher .....	49
Hình 25: Giao diện Payment (Google Pay) .....	50
Hình 26: Giao diện danh sách tài khoản .....	51
Hình 27: Giao diện chỉnh sửa tài khoản (Chỉ admin) .....	52
Hình 28: Giao diện MenuProfile.....	53
Hình 29: Giao diện sửa tài khoản user và chọn các chức năng đổi mật khẩu và xóa tài khoản .....	54
Hình 30: Giao diện đổi mật khẩu .....	55
Hình 31: Giao diện xác nhận xóa tài khoản.....	56
Hình 32: Giao diện nhập mật khẩu để xóa tài khoản.....	57

## **LỜI CAM KẾT**

Nhóm chúng em xin cam đoan tất cả những thông tin, dữ liệu và kiến thức được trình bày trong bài tập lớn này là hoàn toàn trung thực và đáng tin cậy. Nhóm chúng em đã tự thực hiện công việc nghiên cứu, thu thập dữ liệu, phân tích dữ liệu. Tất cả các nguồn tài liệu tham khảo liên quan đều được chúng em trích dẫn và ghi lại nguồn tài liệu tham khảo đầy đủ và chính xác.

Nhóm chúng em cũng cam kết rằng đã tuân thủ đầy đủ các quy định và hướng dẫn của giảng viên và trường đại học đề ra trong quá trình thực hiện bài tập lớn này.

## LỜI CẢM ƠN

Sau bốn năm học tập và nghiên cứu tại Khoa Công Nghệ Thông Tin, Trường Đại Học Thủy Lợi. Chúng em đã được trải nghiệm trong môi trường đào tạo chất lượng và chuyên nghiệp, chúng em đã được sự chỉ dạy nhiệt tình cùng hướng dẫn tận tâm của các thầy cô trong Khoa và các thầy cô trong Trường.

Trước hết, chúng em xin được bày tỏ lòng biết ơn tới các thầy cô giáo trong Khoa Công Nghệ Thông Tin nói riêng và các thầy cô **Trường Đại học Thủy Lợi** nói chung đã trang bị cho chúng em những kiến thức quý báu làm hành trang giúp sức chúng em trong những năm nghiên cứu và học tập sắp tới.

Chúng em xin được gửi lời cảm ơn chân thành đến **ThS. Kiều Tuấn Dũng** đã hỗ trợ và giúp đỡ chúng em, trong quá trình thực hiện bài tập lớn này. Nhờ sự cung cấp những kiến thức và kỹ năng quý báu để thực hiện bài tập lớn một cách hiệu quả. Những lời khuyên, định hướng và phản hồi của cô đã giúp chúng em giải quyết các khó khăn và thắc mắc trong quá trình thực hiện bài tập.

Trong quá trình làm bài tập này, mặc dù cá nhân chúng em đã rất nỗ lực nhưng đồ án cũng như sản phẩm không thể tránh khỏi những thiếu sót do còn hạn chế về kiến thức và kinh nghiệm thực tế cũng như thời gian. Chúng em rất mong nhận được sự cảm thông và những góp ý từ Thầy Cô và các bạn để đề tài được hoàn thiện và phát triển hơn.

Nhóm chúng em xin chân thành cảm ơn.

## MỞ ĐẦU

Trong thời đại công nghệ số bùng nổ, thiết bị di động đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. Việc phát triển ứng dụng cho các thiết bị di động không chỉ đáp ứng nhu cầu giải trí, giao tiếp mà còn mở ra nhiều cơ hội kinh doanh và sáng tạo trong lĩnh vực công nghệ thông tin.

Bài báo cáo này được thực hiện trong khuôn khổ môn học "**Phát triển ứng dụng cho các thiết bị di động**", nhằm mục đích áp dụng những kiến thức và kỹ năng đã học vào thực tế. Chúng em đã chọn dự án "**Ứng dụng FastCinema**" để phát triển, với mong muốn tạo ra một nền tảng đặt vé xem phim trực tuyến tiện lợi, nhanh chóng và thân thiện với người dùng.

Thông qua dự án này, chúng em không chỉ củng cố kiến thức về lập trình Android mà còn hiểu sâu hơn về quy trình phân tích yêu cầu, thiết kế giao diện, xử lý dữ liệu và tối ưu hóa trải nghiệm người dùng. Quá trình phát triển ứng dụng đã giúp chúng em rèn luyện kỹ năng làm việc nhóm, giải quyết vấn đề và quản lý thời gian hiệu quả.

Chúng em xin gửi lời cảm ơn chân thành đến giảng viên đã tận tình hướng dẫn, cung cấp kiến thức quý báu và hỗ trợ chúng em trong suốt quá trình học tập và thực hiện dự án. Đồng thời, chúng em cũng biết ơn sự đóng góp và hợp tác của tất cả các thành viên trong nhóm, những người đã nỗ lực không ngừng để hoàn thành bài tập lớn này.

## PHÂN CÔNG CÔNG VIỆC

Thành viên	Công việc
Đặng Quốc Hiệp (Nhóm trưởng)	<p>Thực hiện thiết kế giao diện cho các chức năng:</p> <ul style="list-style-type: none"> <li>● Choose Date and Show Screen</li> <li>● Choose Seat</li> <li>● Admin ShowScreen management</li> </ul> <p>Thực hiện code các chức năng:</p> <ul style="list-style-type: none"> <li>● Admin <ul style="list-style-type: none"> <li>○ Đăng nhập</li> <li>○ Đăng ký</li> <li>○ Forgot password</li> <li>○ Hiện thị thêm sửa xóa Voucher</li> </ul> </li> <li>● App FastCinema <ul style="list-style-type: none"> <li>○ Đăng nhập</li> <li>○ Đăng ký</li> <li>○ Forgot password</li> <li>○ Call Api hiện thị phim, chi tiết phim</li> <li>○ Hiện thị ngày, chọn ngày</li> <li>○ Xem được movie's trailer</li> </ul> </li> </ul>
Phí Văn Đức	<p>Thực hiện thiết kế giao diện cho các chức năng:</p> <ul style="list-style-type: none"> <li>● Sign In</li> <li>● Create Account</li> <li>● Forgot Password</li> <li>● Home Show Movie</li> <li>● Movie Detail</li> <li>● Admin Account management</li> <li>● Admin Voucher management</li> </ul> <p>Thực hiện code các chức năng:</p> <ul style="list-style-type: none"> <li>● Admin <ul style="list-style-type: none"> <li>○ Hiện thị thêm sửa xóa vé</li> </ul> </li> <li>● App FastCinema <ul style="list-style-type: none"> <li>○ Hiện thị ghế, chọn ghế (multiple choice)</li> <li>○ Hiện thị vé đã đặt</li> </ul> </li> </ul>
Lê Đình Dũng	<p>Thực hiện thiết kế giao diện cho các chức năng:</p> <ul style="list-style-type: none"> <li>● Canceled &amp; Detail Ticket</li> <li>● Ticket History</li> <li>● Personal Information</li> <li>● Change Password</li> </ul>



	<ul style="list-style-type: none"> <li>● Admin Ticket Management</li> </ul> <p>Thực hiện code các chức năng:</p> <ul style="list-style-type: none"> <li>● Admin <ul style="list-style-type: none"> <li>○ Hiện thị thêm sửa xóa phòng chiếu</li> </ul> </li> <li>● App FastCinema <ul style="list-style-type: none"> <li>○ Hiện thị giá, vé phim (Tên, ngày mua, giá, phòng chiếu, ghế), phương thức thanh toán, chọn phương thức thanh toán</li> <li>○ Thanh toán, cho khách hàng phút để giữ vé và thanh toán</li> </ul> </li> </ul>
Cù Tiến Thịnh	<p>Thực hiện thiết kế giao diện cho các chức năng:</p> <ul style="list-style-type: none"> <li>● Voucher</li> <li>● Payment</li> <li>● Personal information</li> <li>● Delete account</li> <li>● Profile menu activity</li> </ul> <p>Thực hiện code các chức năng:</p> <ul style="list-style-type: none"> <li>● Admin <ul style="list-style-type: none"> <li>○ Sửa Account</li> <li>○ Hiện thị danh sách tài khoản</li> </ul> </li> <li>● App FastCinema <ul style="list-style-type: none"> <li>○ Hiện thị activity profile bao gồm My ticket và profile</li> <li>○ Sửa đổi thông tin cá nhân, thay đổi mật khẩu, xóa tài khoản, đăng xuất</li> </ul> </li> </ul>

# CHƯƠNG 1: GIỚI THIỆU – MÔ TẢ BÀI TOÁN

## 1.1. Giới thiệu

Ứng dụng FastCinema là ứng dụng đặt vé xem phim online. Ứng dụng được chia làm 2 ứng dụng riêng lẻ. Một ứng dụng dành cho người dùng và ứng dụng còn lại dành cho admin.

## 1.2. Mô tả chức năng

Ứng dụng FastCinema sẽ có các chức năng sau:

### 1.2.1. *Ứng dụng FastCinema dành cho quản trị viên:*

- Quản lý (Thêm sửa xóa) tài khoản người dùng và admin
- Quản lý (Thêm sửa xóa) voucher
- Quản lý (Thêm sửa xóa) vé xem phim
- Quản lý (Thêm sửa xóa) phòng chiếu

### 1.2.2. *Ứng dụng FastCinema dành cho người dùng:*

- Hiện thị danh sách các phim hiện đang chiếu và sắp chiếu.
- Hiện thị thông tin chi tiết của phim khi chọn vào một phim nhất định.
- Trước khi đặt, quản lý vé xem phim. Người dùng sẽ phải đăng nhập hoặc nếu chưa có tài khoản có thể tạo một tài khoản mới (Lưu ý nếu đã có 1 tài khoản được đăng kí bởi một email xác định, người dùng không thể dùng email đó đăng kí tài khoản mới). Nếu quên mật khẩu, người dùng có thể đổi lại mật khẩu dựa trên email đăng kí lên tài khoản đó.
- Đăng nhập, người dùng cần nhập 2 trường “Email, Password”.
- Tạo tài khoản mới, người dùng cần nhập các trường “Phone number, Full name, Email, Password” (Xác thực dữ liệu đầu vào sẽ dựa vào quy tắc trường nhóm).
- Đổi lại mật khẩu, người dùng cần nhập Email, người dùng sẽ nhận được email khôi phục mật khẩu và thực hiện nhập mật khẩu mới
- Đặt vé xem phim, người dùng cần chọn ngày xem – giờ xem và phòng chiếu (Qua mỗi ngày, số giờ chiếu cho từng phim trong từng phòng sẽ khác nhau). Tiếp đến, người dùng chọn 1 hoặc nhiều chỗ ngồi khác nhau, mỗi 1 chỗ ngồi sẽ tương ứng với 1 vé và tùy loại ghế sẽ có giá khác nhau.

Cuối cùng, người dùng chọn 1 hoặc nhiều combo bóng và nước uống và áp mã giảm giá (Nếu hợp lệ điều kiện được áp dụng).

- Thanh toán vé xem phim bằng Google Pay, sau khi thanh toán thành công sẽ chuyển sang giao diện lịch sử mua vé và thông báo cho khách hàng
- Quản lý tài khoản, người dùng có thể xem chi tiết thông tin tài khoản, thay đổi thông tin tài khoản và xóa tài khoản đang dùng.
- Quản lý vé xem phim, được chia làm 2 mục: Quản lý vé đã mua, quản lý vé đã hủy. Phần quản lý vé đã mua sẽ hiển thị ra tất cả các vé chưa sử dụng và còn thời hạn sử dụng. Người dùng ấn vào 1 trong số các vé sẽ hiển thị ra chi tiết thông tin của phim, giờ chiếu, phòng chiếu, số ghế ngồi, giá vé. Ngoài ra còn có thể hủy vé và được hoàn tiền (Tùy thuộc vào chính sách của rạp).
- Hiển thị tất cả các chương trình giảm giá hiện đang có hiệu lực
  - Giảm giá theo phần trăm

### ***1.2.3. Yêu cầu phi chức năng:***

- **Dễ Sử Dụng:**
  - Giao diện thân thiện và dễ sử dụng cho cả hai ứng dụng.
  - Thiết kế đáp ứng trên nhiều kích thước màn hình và thiết bị.
- **Hiệu Năng:**
  - Thời gian tải nhanh cho danh sách phim và kết quả tìm kiếm.
  - Xử lý hiệu quả trong quá trình chọn ghế và thanh toán.
- **Bảo Mật:**
  - Xử lý an toàn dữ liệu người dùng, đặc biệt là mật khẩu và thông tin thanh toán.
  - Xác thực email để ngăn chặn việc tạo nhiều tài khoản.
- **Độ Tin Cậy:**
  - Khả năng hoạt động ổn định với thời gian ngừng hoạt động tối thiểu.
  - Cập nhật chính xác tình trạng vé và chỗ ngồi để tránh đặt trùng.

## CHƯƠNG 2: PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG

### 2.1. Phân tích yêu cầu

#### 2.1.1. Xác Định Bên Liên Quan

- **Người Dùng Cuối:**
  - Khách hàng sẽ sử dụng ứng dụng để đặt vé.
- **Quản Trị Viên:**
  - Nhân viên chịu trách nhiệm quản lý phim, vé, voucher và tài khoản người dùng.

#### 2.1.2. Phương Pháp Thu Thập Yêu Cầu

##### a) Phỏng Vấn:

- Tiến hành phỏng vấn với người dùng tiềm năng để hiểu nhu cầu và sở thích của họ.
- Thảo luận với nhân viên admin để xác định yêu cầu và thách thức hậu trường.

##### b) Khảo Sát:

- Phân phối khảo sát để thu thập dữ liệu định lượng về kỳ vọng của người dùng.
- Thu thập phản hồi về các tính năng và khía cạnh sử dụng mong muốn.

##### c) Quan Sát:

- Phân tích cách người dùng tương tác với các hệ thống đặt vé hiện có.
- Quan sát quy trình làm việc của admin trong việc quản lý hoạt động rạp.

##### d) Hội Thảo Và Nhóm Tập Trung:

- Tổ chức các buổi họp với các bên liên quan để thông qua các tính năng và giải pháp.
- Khuyến khích sự hợp tác giữa người dùng và nhà phát triển để có cái nhìn sâu sắc hơn.

##### e) Phân Tích Tài Liệu:

- Xem xét tài liệu hiện có từ các hệ thống hiện tại.
- Phân tích ứng dụng của đối thủ để xác định thực tiễn tốt nhất và khoảng trống.

##### f) Trường Hợp Sử Dụng Và Kịch Bản:

- Phát triển sơ đồ use case để hình dung tương tác của người dùng.
- Tạo câu chuyện người dùng để nắm bắt các chức năng cụ thể từ góc nhìn của người dùng.

### 2.1.3. Phân Tích Yêu Cầu

#### a) Ứng Dụng Dành Cho Người Dùng

- **Đăng ký tài khoản mới:**

- Người dùng cần nhập các thông tin sau:
  - **Số điện thoại** (bắt buộc)
  - **Họ tên** (bắt buộc)
  - **Email** (bắt buộc, phải là email hợp lệ và chưa được đăng ký trước đó)
  - **Mật khẩu** (bắt buộc, xác thực dữ liệu đầu vào dựa vào quy tắc trường nhóm)
- Hệ thống kiểm tra và xác thực thông tin trước khi tạo tài khoản.
- Nếu email đã được sử dụng, thông báo cho người dùng và yêu cầu sử dụng email khác.

- **Đăng nhập tài khoản:**

- Người dùng nhập:
  - **Email** (bắt buộc)
  - **Mật khẩu** (bắt buộc)
- Hệ thống xác thực thông tin và cho phép truy cập nếu hợp lệ.

- **Khôi phục mật khẩu:**

- Người dùng nhập **Email** đã đăng ký.
- Hệ thống gửi email khôi phục mật khẩu đến địa chỉ email của người dùng.
- Người dùng theo hướng dẫn trong email để đặt lại mật khẩu mới.

- **Hiển thị danh sách phim:**

- Hiển thị danh sách **phim đang chiếu** và **phim sắp chiếu**.
- Mỗi phim hiển thị thông tin cơ bản:
  - **Tên phim**
  - **Hình ảnh poster**
  - **Thể loại**

- **Ngày phát hành**
- **Xem thông tin chi tiết phim:**
  - Khi người dùng chọn một phim cụ thể, hiển thị thông tin chi tiết:
    - **Tên phim**
    - **Thời lượng**
    - **Đạo diễn**
    - **Diễn viên**
    - **Mô tả nội dung**
    - **Lịch chiếu**
- **Đặt vé xem phim:**
  - Người dùng chọn:
    - **Ngày xem** (bắt buộc)
    - **Giờ xem** (bắt buộc, giờ chiếu khác nhau cho từng ngày)
    - **Phòng chiếu** (bắt buộc)
  - Chọn **ghế ngồi**:
    - Hiển thị sơ đồ chỗ ngồi của phòng chiếu.
    - Người dùng có thể chọn một hoặc nhiều ghế.
    - Giá vé thay đổi tùy theo loại ghế (thường, VIP, đôi, v.v.).
  - Chọn **combo bỏng và nước uống** (tùy chọn):
    - Danh sách các combo có sẵn để người dùng lựa chọn.
  - Áp dụng **mã giảm giá** (tùy chọn):
    - Người dùng nhập mã voucher.
    - Hệ thống kiểm tra tính hợp lệ và điều kiện áp dụng của mã.
  - **Thanh toán**:
    - Thực hiện thanh toán qua Google Pay.
    - Sau khi thanh toán thành công, hiển thị xác nhận và gửi thông báo cho người dùng.
    - Chuyển đến giao diện **lịch sử mua vé**.
- **Quản lý tài khoản cá nhân:**
  - **Xem thông tin tài khoản**:
    - Hiển thị chi tiết thông tin cá nhân của người dùng.
  - **Chỉnh sửa thông tin tài khoản**:

- Người dùng có thể cập nhật các thông tin như số điện thoại, họ tên, mật khẩu.
- **Xóa tài khoản:**
  - Người dùng có thể xóa tài khoản của mình (cần xác nhận trước khi xóa).
- **Quản lý vé xem phim:**
  - **Vé đã mua:**
    - Hiện thị danh sách tất cả các vé đã mua, còn thời hạn sử dụng và chưa sử dụng.
    - Khi chọn một vé, hiển thị chi tiết:
      - **Thông tin phim**
      - **Ngày giờ chiếu**
      - **Phòng chiếu**
      - **Số ghế ngồi**
      - **Giá vé**
    - **Hủy vé:**
      - Người dùng có thể hủy vé và yêu cầu hoàn tiền (tùy thuộc vào chính sách của rạp).
      - Hệ thống thông báo về số tiền được hoàn và thời gian xử lý.
  - **Vé đã hủy:**
    - Hiện thị danh sách các vé đã hủy cùng với thông tin liên quan.
- **Hiện thị chương trình giảm giá và voucher:**
  - Hiện thị danh sách các voucher và chương trình giảm giá hiện có hiệu lực.
  - Thông tin chi tiết bao gồm:
    - **Mã giảm giá**
    - **Mức giảm giá** (theo phần trăm)
    - **Điều kiện áp dụng**
    - **Thời gian hiệu lực**

## **b) Ứng Dụng Dành Cho Quản Trị Viên (Admin)**

- **Quản lý tài khoản người dùng và admin:**
  - **Thêm tài khoản mới:**
    - Nhập thông tin cần thiết để tạo tài khoản người dùng hoặc admin.
  - **Chỉnh sửa tài khoản:**
    - Cập nhật thông tin của tài khoản người dùng hoặc admin.
  - **Xóa tài khoản:**
    - Xóa tài khoản không còn sử dụng hoặc vi phạm quy định.
- **Quản lý voucher:**
  - **Thêm voucher mới:**
    - Nhập thông tin:
      - **Mã voucher**
      - **Mức giảm giá** (phần trăm)
      - **Điều kiện áp dụng**
      - **Thời gian hiệu lực**
  - **Chỉnh sửa voucher:**
    - Cập nhật thông tin của voucher hiện có.
  - **Xóa voucher:**
    - Xóa voucher hết hạn hoặc không còn sử dụng.
- **Quản lý vé xem phim:**
  - **Thêm lịch chiếu mới:**
    - Chọn phim, ngày chiếu, giờ chiếu, phòng chiếu.
    - Cấu hình sơ đồ ghế ngồi và giá vé cho từng loại ghế.
  - **Chỉnh sửa lịch chiếu:**
    - Thay đổi thông tin về thời gian, phòng chiếu, giá vé.
  - **Xóa lịch chiếu:**
    - Xóa lịch chiếu không còn áp dụng hoặc bị hủy.
- **Quản lý phòng chiếu:**
  - **Thêm phòng chiếu mới:**
    - Nhập thông tin:
      - **Tên phòng**



- Số lượng ghế
- Sơ đồ chỗ ngồi
- **Chỉnh sửa phòng chiếu:**
  - Cập nhật thông tin phòng chiếu như số ghế, sơ đồ ghế ngồi.
- **Xóa phòng chiếu:**
  - Xóa phòng chiếu không còn sử dụng hoặc đang bảo trì.

#### c) Yêu Cầu Phi Chức Năng Chung

- **Xác thực dữ liệu đầu vào:**
  - Kiểm tra tính hợp lệ của tất cả các trường nhập liệu.
  - Áp dụng quy tắc trường nhóm đối với mật khẩu và các thông tin quan trọng.
- **Bảo mật thông tin:**
  - Mã hóa mật khẩu và thông tin nhạy cảm.
  - Sử dụng kết nối an toàn khi truyền dữ liệu (HTTPS).
- **Giao diện người dùng thân thiện:**
  - Thiết kế giao diện trực quan, dễ sử dụng.
  - Hỗ trợ đa dạng thiết bị và kích thước màn hình.
- **Hiệu suất và tốc độ:**
  - Ứng dụng hoạt động mượt mà, phản hồi nhanh.
  - Tối ưu hóa quá trình tải dữ liệu và hình ảnh.
- **Thông báo và phản hồi:**
  - Cung cấp thông báo rõ ràng khi người dùng thực hiện hành động (ví dụ: đặt vé thành công, lỗi khi nhập liệu).
  - Hỗ trợ thông báo đẩy (push notification) cho các sự kiện quan trọng.

#### d) Lưu Ý Khác

- **Chính sách hoàn tiền và hủy vé:**
  - Hệ thống cần tuân thủ chính sách của rạp về việc hoàn tiền khi hủy vé.
  - Thông báo rõ ràng cho người dùng về các điều kiện và thời gian hoàn tiền.
- **Tích hợp Google Pay:**

- Đảm bảo tích hợp an toàn và tuân thủ các yêu cầu kỹ thuật của Google Pay.
- Xử lý các trường hợp thanh toán thất bại hoặc bị gián đoạn để đảm bảo trải nghiệm liền mạch cho người dùng.
- **Quy tắc trường nhóm về xác thực dữ liệu:**
  - Áp dụng các quy tắc về độ mạnh mật khẩu (ví dụ: độ dài tối thiểu, bao gồm chữ hoa, chữ thường, số và ký tự đặc biệt).
  - Kiểm tra định dạng email hợp lệ.
  - Kiểm tra định dạng số điện thoại.

## 2.2. Thiết kế hệ thống

### Biểu đồ lớp cho mô hình miền

Mô hình miền của FastCinema tập trung vào các thực thể nghiệp vụ và mối quan hệ giữa chúng. Biểu đồ lớp mô tả các thực thể chính như:

- **User:** Đại diện cho tài khoản người dùng.
- **Movie:** Đại diện cho các bộ phim.
- **Ticket:** Thông tin về vé xem phim.
- **Voucher:** Các mã giảm giá.
- **ScreenRoom:** Thông tin về phòng chiếu và ghế ngồi.
- **Bill:** Thông tin về hóa đơn, lưu trữ chi tiết thanh toán của người dùng cho các giao dịch vé xem phim.
- **Seat:** Thông tin về chỗ ngồi trong phòng chiếu, bao gồm loại ghế và số ghế.
- **Genre:** Thể loại phim, giúp phân loại các bộ phim theo các chủ đề như hành động, lãng mạn, kinh dị, v.v.
- **TicketHistory:** Lịch sử giao dịch vé của người dùng, bao gồm các vé đã sử dụng và các giao dịch đã thực hiện.

Biểu đồ lớp cho mô hình miền chỉ chứa các thuộc tính và mối quan hệ nghiệp vụ chính, không bao gồm các lớp liên quan đến giao diện người dùng hay chi tiết kỹ thuật khác.

## Biểu đồ lớp cho MVVM

Trong mô hình MVVM, ứng dụng được chia thành ba phần chính: **Model**, **View**, và **ViewModel**. Cụ thể trong ứng dụng FastCinema, các lớp sẽ được tổ chức như sau:

### 1. **Model**:

- **Chứa các lớp dữ liệu và nghiệp vụ chính** như User, Movie, Ticket, ShowTime, Voucher, và ScreenRoom.
- **Model** sẽ chịu trách nhiệm quản lý và cung cấp dữ liệu cần thiết cho ViewModel, đồng thời xử lý logic nghiệp vụ.

### 2. **ViewModel**:

- **ViewModel** là lớp trung gian giữa **Model** và **View**. Nó lấy dữ liệu từ **Model** và xử lý dữ liệu theo nhu cầu của **View**. Các **ViewModel** có thể bao gồm các lớp có trong dự án như:
  - SignInViewModel, SignUpViewModel: Xử lý logic đăng nhập, đăng ký cho người dùng.
  - EditAccountViewModel, DiscountViewModel: Xử lý các yêu cầu chỉnh sửa tài khoản và quản lý khuyến mãi.
  - ShowTimeViewModel, TicketViewModel: Quản lý logic liên quan đến hiển thị thời gian chiếu phim và đặt vé.
  - CheckoutActivity, CheckoutSuccessActivity: Xử lý logic thanh toán, như việc chuẩn bị dữ liệu cho quá trình thanh toán qua Google Pay và cập nhật trạng thái sau khi thanh toán.
- Những **ViewModel** này sẽ chịu trách nhiệm lấy dữ liệu từ **Model** và cung cấp cho **View**, nhưng không biết trực tiếp về **View** nào đang sử dụng dữ liệu đó.

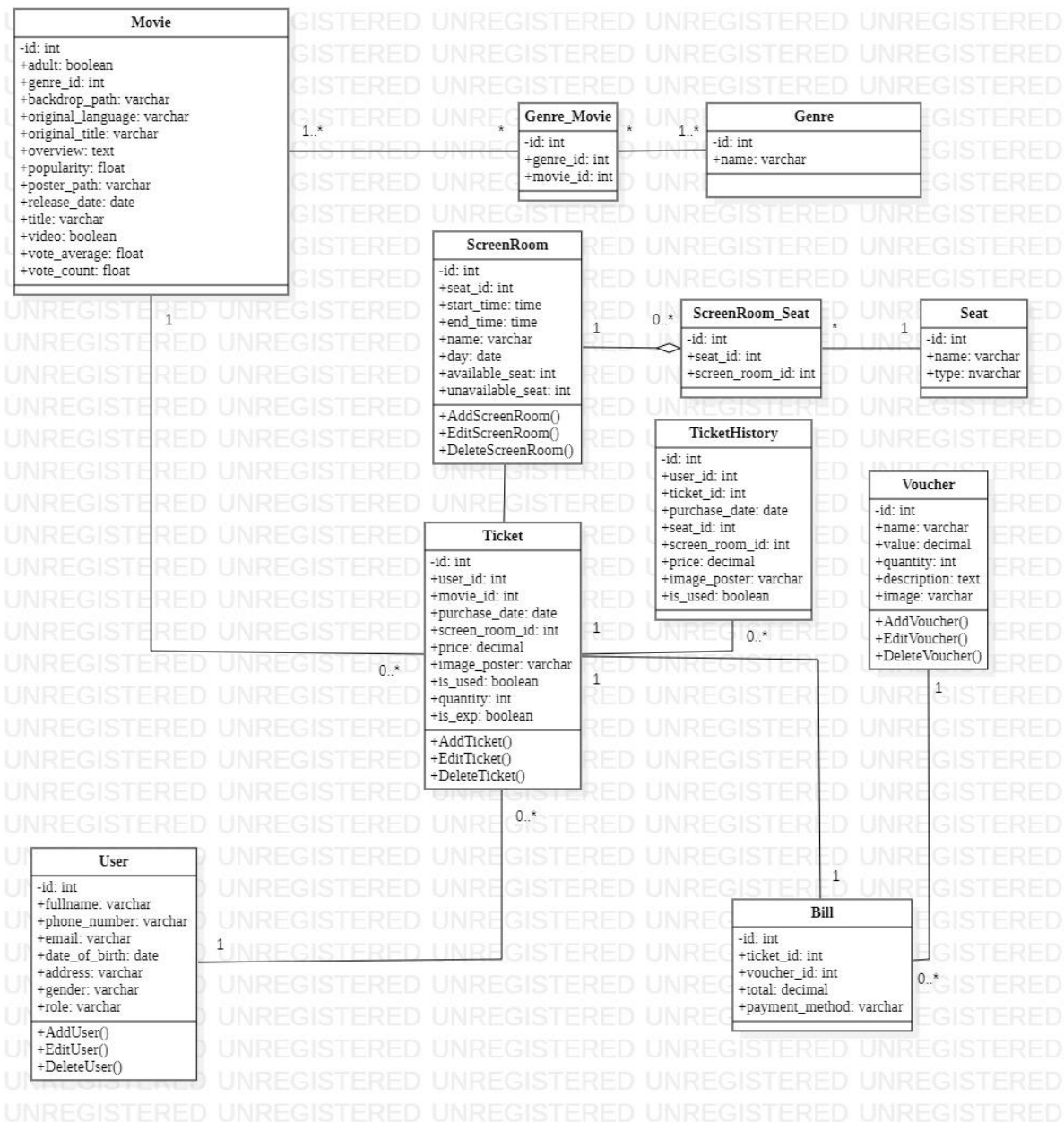
### 3. **View:**

- **View** bao gồm các lớp hiển thị giao diện người dùng có trong dự án, chẳng hạn như:
  - SignInActivity, SignUpActivity: Giao diện cho người dùng đăng nhập và đăng ký.
  - EditAccountActivity, AddDiscountActivity, DiscountsListFragment: Giao diện quản lý tài khoản và khuyến mãi.
  - MainActivity, TicketListFragment: Giao diện chính và hiển thị danh sách vé.
  - CheckoutActivity, CheckoutSuccessActivity: Giao diện cho người dùng thực hiện thanh toán và hiển thị khi thanh toán thành công.
- **View** nhận dữ liệu từ **ViewModel** và hiển thị cho người dùng, đồng thời nhận các thao tác người dùng và chuyển yêu cầu đến **ViewModel**.

### **Mối quan hệ giữa các lớp trong MVVM**

- **Model** quản lý và lưu trữ dữ liệu. **ViewModel** sử dụng **Model** để lấy hoặc lưu dữ liệu.
- **ViewModel** cung cấp dữ liệu cho **View** qua các thuộc tính observable, và **View** lắng nghe các thay đổi này để cập nhật giao diện.
- **View** nhận tương tác của người dùng và chuyển yêu cầu đến **ViewModel** để xử lý, giữ cho logic nghiệp vụ tách biệt khỏi **View**.

## Biểu đồ lớp cho dự án



Hình 1: Biểu đồ lớp

### 2.3. Triển khai

- **Viết code:** Sử dụng Java để cài đặt các lớp theo mô hình MVVM, kết nối cơ sở dữ liệu với Firebase, xử lý dữ liệu và hiển thị giao diện người dùng trên Android.
  - **Cài đặt FirebaseUtils:** Sử dụng thư viện Firebase SDK để quản lý dữ liệu và xác thực người dùng, kết nối với các dịch vụ Firebase như Firestore, Authentication, và Cloud Storage.
  - **Xử lý xác thực:** Cấu hình xác thực Firebase để ứng dụng có thể truy cập và thao tác trên cơ sở dữ liệu, bao gồm các thao tác đăng nhập, đăng ký, và xác thực người dùng. Firebase Authentication được sử dụng để đảm bảo rằng chỉ người dùng đã đăng ký mới có thể truy cập ứng dụng.
  - **Tích hợp API của bên thứ ba:** Sử dụng Retrofit để gọi các API từ bên thứ ba nhằm hiển thị thông tin phim mới nhất. Dữ liệu trả về được xử lý và hiển thị trong các thành phần giao diện như danh sách phim và chi tiết phim.
  - **Thay đổi code trong ViewModel:** Các lớp ViewModel được cài đặt để kết nối giữa tầng giao diện và tầng dữ liệu, đảm bảo rằng mọi thay đổi trong dữ liệu đều được cập nhật trên giao diện người dùng một cách nhất quán.
- **Kiểm thử:** Viết các test case để kiểm tra từng chức năng của ứng dụng nhằm đảm bảo rằng mọi tính năng đều hoạt động đúng theo yêu cầu. Các chức năng chính được kiểm thử bao gồm:
  - Đăng ký, đăng nhập, và khôi phục mật khẩu.
  - Tìm kiếm và xem chi tiết phim.
  - Đặt vé xem phim và thanh toán.
  - Quản lý tài khoản người dùng, bao gồm cập nhật thông tin cá nhân và thay đổi mật khẩu.
  - Quản lý vé và hủy vé.

## 2.4. Vận hành và bảo trì

### - Cài đặt và triển khai:

- Hướng dẫn người dùng cách cài đặt và chạy ứng dụng trên thiết bị Android bằng cách tải về file APK từ Google Play hoặc từ nguồn phân phối khác.
- Cấu hình Firebase: Cung cấp hướng dẫn cho người dùng hoặc quản trị viên về cách thiết lập Firebase nếu có thay đổi trong dự án, bao gồm việc tạo Firebase project, cấu hình Firestore, và thiết lập xác thực người dùng.
- Hướng dẫn người dùng về việc cấp quyền truy cập: Cần có hướng dẫn cho người dùng hoặc admin để cấp quyền truy cập ứng dụng tới Firebase và các API bên ngoài.
- Cài đặt thư viện: Khi tải ứng dụng từ nguồn ngoài, người dùng cần cài đặt các thư viện cần thiết, như Firebase SDK, bằng cách cập nhật Gradle (Android Studio sẽ tự động quản lý các thư viện khi triển khai trên Google Play).

### - Bảo trì:

- **Sửa lỗi phát sinh:** Khắc phục các lỗi phát sinh trong quá trình sử dụng, bao gồm các lỗi về giao diện, lỗi kết nối, và lỗi xác thực người dùng.
- **Cập nhật tính năng mới:** Khi có yêu cầu hoặc phản hồi từ người dùng, cập nhật thêm các tính năng mới như chương trình giảm giá, tính năng hoàn tiền tự động hoặc các hình thức thanh toán khác.
- **Cải thiện hiệu năng:** Tối ưu hóa ứng dụng để giảm thời gian tải và cải thiện hiệu suất khi xử lý dữ liệu lớn, đảm bảo ứng dụng chạy mượt mà ngay cả khi có nhiều người dùng đồng thời.
- **Bảo mật:** Thường xuyên kiểm tra và cập nhật các quy tắc bảo mật trên Firebase để đảm bảo an toàn dữ liệu người dùng, đặc biệt là dữ liệu nhạy cảm như thông tin cá nhân và tài khoản.

## CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

### 3.1. Công nghệ đã sử dụng

- **Ngôn ngữ lập trình:** Java
- **Công cụ phát triển:** Android Studio
- **Thư viện và công nghệ hỗ trợ:**
  - **Firestore:** để quản lý cơ sở dữ liệu người dùng, quản lý tài khoản và các dịch vụ xác thực.
  - **Retrofit:** để thực hiện các kết nối API cho việc truy xuất dữ liệu phim.
  - **ViewModel và LiveData:** hỗ trợ cho việc quản lý và cập nhật giao diện một cách hiệu quả và linh hoạt.
  - **Google Cloud Services:** hỗ trợ lưu trữ hình ảnh và dữ liệu cần thiết khác.
  - **Các thư viện hỗ trợ khác:** thư viện đọc/ghi file, xử lý dữ liệu JSON.

### 3.2. Tiến độ thực hiện

- **Link Github tới dự án:** [danghiep101/CSE441\\_PROJECT: Đặng Quốc Hiệp, Phí Văn Đức, Lê Đình Dũng, Cù Tiên Thịnh](https://github.com/danghiep101/CSE441_PROJECT)

**Hướng dẫn các bước đã thực hiện:**

#### **B1. Tạo dự án mới:**

- Mở Android Studio.
- Chọn "Start a new Android Studio project".
- Chọn loại dự án phù hợp và cấu hình tên dự án (ví dụ: "CinemaTicketApp").
- Cấu hình SDK phiên bản phù hợp (ví dụ: Android API 29 hoặc mới hơn).
- Nhấn "Finish" để tạo dự án.

#### **B2. Tạo các package và cấu trúc dự án:**

- Trong cửa sổ "Project", tạo các package để tổ chức code:



- data: chứa các lớp mô hình và xử lý dữ liệu.
  - model: chứa các lớp User, Movie, ShowTime, Ticket, Account, Discount, Seat.
  - repository: chứa MovieRepository, MovieRepositoryImpl.
  - resource: chứa MovieApi, RetrofitHelper để hỗ trợ gọi API.
- ui: chứa các package giao diện người dùng, gồm:
  - auth: quản lý đăng nhập, đăng ký và quên mật khẩu.
  - home: quản lý giao diện chính của người dùng.
  - profileoverlay: quản lý các chức năng liên quan đến tài khoản như thay đổi mật khẩu, xóa tài khoản.
  - ticketmanagement: quản lý các chức năng đặt vé.
  - showtimemanagement: quản lý các suất chiếu và lịch chiếu.
- utils: chứa các lớp hỗ trợ như FirebaseUtils.

### **B3. Cài đặt thư viện Firebase và Retrofit:**

- Mở file build.gradle của module và thêm các dependency cho Firebase, Retrofit, ViewModel, LiveData.
- Đồng bộ lại dự án để tải về và cài đặt các thư viện.

### **B4. Viết code cho các lớp chính trong dự án:**

- **User và Account:** cài đặt các thuộc tính và phương thức cần thiết để quản lý thông tin người dùng.
- **Movie, ShowTime, Ticket:** chứa thông tin chi tiết về phim, suất chiếu và vé.
- **FirebaseUtils:** cung cấp các phương thức kết nối và xử lý dữ liệu với Firebase.
- **AuthViewModel, HomeViewModel:** chứa logic điều khiển cho các thành phần giao diện người dùng.

### **B5. Cài đặt chức năng xác thực với Firebase:**

- Tích hợp các phương thức xác thực của Firebase như đăng nhập, đăng ký, và đặt lại mật khẩu.
- Cấu hình Google Cloud để sử dụng các API của Firebase, như Firestore và Firebase Authentication.

### **B6. Phát triển giao diện người dùng:**

- Sử dụng XML để thiết kế giao diện cho các màn hình đăng nhập, đăng ký, xem phim, đặt vé, quản lý tài khoản.
- Sử dụng RecyclerView và các adapter cho danh sách phim, suất chiếu, vé và quản lý voucher.
- Sử dụng ViewModel và LiveData để đồng bộ dữ liệu giữa các thành phần giao diện.

### **B7. Kiểm thử và chạy ứng dụng:**

- Chạy ứng dụng từ Android Studio để kiểm tra các chức năng.
- Kiểm thử các tính năng chính:
  - Đăng nhập và đăng ký tài khoản.
  - Tìm kiếm phim và xem chi tiết.
  - Đặt vé và quản lý vé.
  - Chỉnh sửa thông tin tài khoản và xóa tài khoản.
- Sửa lỗi và tối ưu hiệu suất để đảm bảo ứng dụng hoạt động mượt mà.

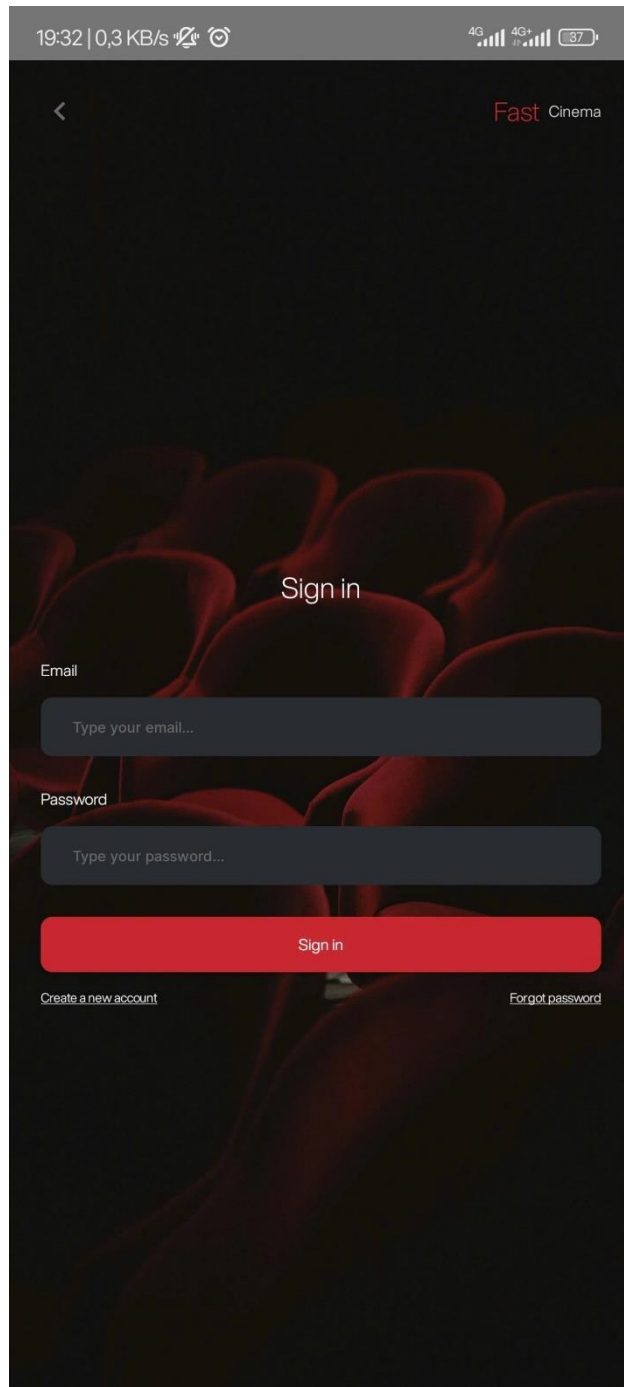
### **B8. Triển khai (tùy chọn):**

- Nếu muốn chia sẻ ứng dụng, sử dụng chức năng "Build Bundle(s) / APK(s)" để tạo file .apk.
- Chia sẻ file .apk cho người dùng để cài đặt và sử dụng trên thiết bị Android.

### 3.3. Hình ảnh sản phẩm

#### 3.3.1. Đặng Quốc Hiệp

##### a) Admin



Hình 2: Giao diện đăng nhập

19:32 | 66 KB/s | 5G | 100%

Fast Cinema

### Create account


Phone number

Email

Full name

Password

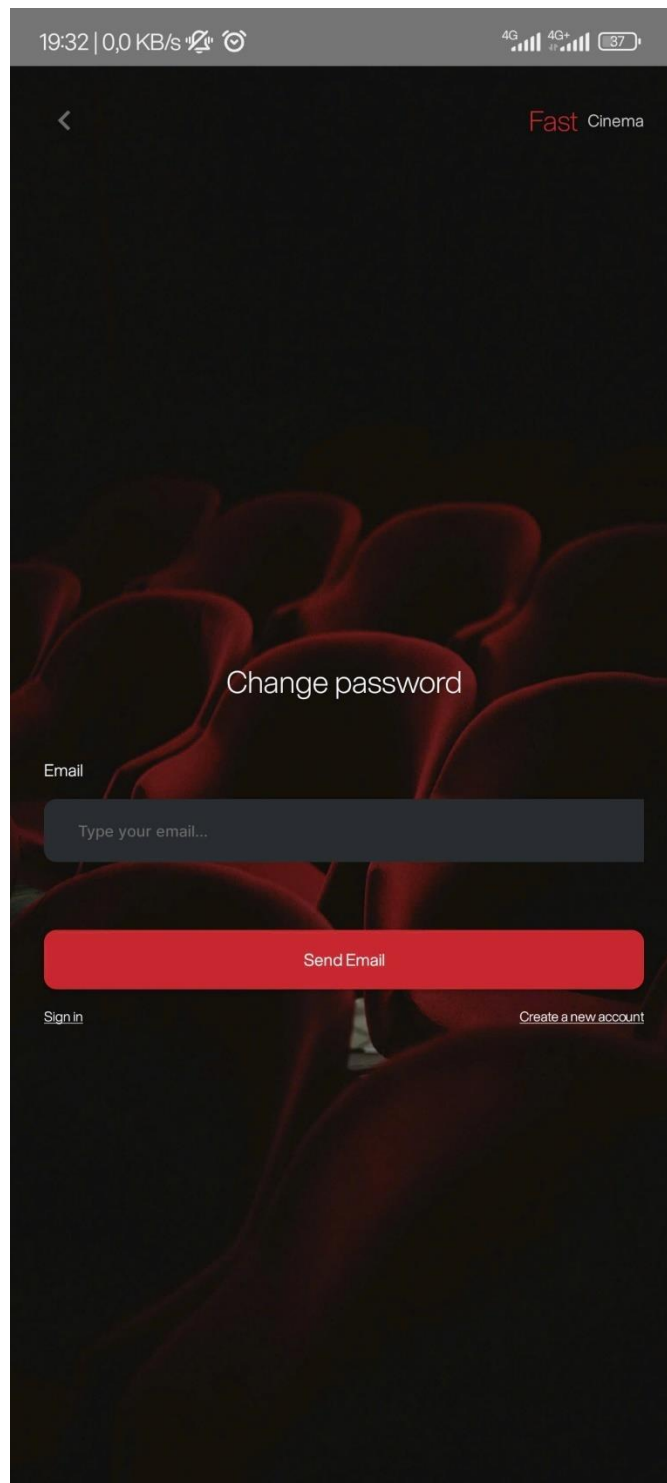
Date of birth



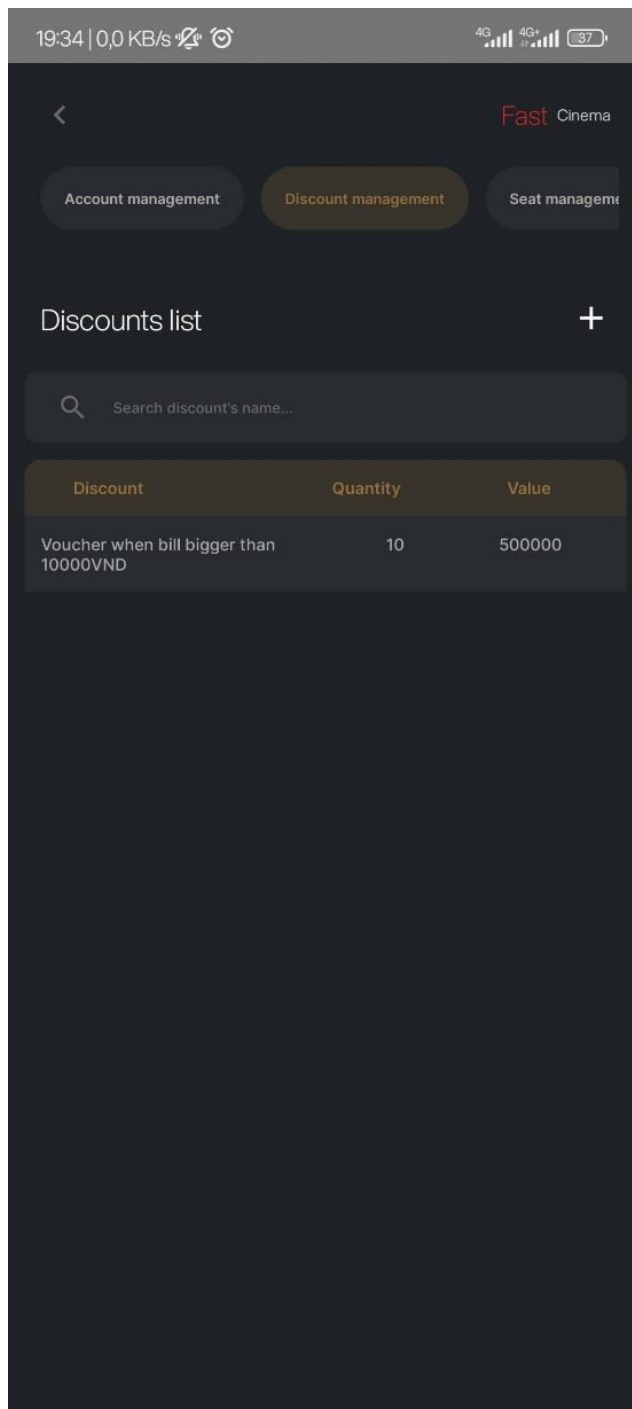
Create

[Sign in](#)

*Hình 3: Giao diện đăng ký tài khoản*




*Hình 4: Giao diện quên mật khẩu*



Hình 5: Giao diện hiển thị danh sách mã giảm giá

19:34 | 0,0 KB/s 4G+ 4G+ 37

< Add discount



Discount

Type discount's name...

Discount quantity Ticket value

Type discount quantity... Ex: 30%


Discount description


Type discount's description...

Add

Hình 6: Giao diện thêm mã giảm giá

19:34 | 0,2 KB/s 4G 4G+ 37

< Edit discount 



Discount

Voucher when bill bigger than 10000VND

Discount quantity

Discount value

10

500000

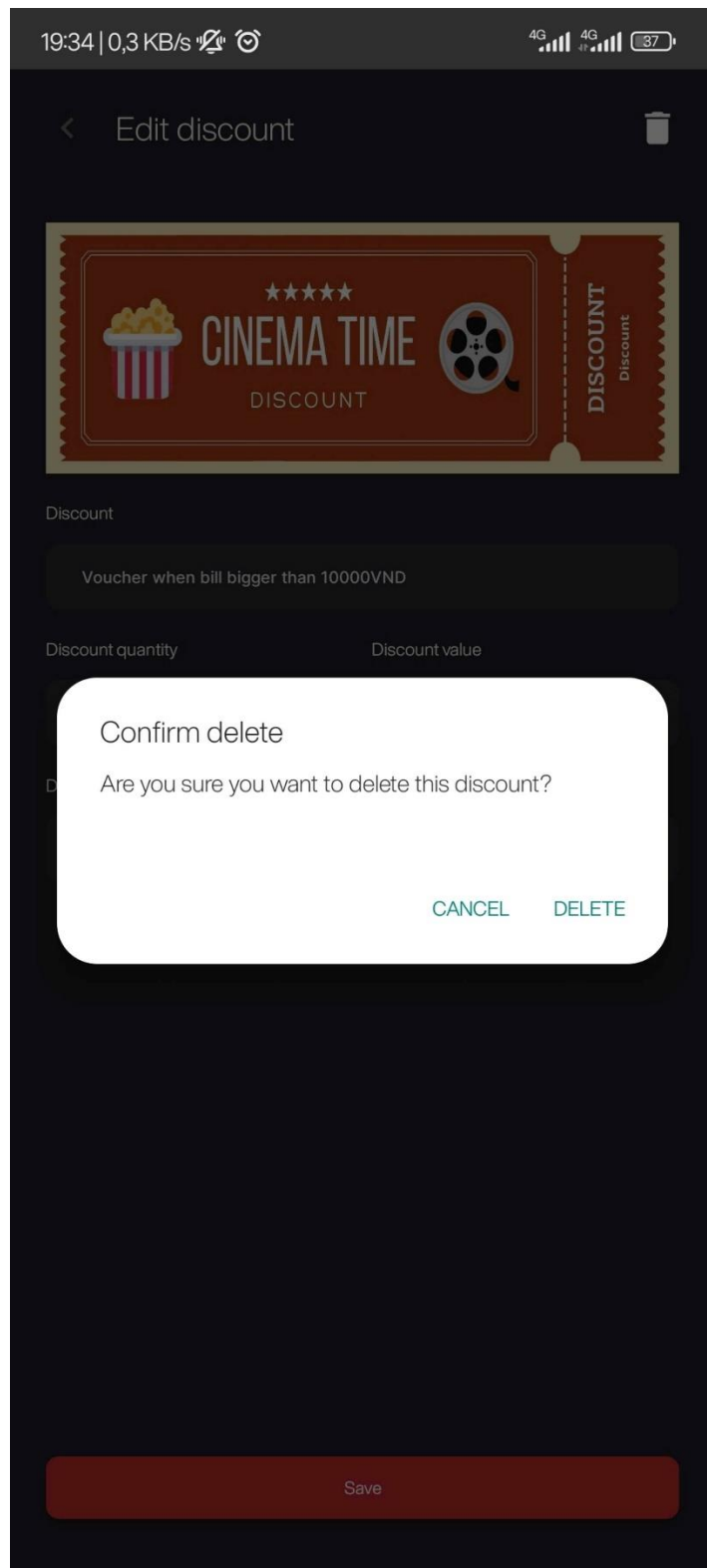
Discount description

Voucher when bill bigger than 10000VND

Save

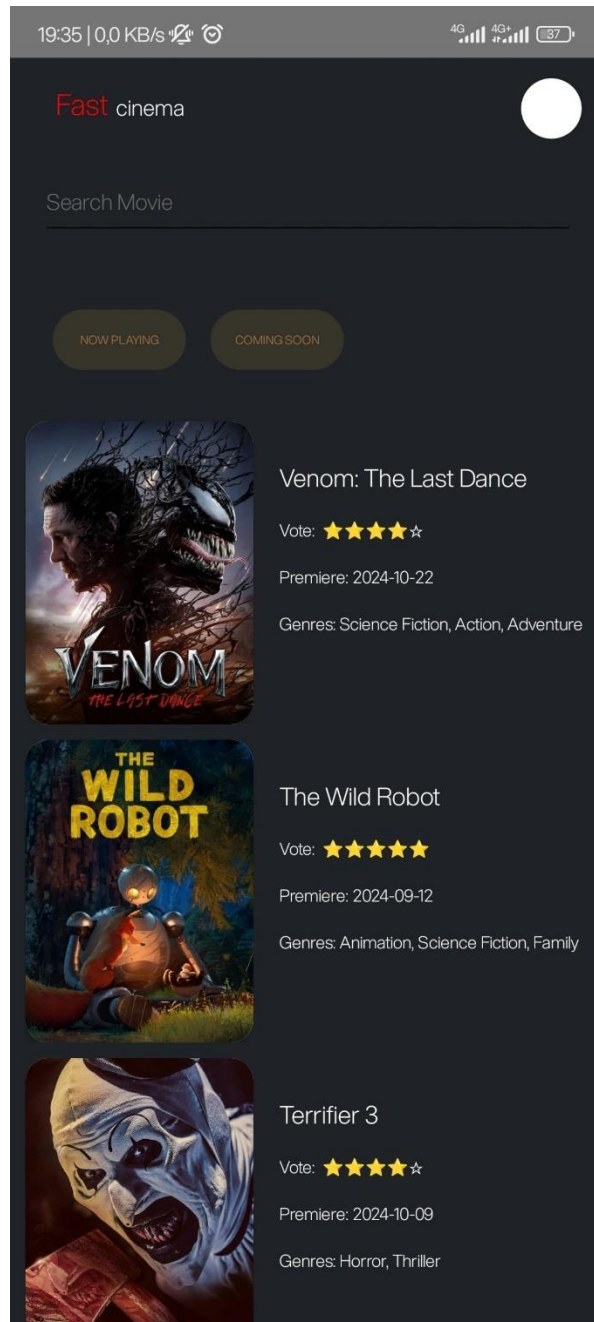
Hình 7: Giao diện sửa mã giảm giá



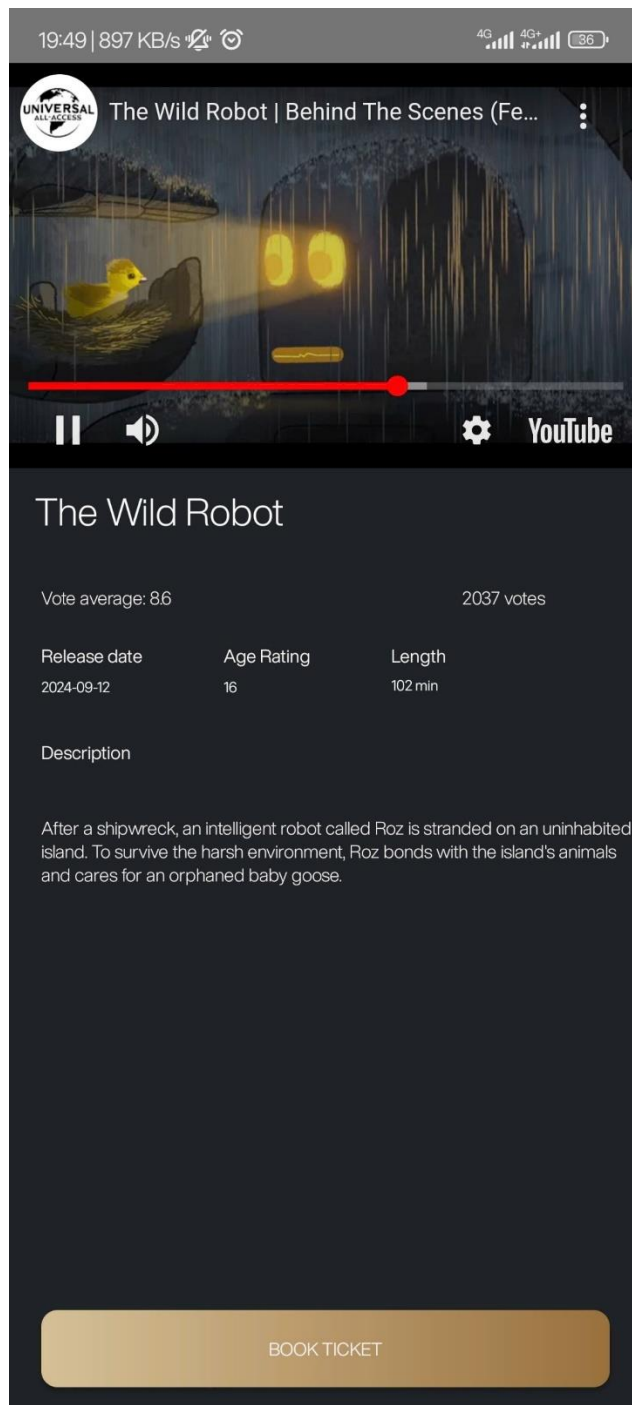


Hình 8: Giao diện xác nhận xóa mã giảm giá

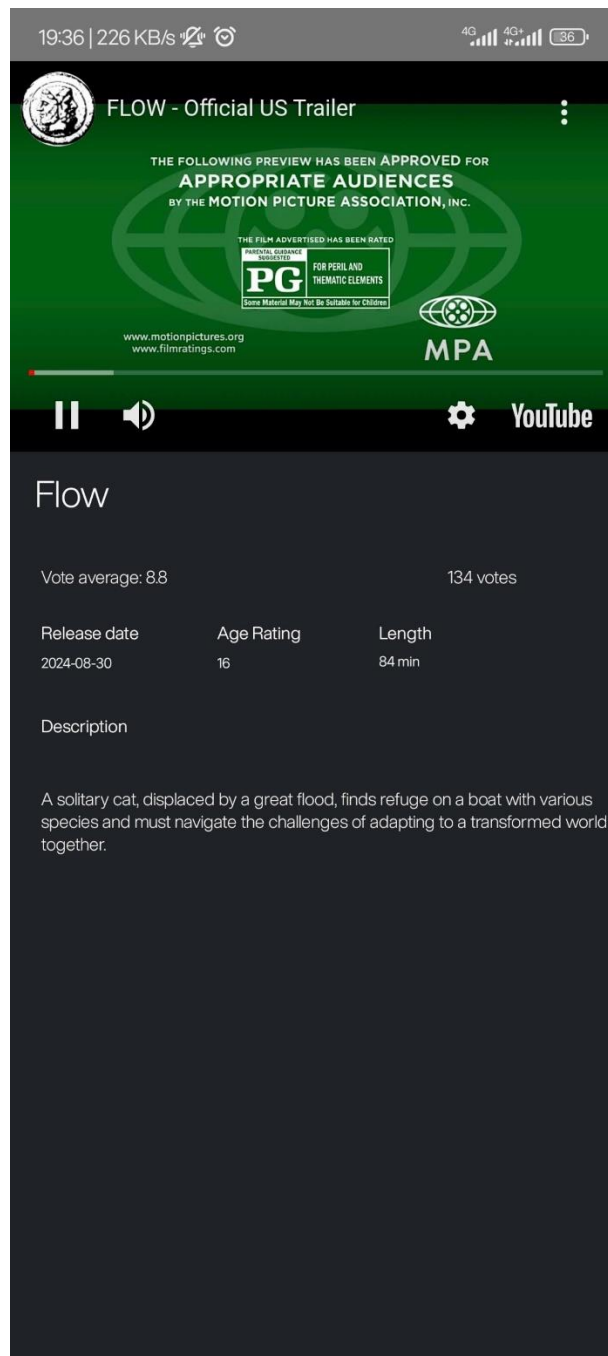
**b) User**



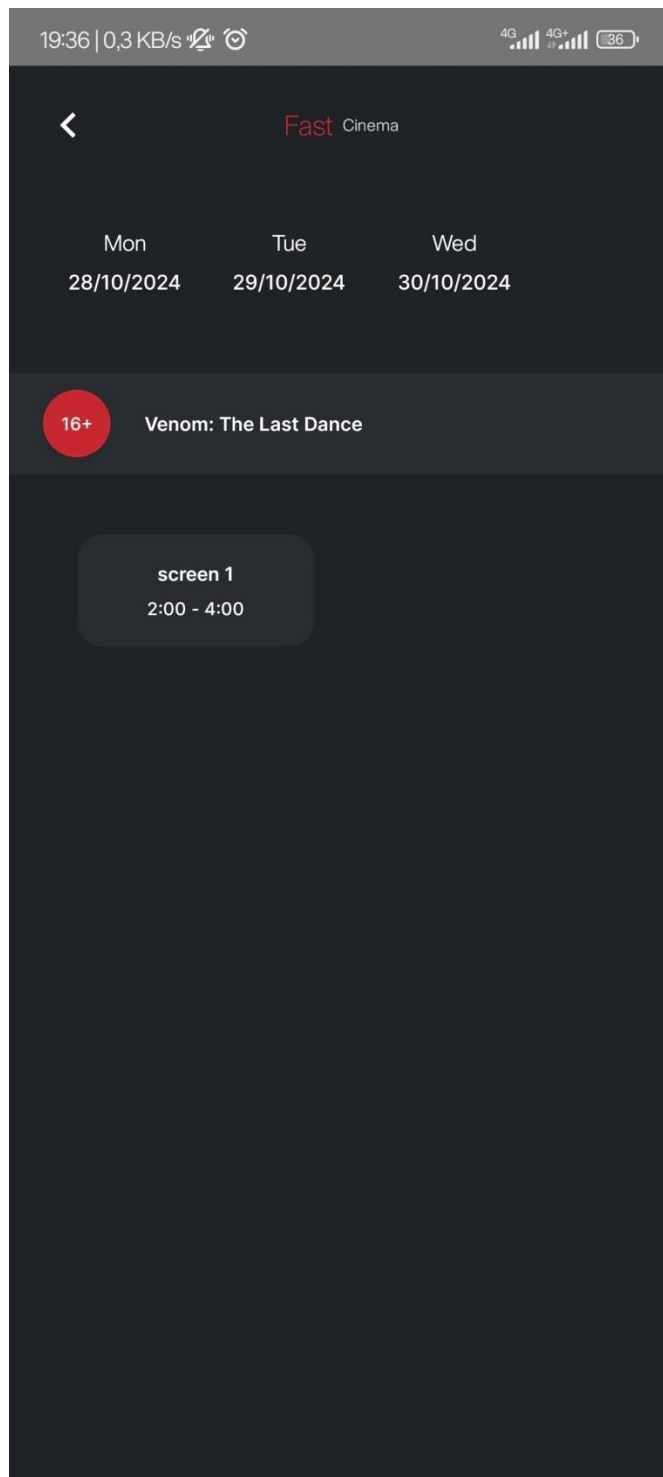
*Hình 9: Giao diện hiển thị phim đang chiếu và sắp chiếu*



Hình 10: Giao diện chi tiết phim đang chiếu



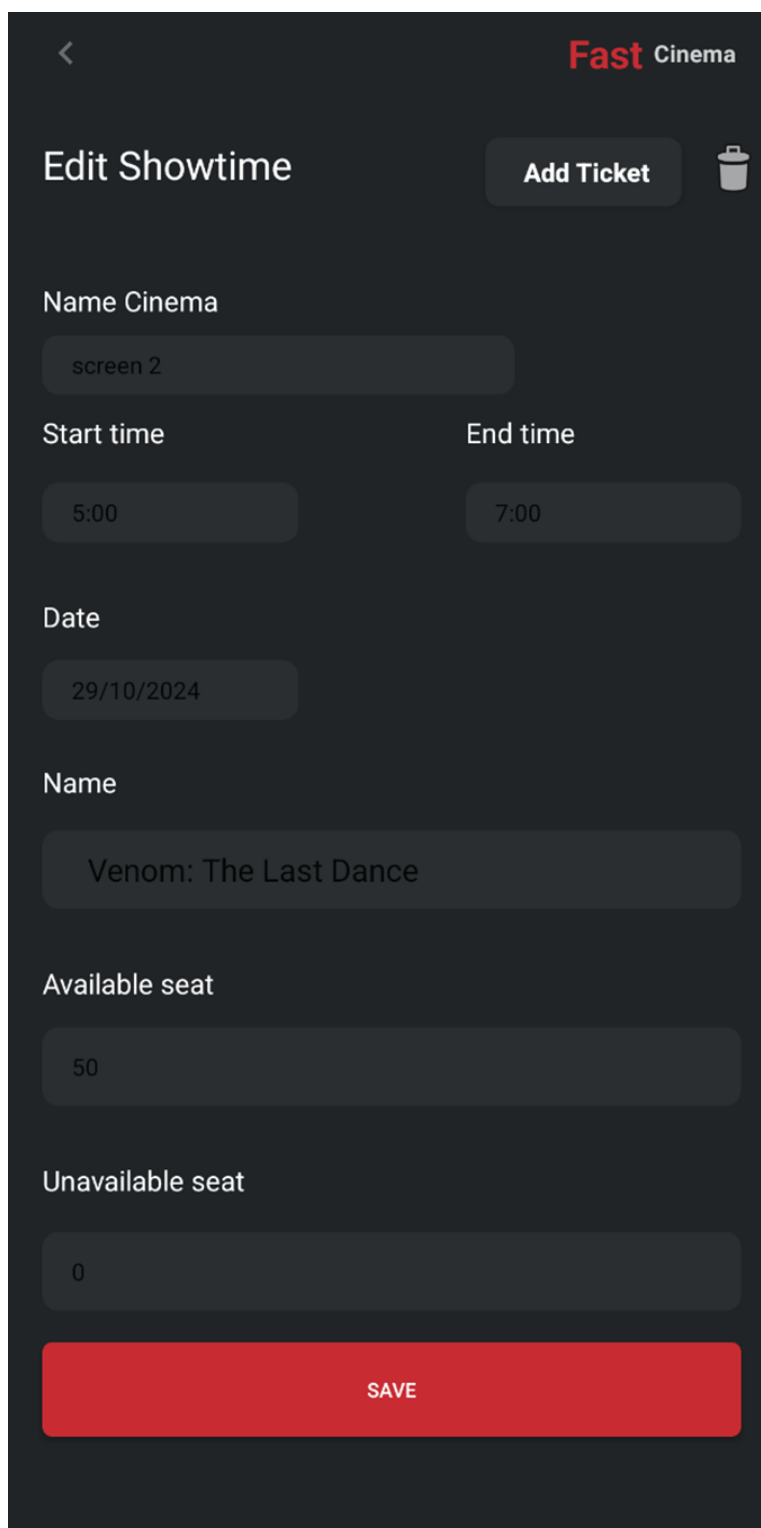
*Hình 11: Giao diện chi tiết phim sắp chiếu*




*Hình 12: Giao diện hiển thị ngày chiếu và xuất chiếu*

### 3.3.2. *Phí Văn Đức*

#### a) Admin



< Fast Cinema

Edit Showtime Add Ticket 

Name Cinema

screen 2

Start time End time

5:00 7:00

Date

29/10/2024

Name

Venom: The Last Dance

Available seat

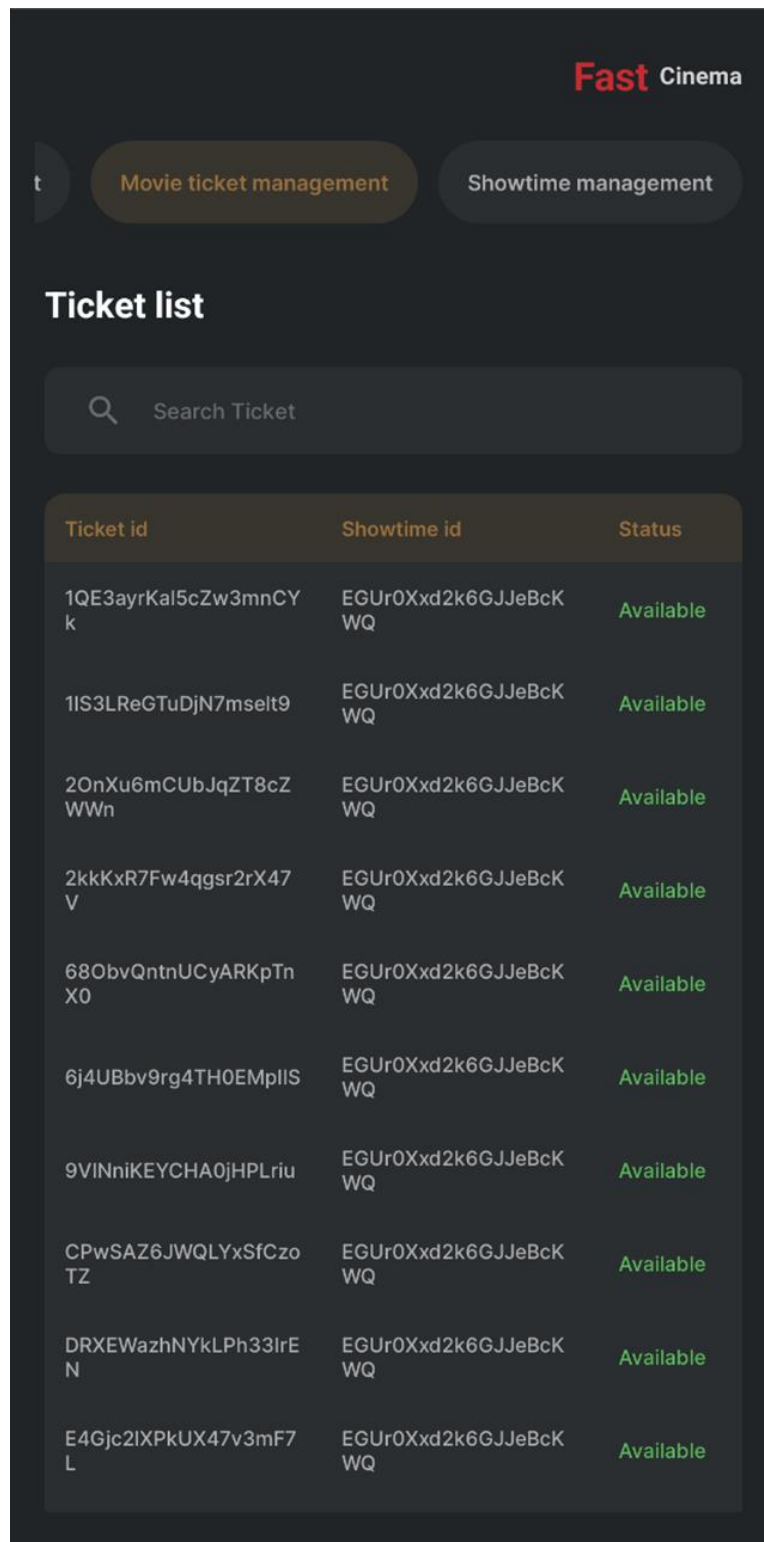
50

Unavailable seat

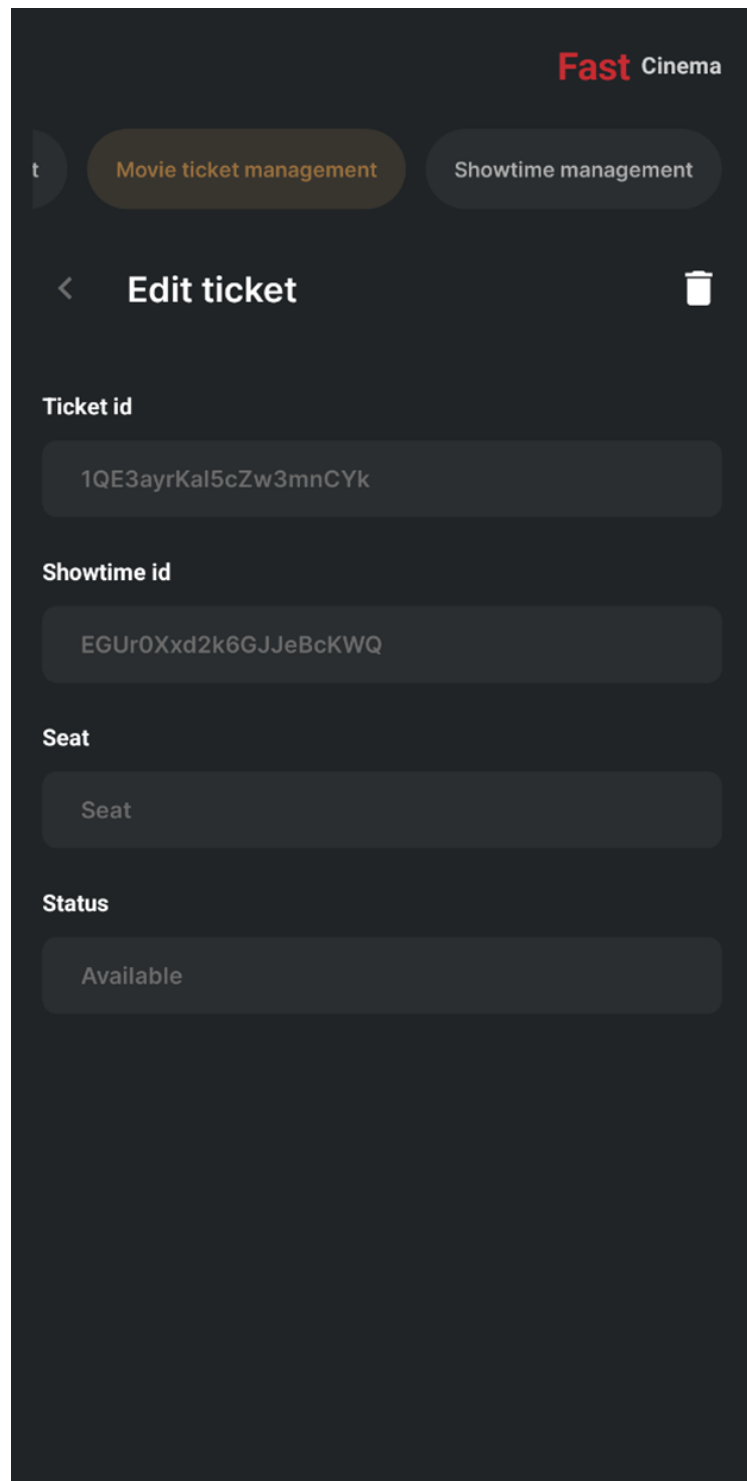
0

SAVE

Hình 13: Giao diện thêm vé



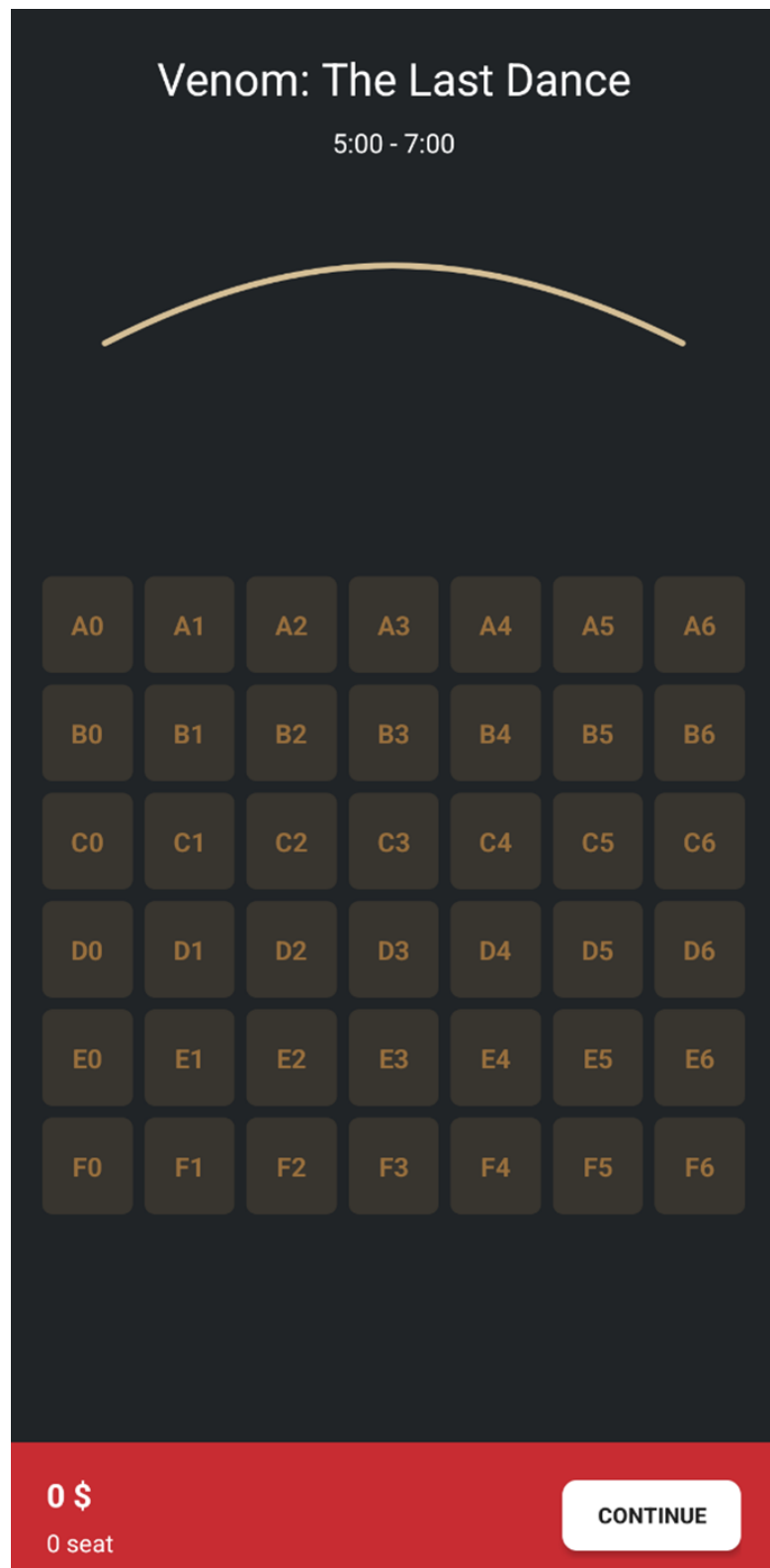
Hình 14: Giao diện danh sách các vé



*Hình 15: Giao diện hiển thị chi tiết vé và xóa vé*




b) User



Hình 16: Giao diện danh sách các ghế ngồi

# Venom: The Last Dance

5:00 - 7:00



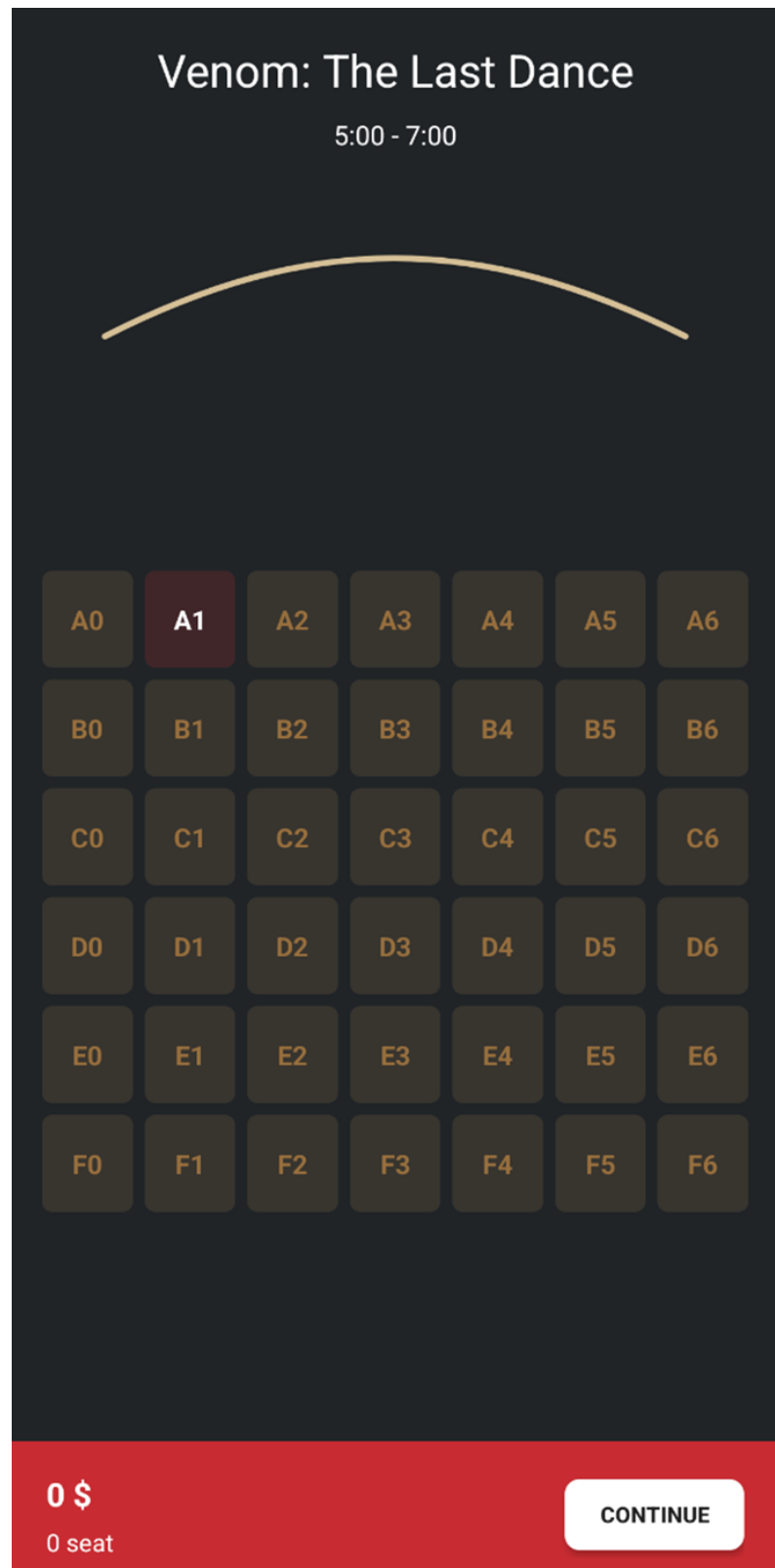
A0	A1	A2	A3	A4	A5	A6
B0	B1	B2	B3	B4	B5	B6
C0	C1	C2	C3	C4	C5	C6
D0	D1	D2	D3	D4	D5	D6
E0	E1	E2	E3	E4	E5	E6
F0	F1	F2	F3	F4	F5	F6

**320000 vnd**

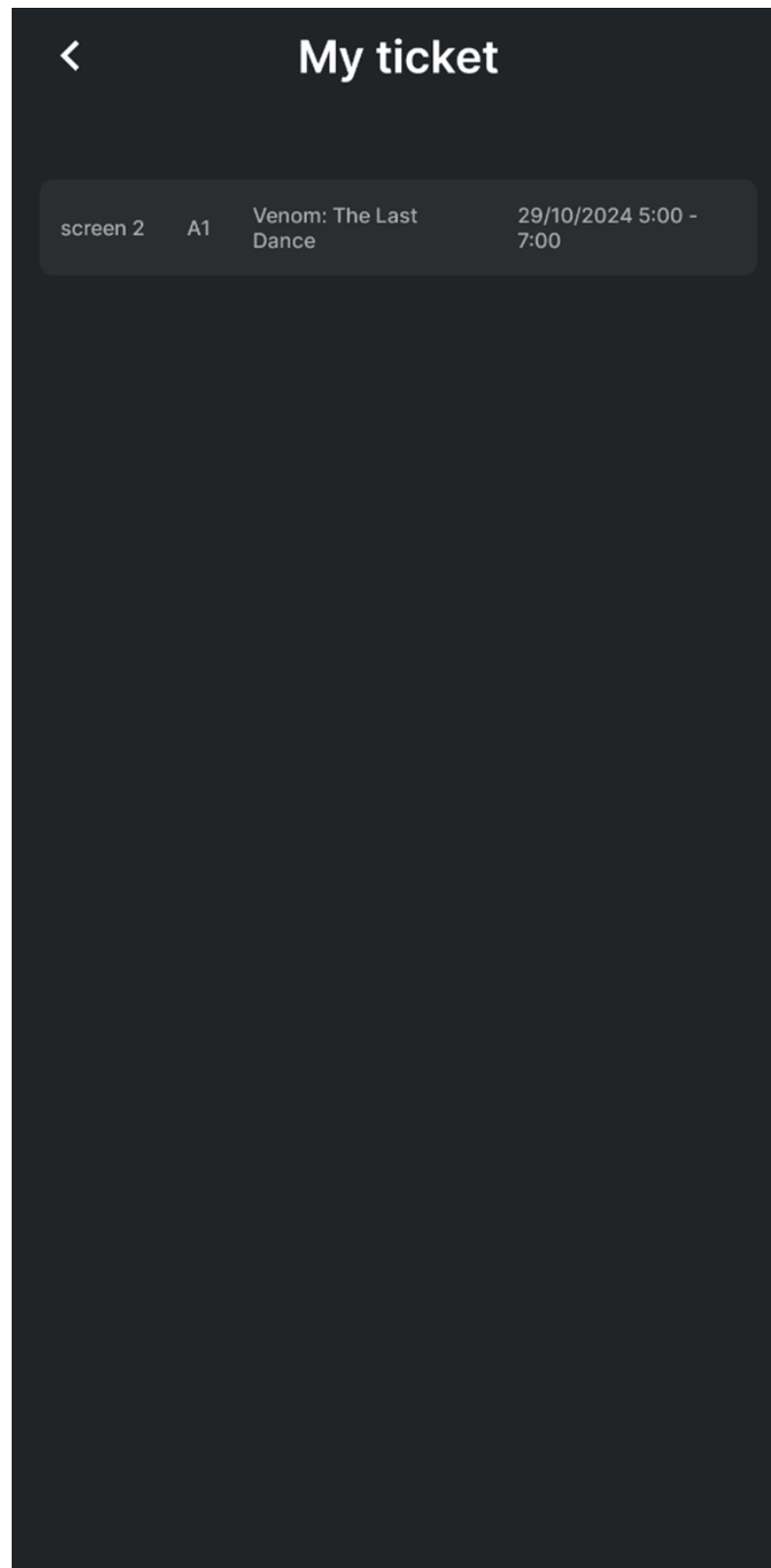
4 seats

CONTINUE

Hình 17: Giao diện chọn ghế ngồi



Hình 18: Giao diện danh sách ghế bao gồm ghế đã đặt



*Hình 19: Giao diện danh sách các vé đã mua*

### 3.3.3. Lê Đình Dũng

#### a) Admin

Showtime list				
<div><div></div><div>Search</div></div>				
No.	Name Movie	Name cinema	Time	Date
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024
1	Screen 1	Available	1:30 - 3:00	01/01/2024

Hình 20: Giao diện Showtime List

Fast Cinema

Add Showtime

Name cinema

Name Cinema

Start time

Time

End time

Time

Date

Date

Name

Item 1

Available seat

Available seat

Unavailable seat

Unavailable seat

ADD

Hình 21: Giao diện Add Showtime

<

Fast Cinema

Edit Showtime

ADD TICKET

Name Cinema

Name Cinema

Start time

End time

Time

Time

Date

Date

Name

Item 1

Available seat

Available seat

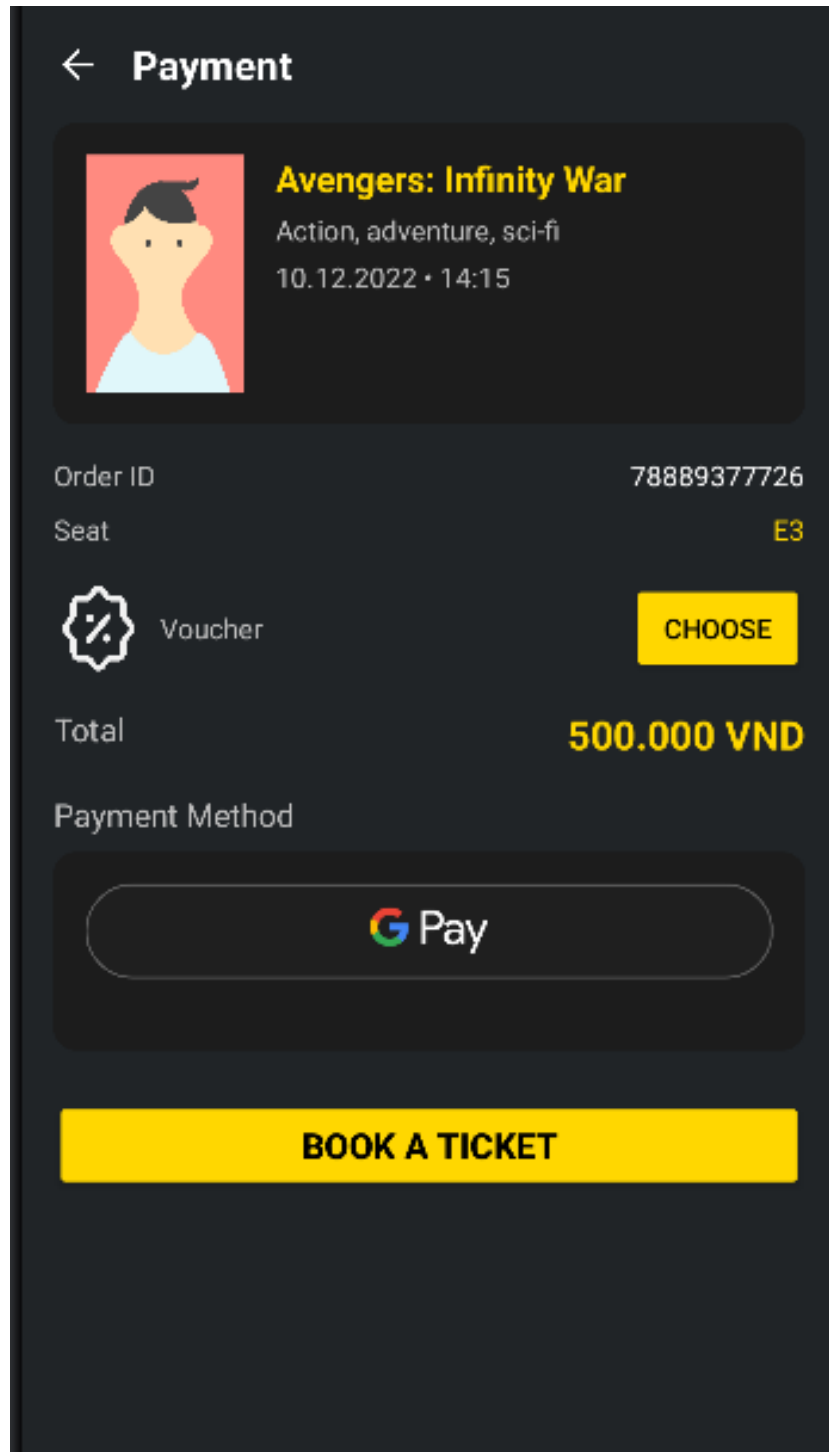
Unavailable seat

Unavailable seat

SAVE

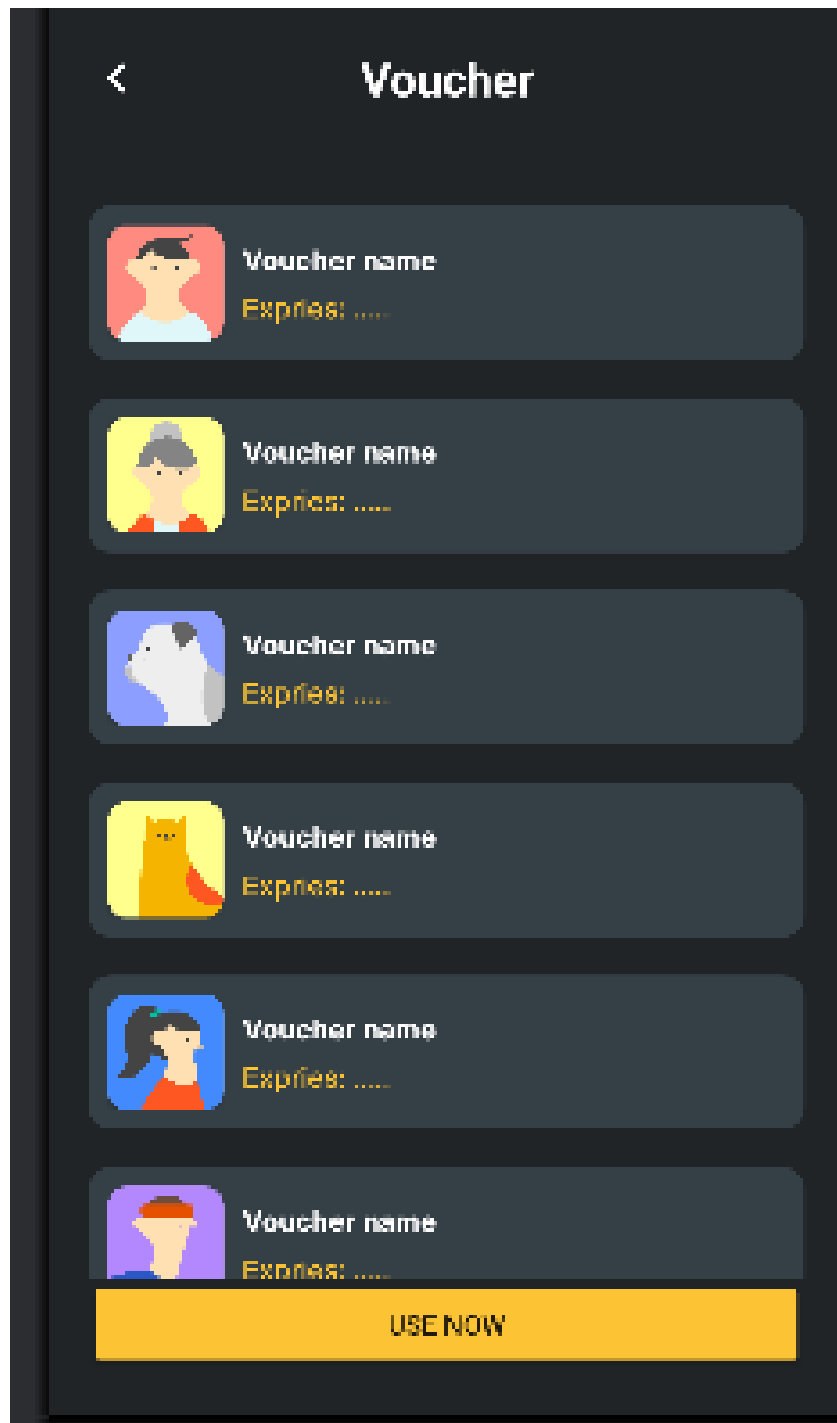
Hình 22: Giao diện Edit Showtime

b) User

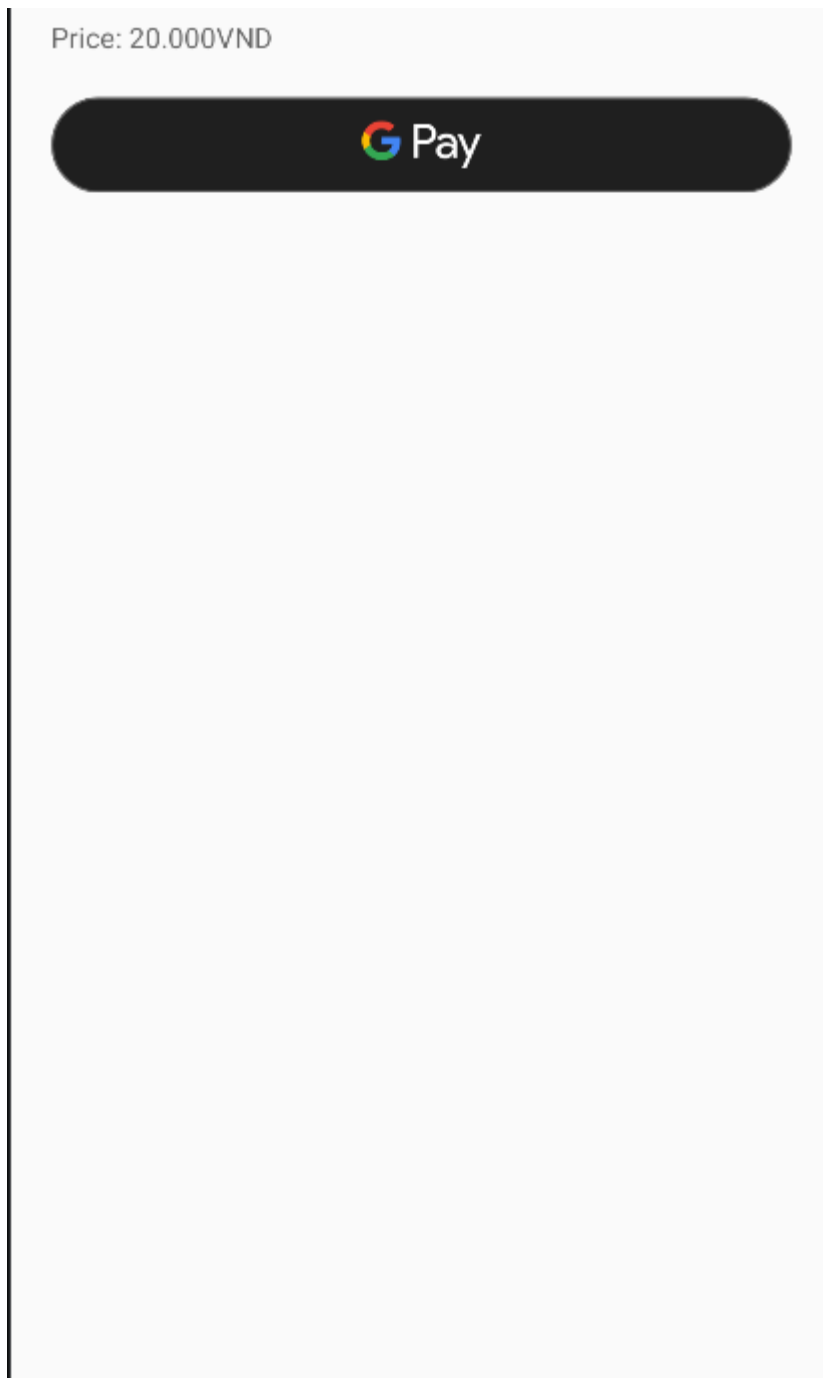


Hình 23: Giao diện Payment





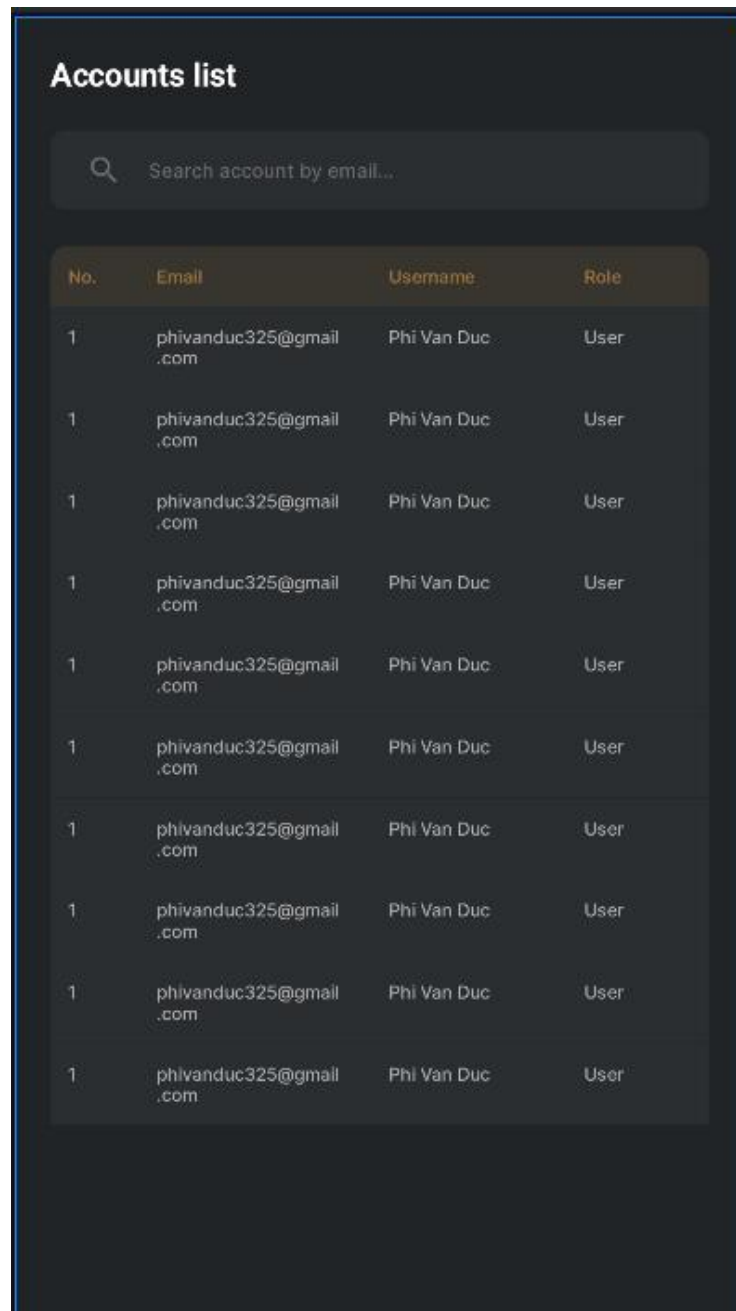
Hình 24: Giao diện Voucher



*Hình 25: Giao diện Payment (Google Pay)*

### 3.3.4. Cù Tiến Thịnh

#### a) Admin



No.	Email	Username	Role
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User
1	phivanduc325@gmail.com	Phi Van Duc	User

Hình 26: Giao diện danh sách tài khoản

<

Edit account

Username

Enter username

Email

Enter email

Phone number

Enter phone number

Address

Enter address

Date of birth

Enter date of birth

Gender

☐ Male

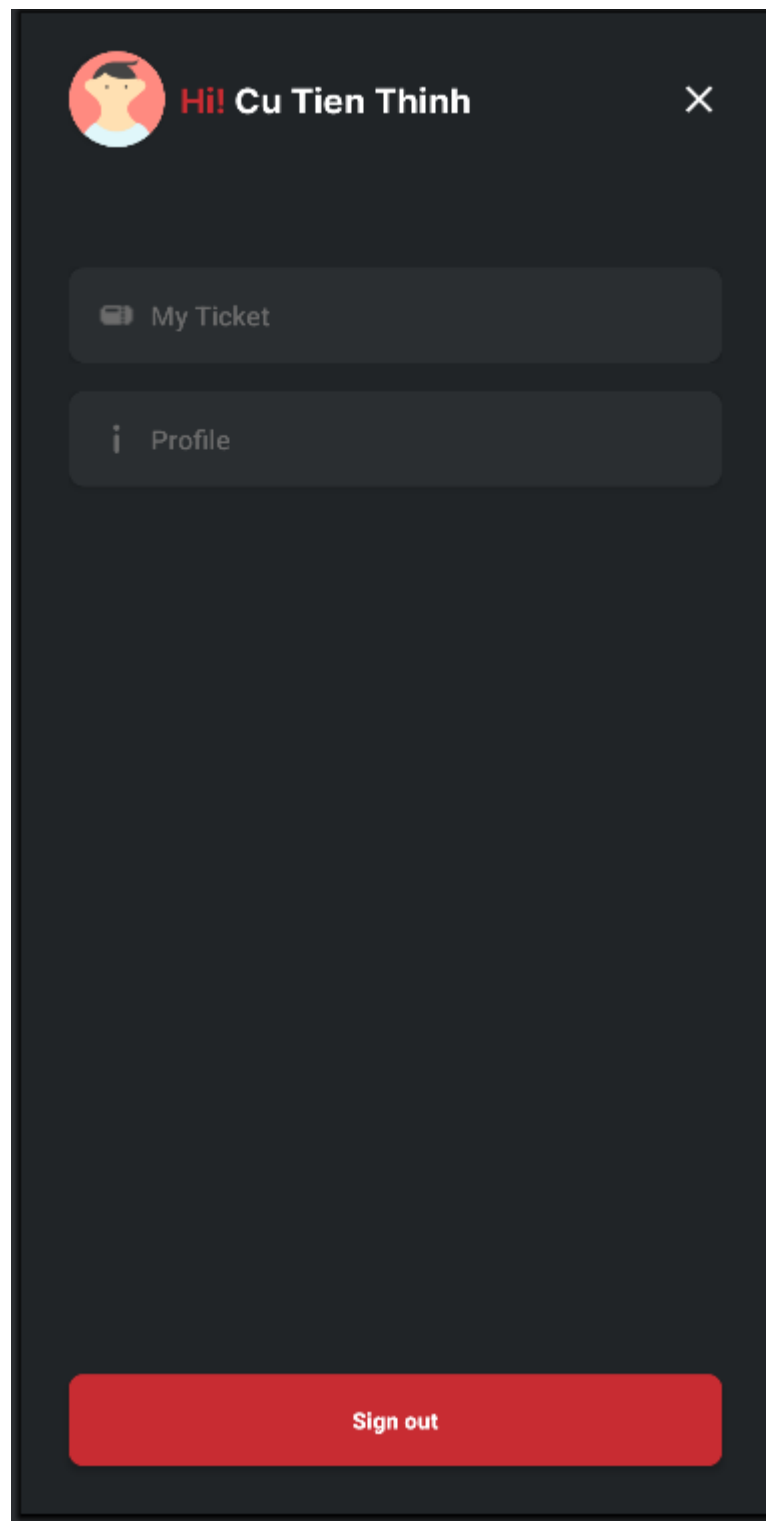
☐ Female

☐ Other

Save

Hình 27: Giao diện chỉnh sửa tài khoản (Chỉ admin)

**b) User**



*Hình 28: Giao diện MenuProfile*

Profile

Username

Enter username

Email

Enter email

Phone number

Enter phone number

Address

Enter address

Date of birth

Enter date of birth

Gender

☐ Male

☐ Female

☐ Other

Password

CHANGE PASSWORD

Delete Account

DELETE

Save

Hình 29: Giao diện sửa tài khoản user và chọn các chức năng đổi mật khẩu và xóa tài khoản

< **Change pasword**

**Current password**

Current your password

**New password**

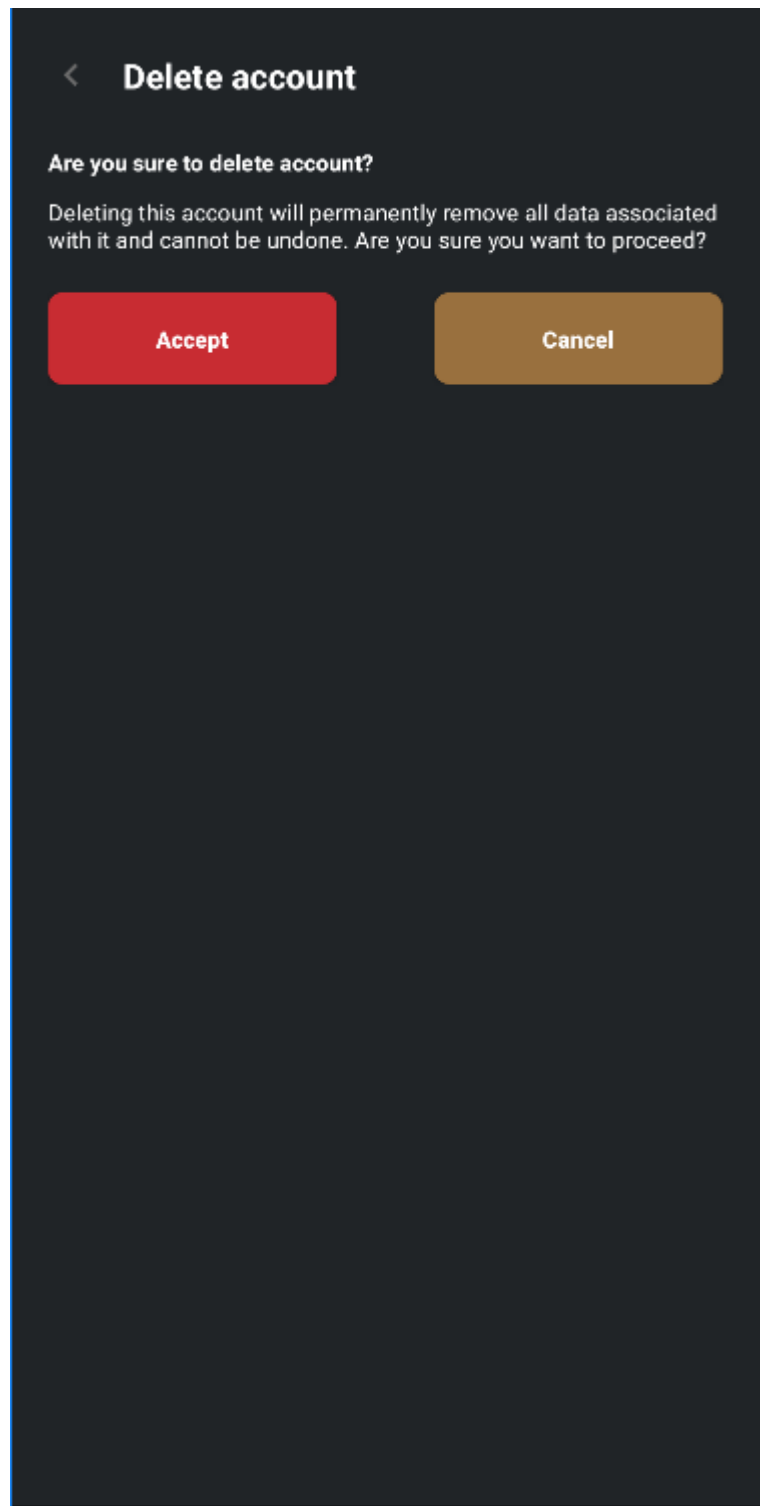
Enter your new password

**Confirm New Password**

Confirm your new password

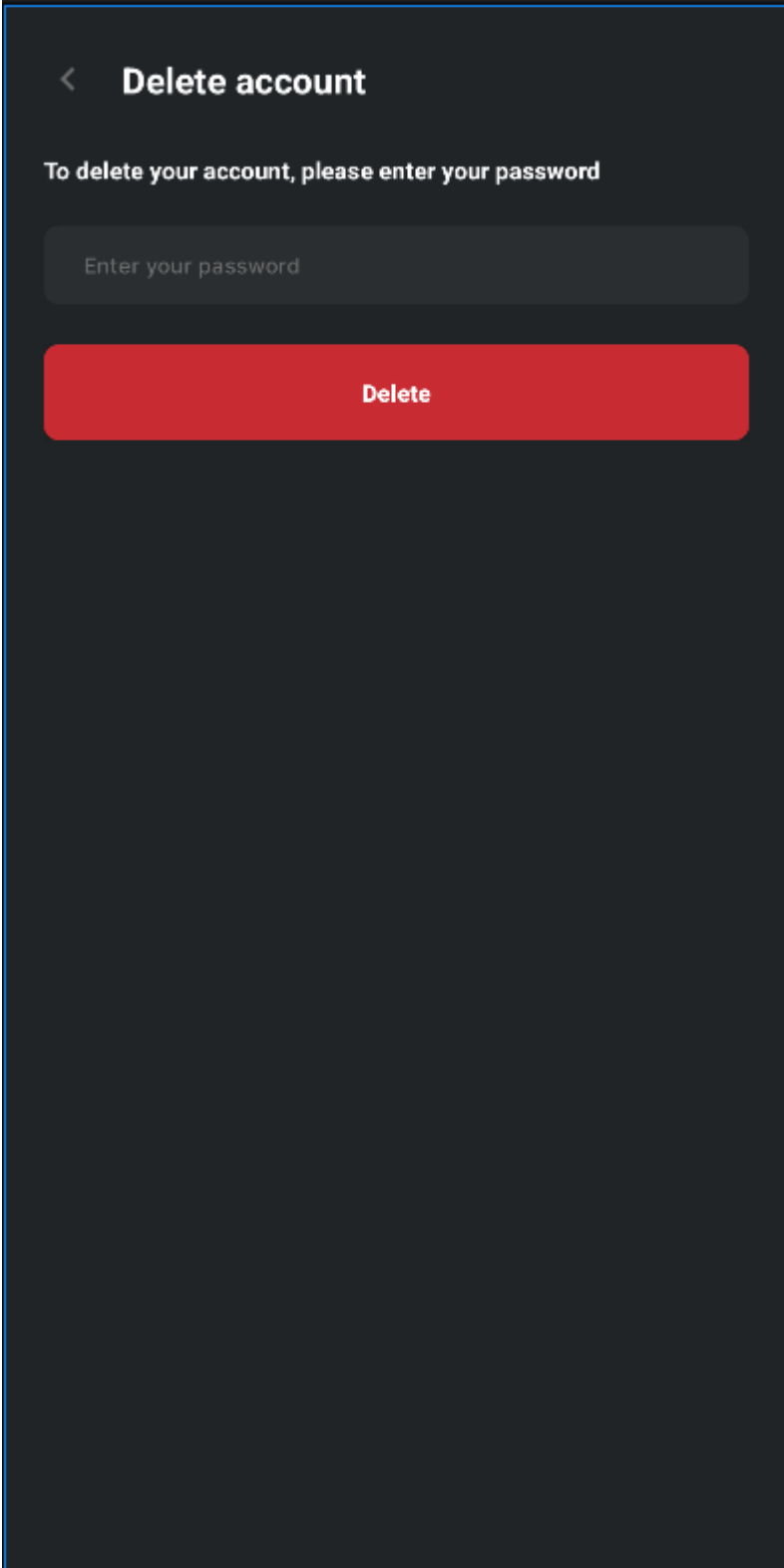
**Change Password**

*Hình 30: Giao diện đổi mật khẩu*



*Hình 31: Giao diện xác nhận xóa tài khoản*



A mobile application screen with a dark blue background. At the top left is a white back arrow icon. To its right is the title 'Delete account' in white. Below the title is a white instruction: 'To delete your account, please enter your password'. Underneath is a dark grey rounded rectangular input field with the placeholder text 'Enter your password' in light grey. Below the input field is a large red rounded rectangular button with the word 'Delete' in white.

< Delete account

To delete your account, please enter your password

Enter your password

Delete

*Hình 32: Giao diện nhập mật khẩu để xóa tài khoản*

## KẾT LUẬN

### 1. Kết quả đạt được

Sau thời gian thực hiện nghiên cứu, chúng em đã hoàn thành được những mục tiêu được đề ra như sau:

- Nghiên cứu phương pháp và các vấn đề xây dựng một phần mềm ứng dụng quản lý hoạt động đặt vé xem phim.
- Khảo sát và tìm hiểu những đặc trưng của quản lý và kinh doanh trong lĩnh vực điện ảnh
- Phát triển ứng dụng cho các thiết bị di động
- Xây dựng phần mềm ứng dụng quản lý hoạt động đặt vé xem phim

### 2. Hướng phát triển

Sau thời gian làm bài tập lớn, mặc dù nhóm đã nỗ lực làm việc và nghiên cứu nhưng vì thời gian, kiến thức, cũng như kinh nghiệm thực tế mà sản phẩm chưa thể hoàn thành một cách tốt nhất.

Chúng em sẽ cố gắng, tiếp tục nghiên cứu để sản phẩm được hoàn thành tốt hơn trong tương lai.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1]. Raymond Gallardo, Scott Hommel, Sowmya Kannan, Joni Gordon, Sharon Biocca Zakhour. *The Java Tutorial: A Short Course on the Basics, 6th edition.*
- [2]. Dave MacLean, Satya Komatineni, Grant Allen. *Pro Android 5.*
- [3]. Lê Hoàng Sơn, Nguyễn Thọ Thông. NXB: Xây dựng. *Giáo trình Lập trình Android: Giáo trình dành cho bậc đại học ngành công nghệ thông tin.*

## PHỤ LỤC

- **Xác thực:** Sử dụng Firebase Authentication để xác thực người dùng và đảm bảo chỉ người dùng đã đăng ký mới có thể truy cập vào ứng dụng. Firebase Authentication cung cấp nhiều phương thức đăng nhập, bao gồm email và mật khẩu, Google, Facebook, và các tài khoản mạng xã hội khác.
- **Firebase Firestore:** Firebase Firestore được sử dụng làm cơ sở dữ liệu chính cho ứng dụng. Firestore cung cấp khả năng đọc, ghi, cập nhật và xóa dữ liệu theo thời gian thực, giúp ứng dụng đồng bộ hóa dữ liệu của người dùng trên nhiều thiết bị và đảm bảo trải nghiệm mượt mà.
- **Hiệu năng:** Firebase Firestore cung cấp các tính năng tối ưu hiệu năng cho ứng dụng, bao gồm chế độ offline cho phép người dùng tiếp tục thao tác ngay cả khi mất kết nối. Việc tải dữ liệu từ Firestore và xử lý các yêu cầu có thể bị ảnh hưởng bởi tốc độ mạng, nên việc kiểm tra và tối ưu hóa hiệu năng là rất quan trọng.
- **Bảo mật:** Firebase cung cấp các quy tắc bảo mật tùy chỉnh cho Firestore, cho phép kiểm soát quyền truy cập vào cơ sở dữ liệu dựa trên xác thực người dùng. Các quy tắc bảo mật này giúp bảo vệ dữ liệu người dùng khỏi các truy cập trái phép. Đồng thời, mã hóa dữ liệu và bảo vệ thông tin xác thực là điều cần thiết để ngăn ngừa các rủi ro về bảo mật.
- **Tích hợp Google Pay:** Ứng dụng tích hợp Google Pay để cung cấp cho người dùng phương thức thanh toán trực tuyến. Google Pay giúp đảm bảo an toàn cho các giao dịch và cho phép người dùng thanh toán nhanh chóng. Đảm bảo thực hiện đầy đủ các bước xác thực và mã hóa dữ liệu khi tích hợp với Google Pay để bảo vệ thông tin tài khoản người dùng.

- **Code đầy đủ đã thực hiện**

- a) **Đăng Quốc Hiệp**

- **App Admin**

- **Admin.java**

```
package com.example.admincse441project.data.model;

public class Admin {
    private String uid;
    private String email;
    private String username;
    private String phoneNumber;
    private String password;
    private String dateOfBirth;

    public String getUid() {
        return uid;
    }

    public void setUid(String uid) {
        this.uid = uid;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getDateOfBirth() {
        return dateOfBirth;
    }
}
```

```

        public void setDateOfBirth(String dateOfBirth) {
            this.dateOfBirth = dateOfBirth;
        }

        public Admin() {
        }

        public Admin(String uid, String email, String username, String
phoneNumber, String password, String dateOfBirth) {
            this.uid = uid;
            this.email = email;
            this.username = username;
            this.phoneNumber = phoneNumber;
            this.password = password;
            this.dateOfBirth = dateOfBirth;
        }
    }
}

```

#### o **SignInActivity.java**

```

package com.example.admncse441project.ui.auth.signin;

import android.content.Intent;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.example.admncse441project.R;
import com.example.admncse441project.databinding.ActivitySignInBinding;
import com.example.admncse441project.databinding.ActivitySignUpBinding;
import com.example.admncse441project.ui.auth.forgotpassword.ForgotPasswordActi
vity;
import com.example.admncse441project.ui.auth.signup.SignUpActivity;
import com.example.admncse441project.ui.home.MainActivity;

public class SignInActivity extends AppCompatActivity {
    private ActivitySignInBinding binding;
    private SignInViewModel viewModel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivitySignInBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new
ViewModelProvider(this).get(SignInViewModel.class);
        onClickView();
        observeViewModel();
    }

    private void onClickView() {
        binding.txtSignUp.setOnClickListener( v -> {
            Intent intent = new Intent(SignInActivity.this,
SignUpActivity.class);
            startActivity(intent);
            finish();
        });
    }
}

```

```

        binding.btnSignIn.setOnClickListener(v -> {
            signIn();
        });
        binding.txtForgotpassword.setOnClickListener(v->{
            Intent intent = new Intent(SignInActivity.this,
ForgotPasswordActivity.class);
            startActivity(intent);
            finish();
        });
    }

    private void signIn() {
        String adminEmail = binding.etEmail.getText().toString().trim();
        String adminPassword =
binding.etPassword.getText().toString().trim();

        if(viewModel.notEmpty(adminEmail, adminPassword)){
            viewModel.login(adminEmail, adminPassword);
        }else {
            for (EditText item: new EditText[]{binding.etEmail,
binding.etPassword}) {
                item.setError(item.getHint() + " is required");
            }
        }
    }

    private void observeViewModel() {
        viewModel.loginStatus.observe(this, isSuccess -> {
            if (isSuccess){
                Intent intent = new Intent(SignInActivity.this,
MainActivity.class);
                startActivity(intent);
                finish();
                Toast.makeText(this, "Login Successfully",
Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(this, "Login failed",
Toast.LENGTH_LONG).show();
            }
        });
    }
}

```

#### o **SignInViewModel.java**

```

package com.example.admincse441project.ui.auth.signin;

import android.view.View;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.utils.FirebaseUtils;

public class SignInViewModel extends ViewModel {
    private MutableLiveData<Boolean> _loginStatus = new
MutableLiveData<>();
    LiveData<Boolean> loginStatus = _loginStatus;

    void login(String email, String password){
        FirebaseUtils.firebaseAuth.signInWithEmailAndPassword(email,

```

```

password).addOnCompleteListener(task -> {
    _loginStatus.postValue(task.isSuccessful());
});
}
boolean notEmpty(String email, String password){
    return !email.isEmpty() && !password.isEmpty();
}
}

```

### o **SignUpActivity.java**

```

package com.example.admincse441project.ui.auth.signup;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.example.admincse441project.R;
import com.example.admincse441project.databinding.ActivitySignUpBinding;
import com.example.admincse441project.ui.auth.signin.SignInActivity;

public class SignUpActivity extends AppCompatActivity {
    private ActivitySignUpBinding binding;
    private SignUpViewModel viewModel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        binding = ActivitySignUpBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new
        ViewModelProvider(this).get(SignUpViewModel.class);
        setContentView(binding.getRoot());

        onClickView();
        observeViewModel();
    }

    private void observeViewModel() {
        viewModel.signUpStatus.observe(this, isSuccess -> {
            if (isSuccess) {
                String adminEmail =
                binding.etEmail.getText().toString().trim();
                String adminName =
                binding.etFullname.getText().toString().trim();
                String phoneNumber =
                binding.etPhoneNumber.getText().toString().trim();
                String dateOfBirth =
                binding.etDateOfBirth.getText().toString().trim();
                viewModel.saveUserDetail(adminName, adminEmail,
                dateOfBirth, phoneNumber);
            } else {
                Toast.makeText(this, "Cannot Sign Up",
                Toast.LENGTH_LONG).show();
            }
        });
    }
}

```



```

    });

    viewModel.saveAdminDetailStatus.observe(this, isSuccess -> {
        if (isSuccess) {
            Intent intent = new Intent(SignUpActivity.this,
SignInActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            Toast.makeText(this, "Sign Up Successfully",
Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(this, "Sign Up failed",
Toast.LENGTH_LONG).show();
        }
    });
}

private void onClickView() {
    binding.txtSignIn.setOnClickListener(new View.OnClickListener()
{
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(SignUpActivity.this,
SignInActivity.class);
            startActivity(intent);
        }
    });

    binding.btnCreate.setOnClickListener(v -> {
        signUp();
    });

    private void signUp() {
        String userEmail = binding.etEmail.getText().toString().trim();
        String userPassword =
binding.etPassword.getText().toString().trim();
        String userName =
binding.etFullname.getText().toString().trim();
        String dateOfBirth =
binding.etDateOfBirth.getText().toString().trim();
        String phoneNumber =
binding.etPhoneNumber.getText().toString().trim();

        if (viewModel.notEmpty(userEmail, userName, userPassword,
dateOfBirth, phoneNumber)) {
            viewModel.signUp(userEmail, userPassword);
        } else {
            for (EditText item : new EditText[]{
                binding.etEmail, binding.etFullname,
binding.etPassword, binding.etDateOfBirth, binding.etPhoneNumber
            }) {
                if (item != null) {
                    item.setError(item.getHint() + " is required");
                }
            }
        }
    }
}
}

```

### o **SignUpViewModel.java**

```
package com.example.admncse441project.ui.auth.signup;

import android.util.Log;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admncse441project.data.model.Admin;
import com.example.admncse441project.utils.FirebaseUtils;

public class SignUpViewModel extends ViewModel {
    private MutableLiveData<Boolean> _signUpStatus = new
MutableLiveData<>();
    LiveData<Boolean> signUpStatus = _signUpStatus;

    private MutableLiveData<Boolean> _saveAdminDetailStatus = new
MutableLiveData<>();
    LiveData<Boolean> saveAdminDetailStatus = _saveAdminDetailStatus;

    void signUp(String adminEmail, String adminPassword) {
        FirebaseUtils.firebaseAuth.createUserWithEmailAndPassword(adminEmail,
adminPassword)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    _signUpStatus.postValue(true);
                } else {
                    Log.e("SignUpViewModel", "Sign up failed: " +
task.getException().getMessage());
                    _signUpStatus.postValue(false);
                }
            });
    }

    void saveUserDetail(String adminName, String adminEmail, String
dateOfBirth, String phoneNumber) {
        String userId = FirebaseUtils.currentUserId();

        if (userId != null) {
            Admin admin = new Admin(userId, adminEmail, adminName,
phoneNumber, "null", dateOfBirth);
            FirebaseUtils.currentUserDetail().set(admin).addOnCompleteListener(task
-> {
                _saveAdminDetailStatus.postValue(task.isSuccessful());
            });
        } else {
            _saveAdminDetailStatus.postValue(false);
        }
    }

    Boolean notEmpty(String adminEmail, String adminName, String
userPassword, String dateOfBirth, String phoneNumber) {
        return !adminEmail.isEmpty() && !adminName.isEmpty() &&
!userPassword.isEmpty() && !dateOfBirth.isEmpty() &&
!phoneNumber.isEmpty();
    }
}
```

### o **ForgotPasswordActivity.java**

```
package com.example.admincse441project.ui.auth.forgotpassword;

import android.os.Bundle;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.example.admincse441project.R;
import com.example.admincse441project.databinding.ActivityForgotPasswordBinding;

public class ForgotPasswordActivity extends AppCompatActivity {
    private ActivityForgotPasswordBinding binding;
    private ForgotPasswordViewModel viewModel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityForgotPasswordBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new
ViewModelProvider(this).get(ForgotPasswordViewModel.class);
        onClickView();
        observeViewModel();
    }

    private void onClickView() {
        binding.btnSend.setOnClickListener(v -> {
            String userEmail =
binding.etEmail.getText().toString().trim();
            if(!userEmail.isEmpty()){
                viewModel.resetPassword(userEmail);
            }else {
                binding.etEmail.setError(binding.etEmail.getHint() + "is
required");
            }
        });
        binding.btnBack.setOnClickListener(v-> {
            finish();
        });
    }

    private void observeViewModel() {
        viewModel.resetStatus.observe(this, isSuccess ->{
            if (isSuccess){
                Toast.makeText(this, "Password reset email sent",
Toast.LENGTH_LONG).show();
                binding.btnSend.setActivated(false);
            }else{
                Toast.makeText(this, "Failed to send reset email",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

```

    }
}

```

#### o **ForgotPasswordViewModel.java**

```

package com.example.admincse441project.ui.auth.forgotpassword;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.utils.FirebaseUtils;

public class ForgotPasswordViewModel extends ViewModel {
    private MutableLiveData<Boolean> _resetStatus = new
MutableLiveData<>();
    LiveData<Boolean> resetStatus = _resetStatus;

    void resetPassword(String email){

FirebaseUtils.firebaseAuth.sendPasswordResetEmail(email).addOnCompleteLi
stener(task -> {
        if (task.isSuccessful()){
            _resetStatus.postValue(task.isSuccessful());
        }else {
            _resetStatus.postValue(false);
        }
    });
}
}

```

#### o **Discount.java**

```

package com.example.admincse441project.data.model.discount;

public class Discount {
    private String id;
    private String name;
    private int value;
    private int quantity;
    private String description;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }
}

```

```

    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Discount() {
    }

    public Discount(String id, String name, int value, int quantity,
String description) {
        this.id = id;
        this.name = name;
        this.value = value;
        this.quantity = quantity;
        this.description = description;
    }
}

```

#### o **DiscountListFragment.java**

```

package
com.example.admncse441project.ui.discountmanagement.showdiscount;

import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.admncse441project.data.model.discount.Discount;
import
com.example.admncse441project.databinding.FragmentDiscountsListBinding;
import
com.example.admncse441project.ui.discountmanagement.add.AddDiscountActi
vity;
import
com.example.admncse441project.ui.discountmanagement.edit.EditDiscountAc
tivity;

import java.util.List;

public class DiscountsListFragment extends Fragment {
    private FragmentDiscountsListBinding binding;

```

```

        private DiscountViewModel viewModel;
        private DiscountAdapter adapter;

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                                Bundle savedInstanceState) {
            binding = FragmentDiscountsListBinding.inflate(inflater,
container, false);
            viewModel = new
ViewModelProvider(this).get(DiscountViewModel.class);

            setupRecyclerView();
            setupObservers();
            onViewClickListeners();
            return binding.getRoot();
        }

        private void setupRecyclerView() {
            adapter = new DiscountAdapter(null, discount -> {
                Intent intent = new Intent(getActivity(),
EditDiscountActivity.class);
                intent.putExtra("DISCOUNT_ID", discount.getId());
                startActivity(intent);
            });
            binding.recyclerViewDiscount.setLayoutManager(new
LinearLayoutManager(getContext()));
            binding.recyclerViewDiscount.setAdapter(adapter);
        }

        private void setupObservers() {
            viewModel.discounts.observe(getViewLifecycleOwner(), discounts -
> {
                adapter.setDiscountList(discounts);
            });
        }

        private void onViewClickListeners() {
            binding.btnAddDiscount.setOnClickListener(view -> {
                Intent intent = new Intent(requireContext(),
AddDiscountActivity.class);
                startActivity(intent);
            });
        }

        @Override
        public void onResume() {
            super.onResume();
            viewModel.loadDiscounts();
        }
    }
}

```

#### o DiscountAdapter.java

```

package
com.example.admncse441project.ui.discountmanagement.showdiscount;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.admncse441project.data.model.discount.Discount;

```

```

import
com.example.admncse441project.databinding.ItemDiscountRowBinding;

import java.util.List;

public class DiscountAdapter extends
RecyclerView.Adapter<DiscountAdapter.ViewHolder> {
    private List<Discount> discountList;
    private OnDiscountClickListener listener;
    public DiscountAdapter(List<Discount> discountList,
OnDiscountClickListener listener) {
        this.discountList = discountList;
        this.listener = listener;
    }
    @NonNull
    @Override
    public DiscountAdapter.ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        LayoutInflater inflater =
LayoutInflater.from(parent.getContext());
        ItemDiscountRowBinding binding =
ItemDiscountRowBinding.inflate(inflater, parent, false);
        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull DiscountAdapter.ViewHolder
holder, int position) {
        Discount discount = discountList.get(position);
        holder.bind(discount);
        holder.itemView.setOnClickListener(v ->
listener.onDiscountClick(discount));
    }

    @Override
    public int getItemCount() {
        return discountList != null ? discountList.size() : 0;
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final ItemDiscountRowBinding binding;

        public ViewHolder(ItemDiscountRowBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }
        public void bind(Discount discount) {
            binding.txtDiscountName.setText(discount.getName());

binding.txtDiscountQuantity.setText(String.valueOf(discount.getQuantity(
)));

binding.txtDiscountValue.setText(String.valueOf(discount.getValue()));
        }
        public void setDiscountList(List<Discount> discountList) {
            this.discountList = discountList;
            notifyDataSetChanged();
        }
        public interface OnDiscountClickListener {
            void onDiscountClick(Discount discount);
        }
    }
}

```

### ○ DiscountViewModel.java

```
package
com.example.admincse441project.ui.discountmanagement.showdiscount;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.discount.Discount;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class DiscountViewModel extends ViewModel {
    private final MutableLiveData<List<Discount>> _discounts = new
MutableLiveData<>();
    public LiveData<List<Discount>> discounts = _discounts;
    public DiscountViewModel() {
        loadDiscounts();
    }
    void loadDiscounts() {
        FirebaseUtils.getDiscountsCollection().get().addOnCompleteListener(task
-> {
            if (task.isSuccessful()) {
                List<Discount> discountList = new ArrayList<>();
                QuerySnapshot querySnapshot = task.getResult();
                if (querySnapshot != null) {
                    for (DocumentSnapshot document :
querySnapshot.getDocuments()) {
                        Discount discount =
document.toObject(Discount.class);
                        if (discount != null) {
                            discountList.add(discount);
                        }
                    }
                    _discounts.setValue(discountList);
                }
            }
        });
    }
}
```

### ○ EditDiscountActivity.java

```
package com.example.admincse441project.ui.discountmanagement.edit;

import static
com.example.admincse441project.utils.FirebaseUtils.deleteDiscount;

import android.app.AlertDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
```



```

import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.example.admncse441project.R;
import com.example.admncse441project.data.model.discount.Discount;
import
com.example.admncse441project.databinding.ActivityEditDiscountBinding;

public class EditDiscountActivity extends AppCompatActivity {
    private ActivityEditDiscountBinding binding;
    private EditDiscountViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityEditDiscountBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new
ViewModelProvider(this).get(EditDiscountViewModel.class);
        Intent intent = getIntent();
        String discountId = intent.getStringExtra("DISCOUNT_ID");

        fetchDiscount(discountId);
        setupValidation();
        onViewClickListeners();

    }

    private void setupValidation() {
        binding.edtEditDiscountQuantity.addTextChangedListener(new
TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {}

            @Override
            public void afterTextChanged(Editable input) {
                validateIntegerInput(input.toString(),
binding.edtEditDiscountQuantity, "Quantity");
            }
        });

        binding.edtEditDiscountValue.addTextChangedListener(new
TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {}

            @Override
            public void afterTextChanged(Editable input) {
                validateIntegerInput(input.toString(),
binding.edtEditDiscountValue, "Value");
            }
        });
    }
}

```

```

    }
    });
}

private void validateIntegerInput(String input, EditText editText,
String fieldName) {
    if (!input.matches("\\d+")) {
        editText.setError(fieldName + " must be a integer.");
    } else {
        editText.setError(null);
    }
}

private void onViewClickListeners() {
    Intent intent = getIntent();
    String discountId = intent.getStringExtra("DISCOUNT_ID");
    binding.btnBack.setOnClickListener(v -> finish());
    binding.btnSave.setOnClickListener(v ->
saveDiscount(discountId));
    binding.btnDelete.setOnClickListener(v -> {
        new AlertDialog.Builder(this)
            .setTitle("Confirm delete")
            .setMessage("Are you sure you want to delete this
discount?")
            .setPositiveButton("Delete", (dialog, which) -> {
                deleteDiscount(discountId);
                Toast.makeText(this, "Discount deleted",
Toast.LENGTH_SHORT).show();
                finish();
            })
            .setNegativeButton("Cancel", (dialog, which) ->
dialog.dismiss())
            .show();
    });
}

private void saveDiscount(String discountId) {
    String name = binding.edtEditDiscountName.getText().toString();
    int value =
Integer.parseInt(binding.edtEditDiscountValue.getText().toString());
    int quantity =
Integer.parseInt(binding.edtEditDiscountQuantity.getText().toString());
    String description =
binding.edtEditDiscountDescription.getText().toString();

    Discount discount = new Discount( discountId, name, value,
quantity, description);
    viewModel.updateDiscount(discount, task -> {
        if (task.isSuccessful()) {
            Toast.makeText(this, "Discount updated successfully",
Toast.LENGTH_SHORT).show();
            finish();
        } else {
            Toast.makeText(this, "Failed to update discount",
Toast.LENGTH_SHORT).show();
        }
    });
}

private void fetchDiscount(String discountId) {
    viewModel.getDiscount(discountId).observe(this, discount -> {

```

```

        binding.edtEditDiscountName.setText(discount.getName());

binding.edtEditDiscountValue.setText(String.valueOf(discount.getValue()));

binding.edtEditDiscountQuantity.setText(String.valueOf(discount.getQuantity()));

binding.edtEditDiscountDescription.setText(discount.getDescription());
    });
}
}

```

#### o **EditDiscountViewModel.java**

```

package com.example.admincse441project.ui.discountmanagement.edit;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.discount.Discount;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.firebase.firestore.DocumentSnapshot;

public class EditDiscountViewModel extends ViewModel {
    private MutableLiveData<Discount> _discount = new
MutableLiveData<>();
    public LiveData<Discount> getDiscount(String discountId) {
        FirebaseUtils.getDiscountsCollection().document(discountId)
            .get().addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document.exists()) {
                        Discount discount =
document.toObject(Discount.class);
                        _discount.setValue(discount);
                    }
                }
            });
        return _discount;
    }

    public void updateDiscount(Discount discount,
OnCompleteListener<Void> onCompleteListener) {

        FirebaseUtils.updateDiscount(discount).addOnCompleteListener(onCompleteListener);
    }
}

```

#### o **AddDiscountActivity.java**

```

package com.example.admincse441project.ui.discountmanagement.add;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;

```

```

import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.bumptech.glide.Glide;
import com.example.admncse441project.R;
import com.example.admncse441project.data.model.discount.Discount;
import
com.example.admncse441project.databinding.ActivityAddDiscountBinding;

public class AddDiscountActivity extends AppCompatActivity {
    private ActivityAddDiscountBinding binding;
    private AddDiscountViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityAddDiscountBinding.inflate(getLayoutInflater());
        viewModel = new
ViewModelProvider(this).get(AddDiscountViewModel.class);
        setContentView(binding.getRoot());

        setupValidation();
        onClickView();
    }

    private void setupValidation() {
        binding.edtDiscountQuantity.addTextChangedListener(new
TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {}

            @Override
            public void afterTextChanged(Editable input) {
                validateIntegerInput(input.toString(),
binding.edtDiscountQuantity, "Quantity");
            }
        });

        binding.edtDiscountValue.addTextChangedListener(new
TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {}

            @Override
            public void afterTextChanged(Editable input) {
                validateIntegerInput(input.toString(),
binding.edtDiscountValue, "Value");
            }
        });
    }
}

```

```

        private void validateIntegerInput(String input, EditText editText,
String fieldName) {
            if (!input.matches("\\d+")) {
                editText.setError(fieldName + " must be a integer.");
            } else {
                editText.setError(null);
            }
        }

        private void onClickView() {
            binding.btnBackAdd.setOnClickListener(v -> finish());
            binding.btnAdd.setOnClickListener(v -> {
                String discountName =
binding.edtDiscountName.getText().toString();
                String discountQuantityStr =
binding.edtDiscountQuantity.getText().toString();
                String discountValueStr =
binding.edtDiscountValue.getText().toString();
                String discountDescription =
binding.edtDiscountDescription.getText().toString();

                if (discountName.isEmpty() || discountQuantityStr.isEmpty()
|| discountValueStr.isEmpty() || discountDescription.isEmpty()) {
                    Toast.makeText(this, "Please fill in all fields",
Toast.LENGTH_SHORT).show();
                    return;
                }

                int discountQuantity =
Integer.parseInt(discountQuantityStr);
                int discountValue = Integer.parseInt(discountValueStr);
                Discount discount = new Discount(null, discountName,
discountValue, discountQuantity, discountDescription);

viewModel.addDiscount(discount).addOnSuccessListener(documentReference -
> {
                    Toast.makeText(this, "Successfully added discount",
Toast.LENGTH_SHORT).show();
                }).addOnFailureListener(e -> {
                    Log.e("AddDiscountActivity", "Error adding discount",
e);
                    Toast.makeText(this, "Add discount failed",
Toast.LENGTH_SHORT).show();
                });
            });
        }
    }
}

```

#### o AddDiscountViewmodel.java

```

package com.example.admncse441project.ui.discountmanagement.add;

import androidx.lifecycle.ViewModel;

import com.example.admncse441project.data.model.discount.Discount;
import com.example.admncse441project.utils.FirebaseUtils;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;

public class AddDiscountViewModel extends ViewModel {
    private final FirebaseUtils firebaseUtils = new FirebaseUtils();
    public Task<DocumentReference> addDiscount(Discount discount) {

```

```

        return
    FirebaseUtils.addDiscount(discount).addOnSuccessListener(documentReference
ce -> {
        String generatedId = documentReference.getId();
        discount.setId(generatedId);
        FirebaseUtils.updateDiscount(discount);
    });
}
}

```

### o **FirebaseUtils**

```

package com.example.admncse441project.utils;

import com.example.admncse441project.data.model.discount.Discount;
import com.example.admncse441project.data.model.ticket.Ticket;
import com.example.admncse441project.data.model.discount.Discount;
import com.example.admncse441project.data.model.account.Account;
import com.google.android.gms.tasks.Task;
import com.example.admncse441project.data.model.showtime.ShowTime;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

public class FirebaseUtils {
    public static FirebaseAuth firebaseAuth =
    FirebaseAuth.getInstance();

    public static String currentUserId() {
        FirebaseUser firebaseAdmin = firebaseAuth.getCurrentUser();
        if (firebaseAdmin != null) {
            return firebaseAdmin.getId();
        } else {
            return null;
        }
    }

    public static DocumentReference currentUserDetail() {
        String adminId = currentUserId();
        if (adminId != null) {
            return
    FirebaseFirestore.getInstance().collection("admin").document(adminId);
        } else {
            throw new IllegalStateException("Admin is not authenticated.
    Cannot access Firestore.");
        }
    }

    public static CollectionReference getShowtimesCollection() {
        return FirebaseFirestore.getInstance().collection("showtimes");
    }

    public static Task<DocumentReference> addShowtime(ShowTime showtime)
    {
        return getShowtimesCollection().add(showtime);
    }

    public static Task<Void> updateShowtime(ShowTime showtime) {
        return
    getShowtimesCollection().document(showtime.getId()).set(showtime);
    }
}

```

```

    }

    public static Task<Void> deleteShowTime(String showId) {
        return getShowtimesCollection().document(showId).delete();
    }

    public static CollectionReference getDiscountsCollection() {
        return FirebaseFirestore.getInstance().collection("discounts");
    }

    public static Task<DocumentReference> addDiscount(Discount discount)
{
    return getDiscountsCollection().add(discount);
}

    public static Task<Void> updateDiscount(Discount discount) {
        return
getDiscountsCollection().document(discount.getId()).set(discount);
    }

    public static Task<Void> deleteDiscount(String discountId) {
        return getDiscountsCollection().document(discountId).delete();
    }
    public static CollectionReference getTicketsCollection() {
        return FirebaseFirestore.getInstance().collection("tickets");
    }

    public static Task<DocumentReference> addTicket(Ticket ticket) {
        return getTicketsCollection().add(ticket);
    }

    public static Task<Void> updateTicket(Ticket ticket) {
        return
getTicketsCollection().document(ticket.getId()).set(ticket);
    }

    public static Task<Void> deleteTicket(String ticketId) {
        return getTicketsCollection().document(ticketId).delete();
    }

    public static Task<DocumentSnapshot> getTicketById(String ticketId)
{
    return getTicketsCollection().document(ticketId).get();
}

}

```

## • App User

### ◦ NowPlayingMovie.java

```
package com.example.cse441_project.data.model.movie;

import com.google.gson.annotations.SerializedName;

import java.util.List;

public class NowPlayingMovie {
    private Dates dates;
    private int page;
    private int totalPages;
    @SerializedName("results") private List<ResultsItem> results;
    private int totalResults;

    public void setDates(Dates dates) {
        this.dates = dates;
    }

    public Dates getDates() {
        return dates;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public int getPage() {
        return page;
    }

    public void setTotalPages(int totalPages) {
        this.totalPages = totalPages;
    }

    public int getTotalPages() {
        return totalPages;
    }

    public void setResults(List<ResultsItem> results) {
        this.results = results;
    }

    public List<ResultsItem> getResults() {
        return results;
    }

    public void setTotalResults(int totalResults) {
        this.totalResults = totalResults;
    }

    public int getTotalResults() {
        return totalResults;
    }
}
```



- **ResultItem.java**

```
package com.example.cse441_project.data.model.movie;

import com.google.gson.annotations.SerializedName;

import java.util.List;

public class ResultsItem {
    private String overview;
    private String originalLanguage;
    private String originalTitle;
    private boolean video;
    @SerializedName("title")
    private String title;
    @SerializedName("genre_ids") private List<Integer> genreIds;
    @SerializedName("poster_path")
    private String posterPath;
    private String backdropPath;
    @SerializedName("release_date") private String releaseDate;
    private Object popularity;
    @SerializedName("vote_average")
    private float voteAverage;
    private int id;
    private boolean adult;
    private int voteCount;
    private List<String> genreNames;

    public List<String> getGenreNames() {
        return genreNames;
    }

    public void setGenreNames(List<String> genreNames) {
        this.genreNames = genreNames;
    }

    public void setOverview(String overview) {
        this.overview = overview;
    }

    public String getOverview() {
        return overview;
    }

    public void setOriginalLanguage(String originalLanguage) {
        this.originalLanguage = originalLanguage;
    }

    public String getOriginalLanguage() {
        return originalLanguage;
    }

    public void setOriginalTitle(String originalTitle) {
        this.originalTitle = originalTitle;
    }

    public String getOriginalTitle() {
        return originalTitle;
    }

    public void setVideo(boolean video) {
        this.video = video;
    }
}
```

```

public boolean isVideo() {
    return video;
}

public void setTitle(String title) {
    this.title = title;
}

public String getTitle() {
    return title;
}

public void setGenreIds(List<Integer> genreIds) {
    this.genreIds = genreIds;
}

public List<Integer> getGenreIds() {
    return genreIds;
}

public void setPosterPath(String posterPath) {
    this.posterPath = posterPath;
}

public String getPosterPath() {
    return posterPath;
}

public void setBackdropPath(String backdropPath) {
    this.backdropPath = backdropPath;
}

public String getBackdropPath() {
    return backdropPath;
}

public void setReleaseDate(String releaseDate) {
    this.releaseDate = releaseDate;
}

public String getReleaseDate() {
    return releaseDate;
}

public void setPopularity(Object popularity) {
    this.popularity = popularity;
}

public Object getPopularity() {
    return popularity;
}

public void setVoteAverage(float voteAverage) {
    this.voteAverage = voteAverage;
}

public float getVoteAverage() {
    return voteAverage;
}

public void setId(int id) {
    this.id = id;
}

```

```

    public int getId() {
        return id;
    }

    public void setAdult(boolean adult) {
        this.adult = adult;
    }

    public boolean isAdult() {
        return adult;
    }

    public void setVoteCount(int voteCount) {
        this.voteCount = voteCount;
    }

    public int getVoteCount() {
        return voteCount;
    }
}

```

#### o **NowPlayingFragment.java**

```

package com.example.cse441_project.ui.home.nowplaying;

import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.data.repository.MovieRepository;
import com.example.cse441_project.data.repository.MovieRepositoryImp;
import com.example.cse441_project.databinding.FragmentNowPlayingBinding;
import com.example.cse441_project.ui.home.moviedetail.MovieCinemaDetailActivity;

import java.util.ArrayList;

public class NowPlayingFragment extends Fragment implements
MovieRecyclerViewAdapter.OnItemClickListener {
    private FragmentNowPlayingBinding binding;
    private MovieRecyclerViewAdapter moviesAdapter;
    private NowPlayingMovieViewModel viewModel;
    private MovieRepository movieRepository;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                                Bundle savedInstanceState) {
        movieRepository = new MovieRepositoryImp();
        NowPlayingViewModelFactory factory = new

```

```

NowPlayingViewModelFactory(movieRepository);
    viewModel = new ViewModelProvider(this,
factory).get(NowPlayingMovieViewModel.class);

    binding = FragmentNowPlayingBinding.inflate(inflater, container,
false);
    setupRecyclerView();
    observeViewModel();
    return binding.getRoot();
}

private void setupRecyclerView() {
    moviesAdapter = new MovieRecyclerViewAdapter(new ArrayList(),
this);
    binding.recyclerViewMovie.setAdapter(moviesAdapter);
    binding.recyclerViewMovie.setLayoutManager(new
LinearLayoutManager(binding.getRoot().getContext()));
    viewModel.loadMovies();
}

private void observeViewModel() {
    viewModel.movies.observe(getViewLifecycleOwner(), movieList -> {
        if (movieList != null) {
            Log.e("NowPlayingFragment", "Received movies: " +
movieList.size());
            moviesAdapter.addMovies(movieList);
        } else {
            Log.e("NowPlayingFragment", "movieList is null");
        }
    });

    viewModel.error.observe(getViewLifecycleOwner(), exception -> {
        if (exception != null) {
            Toast.makeText(requireContext(), exception.getMessage(),
Toast.LENGTH_LONG).show();
        }
    });
}

@Override
public void onItemClick(ResultsItem movie) {
    Log.d("NowPlayingFragment", "Movie ID: " + movie.getId());
    Intent intent = new Intent(requireContext(),
MovieCinemaDetailActivity.class);
    intent.putExtra("MOVIE_ID", movie.getId());
    intent.putExtra("SOURCE", "NowPlayingMovieFragment");
    startActivity(intent);
}
}

```

#### o **NowPlayingMovieViewModel.java**

```

package com.example.cse441_project.ui.home.nowplaying;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.cse441_project.data.model.movie.NowPlayingMovie;
import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.data.repository.MovieRepository;
import com.example.cse441_project.data.repository.MovieRepositoryImp;

```

```

import com.example.cse441_project.data.resource.Result;

import java.util.List;

public class NowPlayingMovieViewModel extends ViewModel {
    private final MovieRepository movieRepository;
    private MutableLiveData<List<ResultsItem>> _movies = new
MutableLiveData<>();
    public LiveData<List<ResultsItem>> movies = _movies;

    private final MutableLiveData<Exception> _error = new
MutableLiveData<>();
    public LiveData<Exception> error = _error;

    public NowPlayingMovieViewModel(MovieRepository movieRepository) {
        this.movieRepository = new MovieRepositoryImp();
    }

    void loadMovies() {
        new Thread(() -> {
            try {
                Result<NowPlayingMovie> result =
movieRepository.getMovie();
                if (result instanceof Result.Success) {
                    _movies.postValue((Result.Success<NowPlayingMovie>)
result).getData().getResults());
                } else if (result instanceof Result.Error) {
                    _movies.postValue(null);
                    _error.postValue((Result.Error<NowPlayingMovie>)
result).getException());
                }
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }).start();
    }
}

```

#### o **NowPlayingMovieViewModelFactory.java**

```

package com.example.cse441_project.ui.home.nowplaying;

import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

import com.example.cse441_project.data.repository.MovieRepository;

public class NowPlayingViewModelFactory implements
ViewModelProvider.Factory {
    private final MovieRepository movieRepository;

    public NowPlayingViewModelFactory(MovieRepository movieRepository) {
        this.movieRepository = movieRepository;
    }

    @NonNull
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass)
    {
        if (modelClass.isAssignableFrom(NowPlayingMovieViewModel.class))
        {
            return (T) new NowPlayingMovieViewModel(movieRepository);
        }
    }
}

```

```

        throw new IllegalArgumentException("Unknown ViewModel class");
    }
}

```

### o **MovieRecyclerViewAdapter.java**

```

package com.example.cse441_project.ui.home.nowplaying;

import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.databinding.ItemShowingMovieBinding;

import java.util.ArrayList;
import java.util.List;

public class MovieRecyclerViewAdapter extends
    RecyclerView.Adapter<MovieRecyclerViewAdapter.ViewHolder> {
    private final ArrayList<ResultsItem> movieList;
    private final OnItemClickListener listener;

    public MovieRecyclerViewAdapter(ArrayList<ResultsItem> movieList,
        OnItemClickListener listener) {
        this.movieList = movieList;
        this.listener = listener;
    }

    @NonNull
    @Override
    public MovieRecyclerViewAdapter.ViewHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        ItemShowingMovieBinding binding =
            ItemShowingMovieBinding.inflate(LayoutInflater.from(parent.getContext()),
                parent, false);
        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull
        MovieRecyclerViewAdapter.ViewHolder holder, int position) {
        holder.bindData(movieList.get(position));
        holder.itemView.setOnClickListener(v -> {
            listener.onItemClick(movieList.get(position));
        });
    }

    @Override
    public int getItemCount() {
        return movieList.size();
    }

    public void addMovies(List<ResultsItem> newMovies) {
        int startPosition = movieList.size();
        movieList.addAll(newMovies);
        notifyDataSetChanged();
    }
}

```

```

    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final ItemShowingMovieBinding binding;
        public ViewHolder(ItemShowingMovieBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }
        public void bindData(ResultsItem resultsItem) {
            String postUrl = "https://image.tmbd.org/t/p/w500" +
resultsItem.getPosterPath();

Glide.with(binding.getRoot().getContext()).load(postUrl).into(binding.im
gPoster);

            binding.txtTitle.setText(resultsItem.getTitle());

            int voteAverage = Math.round(resultsItem.getVoteAverage());
            int starCount = (int) Math.floor(voteAverage / 1.5);
            StringBuilder stars = new StringBuilder();
            for (int i = 0; i < 5; i++) {
                if (i < starCount) {
                    stars.append("★");
                } else {
                    stars.append("☆");
                }
            }
            binding.txtVote.setText(stars.toString());

            binding.txtReleaseDate.setText(resultsItem.getReleaseDate().toString());

            String genres = TextUtils.join(", ",
resultsItem.getGenreNames());
            binding.txtGenre.setText(genres);
        }
    }

    public interface OnItemClickListener{
        void onItemClick(ResultsItem movie);
    }
}

```

### o ComingSoonFragment.java

```

package com.example.cse441_project.ui.home.comingsoon;

import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.data.repository.MovieRepository;
import com.example.cse441_project.data.repository.MovieRepositoryImp;
import com.example.cse441_project.databinding.FragmentComingSoonBinding;

```

```

import
com.example.cse441_project.ui.home.moviedetail.MovieCinemaDetailActivity
;

import java.util.ArrayList;

public class ComingSoonFragment extends Fragment implements
ComingSoonRecyclerViewAdapter.OnItemClickListener {
    private ComingSoonRecyclerViewAdapter moviesAdapter;
    private ComingSoonMovieViewModel viewModel;
    private MovieRepository movieRepository;
    private FragmentComingSoonBinding binding;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                            Bundle savedInstanceState) {
        movieRepository = new MovieRepositoryImp();
        ComingSoonMovieViewModelFactory factory = new
ComingSoonMovieViewModelFactory(movieRepository);
        viewModel = new ViewModelProvider(this,
factory).get(ComingSoonMovieViewModel.class);
        binding = FragmentComingSoonBinding.inflate(inflater, container,
false);
        setupRecyclerView();
        observeViewModel();
        return binding.getRoot();
    }

    private void observeViewModel() {
        viewModel.movies.observe(getViewLifecycleOwner(), movieList -> {
            if (movieList != null) {
                Log.e("ComingSoonFragment", "Received movies: " +
movieList.size());
                moviesAdapter.addMovies(movieList);
            } else {
                Log.e("ComingSoonFragment", "movieList is null");
            }
        });

        viewModel.error.observe(getViewLifecycleOwner(), exception -> {
            if (exception != null) {
                Toast.makeText(requireContext(), exception.getMessage(),
Toast.LENGTH_LONG).show();
            }
        });
    }

    private void setupRecyclerView() {
        moviesAdapter = new ComingSoonRecyclerViewAdapter(new
ArrayList(), this);
        binding.recyclerView.setAdapter(moviesAdapter);
        binding.recyclerView.setLayoutManager(new
LinearLayoutManager(binding.getRoot().getContext()));
        viewModel.loadMovies();
    }

    @Override
    public void onItemClick(ResultsItem movie) {
        Log.d("ComingSoonFragment", "Movie ID: " + movie.getId());
        Intent intent = new Intent(requireContext(),
MovieCinemaDetailActivity.class);
    }

```



```

        intent.putExtra("MOVIE_ID", movie.getId());
        intent.putExtra("SOURCE", "ComingSoonFragment");
        startActivity(intent);
    }
}

```

### o **ComingSoonMovieViewModel.java**

```

package com.example.cse441_project.ui.home.comingsoon;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.cse441_project.data.model.movie.NowPlayingMovie;
import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.data.repository.MovieRepository;
import com.example.cse441_project.data.repository.MovieRepositoryImp;
import com.example.cse441_project.data.resource.Result;

import java.util.List;

public class ComingSoonMovieViewModel extends ViewModel {
    private final MovieRepository movieRepository;
    private MutableLiveData<List<ResultsItem>> _movies = new
MutableLiveData<>();
    public LiveData<List<ResultsItem>> movies = _movies;

    private final MutableLiveData<Exception> _error = new
MutableLiveData<>();
    public LiveData<Exception> error = _error;

    public ComingSoonMovieViewModel(MovieRepository movieRepository) {
        this.movieRepository = new MovieRepositoryImp();
    }

    void loadMovies() {
        new Thread(() -> {
            try {
                Result<NowPlayingMovie> result =
movieRepository.getUpComingMovie();
                if (result instanceof Result.Success) {
                    _movies.postValue(((Result.Success<NowPlayingMovie>)
result).getData().getResults());
                } else if (result instanceof Result.Error) {
                    _movies.postValue(null);
                    _error.postValue(((Result.Error<NowPlayingMovie>)
result).getException());
                }
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }).start();
    }
}

```

### o **ComingSoonMovieViewModelFactory.java**

```

package com.example.cse441_project.ui.home.comingsoon;

import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

import com.example.cse441_project.data.repository.MovieRepository;

```

```

public class ComingSoonMovieViewModelFactory implements
ViewModelProvider.Factory {
    private final MovieRepository movieRepository;

    public ComingSoonMovieViewModelFactory(MovieRepository
movieRepository) {
        this.movieRepository = movieRepository;
    }

    @NonNull
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass)
{
        if (modelClass.isAssignableFrom(ComingSoonMovieViewModel.class))
        {
            return (T) new ComingSoonMovieViewModel(movieRepository);
        }
        throw new IllegalArgumentException("Unknown ViewModel class");
    }
}

```

#### o **ComingSoonRecyclerViewAdapter.java**

```

package com.example.cse441_project.ui.home.comingsoon;

import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.cse441_project.data.model.movie.ResultsItem;
import
com.example.cse441_project.databinding.ItemComingSoonMovieBinding;

import java.util.ArrayList;
import java.util.List;

public class ComingSoonRecyclerViewAdapter extends
RecyclerView.Adapter<ComingSoonRecyclerViewAdapter.ViewHolder> {
    private final ArrayList<ResultsItem> movieList;
    private final ComingSoonRecyclerViewAdapter.OnItemClickListener
listener;

    public ComingSoonRecyclerViewAdapter(ArrayList<ResultsItem>
movieList, ComingSoonRecyclerViewAdapter.OnItemClickListener listener) {
        this.movieList = movieList;
        this.listener = listener;
    }

    @NonNull
    @Override
    public ComingSoonRecyclerViewAdapter.ViewHolder
onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        ItemComingSoonMovieBinding binding =
ItemComingSoonMovieBinding.inflate(LayoutInflater.from(parent.getContext
()), parent, false);
        return new ComingSoonRecyclerViewAdapter.ViewHolder(binding);
    }
}

```

```

@Override
public void onBindViewHolder(@NonNull
ComingSoonRecyclerViewAdapter.ViewHolder holder, int position) {
    holder.bindData(movieList.get(position));
    holder.itemView.setOnClickListener(v -> {
        listener.onClick(movieList.get(position));
    });
}

@Override
public int getItemCount() {
    return movieList.size();
}

public void addMovies(List<ResultsItem> newMovies){
    int startPosition = movieList.size();
    movieList.addAll(newMovies);
    notifyDataSetChanged();
}

public class ViewHolder extends RecyclerView.ViewHolder {
    private final ItemComingSoonMovieBinding binding;
    public ViewHolder(ItemComingSoonMovieBinding binding) {
        super(binding.getRoot());
        this.binding = binding;
    }
    public void bindData(ResultsItem resultsItem){
        String postUrl = "https://image.tmdb.org/t/p/w500" +
resultsItem.getPosterPath();

Glide.with(binding.getRoot().getContext()).load(postUrl).into(binding.im
gPoster);

        binding.txtTitle.setText(resultsItem.getTitle());

        int voteAverage = Math.round(resultsItem.getVoteAverage());
        int starCount = (int) Math.floor(voteAverage / 1.5);
        StringBuilder stars = new StringBuilder();
        for (int i = 0; i < 5; i++) {
            if (i < starCount) {
                stars.append("★");
            } else {
                stars.append("☆");
            }
        }
        binding.txtVote.setText(stars.toString());

        binding.txtReleaseDate.setText(resultsItem.getReleaseDate().toString());

        String genres = TextUtils.join(", ",
resultsItem.getGenreNames());
        binding.txtGenre.setText(genres);
    }
}

public interface OnItemClickListener{
    void onItemClick(ResultsItem movie);
}
}

```

- **GenreMovie.java**

```
package com.example.cse441_project.data.model.genre;

import java.util.List;

public class GenreMovie{
    private List<GenresItem> genres;

    public List<GenresItem> getGenres(){
        return genres;
    }
}
```

- **GenresItem.java**

```
package com.example.cse441_project.data.model.genre;

public class GenresItem{
    private String name;
    private int id;

    public String getName(){
        return name;
    }

    public int getId(){
        return id;
    }
}
```

- **MovieDetail.java**

```
package com.example.cse441_project.data.model.moviedetail;

import com.google.gson.annotations.SerializedName;

import java.util.List;

public class MovieDetail{
    private String originalLanguage;
    private String imdbId;
    private boolean video;
    @SerializedName("title")private String title;
    private String backdropPath;
    private int revenue;
    private List<GenresItem> genres;
    private Object popularity;
    private List<ProductionCountriesItem> productionCountries;
    @SerializedName("id") private int id;
    @SerializedName("vote_count")private int voteCount;
    private int budget;
    @SerializedName("overview")private String overview;
    private String originalTitle;
    @SerializedName("runtime") private int runtime;
    private String posterPath;
    private List<String> originCountry;
    private List<SpokenLanguagesItem> spokenLanguages;
    private List<ProductionCompaniesItem> productionCompanies;
    @SerializedName("release_date")private String releaseDate;
    @SerializedName("vote_average")private float voteAverage;
    private Object belongsToCollection;
    private String tagline;
    @SerializedName("adult") private boolean adult;
    private String homepage;
}
```

```

private String status;

public void setOriginalLanguage(String originalLanguage) {
    this.originalLanguage = originalLanguage;
}

public String getOriginalLanguage() {
    return originalLanguage;
}

public void setImdbId(String imdbId) {
    this.imdbId = imdbId;
}

public String getImdbId() {
    return imdbId;
}

public void setVideo(boolean video) {
    this.video = video;
}

public boolean isVideo() {
    return video;
}

public void setTitle(String title) {
    this.title = title;
}

public String getTitle() {
    return title;
}

public void setBackdropPath(String backdropPath) {
    this.backdropPath = backdropPath;
}

public String getBackdropPath() {
    return backdropPath;
}

public void setRevenue(int revenue) {
    this.revenue = revenue;
}

public int getRevenue() {
    return revenue;
}

public void setGenres(List<GenresItem> genres) {
    this.genres = genres;
}

public List<GenresItem> getGenres() {
    return genres;
}

public void setPopularity(Object popularity) {
    this.popularity = popularity;
}

public Object getPopularity() {

```

```

        return popularity;
    }

    public void setProductionCountries(List<ProductionCountriesItem>
productionCountries){
        this.productionCountries = productionCountries;
    }

    public List<ProductionCountriesItem> getProductionCountries(){
        return productionCountries;
    }

    public void setId(int id){
        this.id = id;
    }

    public int getId(){
        return id;
    }

    public void setVoteCount(int voteCount){
        this.voteCount = voteCount;
    }

    public int getVoteCount(){
        return voteCount;
    }

    public void setBudget(int budget){
        this.budget = budget;
    }

    public int getBudget(){
        return budget;
    }

    public void setOverview(String overview){
        this.overview = overview;
    }

    public String getOverview(){
        return overview;
    }

    public void setOriginalTitle(String originalTitle){
        this.originalTitle = originalTitle;
    }

    public String getOriginalTitle(){
        return originalTitle;
    }

    public void setRuntime(int runtime){
        this.runtime = runtime;
    }

    public int getRuntime(){
        return runtime;
    }

    public void setPosterPath(String posterPath){
        this.posterPath = posterPath;
    }

```

```

    public String getPosterPath() {
        return posterPath;
    }

    public void setOriginCountry(List<String> originCountry) {
        this.originCountry = originCountry;
    }

    public List<String> getOriginCountry() {
        return originCountry;
    }

    public void setSpokenLanguages(List<SpokenLanguagesItem>
spokenLanguages) {
        this.spokenLanguages = spokenLanguages;
    }

    public List<SpokenLanguagesItem> getSpokenLanguages() {
        return spokenLanguages;
    }

    public void setProductionCompanies(List<ProductionCompaniesItem>
productionCompanies) {
        this.productionCompanies = productionCompanies;
    }

    public List<ProductionCompaniesItem> getProductionCompanies() {
        return productionCompanies;
    }

    public void setReleaseDate(String releaseDate) {
        this.releaseDate = releaseDate;
    }

    public String getReleaseDate() {
        return releaseDate;
    }

    public void setVoteAverage(float voteAverage) {
        this.voteAverage = voteAverage;
    }

    public Object getVoteAverage() {
        return voteAverage;
    }

    public void setBelongsToCollection(Object belongsToCollection) {
        this.belongsToCollection = belongsToCollection;
    }

    public Object getBelongsToCollection() {
        return belongsToCollection;
    }

    public void setTagline(String tagline) {
        this.tagline = tagline;
    }

    public String getTagline() {
        return tagline;
    }

```

```

    public void setAdult(boolean adult) {
        this.adult = adult;
    }

    public boolean isAdult() {
        return adult;
    }

    public void setHomepage(String homepage) {
        this.homepage = homepage;
    }

    public String getHomepage() {
        return homepage;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getStatus() {
        return status;
    }
}

```

#### o **MovieTrailer.java**

```

package com.example.cse441_project.data.model.movietrailer;

import java.util.List;

public class MovieTrailer{
    private int id;
    private List<MovieTrailerItem> results;

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setResults(List<MovieTrailerItem> results) {
        this.results = results;
    }

    public List<MovieTrailerItem> getResults() {
        return results;
    }
}

```

#### o **MovieTrailerItem.java**

```

package com.example.cse441_project.data.model.movietrailer;

import com.google.gson.annotations.SerializedName;

public class MovieTrailerItem {
    private String site;
    private int size;
    private String iso31661;
    private String name;
    private boolean official;
    @SerializedName("id") private String id;
}

```



```

private String type;
private String publishedAt;
private String iso6391;
@SerializedName("key") private String key;

public void setSite(String site){
    this.site = site;
}

public String getSite(){
    return site;
}

public void setSize(int size){
    this.size = size;
}

public int getSize(){
    return size;
}

public void setIso31661(String iso31661){
    this.iso31661 = iso31661;
}

public String getIso31661(){
    return iso31661;
}

public void setName(String name){
    this.name = name;
}

public String getName(){
    return name;
}

public void setOfficial(boolean official){
    this.official = official;
}

public boolean isOfficial(){
    return official;
}

public void setId(String id){
    this.id = id;
}

public String getId(){
    return id;
}

public void setType(String type){
    this.type = type;
}

public String getType(){
    return type;
}

public void setPublishedAt(String publishedAt){
    this.publishedAt = publishedAt;
}

```

```

    }

    public String getPublishedAt() {
        return publishedAt;
    }

    public void setIso6391(String iso6391) {
        this.iso6391 = iso6391;
    }

    public String getIso6391() {
        return iso6391;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public String getKey() {
        return key;
    }
}

```

#### o **MovieCinemaDetailActivity.java**

```

package com.example.cse441_project.ui.home.moviedetail;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;

import com.example.cse441_project.data.model.moviedetail.MovieDetail;
import com.example.cse441_project.data.model.movietrailer.MovieTrailerItem;
import com.example.cse441_project.databinding.ActivityMovieDetailBinding;
import com.example.cse441_project.ui.bookticket.showscreen.ChooseDateAndTimeActivity;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.YouTubePlayer;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.listeners.AbstractYouTubePlayerListener;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView;

public class MovieCinemaDetailActivity extends AppCompatActivity {
    private ActivityMovieDetailBinding binding;
    private DetailMovieViewModel viewModel;
    private int id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        binding =
ActivityMovieDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        viewModel = new
ViewModelProvider(this).get(DetailMovieViewModel.class);
        Intent intent = getIntent();
        id = intent.getIntExtra("MOVIE_ID", -1);
        String source = intent.getStringExtra("SOURCE");

        if (id != -1) {
            observeViewModel();
            viewModel.loadMovie(id);

binding.btnBookTicket.setVisibility("NowPlayingMovieFragment".equals(sou
rce) ? View.VISIBLE : View.GONE);
        } else {
            Toast.makeText(this, "Movie is not available",
Toast.LENGTH_SHORT).show();
        }

        private void observeViewModel() {
            viewModel.movieDetail.observe(this, detailMovie -> {
                if (detailMovie != null) {
                    bindData(detailMovie);
                }
            });

            viewModel.movieTrailer.observe(this, movieTrailers -> {
                if (movieTrailers != null && !movieTrailers.isEmpty()) {
                    MovieTrailerItem firstTrailer = movieTrailers.get(0);
                    setupPlayer(firstTrailer);
                } else {
                    Toast.makeText(this, "No trailer available",
Toast.LENGTH_SHORT).show();
                }
            });

            viewModel.error.observe(this, error -> {
                if (error != null) {
                    Toast.makeText(this, "Error: " + error.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            });
        }

        private void bindData(MovieDetail movieDetail) {
            binding.txtName.setText(movieDetail.getTitle());
            binding.txtVoteAvarage.setText("Vote average: " +
movieDetail.getVoteAverage());
            binding.txtTotalVote.setText(movieDetail.getVoteCount() + "
votes");
            binding.txtDate.setText(movieDetail.getReleaseDate());
            binding.txtAge.setText(movieDetail.isAdult() ? "18" : "16");
            binding.txtDuration.setText(movieDetail.getRuntime() + " min");
            binding.txtDescription.setText(movieDetail.getOverview());
            binding.btnBookTicket.setOnClickListener(v -> {
                Log.d("MovieCinemaDetailActivity", "Received Movie ID: " +
id);

```

```

        Intent intent = new Intent(this,
ChooseDateAndTimeActivity.class);
        intent.putExtra("MOVIE_ID", id);
        intent.putExtra("MOVIE_NAME", movieDetail.getTitle());
        startActivity(intent);
    });
}

private void setupPlayer(MovieTrailerItem movieTrailerItem) {
    YouTubePlayerView youTubePlayerView = binding.youtubePlayerView;
    getLifecycle().addObserver(youTubePlayerView);

    youTubePlayerView.addYouTubePlayerListener(new
AbstractYouTubePlayerListener() {
        @Override
        public void onReady(@NonNull YouTubePlayer youTubePlayer) {
            String videoId = movieTrailerItem.getKey();
            youTubePlayer.loadVideo(videoId, 0);
        }
    });
}
}

```

#### o **DetailMovieViewModel.java**

```

package com.example.cse441_project.ui.home.moviedetail;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.cse441_project.data.model.moviedetail.MovieDetail;
import com.example.cse441_project.data.model.movietrailer.MovieTrailer;
import
com.example.cse441_project.data.model.movietrailer.MovieTrailerItem;
import com.example.cse441_project.data.repository.MovieRepository;
import com.example.cse441_project.data.repository.MovieRepositoryImp;
import com.example.cse441_project.data.resource.Result;

import java.util.List;

public class DetailMovieViewModel extends ViewModel {
    private MovieRepository movieRepository;

    private MutableLiveData<MovieDetail> _movieDetail = new
MutableLiveData<>();
    public LiveData<MovieDetail> movieDetail = _movieDetail;

    private MutableLiveData<List<MovieTrailerItem>> _movieTrailer = new
MutableLiveData<>();
    public LiveData<List<MovieTrailerItem>> movieTrailer =
_movieTrailer;

    private MutableLiveData<Exception> _error = new MutableLiveData<>();
    public LiveData<Exception> error = _error;

    public DetailMovieViewModel() {
        this.movieRepository = new MovieRepositoryImp();
    }

    public void loadMovie(int id) {
        new Thread(() -> {

```

```

        try {
            Result<MovieDetail> detailMovie =
movieRepository.getDetailMovie(id);
            if(detailMovie instanceof Result.Success){

_movieDetail.postValue(((Result.Success<MovieDetail>)
detailMovie).getData());
            } else if (detailMovie instanceof Result.Error) {
                _movieDetail.postValue(null);
                _error.postValue(((Result.Error< MovieDetail>)
detailMovie).getException());
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

    }).start();

    new Thread(() -> {
        try {
            Result<MovieTrailer> movieTrailerResult =
movieRepository.getTrailerMovie(id);
            if (movieTrailerResult instanceof Result.Success) {
                List<MovieTrailerItem> trailers =
((Result.Success<MovieTrailer>)
movieTrailerResult).getData().getResults();
                _movieTrailer.postValue(trailers);
            } else if (movieTrailerResult instanceof Result.Error) {
                _movieTrailer.postValue(null);
                _error.postValue(((Result.Error<MovieTrailer>)
movieTrailerResult).getException());
            }
        } catch (Exception e) {
            _error.postValue(e);
        }
    }).start();
    }
}

```

#### o Showtime.java

```
package com.example.cse441_project.data.model.showtime;
```

```

public class ShowTime {
    private String id;
    private String name;
    private String availableSeat;
    private String unavailableSeat;
    private String startTime;
    private String endTime;
    private String date;
    private String idMovie;
    private String nameCinema;

    public ShowTime() {
    }

    public ShowTime(String id, String name, String availableSeat, String
unavailableSeat, String startTime, String endTime, String date, String
idMovie) {
        this.id = id;
        this.name = name;
        this.availableSeat = availableSeat;
    }
}

```

```

        this.unavailableSeat = unavailableSeat;
        this.startTime = startTime;
        this.endTime = endTime;
        this.date = date;
        this.idMovie = idMovie;
    }

    public String getNameCinema() {
        return nameCinema;
    }

    public void setNameCinema(String nameMovie) {
        this.nameCinema = nameMovie;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAvailableSeat() {
        return availableSeat;
    }

    public void setAvailableSeat(String availableSeat) {
        this.availableSeat = availableSeat;
    }

    public String getUnavailableSeat() {
        return unavailableSeat;
    }

    public void setUnavailableSeat(String unavailableSeat) {
        this.unavailableSeat = unavailableSeat;
    }

    public String getStartTime() {
        return startTime;
    }

    public void setStartTime(String startTime) {
        this.startTime = startTime;
    }

    public String getEndTime() {
        return endTime;
    }

    public void setEndTime(String endTime) {
        this.endTime = endTime;
    }

    public String getDate() {

```

```

        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getIdMovie() {
        return idMovie;
    }

    public void setIdMovie(String idMovie) {
        this.idMovie = idMovie;
    }
}

```

#### o ChooseDateAndTimeActivity.java

```

package com.example.cse441_project.ui.bookticket.showscreen;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.R;
import com.example.cse441_project.data.model.showtime.ShowTime;
import com.example.cse441_project.databinding.ActivityChooseDateAndTimeBinding;
import com.example.cse441_project.ui.bookticket.chooseseat.ChooseSeatActivity;
import com.example.cse441_project.utils.FirebaseUtils;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class ChooseDateAndTimeActivity extends AppCompatActivity {

    private ActivityChooseDateAndTimeBinding binding;
    private ChooseDateAndTimeViewModel viewModel;
    private ChooseDateAdapter chooseDateAdapter;
    private ChooseScreenTimeAdapter chooseScreenTimeAdapter;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding =
ActivityChooseDateAndTimeBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new

```

```

ViewModelProvider(this).get(ChooseDateAndTimeViewModel.class);
Intent intent = getIntent();
int movieId = intent.getIntExtra("MOVIE_ID", -1);

binding.txtNameMovie.setText(intent.getStringExtra("MOVIE_NAME"));
if (movieId != -1) {

FirebaseUtils.getShowtimeIdmovie(String.valueOf(movieId)).get().addOnCom
pleteListener(task -> {
    if (task.isSuccessful() && task.getResult() != null &&
!task.getResult().isEmpty()) {
        viewModel.loadShowtime(String.valueOf(movieId));
    } else {
        Toast.makeText(this, "Movie is not available",
Toast.LENGTH_SHORT).show();
    }
});
} else {
    Toast.makeText(this, "Invalid Movie ID",
Toast.LENGTH_SHORT).show();
}

    setupRecyclerViews();
    observeViewModel();
    onClickView();
}

private void onClickView() {
    binding.imgPoster.setOnClickListener(v -> {
        finish();
    });
}

private void setupRecyclerViews() {
    chooseDateAdapter = new ChooseDateAdapter(this::onDateClick);
    binding.rcvDate.setLayoutManager(new LinearLayoutManager(this,
RecyclerView.HORIZONTAL, false));
    binding.rcvDate.setAdapter(chooseDateAdapter);

    chooseScreenTimeAdapter = new
ChooseScreenTimeAdapter(this::onShowTimeClick);
    binding.rcvTimeAndScreen.setLayoutManager(new
GridLayoutManager(this, 2));
    binding.rcvTimeAndScreen.setAdapter(chooseScreenTimeAdapter);
}

private void observeViewModel() {
    viewModel.showTimeList.observe(this, showTimeList -> {
        Log.d("ChooseDateAndTimeActivity", "Showtime List Observed:
" + showTimeList);
        chooseScreenTimeAdapter.submitList(showTimeList);
        chooseDateAdapter.submitList(showTimeList);
    });
}

private void onDateClick(String date) {
    SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy");
    Date selectedDate;
    try {
        selectedDate = dateFormat.parse(date);
    } catch (ParseException e) {

```



```

        e.printStackTrace();
        return;
    }

    if (viewModel.showTimeList.getValue() != null) {
        List<ShowTime> filteredShowTimes = new ArrayList<>();
        for (ShowTime showTime : viewModel.showTimeList.getValue())
        {
            Log.d("FilteringShowTimes", "ShowTime Date: " +
showTime.getDate());
            try {
                Date showTimeDate =
dateFormat.parse(showTime.getDate());
                if (showTimeDate != null &&
showTimeDate.equals(selectedDate)) {

                    filteredShowTimes.add(showTime);
                }
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }

        Log.d("FilteredShowTimes", "Filtered ShowTimes: " +
filteredShowTimes);
        chooseScreenTimeAdapter.submitList(filteredShowTimes);
    } else {
        Log.d("ShowTimeList", "showTimeList is null or empty");
    }
}

private void onShowTimeClick(ShowTime showTime) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Confirm");
    builder.setMessage("Are you sure want to choose this showtime?
\n" + showTime.getNameCinema() + " \n" + showTime.getStartTime() +
        " - " + showTime.getEndTime() + showTime.getDate() + "
\n " );
    builder.setPositiveButton("Ok", (dialog, which) -> {
        Intent intent = new Intent(this, ChooseSeatActivity.class);
        intent.putExtra("SHOWTIME_ID", showTime.getId());
        intent.putExtra("SHOWTIME_AVAILABLE_SEAT",
showTime.getAvailableSeat());
        intent.putExtra("SHOWTIME_MOVIE", showTime.getName());
        intent.putExtra("SHOWTIME_START", showTime.getStartTime());
        intent.putExtra("SHOWTIME_END", showTime.getEndTime());

        startActivity(intent);
        setResult(RESULT_OK, intent);

        finish();
    });
    builder.setNegativeButton("Cancel", (dialog, which) ->
dialog.dismiss());
    builder.show();
}
}

```

### o ChooseDateTimeViewModel.java

```
package com.example.cse441_project.ui.bookticket.showscreen;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.cse441_project.data.model.showtime.ShowTime;
import com.example.cse441_project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class ChooseDateAndTimeViewModel extends ViewModel {

    private final MutableLiveData<List<ShowTime>> _showTimeList = new
MutableLiveData<>();
    public LiveData<List<ShowTime>> showTimeList = _showTimeList;
    public final MutableLiveData<Exception> error = new
MutableLiveData<>();

    void loadShowtime(String movieId) {
        if (movieId == null || movieId.isEmpty()) {
            error.setValue(new Exception("Invalid Movie ID"));
            return;
        }

        FirebaseUtils.getShowtimeCollection(movieId).get().addOnCompleteListener
(task -> {
            if (task.isSuccessful()) {
                List<ShowTime> showtimeList = new ArrayList<>();
                QuerySnapshot querySnapshot = task.getResult();
                Log.e("ChooseDateAndTimeViewModel", "Showtime List Size:
" + (querySnapshot != null ? querySnapshot.size() : 0));

                if (querySnapshot != null) {
                    for (DocumentSnapshot document :
querySnapshot.getDocuments()) {
                        ShowTime showTime =
document.toObject(ShowTime.class);
                        if (showTime != null) {
                            showtimeList.add(showTime);
                        } else {
                            Log.e("ChooseDateAndTimeViewModel",
"Showtime is null for document: " + document.getId());
                        }
                    }
                    _showTimeList.setValue(showtimeList);
                }
            } else {
                error.setValue(task.getException());
                Log.e("ChooseDateAndTimeViewModel", "Error getting
showtimes: ", task.getException());
            }
        })
    }
}
```

```

    });
}
}

```

### o ChooseScreenTimeAdapter.java

```

package com.example.cse441_project.ui.bookticket.showscreen;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.data.model.showtime.ShowTime;
import com.example.cse441_project.databinding.ItemTimeAndScreenBinding;

import java.util.ArrayList;
import java.util.List;

public class ChooseScreenTimeAdapter extends
RecyclerView.Adapter<ChooseScreenTimeAdapter.ShowTimeViewHolder> {
    private List<ShowTime> showTimeList = new ArrayList<>();
    private final OnShowTimeClickListener onShowTimeClickListener;

    public interface OnShowTimeClickListener {
        void onShowTimeClick(ShowTime showTime);
    }

    public ChooseScreenTimeAdapter(OnShowTimeClickListener
onShowTimeClickListener) {
        this.onShowTimeClickListener = onShowTimeClickListener;
    }

    @NonNull
    @Override
    public ShowTimeViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        ItemTimeAndScreenBinding binding =
ItemTimeAndScreenBinding.inflate(LayoutInflater.from(parent.getContext()),
parent, false);
        return new ShowTimeViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull ShowTimeViewHolder holder, int
position) {
        ShowTime showTime = showTimeList.get(position);
        holder.bind(showTime);
    }

    @Override
    public int getItemCount() {
        return showTimeList.size();
    }

    public void submitList(List<ShowTime> newList) {
        this.showTimeList = newList;
        Log.d("ChooseScreenTimeAdapter", "Dữ liệu trong submitList: " +
newList);
        notifyDataSetChanged();
    }
}

```

```

class ShowTimeViewHolder extends RecyclerView.ViewHolder {
    private final ItemTimeAndScreenBinding binding;

    public ShowTimeViewHolder(ItemTimeAndScreenBinding binding) {
        super(binding.getRoot());
        this.binding = binding;
    }

    public void bind(ShowTime showTime) {;
        binding.txtPlayTime.setText(showTime.getStartTime() + " - "
+ showTime.getEndTime());
        binding.txtScreen.setText(showTime.getNameCinema());
        binding.getRoot().setOnClickListener(v ->
onShowTimeClickListener.onShowTimeClick(showTime));
    }
}

```

#### o ChooseDateAdapter.java

```

package com.example.cse441_project.ui.bookticket.showscreen;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.data.model.showtime.ShowTime;
import com.example.cse441_project.databinding.ItemDateAndTimeBinding;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Comparator;
import java.util.Date;
import java.util.List;
import java.util.Locale;

public class ChooseDateAdapter extends
RecyclerView.Adapter<ChooseDateAdapter.ViewHolder> {
    private List<ShowTime> showTimeList = new ArrayList<>();
    private final OnDateClickListener onDateClickListener;

    public ChooseDateAdapter(OnDateClickListener listener) {
        this.onDateClickListener = listener;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        ItemDateAndTimeBinding binding =
ItemDateAndTimeBinding.inflate(LayoutInflater.from(parent.getContext()),
parent, false);
        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int

```

```

position) {
    ShowTime showTime = showTimeList.get(position);
    holder.bind(showTime);
}

@Override
public int getItemCount() {
    return showTimeList.size();
}

public void submitList(List<ShowTime> newList) {
    showTimeList.clear();
    showTimeList.addAll(newList);
    sortShowTimesByDay();
    Log.d("ChooseDateAdapter", "Dữ liệu trong submitList: " +
newList);
    notifyDataSetChanged();
}

public class ViewHolder extends RecyclerView.ViewHolder {
    private final ItemDateAndTimeBinding binding;

    public ViewHolder(@NonNull ItemDateAndTimeBinding binding) {
        super(binding.getRoot());
        this.binding = binding;

        binding.getRoot().setOnClickListener(v -> {
            String date =
binding.txtDayOfMonth.getText().toString();
            onClickListener.onDateClick(date);
        });
    }

    public void bind>ShowTime showTime) {
        SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy",
Locale.US);

        try {
            Date date = format.parse(showTime.getDate());

            Calendar calendar = Calendar.getInstance();
            calendar.setTime(date);
            int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
            String[] daysOfWeek = new String[]{"Sun", "Mon", "Tue",
"Wed", "Thu", "Fri", "Sat"};
            String dayName = daysOfWeek[dayOfWeek - 1];

            binding.txtDayOfMonth.setText(showTime.getDate());
            binding.txtDayOfWeek.setText(dayName);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }

    public void sortShowTimesByDay() {
        showTimeList.sort(new Comparator<ShowTime>() {
            @Override
            public int compare>ShowTime o1, ShowTime o2) {
                SimpleDateFormat format = new
SimpleDateFormat("dd/MM/yyyy", Locale.US);
                try {
                    Date date1 = format.parse(o1.getDate());

```

```

        Date date2 = format.parse(o2.getDate());
        Calendar calendar1 = Calendar.getInstance();
        calendar1.setTime(date1);
        int dayOfWeek1 =
calendar1.get(Calendar.DAY_OF_WEEK);
        Calendar calendar2 = Calendar.getInstance();
        calendar2.setTime(date2);
        int dayOfWeek2 =
calendar2.get(Calendar.DAY_OF_WEEK);

        return Integer.compare(dayOfWeek1 == 1 ? 7 :
dayOfWeek1 - 1, dayOfWeek2 == 1 ? 7 : dayOfWeek2 - 1);
    } catch (ParseException e) {
        e.printStackTrace();
        return 0;
    }
}
});
}
public interface OnDateClickListener {
    void onDateClick(String date);
}
}

```

#### o **FirestoreUtils.java**

```

package com.example.cse441_project.utils;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;

public class FirestoreUtils {
    public static FirebaseAuth firebaseAuth =
FirebaseAuth.getInstance();
    private static final FirebaseFirestore firestore =
FirebaseFirestore.getInstance();

    public static String currentUserId() {
        FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
        if (firebaseUser != null) {
            return firebaseUser.getId();
        } else {
            return null;
        }
    }

    public static DocumentReference currentUserDetail() {
        String userId = currentUserId();
        if (userId != null) {
            return
Firestore.getInstance().collection("users").document(userId);
        } else {
            throw new IllegalStateException("User is not authenticated.
Cannot access Firestore.");
        }
    }
}

```

```

        public static CollectionReference getShowtimeCollection(String
movieId) {
            return FirebaseFirestore.getInstance().collection("showtimes");
        }

        public static Query getShowtimeIdmovie(String idMovie) {
            return
FirebaseFirestore.getInstance().collection("showtimes").whereEqualTo("id
Movie", idMovie);
        }

        public static Query getTicketsByShowtimeAndSeat(String showtimeId) {
            return FirebaseFirestore.getInstance()
                .collection("tickets")
                .whereEqualTo("showtimeId", showtimeId)
                .whereNotEqualTo("seat", "");
        }
    }
}

```

#### o **MovieRepositoryInterface.java**

```

package com.example.cse441_project.data.repository;

import com.example.cse441_project.data.model.movie.NowPlayingMovie;
import com.example.cse441_project.data.model.movedetail.MovieDetail;
import com.example.cse441_project.data.model.movietrailer.MovieTrailer;
import
com.example.cse441_project.data.model.movietrailer.MovieTrailerItem;
import com.example.cse441_project.data.resource.Result;

public interface MovieRepository {
    Result<NowPlayingMovie> getMovie() throws Exception;
    Result<NowPlayingMovie> getUpComingMovie() throws Exception;
    Result<MovieDetail> getDetailMovie(int id) throws Exception;
    Result<MovieTrailer> getTrailerMovie(int id) throws Exception;
}

```

#### o **MovieRepositoryImp.java**

```

package com.example.cse441_project.data.repository;

import android.util.Log;

import com.example.cse441_project.data.model.genre.GenreMovie;
import com.example.cse441_project.data.model.genre.GenresItem;
import com.example.cse441_project.data.model.movie.NowPlayingMovie;
import com.example.cse441_project.data.model.movie.ResultsItem;
import com.example.cse441_project.data.model.movedetail.MovieDetail;
import com.example.cse441_project.data.model.movietrailer.MovieTrailer;
import
com.example.cse441_project.data.model.movietrailer.MovieTrailerItem;
import com.example.cse441_project.data.resource.MovieApi;
import com.example.cse441_project.data.resource.Result;
import com.example.cse441_project.data.resource.RetrofitHelper;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import retrofit2.Response;

public class MovieRepositoryImp implements MovieRepository{

```

```

private final MovieApi movieApi;
private Map<Integer, String> genreMap = new HashMap<>();

public MovieRepositoryImp(){
    movieApi = RetrofitHelper.getInstance().create(MovieApi.class);
}

@Override
public Result<NowPlayingMovie> getMovie() throws Exception {
    try {
        Response<GenreMovie> genreMovieResponse =
movieApi.getGenres().execute();
        if (genreMovieResponse.isSuccessful() &&
genreMovieResponse.body() != null) {
            for (GenresItem genreMovie :
genreMovieResponse.body().getGenres()) {
                genreMap.put(genreMovie.getId(),
genreMovie.getName());
            }
        } else {
            return new Result.Error<>(new Exception("Failed to fetch
Genre"));
        }

        Response<NowPlayingMovie> movieResponse =
movieApi.getMovies().execute();
        if (movieResponse.isSuccessful() && movieResponse.body() !=
null) {
            List<ResultsItem> movieItems =
movieResponse.body().getResults();
            if (movieItems != null) {
                for (ResultsItem movie : movieItems) {
                    List<Integer> genreIds = movie.getGenreIds();
                    if (genreIds != null) {
                        List<String> genreNames = new ArrayList<>();
                        for (Integer genreId : genreIds) {
                            String genreName =
genreMap.get(genreId);

                            if (genreName != null) {
                                genreNames.add(genreName);
                            }
                        }
                        if (genreNames.size() > 3) {
                            genreNames = genreNames.subList(0, 3);
                        }
                        movie.setGenreNames(genreNames);
                    } else {
                        Log.e("MovieRepositoryImp", "Genre IDs are
null for movie: " + movie.getTitle());
                    }
                }
            } else {
                Log.e("MovieRepositoryImp", "Movie results are
null");
            }

            Log.d("MovieRepositoryImp", "Success: " +
movieResponse.body().getResults());
            return new Result.Success<>(movieResponse.body());
        } else {
            Log.e("MovieRepositoryImp", "Error fetching movies: " +
movieResponse.message());
            return new Result.Error<>(new

```



```

Exception(movieResponse.message()));
    }
    } catch (IOException e) {
        return new Result.Error<>(e);
    }
}

@Override
public Result<NowPlayingMovie> getUpComingMovie() throws Exception {
    try {
        Response<GenreMovie> genreMovieResponse =
movieApi.getGenres().execute();
        if (genreMovieResponse.isSuccessful() &&
genreMovieResponse.body() != null) {
            for (GenresItem genreMovie :
genreMovieResponse.body().getGenres()) {
                genreMap.put(genreMovie.getId(),
genreMovie.getName());
            }
        } else {
            return new Result.Error<>(new Exception("Failed to fetch
Genre"));
        }

        Response<NowPlayingMovie> movieResponse =
movieApi.getUpComingMovie().execute();
        if (movieResponse.isSuccessful() && movieResponse.body() !=
null) {
            List<ResultsItem> movieItems =
movieResponse.body().getResults();
            if (movieItems != null) {
                for (ResultsItem movie : movieItems) {
                    List<Integer> genreIds = movie.getGenreIds();
                    if (genreIds != null) {
                        List<String> genreNames = new ArrayList<>();
                        for (Integer genreId : genreIds) {
                            String genreName =
genreMap.get(genreId);

                            if (genreName != null) {
                                genreNames.add(genreName);
                            }
                        }
                        if (genreNames.size() > 3) {
                            genreNames = genreNames.subList(0, 3);
                        }
                        movie.setGenreNames(genreNames);
                    } else {
                        Log.e("MovieRepositoryImp", "Genre IDs are
null for movie: " + movie.getTitle());
                    }
                }
            } else {
                Log.e("MovieRepositoryImp", "Movie results are
null");
            }

            Log.d("MovieRepositoryImp", "Success: " +
movieResponse.body().getResults());
            return new Result.Success<>(movieResponse.body());
        } else {
            Log.e("MovieRepositoryImp", "Error fetching movies: " +
movieResponse.message());
            return new Result.Error<>(new

```

```

Exception(movieResponse.message()));
    }
    } catch (IOException e) {
        return new Result.Error<>(e);
    }
}

@Override
public Result<MovieDetail> getDetailMovie(int id) throws Exception {
    try {
        Response<MovieDetail> movieDetailResponse =
movieApi.getDetailMovie(id).execute();
        if(movieDetailResponse.isSuccessful() &&
movieDetailResponse.body() != null){
            MovieDetail movieDetail = movieDetailResponse.body();
            return new Result.Success(movieDetail);
        }else {
            return new Result.Error<>(new Exception("Failed to get
movie details: " + movieDetailResponse.message()));
        }
    } catch (IOException e){
        return new Result.Error<>(e);
    }
}

@Override
public Result<MovieTrailer> getTrailerMovie(int id) throws Exception
{
    try {
        Response<MovieTrailer> movieTrailerResponse =
movieApi.getVideoTrailer(id).execute();

        if (movieTrailerResponse.isSuccessful() &&
movieTrailerResponse.body() != null) {
            MovieTrailer movieTrailer = movieTrailerResponse.body();
            return new Result.Success<>(movieTrailer);
        } else {
            return new Result.Error<>(new Exception("Failed to fetch
trailer: " + movieTrailerResponse.message()));
        }

    } catch (IOException e) {
        return new Result.Error<>(e);
    }
}
}

```

#### o **MovieApiInterface.java**

```

package com.example.cse441_project.data.resource;

import com.example.cse441_project.data.model.genre.GenreMovie;
import com.example.cse441_project.data.model.movie.NowPlayingMovie;
import com.example.cse441_project.data.model.moviedetail.MovieDetail;
import com.example.cse441_project.data.model.movietrailer.MovieTrailer;
import
com.example.cse441_project.data.model.movietrailer.MovieTrailerItem;

import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;

public interface MovieApi {
    @GET("3/movie/now_playing?api_key=ed5fb4eff85f0e634388e9c3f7a86f5a")

```

```

Call<NowPlayingMovie> getMovies();

@GET("3/genre/movie/list?api_key=ed5fb4eff85f0e634388e9c3f7a86f5a")
Call<GenreMovie> getGenres();

@GET("3/movie/upcoming?api_key=ed5fb4eff85f0e634388e9c3f7a86f5a")
Call<NowPlayingMovie> getUpComingMovie();

@GET("/3/movie/{id}?api_key=ed5fb4eff85f0e634388e9c3f7a86f5a")
Call<MovieDetail> getDetailMovie(@Path("id") int id);

@GET("3/movie/{movieId}/videos?api_key=ed5fb4eff85f0e634388e9c3f7a86f5a")
)
    Call<MovieTrailer> getVideoTrailer(@Path("movieId") int movieId);

}

```

### o Result.java

```

package com.example.cse441_project.data.resource;

public class Result<T> {
    public static final class Success<T> extends Result<T>{
        private final T data;

        public Success (T data){
            this.data = data;
        }

        public T getData(){
            return data;
        }
    }

    public static final class Error<T> extends Result<T>{
        private final Exception exception;

        public Error (Exception exception){
            this.exception = exception;
        }

        public Exception getException() {
            return exception;
        }
    }
}

```

- **RetrofitHelper.java**

```
package com.example.cse441_project.data.resource;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitHelper{
    private static final String BASE_URL =
    "https://api.themoviedb.org/";

    public static Retrofit getInstance(){
        return new Retrofit.Builder().baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();
    }
}
```

## b) Phí Văn Đức

### ○ Ticket.java

```
package com.example.cse441_project.data.model.ticket;

public class Ticket {

    private String id;

    private String showtimeId;

    private String seat;

    private String userId;

    private String status;

    public Ticket() {}

    public Ticket(String id, String showtimeId, String seat, String
userId, String status) {

        this.id = id;

        this.showtimeId = showtimeId;

        this.seat = seat;

        this.userId = userId;

        this.status = status;

    }

    public String getId() {

        return id;

    }

    public void setId(String id) {

        this.id = id;

    }

    public String getSeat() {
```

```

        return seat;
    }

    public void setSeat(String seat) {
        this.seat = seat;
    }

    public String getShowtimeId() {
        return showtimeId;
    }

    public void setShowtimeId(String showtimeId) {
        this.showtimeId = showtimeId;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }
}

```

## ○ EditShowTimeActivity.java

```
package com.example.admincse441project.ui.showtimemaganement;

import static
com.example.admincse441project.utils.FirebaseUtils.deleteShowTime;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.Spinner;

import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.lifecycle.ViewModelProvider;

import com.example.admincse441project.R;

import com.example.admincse441project.data.model.showtime.ShowTime;

import com.example.admincse441project.data.model.movie.ResultsItem; //
Thêm import này

import com.example.admincse441project.data.model.ticket.Ticket;

import com.example.admincse441project.data.repository.MovieRepositoryImp;
// Thêm import này

import
com.example.admincse441project.ui.showtimemaganement.NowPlayingMovieViewM
odel; // Thêm import này

import
com.example.admincse441project.ui.showtimemaganement.NowPlayingViewModelF
actory; // Thêm import này
```

```

import
com.example.admncse441project.ui.ticketmanagement.add.AddTicketViewModel
;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Locale;


public class EditShowtimeActivity extends AppCompatActivity {

    private Spinner nameSpinner;

    private EditText startTimeEditText, endTimeEditText,
availableSeatEditText, unavailableSeatEditText, dateEditText, nameCinema;

    private Button saveButton, addTicketButton;

    private ImageView deleteButton;

    private EditShowTimeViewModel viewModel;

    private AddTicketViewModel addTicketViewModel;

    private NowPlayingMovieViewModel movieViewModel;

    private HashMap<String, String> movieIdMap; // Thêm biến này để lưu
idMovie

    private String showTimeId;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_edit_showtime);


        viewModel = new
ViewModelProvider(this).get(EditShowTimeViewModel.class);

        addTicketViewModel = new
ViewModelProvider(this).get(AddTicketViewModel.class);

```



```

movieIdMap = new HashMap<>(); // Khởi tạo HashMap để lưu idMovie

initializeViews();

setupSpinner();

Intent intent = getIntent();

showTimeId = intent.getStringExtra("SHOWTIME_ID");

fetchShowTime(showTimeId);

saveButton.setOnClickListener(v -> saveShowtime(showTimeId));

addTicketButton.setOnClickListener(v -> addTicket(showTimeId));

setupClickListeners();

// Khởi tạo ViewModel cho Movie

NowPlayingViewModelFactory factory = new
NowPlayingViewModelFactory(new MovieRepositoryImp());

movieViewModel = new ViewModelProvider(this,
factory).get(NowPlayingMovieViewModel.class);

movieViewModel.loadMovies();

observeMovies();
}

private void initializeViews() {

    nameSpinner = findViewById(R.id.editTextText3);

    dateEditText = findViewById(R.id.editTextText6);

    startTimeEditText = findViewById(R.id.editTextText);

    endTimeEditText = findViewById(R.id.editTextText2);

    availableSeatEditText = findViewById(R.id.editTextText4);

    unavailableSeatEditText = findViewById(R.id.editTextText8);

    saveButton = findViewById(R.id.button);

    nameCinema=findViewById(R.id.editTextText10);

    deleteButton = findViewById(R.id.imageButton3);

```

```

        addTicketButton = findViewById(R.id.btn_showtime_add_ticket);
    }

    private void setupSpinner() {

        List<String> showTimeNames = getShowTimeNames();

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
        android.R.layout.simple_spinner_item, showTimeNames);

        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
        item);

        nameSpinner.setAdapter(adapter);
    }

    private void fetchShowTime(String showTimeId) {

        viewModel.getShowTime(showTimeId).observe(this, showTime -> {

            int position = getPositionForShowTimeName(showTime.getName());

            nameSpinner.setSelection(position);

            startTimeEditText.setText(showTime.getStartTime());

            endTimeEditText.setText(showTime.getEndTime());

            availableSeatEditText.setText(showTime.getAvailableSeat());

            nameCinema.setText(showTime.getNameCinema());

            unavailableSeatEditText.setText(showTime.getUnavailableSeat());

            dateEditText.setText(showTime.getDate());

        });
    }

    private int getPositionForShowTimeName(String name) {

        ArrayAdapter<String> adapter = (ArrayAdapter<String>)
        nameSpinner.getAdapter();

        return adapter.getPosition(name);
    }

    private void saveShowtime(String id) {

```

```

        String selectedName = nameSpinner.getSelectedItem().toString();

        String startTimeStr =
startTimeEditText.getText().toString().trim();

        String endTimeStr = endTimeEditText.getText().toString().trim();

        String availableSeat =
availableSeatEditText.getText().toString().trim();

        String unavailableSeat =
unavailableSeatEditText.getText().toString().trim();

        String Cinema=nameCinema.getText().toString().trim();

        String date = dateEditText.getText().toString().trim();


        if (selectedName.isEmpty() || startTimeStr.isEmpty() ||
endTimeStr.isEmpty() || availableSeat.isEmpty() ||
unavailableSeat.isEmpty() || date.isEmpty() || Cinema.isEmpty()) {

            Toast.makeText(this, "Vui lòng điền đầy đủ thông tin.",
Toast.LENGTH_SHORT).show();

            return;

        }


        if (!isValidTimeFormat(startTimeStr) ||
!isValidTimeFormat(endTimeStr)) {

            Toast.makeText(this, "Định dạng thời gian không hợp lệ. Vui
lòng sử dụng HH:mm", Toast.LENGTH_SHORT).show();

            return;

        }


        String idMovie = movieIdMap.get(selectedName); // Lấy idMovie tương ứng

        ShowTime showTime = new ShowTime(id, selectedName, availableSeat,
unavailableSeat, startTimeStr, endTimeStr, date, idMovie,Cinema);

        viewModel.updateShowTime(showTime, task -> {

            if (task.isSuccessful()) {

                Toast.makeText(this, "Showtime updated successfully",
Toast.LENGTH_SHORT).show();

            } else {

```

```

        Toast.makeText(this, "Failed to update Showtime",
Toast.LENGTH_SHORT).show();

    }

    finish();

});

}

private void addTicket(String showtimeId) {

    for(int i = 0; i < 42; i++) {

        Ticket ticket = new Ticket(null, showtimeId, "", "",
"Available");

        if (i == 0) {

addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReferen
ce -> {

            Toast.makeText(this, "Adding ticket...",
Toast.LENGTH_SHORT).show();

            }).addOnFailureListener(e -> {

                Log.e("TicketAddFragment", "Error adding ticket!", e);

                Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();

            });

        }

        else if (i == 41) {

addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReferen
ce -> {

            Toast.makeText(this, "Successfully added ticket!",
Toast.LENGTH_SHORT).show();

            }).addOnFailureListener(e -> {

                Log.e("TicketAddFragment", "Error adding ticket!", e);

                Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();

            });

        }

    }
}

```

```

        else {

addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReference -> {

            }).addOnFailureListener(e -> {

                Log.e("TicketAddFragment", "Error adding ticket!", e);

                Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();

            });

        }

    }

}

```

```

private boolean isValidTimeFormat(String timeStr) {

    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");

    sdf.setLenient(false);

    try {

        sdf.parse(timeStr);

        return true;

    } catch (ParseException e) {

        return false;

    }

}

```

```

private void setupClickListeners() {

    Intent intent = getIntent();

    String showTimeId = intent.getStringExtra("SHOWTIME_ID");

    deleteButton.setOnClickListener(v -> new AlertDialog.Builder(this)

        .setTitle("Xác nhận xóa")

        .setMessage("Bạn có chắc chắn muốn xóa showtime này không?")

        .setPositiveButton("Xóa", (dialog, which) -> {

            deleteShowTime(showTimeId);

}

```

```

        Toast.makeText(this, "ShowTime đã được xóa",
Toast.LENGTH_SHORT).show();

        finish();

    })

    .setNegativeButton("Hủy", (dialog, which) ->
dialog.dismiss())

    .show());

}

private void observeMovies() {

    movieViewModel.movies.observe(this, movies -> {

        if (movies != null) {

            List<String> movieTitles = new ArrayList<>();

            for (ResultsItem movie : movies) {

                movieTitles.add(movie.getTitle());

                movieIdMap.put(movie.getTitle(),
String.valueOf(movie.getId()));

            }

            ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, movieTitles);

            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);

            nameSpinner.setAdapter(adapter);

        }

    });

    movieViewModel.error.observe(this, exception -> {

        Log.e("EditShowtimeActivity", "Error loading movies",
exception);

        Toast.makeText(this, "Lỗi khi tải phim", Toast.LENGTH_SHORT).show();

    });

}

```

```

        private List<String> getShowTimeNames() {

            // Thay đổi phương thức này để lấy dữ liệu thực tế từ nguồn của
            bạn

            return List.of("Showtime 1", "Showtime 2", "Showtime 3"); // Ví dụ

        }
    }
}

```

### ○ TicketListFragment.java

```

package com.example.admncse441project.ui.ticketmanagement.list;

import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.admncse441project.R;
import
com.example.admncse441project.databinding.FragmentTicketListBinding;
import
com.example.admncse441project.ui.ticketmanagement.edit.TicketEditFragmen
t;

public class TicketListFragment extends Fragment {

    private FragmentTicketListBinding binding;

    private TicketViewModel viewModel;

    private TicketAdapter adapter;

```

```

@Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

        binding = FragmentTicketListBinding.inflate(inflater, container,
false);

        viewModel = new
ViewModelProvider(this).get(TicketViewModel.class);

        setupRecyclerView();

        setupObservers();

        return binding.getRoot();
    }

    private void setupRecyclerView() {

        adapter = new TicketAdapter(null, ticket -> {

            TicketEditFragment ticketEditFragment = new
TicketEditFragment();

            Bundle bundle = new Bundle();

            bundle.putString("TICKET_ID", ticket.getId());

            ticketEditFragment.setArguments(bundle);

            getParentFragmentManager().beginTransaction()

                .replace(R.id.fragmentContainerView,
ticketEditFragment)

                .addToBackStack(null)    // Thêm vào backstack nếu muốn quay lại

                .commit();

        });

        binding.rcvTicketList.setLayoutManager(new
LinearLayoutManager(getContext()));

        binding.rcvTicketList.setAdapter(adapter);
    }

```



```

private void setupObservers() {

    viewModel.tickets.observe(getViewLifecycleOwner(), tickets -> {

        adapter.setTicketList(tickets);

    });

}

@Override

public void onResume() {

    super.onResume();

    viewModel.loadTickets();

}

}

```

### ○ TicketViewModel.java

```

package com.example.admincse441project.ui.ticketmanagement.list;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.ticket.Ticket;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class TicketViewModel extends ViewModel {

```

```

        private final MutableLiveData<List<Ticket>> _ticket = new
MutableLiveData<>();

        public LiveData<List<Ticket>> tickets = _ticket;

        public TicketViewModel() {

            loadTickets();

        }

        void loadTickets() {

FirebaseUtils.getTicketsCollection().get().addOnCompleteListener(task ->
{

            if (task.isSuccessful()) {

                List<Ticket> ticketList = new ArrayList<>();

                QuerySnapshot querySnapshot = task.getResult();

                if (querySnapshot != null) {

                    for (DocumentSnapshot document :
querySnapshot.getDocuments()) {

                        Ticket ticket = document.toObject(Ticket.class);

                        if (ticket != null) {

                            ticketList.add(ticket);

                        }

                    }

                    _ticket.setValue(ticketList);

                }

            }

        });

    }

}

```

## ○ TicketAdapter.java

```
package com.example.admincse441project.ui.ticketmanagement.list;

import android.view.LayoutInflater;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.example.admincse441project.R;

import com.example.admincse441project.data.model.ticket.Ticket;

import com.example.admincse441project.databinding.ItemTicketRowBinding;

import java.util.List;

public class TicketAdapter extends
RecyclerView.Adapter<TicketAdapter.ViewHolder> {

    private List<Ticket> ticketList;

    private TicketAdapter.OnTicketClickListener listener;

    public TicketAdapter(List<Ticket> ticketList,
TicketAdapter.OnTicketClickListener listener) {

        this.ticketList = ticketList;

        this.listener = listener;

    }

    @NonNull

    @Override

    public TicketAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {

        LayoutInflater inflater =
LayoutInflater.from(parent.getContext());
```

```

        ItemTicketRowBinding binding =
ItemTicketRowBinding.inflate(inflater, parent, false);

        return new TicketAdapter.ViewHolder(binding);
    }

    @Override

    public void onBindViewHolder(@NonNull TicketAdapter.ViewHolder holder,
int position) {

        Ticket ticket = ticketList.get(position);

        holder.bind(ticket);

        holder.itemView.setOnClickListener(v ->
listener.onTicketClick(ticket));

        TextView txtStatus =
holder.itemView.findViewById(R.id.txt_ticket_status);

        if (txtStatus.getText().toString().equals("Booked") ||
txtStatus.getText().toString().equals("Used"))

txtStatus.setTextColor(holder.itemView.getContext().getColor(R.color.yell
ow_theme));

        else if (txtStatus.getText().toString().equals("Expired"))

txtStatus.setTextColor(holder.itemView.getContext().getColor(R.color.red_
theme));

    }

    @Override

    public int getItemCount() {

        return ticketList != null ? ticketList.size() : 0;

    }

    public class ViewHolder extends RecyclerView.ViewHolder {

        private final ItemTicketRowBinding binding;

```

```

    public ViewHolder(ItemTicketRowBinding binding) {

        super(binding.getRoot());

        this.binding = binding;

    }


    public void bind(Ticket ticket) {

        binding.txtTicketId.setText(String.valueOf(ticket.getId()));

binding.txtTicketSeat.setText(String.valueOf(ticket.getShowtimeId()));

binding.txtTicketStatus.setText(String.valueOf(ticket.getStatus()));

    }

}


    public void setTicketList(List<Ticket> ticketList) {

        this.ticketList = ticketList;

        notifyDataSetChanged();

    }


    public interface OnTicketClickListener {

        void onTicketClick(Ticket ticket);

    }

}

```

- **Seat.java**

```
package com.example.cse441_project.data.model.seat;

public class Seat {

    private String name;

    public Seat(String name) {

        this.name = name;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}
```

## ○ ChooseSeatActivity.java

```
package com.example.cse441_project.ui.bookticket.chooseseat;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.util.TypedValue;
import android.graphics.Rect;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.R;
import com.example.cse441_project.data.model.seat.Seat;
import com.example.cse441_project.data.model.ticket.Ticket;
import com.example.cse441_project.data.showscreen.ShowScreen;
import com.example.cse441_project.ui.bookticket.ChooseVoucherActivity;
import
com.example.cse441_project.ui.bookticket.showscreen.ChooseDateAndTimeActi
vity;
import com.example.cse441_project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
```

```

import java.util.ArrayList;

import java.util.List;

public class ChooseSeatActivity extends Activity {

    private RecyclerView rcvListSeat;

    private TextView txtPrice;

    private TextView txtNumberSeats;

    private Button btnContinue;

    private TextView txtNameMovie;

    private TextView txtTime;

    SeatAdapter adapter;

    private List<String> unavailableSeatList = new ArrayList<>();

    private List<Ticket> listTickets = new ArrayList<>();

    private String showtimeId;

    @Override

    protected void onCreate(@Nullable Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_choose_seat);

        rcvListSeat = findViewById(R.id.rcv_list_seat);

        txtPrice = findViewById(R.id.txt_choose_seat_price);

        txtNumberSeats = findViewById(R.id.txt_choose_seat_number);

        btnContinue = findViewById(R.id.btn_choose_seat_continue);

        txtNameMovie = findViewById(R.id.txt_choose_seat_movie_name);

        txtTime = findViewById(R.id.txt_choose_seat_time);

        // Lấy dữ liệu bên Activity khác

        showtimeId = getIntent().getStringExtra("SHOWTIME_ID");
    }
}

```



```

// Tạo danh sách ghế

List<Seat> list = new ArrayList<>();

for (char letter = 'A'; letter <= 'F'; letter++) {

    for (int number = 0; number <= 6; number++) {

        list.add(new Seat(letter + String.valueOf(number)));

    }

}

// Query vé đã mua => biết được chỗ ngồi đã bị chiếm dụng

FirebaseUtils.getTicketsByShowtimeAndSeat(showtimeId)

    .get()

    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>()

{

    @Override

    public void onComplete(Task<QuerySnapshot> task) {

        if (task.isSuccessful()) {

            for (DocumentSnapshot document : task.getResult())

{

                String seat = document.getString("seat");

                unavailableSeatList.add(seat);

            }

        }

        FirebaseUtils.getTicketsByShowtime(showtimeId)

            .get()

            .addOnCompleteListener(new

OnCompleteListener<QuerySnapshot>() {

                @Override

                public void

onComplete(Task<QuerySnapshot> task) {

                    if (task.isSuccessful()) {

                        for (DocumentSnapshot document

: task.getResult()) {

```

```

Ticket ticket =
document.toObject(Ticket.class);

if (ticket != null) {

listTickets.add(ticket);

}

}

// Xử lý RecyclerView

adapter = new
SeatAdapter(list, unavailableSeatList, ChooseSeatActivity.this, txtPrice,
txtNumberSeats, listTickets);

rcvListSeat.setAdapter(adapter);

} else {

System.err.println("Error
getting tickets: " + task.getException());

}

}

});

} else {

System.err.println("Error getting tickets: " +
task.getException());

}

}

});

rcvListSeat.setLayoutManager(new GridLayoutManager(this, 7));

int spaceInDp = (int) TypedValue.applyDimension(

TypedValue.COMPLEX_UNIT_DIP, 3,
getResources().getDisplayMetrics()

);

rcvListSeat.addItemDecoration(new RecyclerView.ItemDecoration() {

@Override

```

```

        public void getItemOffsets(@NonNull Rect outRect, @NonNull
View view,

                                @NonNull RecyclerView parent,
@NonNull RecyclerView.State state) {

            outRect.left = spaceInDp;

            outRect.right = spaceInDp;

            outRect.top = spaceInDp;

            outRect.bottom = spaceInDp;

        }

    });

    btnContinue.setOnClickListener(v -> continueProcess());

    txtNameMovie.setText(getIntent().getStringExtra("SHOWTIME_MOVIE"));

    txtTime.setText(getIntent().getStringExtra("SHOWTIME_START") + " - " +
getIntent().getStringExtra("SHOWTIME_END"));

    }

    private void continueProcess() {

        List<String> choosedSeats = adapter.getSelectedSeatList();

        String totalPrice = adapter.getTotalPrice();

        if (choosedSeats.size() > 0 && listTickets.size() > 0) {

            Intent intent = new Intent(this, ChooseVoucherActivity.class);

            ArrayList<String> selectedSeatsList = new
ArrayList<>(choosedSeats);

            intent.putStringArrayListExtra("SELECTED_SEATS_LIST",
selectedSeatsList);

            intent.putExtra("TOTAL_PRICE", totalPrice);

            intent.putExtra("SHOWTIME_ID", showtimeId);

            startActivity(intent);

            setResult(RESULT_OK, intent);

```

```

    }

}

}

```

### ○ SeatAdapter.java

```

package com.example.cse441_project.ui.bookticket.chooseseat;

import android.content.Context;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.R;
import com.example.cse441_project.data.model.seat.Seat;
import com.example.cse441_project.data.model.ticket.Ticket;

import java.util.ArrayList;
import java.util.List;

public class SeatAdapter extends
RecyclerView.Adapter<SeatAdapter.ViewHolder> {

    private List<Seat> list;

    private List<String> unavailableSeatList;

    private List<String> selectedSeatList;

```

```

private List<Ticket> listTickets;

private Context context;

private String totalPrice;


private TextView txtPrice;

private TextView txtNumberSeats;


    public SeatAdapter(List<Seat> list, List<String> unavailableSeatList,
Context context, TextView txtPrice, TextView txtNumberSeats, List<Ticket>
listTickets) {

        this.list = list;

        this.unavailableSeatList = unavailableSeatList;

        this.selectedSeatList = new ArrayList<>();

        this.context = context;

        this.txtPrice = txtPrice;

        this.txtNumberSeats = txtNumberSeats;

        this.listTickets = listTickets;

    }


    public List<String> getSelectedSeatList() {

        return selectedSeatList;

    }


    public String getTotalPrice() {

        return totalPrice;

    }


    @NonNull

    @Override

    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_seat,
parent, false);

```

...

```

        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
    {
        Seat seat = list.get(position);

        holder.name.setText(seat.getName());

        if (unavailableSeatList.contains(seat.getName())) {
            holder.name.setTextColor(Color.WHITE);
            holder.name.setBackgroundColor(Color.rgb(50, 200, 44, 50));
        } else if (selectedSeatList.contains(seat.getName())) {
            holder.name.setTextColor(Color.WHITE);
            holder.name.setBackgroundColor(Color.rgb(50, 10, 156, 161));
        } else {

            holder.name.setTextColor(holder.itemView.getContext().getColor(R.color.button_gradient_2));

            holder.name.setBackgroundColor(Color.rgb(50, 152, 120, 81));
        }

        holder.itemView.setOnClickListener(view -> {

            // Kiểm tra nếu listTickets có kích thước bằng 0

            if (listTickets.size() == 0 || selectedSeatList.size() >=
listTickets.size()) {

                Toast.makeText(context, "Cannot select seats because no
tickets are available", Toast.LENGTH_SHORT).show();

                return; // Dừng lại không cho chọn thêm ghế
            }

            if (unavailableSeatList.contains(seat.getName())) {

                Toast.makeText(context, "This seat is already occupied",
Toast.LENGTH_SHORT).show();

```

```

    } else {

        if (!selectedSeatList.contains(seat.getName())) {

            selectedSeatList.add(seat.getName());

        } else {

            selectedSeatList.remove(seat.getName());

        }

        notifyItemChanged(position);

    }

    totalPrice = (selectedSeatList.size() * 80000) + "";

    txtPrice.setText(totalPrice + " vnd");

    String numberSeats = selectedSeatList.size() <= 1 ? " seat" :
" seats";

    txtNumberSeats.setText(selectedSeatList.size() + numberSeats);

    });

}

@Override

public int getItemCount() {

    return list.size();

}

public class ViewHolder extends RecyclerView.ViewHolder {

    private TextView name;

    public ViewHolder(@NonNull View itemView) {

        super(itemView);

        name = itemView.findViewById(R.id.txt_item_seat);

    }

}

}

```

## ○ MyTicketActivity.java

```
package com.example.cse441_project.ui.profileoverlay.myticket;

import android.content.Intent;

import android.nfc.Tag;

import android.os.Bundle;

import android.util.Log;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.R;

import com.example.cse441_project.data.model.ticket.Ticket;

import com.example.cse441_project.ui.profileoverlay.ProfileMenuActivity;

import com.example.cse441_project.utils.FirebaseUtils;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.firestore.DocumentSnapshot;

import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;

import java.util.List;

public class MyTicketActivity extends AppCompatActivity {

    private RecyclerView rcv;

    private ImageView btnBack;
```



```

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    EdgeToEdge.enable(this);

    setContentView(R.layout.activity_my_ticket);

    rcv = findViewById(R.id.rcv_my_ticket_list);
    btnBack = findViewById(R.id.btn_back_my_ticket);

    List<Ticket> list = new ArrayList<>();

    if (FirebaseUtils.currentUserId() != null) {

        FirebaseUtils.getTicketByUserId(FirebaseUtils.currentUserId())

            .get()

            .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

                @Override

                public void onComplete(Task<QuerySnapshot> task) {

                    if (task.isSuccessful() && task.getResult() !=
null) {

                        for (DocumentSnapshot document :
task.getResult()) {

                            Ticket ticket =
document.toObject(Ticket.class);

                            list.add(ticket);

                        }

                        MyTicketAdapter adapter = new
MyTicketAdapter(list);

                        rcv.setAdapter(adapter);

                        rcv.setLayoutManager(new
LinearLayoutManager(MyTicketActivity.this));

                    } else {

```

```

        Toast.makeText(MyTicketActivity.this,
                        "Failed to load tickets.",
                        Toast.LENGTH_SHORT).show();
    }
}

});

    Log.d("Hehe", FirebaseUtils.currentUserId());
}

    btnBack.setOnClickListener(v -> {
        Intent intent = new Intent(MyTicketActivity.this,
ProfileMenuActivity.class);

        startActivity(intent);

        finish();

    });
}
}

```

## ○ MyTicketAdapter.java

```
package com.example.cse441_project.ui.profileoverlay.myticket;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.example.cse441_project.R;

import com.example.cse441_project.data.model.seat.Seat;

import com.example.cse441_project.data.model.ticket.Ticket;

import com.example.cse441_project.ui.bookticket.chooseseat.SeatAdapter;

import com.example.cse441_project.utils.FirebaseUtils;

import java.util.List;

public class MyTicketAdapter extends
RecyclerView.Adapter<MyTicketAdapter.ViewHolder> {

    private List<Ticket> list;

    public MyTicketAdapter(List<Ticket> list) {

        this.list = list;

    }

    @NonNull

    @Override

    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
```

```

        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_my_ticket,
parent, false);

        return new MyTicketAdapter.ViewHolder(view);

    }

    @Override

    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
    {

        Ticket ticket = list.get(position);

        FirebaseUtils.getShowtimeById(ticket.getShowtimeId()).get()

            .addOnCompleteListener(task -> {

                if (task.isSuccessful() && task.getResult() != null) {

                    String screen =
task.getResult().getString("nameCinema");

                    String movie = task.getResult().getString("name");

                    String date = task.getResult().getString("date");

                    String start =
task.getResult().getString("startTime");

                    String end =
task.getResult().getString("endTime");

                    holder.txtScreen.setText(screen != null ? screen :
"N/A");

                    holder.txtMovie.setText(movie != null ? movie :
"N/A");

                    holder.txtTime.setText(date + " " + start + " - "
+ end);

                    holder.txtSeat.setText(ticket.getSeat());

                } else {

                    holder.txtMovie.setText("Showtime not found");

                }

            });

    }

```

```

@Override

public int getItemCount() {

    return list.size();

}


public class ViewHolder extends RecyclerView.ViewHolder {

    private TextView txtScreen;

    private TextView txtSeat;

    private TextView txtMovie;

    private TextView txtTime;


    public ViewHolder(@NonNull View itemView) {

        super(itemView);

        this.txtMovie =
itemView.findViewById(R.id.txt_my_ticket_movie);

        this.txtScreen =
itemView.findViewById(R.id.txt_my_ticket_screen);

        this.txtSeat = itemView.findViewById(R.id.txt_my_ticket_seat);

        this.txtTime = itemView.findViewById(R.id.txt_my_ticket_time);

    }

}

}

```

## c) Lê Đình Dũng

### ○ AddShowTime.java

```
package com.example.admincse441project.ui.showtimemaganement;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.example.admincse441project.R;
import com.example.admincse441project.data.model.showtime.ShowTime;
import com.example.admincse441project.data.model.movie.ResultsItem; //
Thêm import này
import com.example.admincse441project.data.repository.MovieRepositoryImp;
// Thêm import này
import
com.example.admincse441project.ui.showtimemaganement.NowPlayingMovieViewM
odel; // Thêm import này
import
com.example.admincse441project.ui.showtimemaganement.NowPlayingViewModelF
actory; // Thêm import này

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;

public class AddShowtimeActivity extends AppCompatActivity {
    private Spinner spinnerMovie;
    private EditText editTextAvailableSeat, editTextUnavailableSeat,
    editTextStartTime, editTextEndTime, editTextDate, editNameCinema;
    private Button button;
    private AddShowTimeVewModel viewModel;
    private NowPlayingMovieViewModel movieViewModel;

    private HashMap<String, String> movieIdMap; // Thêm biến này để lưu
idMovie

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        viewModel = new
        ViewModelProvider(this).get(AddShowTimeVewModel.class);
        setContentView(R.layout.activity_add_showtime);

        // Khởi tạo các view
        spinnerMovie = findViewById(R.id.editTextText3); // Cập nhật id
```

```

    thành spinnerMovie
    editTextAvailableSeat = findViewById(R.id.editTextText4);
    editTextUnavailableSeat = findViewById(R.id.editTextText5);
    editTextStartTime = findViewById(R.id.editTextText);
    editTextEndTime = findViewById(R.id.editTextText2);
    editTextDate = findViewById(R.id.editTextText12);
    editNameCinema = findViewById(R.id.editTextText9);
    button = findViewById(R.id.buttonAdd);

    // Khởi tạo HashMap
    movieIdMap = new HashMap<>();

    // Thiết lập sự kiện khi nút được nhấn
    button.setOnClickListener(view -> addShowtime());

    // Thiết lập sự kiện cho nút quay lại
    ImageView backButton = findViewById(R.id.back);
    backButton.setOnClickListener(v -> finish());

    // Áp dụng insets cho layout
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v,
insets) -> {
        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
        return insets;
    });

    // Khởi tạo ViewModel cho Movie
    NowPlayingViewModelFactory factory = new
NowPlayingViewModelFactory(new MovieRepositoryImp());
    movieViewModel = new ViewModelProvider(this,
factory).get(NowPlayingMovieViewModel.class);
    movieViewModel.loadMovies();
    observeMovies();
}

private void observeMovies() {
    movieViewModel.movies.observe(this, movies -> {
        if (movies != null) {
            List<String> movieTitles = new ArrayList<>();
            for (ResultsItem movie : movies) {
                movieTitles.add(movie.getTitle());
                movieIdMap.put(movie.getTitle(),
String.valueOf(movie.getId()));
                // Lưu idMovie vào HashMap
            }
            ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, movieTitles);

            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);

            spinnerMovie.setAdapter(adapter);
        }
    });

    movieViewModel.error.observe(this, exception -> {
        Log.e("AddShowtimeActivity", "Error loading movies",
exception);
        Toast.makeText(this, "Error loading movies",
Toast.LENGTH_SHORT).show();
    });
}

```

```

    });
}

private void addTicket() {

}

private void addShowtime() {
    // Lấy dữ liệu từ các EditText và Spinner
    String name = (String) spinnerMovie.getSelectedItem(); // Lấy tên
    phim từ spinner
    String availableSeat =
    editTextAvailableSeat.getText().toString().trim();
    String unavailableSeat =
    editTextUnavailableSeat.getText().toString().trim();
    String startTimeStr =
    editTextStartTime.getText().toString().trim();
    String endTimeStr = editTextEndTime.getText().toString().trim();
    String nameCinema=editNameCinema.getText().toString().trim();
    String dateStr = editTextDate.getText().toString().trim();

    // Kiểm tra xem tất cả các trường đã được điền hay chưa
    if (!name.isEmpty() && !availableSeat.isEmpty() &&
    !unavailableSeat.isEmpty() &&
        !startTimeStr.isEmpty() && !endTimeStr.isEmpty() &&
        !dateStr.isEmpty()) {

        String idMovie = movieIdMap.get(name); // Lấy idMovie tương
        ứng

        // Tạo đối tượng ShowTime
        ShowTime showTime = new ShowTime(null, name, availableSeat,
        unavailableSeat, startTimeStr, endTimeStr, dateStr, idMovie,nameCinema);

        // Gọi ViewModel để thêm showtime

viewModel.addShowTime(showTime).addOnSuccessListener(documentReference ->
{
    Toast.makeText(this, "Successfully added Showtime",
    Toast.LENGTH_SHORT).show();
    finish();
    }).addOnFailureListener(e -> {
        Log.e("AddShowTimeActivity", "Error adding Showtime", e);
        Toast.makeText(this, "Add Showtime failed",
        Toast.LENGTH_SHORT).show();
    });
    } else {
        Toast.makeText(this, "Please fill all fields",
        Toast.LENGTH_SHORT).show();
    }
}

private String parseTimeToString(String timeStr) {
    try {
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm",
        Locale.getDefault());
        sdf.parse(timeStr); // Không cần phải chuyển đổi thành Time
        return timeStr; // Trả về chuỗi thời gian
    } catch (ParseException e) {
        e.printStackTrace();
        return null;
    }
}
}

```



```

    private String parseDateToString(String dateStr) {
        try {
            // Định dạng cho dữ liệu đầu vào
            SimpleDateFormat inputFormat = new
SimpleDateFormat("yyyy/MM/dd", Locale.getDefault());
            // Định dạng cho dữ liệu đầu ra
            SimpleDateFormat outputFormat = new SimpleDateFormat("yyyy-
MM-dd", Locale.getDefault());

            // Phân tích chuỗi ngày
            java.util.Date date = inputFormat.parse(dateStr);
            // Chuyển đổi sang định dạng yyyy-MM-dd
            return outputFormat.format(date);
        } catch (ParseException e) {
            e.printStackTrace();
            return null;
        }
    }
}

```

### ○ AddShowTimeViewModel.java

```

package com.example.admincse441project.ui.showtimemaganement;

import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.showtime.ShowTime;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;

public class AddShowTimeVliewModel extends ViewModel {
    private final FirebaseUtils firebaseUtils = new FirebaseUtils();
    public Task<DocumentReference> addShowTime(ShowTime show) {
        return
FirebaseUtils.addShowtime(show).addOnSuccessListener(documentReference ->
{
            String generatedId = documentReference.getId();
            show.setId(generatedId);
            FirebaseUtils.updateShowtime(show);
        });
    }
}

```

## ○ EditShowTimeActivity.java

```
package com.example.admncse441project.ui.showtimemaganement;

import static
com.example.admncse441project.utils.FirebaseUtils.deleteShowTime;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;

import com.example.admncse441project.R;
import com.example.admncse441project.data.model.showtime.ShowTime;
import com.example.admncse441project.data.model.movie.ResultsItem; //
Thêm import này
import com.example.admncse441project.data.model.ticket.Ticket;
import com.example.admncse441project.data.repository.MovieRepositoryImp;
// Thêm import này
import
com.example.admncse441project.ui.showtimemaganement.NowPlayingMovieViewM
odel; // Thêm import này
import
com.example.admncse441project.ui.showtimemaganement.NowPlayingViewModelF
actory; // Thêm import này
import
com.example.admncse441project.ui.ticketmanagement.add.AddTicketViewModel
;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;

public class EditShowtimeActivity extends AppCompatActivity {
    private Spinner nameSpinner;
    private EditText startTimeEditText, endTimeEditText,
availableSeatEditText, unavailableSeatEditText, dateEditText, nameCinema;
    private Button saveButton, addTicketButton;
    private ImageView deleteButton;
    private EditShowTimeViewModel viewModel;
    private AddTicketViewModel addTicketViewModel;
    private NowPlayingMovieViewModel movieViewModel;
    private HashMap<String, String> movieIdMap; // Thêm biến này để lưu
idMovie

    private String showTimeId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_showtime);
    }
}
```

```

        viewModel = new
ViewModelProvider(this).get(EditShowTimeViewModel.class);
        addTicketViewModel = new
ViewModelProvider(this).get(AddTicketViewModel.class);
        movieIdMap = new HashMap<>(); // Khởi tạo HashMap để lưu idMovie

        initializeViews();
        setupSpinner();

        Intent intent = getIntent();
        showTimeId = intent.getStringExtra("SHOWTIME_ID");
        fetchShowTime(showTimeId);

        saveButton.setOnClickListener(v -> saveShowtime(showTimeId));
        addTicketButton.setOnClickListener(v -> addTicket(showTimeId));
        setupClickListeners();

        // Khởi tạo ViewModel cho Movie
        NowPlayingViewModelFactory factory = new
NowPlayingViewModelFactory(new MovieRepositoryImp());
        movieViewModel = new ViewModelProvider(this,
factory).get(NowPlayingMovieViewModel.class);
        movieViewModel.loadMovies();
        observeMovies();
    }

    private void initializeViews() {
        nameSpinner = findViewById(R.id.editTextText3);
        dateEditText = findViewById(R.id.editTextText6);
        startTimeEditText = findViewById(R.id.editTextText);
        endTimeEditText = findViewById(R.id.editTextText2);
        availableSeatEditText = findViewById(R.id.editTextText4);
        unavailableSeatEditText = findViewById(R.id.editTextText8);
        saveButton = findViewById(R.id.button);
        nameCinema=findViewById(R.id.editTextText10);
        deleteButton = findViewById(R.id.imageButton3);
        addTicketButton = findViewById(R.id.btn_showtime_add_ticket);
    }

    private void setupSpinner() {
        List<String> showTimeNames = getShowTimeNames();
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, showTimeNames);

        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        nameSpinner.setAdapter(adapter);
    }

    private void fetchShowTime(String showTimeId) {
        viewModel.getShowTime(showTimeId).observe(this, showTime -> {
            int position =
getPositionForShowTimeName(showTime.getName());
            nameSpinner.setSelection(position);
            startTimeEditText.setText(showTime.getStartTime());
            endTimeEditText.setText(showTime.getEndTime());
            availableSeatEditText.setText(showTime.getAvailableSeat());
            nameCinema.setText(showTime.getNameCinema());

            unavailableSeatEditText.setText(showTime.getUnavailableSeat());
            dateEditText.setText(showTime.getDate());
        });
    }

```

```

    }

    private int getPositionForShowTimeName(String name) {
        ArrayAdapter<String> adapter = (ArrayAdapter<String>)
nameSpinner.getAdapter();
        return adapter.getPosition(name);
    }

    private void saveShowtime(String id) {
        String selectedName = nameSpinner.getSelectedItem().toString();
        String startTimeStr =
startTimeEditText.getText().toString().trim();
        String endTimeStr = endTimeEditText.getText().toString().trim();
        String availableSeat =
availableSeatEditText.getText().toString().trim();
        String unavailableSeat =
unavailableSeatEditText.getText().toString().trim();
        String Cinema=nameCinema.getText().toString().trim();
        String date = dateEditText.getText().toString().trim();

        if (selectedName.isEmpty() || startTimeStr.isEmpty() ||
endTimeStr.isEmpty() || availableSeat.isEmpty() ||
unavailableSeat.isEmpty() || date.isEmpty() || Cinema.isEmpty()) {
            Toast.makeText(this, "Vui lòng điền đầy đủ thông tin.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidTimeFormat(startTimeStr) ||
!isValidTimeFormat(endTimeStr)) {
            Toast.makeText(this, "Định dạng thời gian không hợp lệ. Vui
lòng sử dụng HH:mm", Toast.LENGTH_SHORT).show();
            return;
        }

        String idMovie = movieIdMap.get(selectedName); // Lấy idMovie
        tương ứng
        ShowTime showTime = new ShowTime(id, selectedName, availableSeat,
unavailableSeat, startTimeStr, endTimeStr, date, idMovie,Cinema);

        viewModel.updateShowTime(showTime, task -> {
            if (task.isSuccessful()) {
                Toast.makeText(this, "Showtime updated successfully",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Failed to update Showtime",
Toast.LENGTH_SHORT).show();
            }
            finish();
        });
    }

    private void addTicket(String showtimeId) {
        for(int i = 0; i <
Integer.parseInt(availableSeatEditText.getText().toString().trim()); i++)
        {
            Ticket ticket = new Ticket(null, showtimeId, "", "",
"Available");

            if (i == 0) {

addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReferen
ce -> {

```

```

        Toast.makeText(this, "Adding ticket...",
Toast.LENGTH_SHORT).show();
        }).addOnFailureListener(e -> {
            Log.e("TicketAddFragment", "Error adding ticket!",
e);
            Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();
        });
    }
    else if (i ==
Integer.parseInt(availableSeatEditText.getText().toString().trim()) - 1)
{
addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReferen
ce -> {
        Toast.makeText(this, "Successfully added ticket!",
Toast.LENGTH_SHORT).show();
        }).addOnFailureListener(e -> {
            Log.e("TicketAddFragment", "Error adding ticket!",
e);
            Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();
        });
    }

addTicketViewModel.addTicket(ticket).addOnSuccessListener(documentReferen
ce -> {
        }).addOnFailureListener(e -> {
            Log.e("TicketAddFragment", "Error adding ticket!", e);
            Toast.makeText(this, "Add ticket failed!",
Toast.LENGTH_SHORT).show();
        });
    }
}

private boolean isValidTimeFormat(String timeStr) {
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
    sdf.setLenient(false);
    try {
        sdf.parse(timeStr);
        return true;
    } catch (ParseException e) {
        return false;
    }
}

private void setupClickListeners() {
    Intent intent = getIntent();
    String showTimeId = intent.getStringExtra("SHOWTIME_ID");

    deleteButton.setOnClickListener(v -> new
AlertDialog.Builder(this)
        .setTitle("Xác nhận xóa")
        .setMessage("Bạn có chắc chắn muốn xóa showtime này
không?")
        .setPositiveButton("Xóa", (dialog, which) -> {
            deleteShowTime(showTimeId);
            Toast.makeText(this, "ShowTime đã được xóa",
Toast.LENGTH_SHORT).show();
            finish();
        })
        .setNegativeButton("Hủy", (dialog, which) ->

```

```

dialog.dismiss()
        .show());
    }

    private void observeMovies() {
        movieViewModel.movies.observe(this, movies -> {
            if (movies != null) {
                List<String> movieTitles = new ArrayList<>();
                for (ResultsItem movie : movies) {
                    movieTitles.add(movie.getTitle());
                    movieIdMap.put(movie.getTitle(),
String.valueOf(movie.getId()));
                }
                ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, movieTitles);

                adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);

                nameSpinner.setAdapter(adapter);
            }
        });

        movieViewModel.error.observe(this, exception -> {
            Log.e("EditShowtimeActivity", "Error loading movies",
exception);
            Toast.makeText(this, "Lỗi khi tải phim",
Toast.LENGTH_SHORT).show();
        });
    }

    private List<String> getShowTimeNames() {
        // Thay đổi phương thức này để lấy dữ liệu thực tế từ nguồn của
bạn
        return List.of("Showtime 1", "Showtime 2", "Showtime 3"); // Ví
dụ
    }
}

```

## ○ EditShowTimeViewModel

```

package com.example.admncse441project.ui.showtimemaganement;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admncse441project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.firebase.firestore.DocumentSnapshot;

import com.example.admncse441project.data.model.showtime.ShowTime;

public class EditShowTimeViewModel extends ViewModel {
    private MutableLiveData<ShowTime> _showTime = new
MutableLiveData<>();
    public LiveData<ShowTime> getShowTime(String discountId) {
        FirebaseUtils.getShowtimesCollection().document(discountId)
            .get().addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document.exists()) {
                        ShowTime discount =

```

```

document.toObject>ShowTime.class);
                _showTime.setValue(discount);
            }
        }
    });
    return _showTime;
}

public void updateShowTime>ShowTime showTime,
OnCompleteListener

```

### ○ NowPlayingMovieViewModel.java

```

package com.example.admincse441project.ui.showtimemaganement;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.movie.NowPlayingMovie;
import com.example.admincse441project.data.model.movie.ResultsItem;
import com.example.admincse441project.data.repository.MovieRepository;
import com.example.admincse441project.data.repository.MovieRepositoryImp;
import com.example.admincse441project.data.resource.Result;

import java.util.List;

public class NowPlayingMovieViewModel extends ViewModel {
    private final MovieRepository movieRepository;
    private MutableLiveData<List<ResultsItem>> _movies = new
MutableLiveData<>();
    public LiveData<List<ResultsItem>> movies = _movies;

    private final MutableLiveData<Exception> _error = new
MutableLiveData<>();
    public LiveData<Exception> error = _error;

    public NowPlayingMovieViewModel(MovieRepository movieRepository) {
        this.movieRepository = new MovieRepositoryImp();
    }

    public void loadMovies() {
        new Thread(() -> {
            try {
                Result<NowPlayingMovie> result =
movieRepository.getMovie();
                if (result instanceof Result.Success) {
                    _movies.postValue(((Result.Success<NowPlayingMovie>)
result).getData().getResults());
                } else if (result instanceof Result.Error) {
                    _movies.postValue(null);
                    _error.postValue(((Result.Error<NowPlayingMovie>)
result).getException());
                }
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        })
    }
}

```

```

        }).start();
    }
}

```

### ○ NowPlayingViewModelFactory.java

```

package com.example.admincse441project.ui.showtimemaganement;

import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

import com.example.admincse441project.data.repository.MovieRepository;

public class NowPlayingViewModelFactory implements
    ViewModelProvider.Factory {
    private final MovieRepository movieRepository;

    public NowPlayingViewModelFactory(MovieRepository movieRepository) {
        this.movieRepository = movieRepository;
    }

    @NonNull
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {
        if (modelClass.isAssignableFrom(NowPlayingMovieViewModel.class))
        {
            return (T) new NowPlayingMovieViewModel(movieRepository);
        }
        throw new IllegalArgumentException("Unknown ViewModel class");
    }
}

```

### ○ ShowTimeAdapter.java

```

package com.example.admincse441project.ui.showtimemaganement;

import android.view.LayoutInflater;
import android.view.ViewGroup;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.admincse441project.data.model.showtime.ShowTime;
import com.example.admincse441project.databinding.ItemShowtimeBinding;
import java.util.List;

public class ShowTimeAdapter extends
    RecyclerView.Adapter<ShowTimeAdapter.ViewHolder> {
    private List<ShowTime> showTimeList;
    private OnShowTimeClickListener listener;

    public ShowTimeAdapter(List<ShowTime> showTimeList,
        OnShowTimeClickListener listener) {
        this.showTimeList = showTimeList;
        this.listener = listener;
    }

    @NonNull
    @Override
    public ShowTimeAdapter.ViewHolder onCreateViewHolder(@NonNull
        ViewGroup parent, int viewType) {

```



```

        LayoutInflater inflater =
LayoutInflater.from(parent.getContext());
        ItemShowtimeBinding binding =
ItemShowtimeBinding.inflate(inflater, parent, false);
        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull ShowTimeAdapter.ViewHolder
holder, int position) {
        ShowTime showtime = showTimeList.get(position);

        // Set STT (Số thứ tự)
        holder.binding.txtTicketNumber.setText(String.valueOf(position +
1));

        // Bind dữ liệu showtime
        holder.bind(showtime);
        holder.itemView.setOnClickListener(v ->
listener.onShowTimeClick(showtime));
    }

    @Override
    public int getItemCount() {
        return showTimeList != null ? showTimeList.size() : 0;
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final ItemShowtimeBinding binding;

        public ViewHolder(ItemShowtimeBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        public void bind(ShowTime showTime) {

            binding.textView23.setText(String.valueOf(showTime.getName()));

            binding.textView33.setText(String.valueOf(showTime.getNameCinema()));

            binding.textView14.setText(String.valueOf(showTime.getStartTime() + " - "
+ showTime.getEndTime()));
            binding.txtDate.setText(String.valueOf(showTime.getDate()));
        }
    }

    public void setShowTimeList(List<ShowTime> showTimeList) {
        this.showTimeList = showTimeList;
        notifyDataSetChanged();
    }

    public interface OnShowTimeClickListener {
        void onShowTimeClick(ShowTime showtime);
    }
}

```

## ○ ShowTimeListFragment.java

```
package com.example.admncse441project.ui.showtimemaganement;

import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.admncse441project.databinding.FragmentDiscountsListBinding;
import com.example.admncse441project.databinding.FragmentShowTimeListBinding;

import java.util.List;

public class ShowTimeListFragment extends Fragment {
    private FragmentShowTimeListBinding binding;
    private ShowTimeViewModel viewModel;
    private ShowTimeAdapter adapter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        binding = FragmentShowTimeListBinding.inflate(inflater, container, false);
        viewModel = new
        ViewModelProvider(this).get(ShowTimeViewModel.class);

        setupRecyclerView();
        setupObservers();
        onViewClickListeners();
        return binding.getRoot();
    }

    private void setupRecyclerView() {
        adapter = new ShowTimeAdapter(null, discount -> {
            Intent intent = new Intent(getActivity(),
            EditShowtimeActivity.class);
            intent.putExtra("SHOWTIME_ID", discount.getId());
            startActivity(intent);
        });
        binding.rcvManagementList.setLayoutManager(new
        LinearLayoutManager(getContext()));
        binding.rcvManagementList.setAdapter(adapter);
    }

    private void setupObservers() {
        viewModel.showtimes.observe(getViewLifecycleOwner(), showTimes ->
        {
            adapter.setShowTimeList(showTimes);
        });
    }

    private void onViewClickListeners() {
```

```

        binding.imageButton2.setOnClickListener(view -> {
            Intent intent = new Intent(requireContext(),
AddShowtimeActivity.class);
            startActivity(intent);
        });
    }

    @Override
    public void onResume() {
        super.onResume();
        viewModel.loadShowTimes();
    }
}

```

## ○ ShowTimeViewModel.java

```

package com.example.admincse441project.ui.showtimemaganement;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.admincse441project.data.model.showtime.ShowTime;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class ShowTimeViewModel extends ViewModel {
    private final MutableLiveData<List<ShowTime>> _showtimes = new
MutableLiveData<>();
    public LiveData<List<ShowTime>> showtimes = _showtimes;
    public ShowTimeViewModel() {
        loadShowTimes();
    }
    void loadShowTimes() {

FirebaseUtils.getShowtimesCollection().get().addOnCompleteListener(task -
> {
        if (task.isSuccessful()) {
            List<ShowTime> showTimeList = new ArrayList<>();
            QuerySnapshot querySnapshot = task.getResult();
            if (querySnapshot != null) {
                for (DocumentSnapshot document :
querySnapshot.getDocuments()) {
                    ShowTime showTime =
document.toObject(ShowTime.class);
                    if (showTime != null) {
                        showTimeList.add(showTime);
                    }
                }
                _showtimes.setValue(showTimeList);
            }
        }
    });
}
}

```

## ○ PaymentActivity.java

```
package com.example.cse441_project.ui.bookticket;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;
import com.example.cse441_project.R;
import com.example.cse441_project.ui.ggpay.activity.CheckoutActivity;

public class PaymentActivity extends AppCompatActivity {
    private String showTimeId;
    private SelectShowTime viewModel;
    private TextView nameMovie, timeMovie, Seat, Total;
    private Button voucher, ticket;
    private double newTotalPrice;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_payment);
        voucher = findViewById(R.id.btn_choose_promotion);
        ticket = findViewById(R.id.btn_book_ticket_payment);

        voucher.setOnClickListener(v -> continueProcess());
        ticket.setOnClickListener(v -> bookTicket());
        viewModel = new
        ViewModelProvider(this).get(SelectShowTime.class);
        showTimeId = getIntent().getStringExtra("SHOWTIME_ID");

        initializeViews();
        fetchShowTime(showTimeId);
    }

    private void bookTicket() {
        Intent intent = new Intent(this, CheckoutActivity.class);
        intent.putExtra("TOTAL_PRICE", newTotalPrice);
        Log.e("CheckoutActivity", "bookTicket: " + newTotalPrice);
        startActivity(intent); // Sử dụng startActivityForResult để nhận
        dữ liệu
    }

    private void continueProcess() {
        Intent intent = new Intent(this, ChooseVoucherActivity.class);
        startActivityForResult(intent, 1); // Sử dụng
        startActivityForResult để nhận dữ liệu
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK && data != null)
        {
            // Nhận giá trị voucher từ ChooseVoucherActivity
            String voucherValue = data.getStringExtra("voucher_value");
            if (voucherValue != null) {

```

```

        updateTotalPrice(voucherValue);
    }
}

private void initializeViews() {
    nameMovie = findViewById(R.id.tvMovieName);
    timeMovie = findViewById(R.id.tvTimeDate);
    Seat = findViewById(R.id.seat);
    Total = findViewById(R.id.total);
}

private void fetchShowTime(String showTimeId) {
    String selectedSeatsString =
    getIntent().getStringExtra("SELECTED_SEATS_LIST");
    String totalPrice = getIntent().getStringExtra("TOTAL_PRICE");
    viewModel.getShowTime(showTimeId).observe(this, showTime -> {
        nameMovie.setText(showTime.getName());
        timeMovie.setText(showTime.getDate() + "-" +
    showTime.getStartTime());
        Seat.setText(selectedSeatsString);
        Total.setText(totalPrice); // Hiển thị tổng tiền ban đầu
    });
}

private void updateTotalPrice(String voucherValue) {
    // Lấy giá trị tổng tiền hiện tại
    String totalPriceText = Total.getText().toString();
    double totalPrice = Double.parseDouble(totalPriceText);

    // Lấy giá trị voucher
    double discountValue = Double.parseDouble(voucherValue);

    // Tính toán lại tổng tiền sau khi trừ đi voucher
    newTotalPrice = totalPrice - discountValue;

    // Cập nhật giao diện
    Total.setText(String.valueOf(newTotalPrice)); // Cập nhật lại
    tổng tiền
}
}

```

### ○ SelectShowTime.java

```

package com.example.cse441_project.ui.bookticket;

import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.cse441_project.data.model.showtime.ShowTime;
import com.example.cse441_project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.firebase.firestore.DocumentSnapshot;
import com.example.cse441_project.utils.FirebaseUtils;
public class SelectShowTime extends ViewModel {
    private MutableLiveData<ShowTime> _showTime = new
    MutableLiveData<>();
    public LiveData<ShowTime> getShowTime(String discountId) {

```

```

        FirebaseUtils.getShowtimesCollection().document(discountId)
            .get().addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document.exists()) {
                        ShowTime discount =
document.toObject(ShowTime.class);
                        _showTime.setValue(discount);
                    }
                }
            });
        return _showTime;
    }

    public void updateShowTime(ShowTime showTime,
        OnCompleteListener<Void> onCompleteListener) {

        FirebaseUtils.updateShowtime(showTime).addOnCompleteListener(onCompleteLi
stener);
    }
}

```

### ○ VoucherAdapter.java

```

package com.example.cse441_project.ui.bookticket;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.cse441_project.R;
import com.example.cse441_project.data.model.discount.Discount;
import com.bumptech.glide.Glide;
import java.util.ArrayList;
import java.util.List;

public class VoucherAdapter extends
RecyclerView.Adapter<VoucherAdapter.VoucherViewHolder> {
    private List<Discount> discounts = new ArrayList<>();
    private int selectedPosition = -1; // Trạng thái chọn
    private String selectedId = "";
    private OnItemSelectedListener onItemSelectedListener; // Listener để
thông báo

    // Interface để thông báo khi một voucher được chọn
    public interface OnItemSelectedListener {
        void onItemSelected(String selectedId, String selectedValue);
    }

    public void setOnItemSelectedListener(OnItemSelectedListener
listener) {
        this.onItemSelectedListener = listener;
    }

    public void setDiscounts(List<Discount> discounts) {
        this.discounts = discounts;
        notifyDataSetChanged();
    }
}

```

```

        @NonNull
        @Override
        public ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
            View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_choose_vou
cher, parent, false);
            return new ViewHolder(view);
        }

        @Override
        public void onBindViewHolder(@NonNull ViewHolder holder, int
position) {
            Discount discount = discounts.get(position);
            holder.voucherName.setText(discount.getName());
            holder.voucherExpiry.setText("Expires: " + discount.getValue());

            // Đổi màu nền khi item được chọn
            if (position == selectedPosition) {
                holder.itemView.setBackgroundColor(0xFFCCCC); // Màu #CCC
                khi chọn
            } else {
                holder.itemView.setBackgroundColor(0xFFFFFFFF); // Màu trắng
                khi không chọn
            }

            holder.itemView.setOnClickListener(v -> {
                selectedPosition = holder.getAdapterPosition();
                selectedId = discount.getId(); // Lưu lại ID của phiếu giảm
                giá đã chọn
                String expiryText =
holder.voucherExpiry.getText().toString();
                String expiryNumber = expiryText.replaceAll("[^0-9]", "");
                // Gọi listener để thông báo về voucher được chọn
                if (onItemSelectedListener != null) {
                    onItemSelectedListener.onItemSelected(selectedId,
expiryNumber); // Gửi ID và giá trị
                }

                notifyDataSetChanged(); // Cập nhật toàn bộ RecyclerView
            });
        }

        @Override
        public int getItemCount() {
            return discounts.size();
        }

        static class ViewHolder extends RecyclerView.ViewHolder {
            TextView voucherName, voucherExpiry;

            ViewHolder(@NonNull View itemView) {
                super(itemView);
                voucherName = itemView.findViewById(R.id.txt_voucher_name);
                voucherExpiry =
itemView.findViewById(R.id.txt_voucher_expires);
            }
        }
    }
}

```

## ○ CheckoutActivity.java

```
/*
 * Copyright 2024 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 * implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.cse441_project.ui.ggpay.activity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import androidx.activity.result.ActivityResultLauncher;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;

import com.example.cse441_project.R;

import com.example.cse441_project.databinding.ActivityCheckoutGgBinding;
import com.example.cse441_project.ui.ggpay.util.PaymentsUtil;
import com.example.cse441_project.ui.ggpay.viewmodel.CheckoutViewModel;
import com.google.android.gms.common.api.CommonStatusCodes;

import com.google.android.gms.tasks.Task;
import com.google.android.gms.wallet.PaymentData;
import com.google.android.gms.wallet.button.ButtonOptions;
import com.google.android.gms.wallet.button.PayButton;
import com.google.android.gms.wallet.contract.TaskResultContracts;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.Locale;

/**
 * Checkout implementation for the app
 */
public class CheckoutActivity extends AppCompatActivity {

    private double priceInVND;
    private CheckoutViewModel model;

    private PayButton googlePayButton;

    private final ActivityResultLauncher<Task<PaymentData>>
    paymentDataLauncher =
```



```

        registerForActivityResult(new
TaskResultContracts.GetPaymentDataResult(), result -> {
    int statusCode = result.getStatus().getStatusCode();
    switch (statusCode) {
        case CommonStatusCodes.SUCCESS:
            handlePaymentSuccess(result.getResult());
            break;
        //case CommonStatusCodes.CANCELED: The user canceled
        case CommonStatusCodes.DEVELOPER_ERROR:
            handleError(statusCode,
result.getStatus().getStatusMessage());
            break;
        default:
            handleError(statusCode, "Unexpected non API" +
                " exception when trying to deliver the task result to an
activity!");
            break;
    }
});

/**
 * Initialize the Google Pay API on creation of the activity
 *
 * @see Activity#onCreate(Bundle)
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    model = new ViewModelProvider(this).get(CheckoutViewModel.class);
    //todo: 2 layout thanh toán là activity_checkout và
activity_checkout_success
    model.canUseGooglePay.observe(this, this::setGooglePayAvailable);
    //todo: Nhận giá trị từ Intent gửi sang, ví dụ như ở dưới
    //
    priceInVND = getIntent().getDoubleExtra("TOTAL_PRICE", 10000); // Mặc
định là 0 nếu không có giá
    Log.e("CheckoutActivity", "bookTicket: " +priceInVND );
    // Sử dụng giá để hiển thị hoặc thực hiện thanh toán
    requestPayment(priceInVND);
    initializeUi();
}

private void initializeUi() {
    // Use view binding to access the UI elements
    ActivityCheckoutGgBinding layoutBinding =
ActivityCheckoutGgBinding.inflate(getLayoutInflater());
    setContentView(layoutBinding.getRoot());
    //todo: dùng cấu trúc nh bên dưới để set giá trị cho các trường
layoutBinding.txtPrice.setText(String.valueOf(priceInVND));

    // The Google Pay button is a layout file - take the root view
    googlePayButton = layoutBinding.googlePayButton;
    try {
        googlePayButton.initialize(
            ButtonOptions.newBuilder()

.setAllowedPaymentMethods(PaymentsUtil.getAllowedPaymentMethods().toStrin
g()).build()
        );
        // Thay đổi cách gọi requestPayment để truyền giá tiền
        googlePayButton.setOnClickListener(v ->

```

```

requestPayment(priceInVND)); // Sử dụng biến giá tiền
    } catch (JSONException e) {
        // Giấu nút Google Pay nếu có lỗi
        googlePayButton.setVisibility(View.GONE);
    }
}

/**
 * If isReadyToPay returned {@code true}, show the button and hide the
 * "checking" text.
 * Otherwise, notify the user that Google Pay is not available. Please
 * adjust to fit in with
 * your current user flow. You are not required to explicitly let the
 * user know if isReadyToPay
 * returns {@code false}.
 *
 * @param available isReadyToPay API response.
 */
private void setGooglePayAvailable(boolean available) {
    if (available) {
        googlePayButton.setVisibility(View.VISIBLE);
    } else {
        Toast.makeText(this, R.string.google_pay_status_unavailable,
            Toast.LENGTH_LONG).show();
    }
}

public void requestPayment(double priceInVND) {
    // Sử dụng giá tiền đã truyền vào
    final Task<PaymentData> task =
model.getLoadPaymentDataTask(priceInVND);
    task.addOnCompleteListener(paymentDataLauncher::launch);
}

/**
 * PaymentData response object contains the payment information, as
 * well as any additional
 * requested information, such as billing and shipping address.
 *
 * @param paymentData A response object returned by Google after a
 * payer approves payment.
 * @see <a
 * href="https://developers.google.com/pay/api/android/reference/
 * object#PaymentData">PaymentData</a>
 */
private void handlePaymentSuccess(PaymentData paymentData) {
    final String paymentInfo = paymentData.toJson();

    try {
        JSONObject paymentMethodData = new
JSONObject(paymentInfo).getJSONObject("paymentMethodData");
        // If the gateway is set to "example", no payment information is
        returned - instead, the
        // token will only consist of "examplePaymentMethodToken".

        final JSONObject info = paymentMethodData.getJSONObject("info");
        // Logging token string.
        Log.d("Google Pay token", paymentMethodData
            .getJSONObject("tokenizationData")
            .getString("token"));

        startActivity(new Intent(this, CheckoutSuccessActivity.class));
        //todo: Nếu thành công, Truyền activity muốn đến ở đây
    }
}

```

```

    } catch (JSONException e) {
        Log.e("handlePaymentSuccess", "Error: " + e);
    }
}

/**
 * At this stage, the user has already seen a popup informing them an
 * error occurred. Normally,
 * only logging is required.
 *
 * @param statusCode holds the value of any constant from
 * CommonStatusCode or one of the
 * WalletConstants.ERROR_CODE_* constants.
 * @see <a
 * href="https://developers.google.com/android/reference/com/google/android/
 * gms/wallet/
 * WalletConstants#constant-summary">Wallet Constants Library</a>
 */
private void handleError(int statusCode, @Nullable String message) {
    Log.e("loadPaymentData failed",
        String.format(Locale.getDefault(), "Error code: %d, Message: %s",
        statusCode, message));
}
}

```

## ○ CheckoutSuccessActivity.java

```
/*
 * Copyright 2024 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 * implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.cse441_project.ui.ggpay.activity;

import android.content.Intent;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.databinding.ActivityCheckoutSuccessBinding;
import com.example.cse441_project.ui.profileoverlay.MyTicketActivity;

public class CheckoutSuccessActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ActivityCheckoutSuccessBinding layoutBinding =
            ActivityCheckoutSuccessBinding.inflate(getLayoutInflater());
        setContentView(layoutBinding.getRoot());
        startActivity(new Intent(this, MyTicketActivity.class));
        finish(); // Đóng CheckoutSuccessActivity để không quay lại
    }
}
```

## d) Cù Tiên Thịnh

- **App Admin**

- **Account.java**

```
package com.example.admincse441project.data.model.account;

import com.google.firebase.firestore.auth.User;

import java.util.Map;

public class Account {
    private String uid;
    private String phoneNumber;
    private String email;
    private String username;
    private String dateOfBirth;
    private String address;
    private String gender;
    private boolean isAdmin;

    // Constructor, getter và setter
    public Account() {}

    public String getUid() {
        return uid;
    }

    public void setUid(String uid) {
        this.uid = uid;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getAddress() {
        return address;
    }
}
```

```

    public void setAddress(String address) {
        this.address = address;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(String dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public boolean isAdmin() {
        return isAdmin;
    }

    public void setAdmin(boolean admin) {
        isAdmin = admin;
    }

    public Account(String uid, String phoneNumber, String email, String
username, String dateOfBirth, String address, String gender) {
        this.uid = uid;
        this.phoneNumber = phoneNumber;
        this.email = email;
        this.username = username;
        this.dateOfBirth = dateOfBirth;
        this.address = address;
        this.gender = gender;
    }

    public String getRole() {
        return isAdmin ? "Admin" : "User";
    }
}

```

### o **EditAccountActivity.java**

```
package com.example.admincse441project.ui.accountmanagement.edit;

import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RadioGroup;
import android.widget.Switch;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.admincse441project.R;
import com.example.admincse441project.data.model.account.Account;
import com.example.admincse441project.utils.FirebaseUtils;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class EditAccountActivity extends AppCompatActivity{

    private EditText edtPhoneNumber, edtEmail, edtUsername,
    edtDateOfBirth, edtAddress;
    private Button btnSave;
    private String accountId;
    private String currentCollection;
    private RadioGroup radioGroupGender;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_account);

        edtPhoneNumber = findViewById(R.id.edtPhoneNumber);
        edtEmail = findViewById(R.id.edtEmail);
        edtUsername = findViewById(R.id.edtUsername);
        edtDateOfBirth = findViewById(R.id.edtDateOfBirth);
        edtAddress = findViewById(R.id.edtAddress);
        radioGroupGender = findViewById(R.id.radioGroupGender);
        btnSave = findViewById(R.id.btnSave);

        accountId = getIntent().getStringExtra("ACCOUNT_ID");
        Log.d("EditAccountActivity", "Received ACCOUNT_ID: " +
        accountId);

        if (accountId != null) {
            displayAccountInfo(accountId);
        } else {
            Toast.makeText(this, "Account ID not found",
            Toast.LENGTH_SHORT).show();
        }

        btnSave.setOnClickListener(v -> updateAccountInfo());
    }
}
```

```

        ImageView btnBack = findViewById(R.id.btnBack);
        btnBack.setOnClickListener(v -> onBackPressed());
    }

    private void displayAccountInfo(String accountId) {
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection("admin").document(accountId).get()
            .addOnSuccessListener(documentSnapshot -> {
                if (documentSnapshot.exists()) {
                    Account account =
documentSnapshot.toObject(Account.class);
                    if (account != null) {
                        populateFields(account);
                        currentCollection = "admin";
                    }
                } else {
                    db.collection("users").document(accountId).get()
                        .addOnSuccessListener(userDoc -> {
                            if (userDoc.exists()) {
                                Account account =
userDoc.toObject(Account.class);

                                if (account != null) {
                                    populateFields(account);
                                    currentCollection = "users";
                                    disableEditing();
                                }
                            } else {
                                Toast.makeText(this, "Account not
found", Toast.LENGTH_SHORT).show();
                            }
                        });
                }
            })
            .addOnFailureListener(e -> {
                Toast.makeText(this, "Error fetching account
details", Toast.LENGTH_SHORT).show();
                Log.e("EditAccountActivity", "Error fetching
account", e);
            });
    }

    private void disableEditing() {
        edtPhoneNumber.setEnabled(false);
        edtEmail.setEnabled(false);
        edtUsername.setEnabled(false);
        edtDateOfBirth.setEnabled(false);
        edtAddress.setEnabled(false);
        radioGroupGender.setEnabled(false);
        btnSave.setVisibility(View.GONE);
    }

    private void populateFields(Account account) {
        edtPhoneNumber.setText(account.getPhoneNumber());
        edtEmail.setText(account.getEmail());
        edtUsername.setText(account.getUsername());
        edtDateOfBirth.setText(account.getDateOfBirth());
        edtAddress.setText(account.getAddress());

        if ("Male".equals(account.getGender())) {
            radioGroupGender.check(R.id.rbMale);
        }
    }

```



```

    } else if ("Female".equals(account.getGender())) {
        radioButtonGender.check(R.id.rbFemale);
    } else {
        radioButtonGender.check(R.id.rbOther);
    }
}

private void updateAccountInfo() {
    String phoneNumber = edtPhoneNumber.getText().toString();
    String email = edtEmail.getText().toString();
    String username = edtUsername.getText().toString();
    String dateOfBirth = edtDateOfBirth.getText().toString();
    String address = edtAddress.getText().toString();

    String gender;
    int selectedGenderId =
radioButtonGender.getCheckedRadioButtonId();
    if (selectedGenderId == R.id.rbMale) {
        gender = "Male";
    } else if (selectedGenderId == R.id.rbFemale) {
        gender = "Female";
    } else {
        gender = "Other";
    }

    if (TextUtils.isEmpty(phoneNumber) || TextUtils.isEmpty(email) ||
TextUtils.isEmpty(username) || TextUtils.isEmpty(dateOfBirth)) {
        Toast.makeText(this, "Please fill in all fields",
Toast.LENGTH_SHORT).show();
        return;
    }

    FirebaseFirestore db = FirebaseFirestore.getInstance();

    db.collection(currentCollection).document(accountId).get().addOnSuccessListener(
documentSnapshot -> {
        if (documentSnapshot.exists()) {
            Map<String, Object> accountData = new
HashMap<>(documentSnapshot.getData());

            accountData.put("phoneNumber", phoneNumber);
            accountData.put("email", email);
            accountData.put("username", username);
            accountData.put("dateOfBirth", dateOfBirth);
            accountData.put("address", address);
            accountData.put("gender", gender);

            db.collection(currentCollection).document(accountId)
                .set(accountData)
                .addOnSuccessListener(aVoid -> {
                    Toast.makeText(this, "Account updated
successfully", Toast.LENGTH_SHORT).show();
                    setResult(RESULT_OK);
                    finish();
                })
                .addOnFailureListener(e -> Toast.makeText(this,
"Failed to update account", Toast.LENGTH_SHORT).show());
            } else {
                Toast.makeText(this, "Account data not found",
Toast.LENGTH_SHORT).show();
            }
        })
        .addOnFailureListener(e -> Toast.makeText(this, "Failed to

```

```

        fetch account data", Toast.LENGTH_SHORT).show());
    }

}

```

### o AccountAdapter.java

```

package com.example.admincse441project.ui.accountmanagement.showaccount;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.admincse441project.R;
import com.example.admincse441project.data.model.account.Account;

import java.util.List;

public class AccountAdapter extends
RecyclerView.Adapter<AccountAdapter.AccountViewHolder> {

    private List<Account> accountList;
    private OnItemClickListener listener;

    public interface OnItemClickListener {
        void onItemClick(Account account);
    }

    public void setOnItemClickListener(OnItemClickListener listener) {
        this.listener = listener;
    }

    public AccountAdapter(List<Account> accountList) {
        this.accountList = accountList;
    }

    @NonNull
    @Override
    public AccountViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_account_ro
w, parent, false);
        return new AccountViewHolder(view);
    }

    @Override
    public int getItemCount() {
        return accountList.size();
    }

    static class AccountViewHolder extends RecyclerView.ViewHolder {
        TextView tvNo, tvEmail, tvUsername, tvRole;

        public AccountViewHolder(@NonNull View itemView) {
            super(itemView);
            tvNo = itemView.findViewById(R.id.tvNo);
            tvEmail = itemView.findViewById(R.id.tvEmail);
            tvUsername = itemView.findViewById(R.id.tvUsername);

```

```

        tvRole = itemView.findViewById(R.id.tvRole);
    }
}

@Override
public void onBindViewHolder(@NonNull AccountViewHolder holder, int
position) {
    Account account = accountList.get(position);
    holder.tvNo.setText(String.valueOf(position + 1));
    holder.tvEmail.setText(account.getEmail());
    holder.tvUsername.setText(account.getUsername());
    holder.tvRole.setText(account.getRole());

    holder.itemView.setOnClickListener(v -> {
        if (listener != null) {
            listener.onItemClick(account);
        }
    });
}
}

```

### o AccountListFragment.java

```

package com.example.admncse441project.ui.accountmanagement.showaccount;

import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.admncse441project.R;
import com.example.admncse441project.data.model.account.Account;
import com.example.admncse441project.ui.accountmanagement.edit.EditAccountActiv
ity;
import com.example.admncse441project.utils.FirebaseUtils;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;

import java.util.ArrayList;
import java.util.List;

public class AccountListFragment extends Fragment {

    private RecyclerView recyclerView;
    private AccountAdapter accountAdapter;
    private List<Account> accountList;
    private EditText edtSearchAccount;

```

```

        private static final int EDIT_ACCOUNT_REQUEST_CODE = 1;

        @Nullable
        @Override
        public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {
            View view = inflater.inflate(R.layout.fragment_account_list,
            container, false);

            recyclerView = view.findViewById(R.id.recyclerView);
            edtSearchAccount = view.findViewById(R.id.edt_search_account2);

            recyclerView.setLayoutManager(new
            LinearLayoutManager(getContext()));
            accountList = new ArrayList<>();
            accountAdapter = new AccountAdapter(accountList);
            recyclerView.setAdapter(accountAdapter);

            loadAccounts("");

            edtSearchAccount.addTextChangedListener(new TextWatcher() {
                @Override
                public void beforeTextChanged(CharSequence s, int start, int
            count, int after) {}

                @Override
                public void onTextChanged(CharSequence s, int start, int
            before, int count) {
                    loadAccounts(s.toString().trim());
                }

                @Override
                public void afterTextChanged(Editable s) {}
            });

            accountAdapter.setOnItemClickListener(account -> {
                if (!account.isAdmin()) {
                    Toast.makeText(getContext(), "This account does not have
            permission to edit", Toast.LENGTH_SHORT).show();
                } else {
                    Intent intent = new Intent(getContext(),
            EditAccountActivity.class);
                    intent.putExtra("ACCOUNT_ID", account.getId());
                    Log.d("AccountListFragment", "Sending ACCOUNT_ID: " +
            account.getId());
                    startActivityForResult(intent,
            EDIT_ACCOUNT_REQUEST_CODE);
                }
            });

            return view;
        }

        private void loadAccounts(String query) {
            FirebaseFirestore db = FirebaseFirestore.getInstance();
            accountList.clear();

            db.collection("admin").get().addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (DocumentSnapshot document : task.getResult()) {
                        Account account = document.toObject(Account.class);
                        if (account != null) {

```

```

        account.setUid(document.getId());
        account.setAdmin(true);
        accountList.add(account);
    }
    accountAdapter.notifyDataSetChanged();
}
});

db.collection("users").get().addOnCompleteListener(userTask -> {
    if (userTask.isSuccessful() && userTask.getResult() != null)
    {
        for (DocumentSnapshot document : userTask.getResult()) {
            Account account = document.toObject(Account.class);
            if (account != null) {
                account.setUid(document.getId());
                account.setAdmin(false);
                accountList.add(account);
            }
        }
        accountAdapter.notifyDataSetChanged();
    }
});
}

@Override
public void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == EDIT_ACCOUNT_REQUEST_CODE && resultCode ==
AppCompatActivity.RESULT_OK) {
        loadAccounts("");
    }
}
}

```

- **App User**

- **ProfileMenuActivity.java**

```
package com.example.cse441_project.ui.profileoverlay;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.R;
import com.example.cse441_project.ui.home.HomeActivity;
import com.example.cse441_project.ui.profileoverlay.ProfileActivity;
import com.example.cse441_project.ui.auth.login.LoginActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

public class ProfileMenuActivity extends AppCompatActivity {

    private FirebaseAuth mAuth;
    private FirebaseFirestore db;
    private TextView tvHi, tvName;
    private Button btnProfile, btnSignOut;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_menu);

        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        tvHi = findViewById(R.id.tvHi);
        tvName = findViewById(R.id.tvName);
        btnProfile = findViewById(R.id.btnProfile);
        btnSignOut = findViewById(R.id.btnSignOut);

        fetchUsername();

        ImageView btnClose = findViewById(R.id.btnClose);
        btnClose.setOnClickListener(v -> onBackPressed());

        btnProfile.setOnClickListener(v -> {
            Intent intent = new Intent(ProfileMenuActivity.this,
ProfileActivity.class);
            startActivityForResult(intent, 1);
        });

        btnSignOut.setOnClickListener(v -> {
            mAuth.signOut();
            Intent intent = new Intent(ProfileMenuActivity.this,
LoginActivity.class);
            startActivity(intent);
            finish();
        });
    }
}
```

```

        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK) {
            String updatedUsername =
data.getStringExtra("updatedUsername");
            if (updatedUsername != null) {
                tvName.setText(updatedUsername);
            }
        }
    }

    private void fetchUsername() {
        String uid = mAuth.getCurrentUser().getUid();
        Log.d("ProfileMenuActivity", "Current User UID: " + uid);

        db.collection("users").whereEqualTo("uid", uid)
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful() &&
!task.getResult().isEmpty()) {
                    DocumentSnapshot document =
task.getResult().getDocuments().get(0);
                    String username = document.getString("username");
                    Log.d("ProfileMenuActivity", "Username from
users: " + username);
                    tvName.setText(username);
                } else {
                    db.collection("admin").whereEqualTo("uid", uid)
                        .get()
                        .addOnCompleteListener(adminTask -> {
                            if (adminTask.isSuccessful() &&
!adminTask.getResult().isEmpty()) {
                                DocumentSnapshot document =
adminTask.getResult().getDocuments().get(0);
                                String username =
document.getString("username");
                                Log.d("ProfileMenuActivity",
"Username from admin: " + username);
                                tvName.setText(username);
                            } else {
                                Log.d("ProfileMenuActivity", "No
matching document in admin collection.");
                            }
                        });
                });
            });
    }
}

```

## o ProfileActivity.java

```
package com.example.cse441_project.ui.profileoverlay;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.R;
import com.example.cse441_project.data.model.user.User;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

public class ProfileActivity extends AppCompatActivity {

    private FirebaseAuth mAuth;
    private FirebaseFirestore db;
    private EditText edtUsername, edtEmail, edtPhoneNumber, edtAddress,
    edtDateOfBirth;
    private RadioGroup radioGroupGender;
    private RadioButton rbMale, rbFemale, rbOther;
    private Button btnSave, btnDelete, btnChangePassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        edtUsername = findViewById(R.id.edtUsername);
        edtEmail = findViewById(R.id.edtEmail);
        edtPhoneNumber = findViewById(R.id.edtPhoneNumber);
        edtAddress = findViewById(R.id.edtAddress);
        edtDateOfBirth = findViewById(R.id.edtDateOfBirth);
        radioGroupGender = findViewById(R.id.radioGroupGender);
        rbMale = findViewById(R.id.rbMale);
        rbFemale = findViewById(R.id.rbFemale);
        rbOther = findViewById(R.id.rbOther);
        btnSave = findViewById(R.id.btnSave);
        btnDelete = findViewById(R.id.btnDeleteAccount);
        btnChangePassword = findViewById(R.id.btnChangePassword);

        loadUserData();

        btnSave.setOnClickListener(v -> {
            if (validateFields()) {
                saveUserData();
            }
        });

        btnDelete.setOnClickListener(v -> {
            Intent intent = new Intent(ProfileActivity.this,
```



```

DeleteAccountActivity.class);
        startActivity(intent);
    });

    btnChangePassword.setOnClickListener(v -> {
        Intent intent = new Intent(ProfileActivity.this,
ChangePasswordActivity.class);
        startActivity(intent);
    });

    ImageView btnBack = findViewById(R.id.btnBack);
    btnBack.setOnClickListener(v -> onBackPressed());
}

private void loadUserData() {
    String uid = mAuth.getCurrentUser().getUid();

    db.collection("users").document(uid)
        .get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful() && task.getResult().exists())
{
                populateFields(task.getResult());
            } else {
                db.collection("admin").document(uid)
                    .get()
                    .addOnCompleteListener(adminTask -> {
                        if (adminTask.isSuccessful() &&
adminTask.getResult().exists()) {
                            populateFields(adminTask.getResult());
                        } else {
                            Log.d("ProfileActivity", "User
data not found.");
                            Toast.makeText(this, "User data
not found.", Toast.LENGTH_SHORT).show();
                        }
                    });
            }
        });
}

private void populateFields(DocumentSnapshot document) {
    edtUsername.setText(document.getString("username"));
    edtEmail.setText(document.getString("email"));
    edtPhoneNumber.setText(document.getString("phoneNumber"));
    edtAddress.setText(document.getString("address"));
    edtDateOfBirth.setText(document.getString("dateOfBirth"));

    String gender = document.getString("gender");
    if (gender != null) {
        switch (gender) {
            case "Male":
                rbMale.setChecked(true);
                break;
            case "Female":
                rbFemale.setChecked(true);
                break;
            case "Other":
                rbOther.setChecked(true);
                break;
        }
    }
}

```

```

    }

    private boolean validateFields() {
        if (TextUtils.isEmpty(edtUsername.getText())) {
            edtUsername.setError("Username cannot be empty");
            return false;
        }
        if (TextUtils.isEmpty(edtEmail.getText())) {
            edtEmail.setError("Email cannot be empty");
            return false;
        }
        if (TextUtils.isEmpty(edtPhoneNumber.getText())) {
            edtPhoneNumber.setError("Phone number cannot be empty");
            return false;
        }
        if (TextUtils.isEmpty(edtDateOfBirth.getText())) {
            edtDateOfBirth.setError("Date of birth cannot be empty");
            return false;
        }
        return true;
    }

    private void saveUserData() {
        String uid = mAuth.getCurrentUser().getUid();
        String username = edtUsername.getText().toString().trim();
        String email = edtEmail.getText().toString().trim();
        String phoneNumber = edtPhoneNumber.getText().toString().trim();
        String address = edtAddress.getText().toString().trim();
        String dateOfBirth = edtDateOfBirth.getText().toString().trim();

        String gender = "";
        int selectedGenderId =
radioGroupGender.getCheckedRadioButtonId();
        if (selectedGenderId == rbMale.getId()) {
            gender = "Male";
        } else if (selectedGenderId == rbFemale.getId()) {
            gender = "Female";
        } else if (selectedGenderId == rbOther.getId()) {
            gender = "Other";
        }

        User updatedUser = new User(uid, email, username, phoneNumber,
null, dateOfBirth);
        updatedUser.setAddress(address);
        updatedUser.setGender(gender);

        db.collection("users").document(uid)
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful() && task.getResult().exists())
{
db.collection("users").document(uid).set(updatedUser)
                    .addOnSuccessListener(aVoid -> {
                        Toast.makeText(ProfileActivity.this,
"Saved successfully!", Toast.LENGTH_SHORT).show();
                        Intent resultIntent = new Intent();

resultIntent.putExtra("updatedUsername", username);
                        setResult(RESULT_OK, resultIntent);
                        finish(); // Close this activity
                    })
                    .addOnFailureListener(e -> {

```

```

        Log.e("ProfileActivity", "Error
saving data: " + e.getMessage());
        Toast.makeText(ProfileActivity.this,
"Error saving data.", Toast.LENGTH_SHORT).show();
    });
    } else {

db.collection("admin").document(uid).set(updatedUser)
        .addOnSuccessListener(aVoid -> {
            Toast.makeText(ProfileActivity.this,
"Saved successfully!", Toast.LENGTH_SHORT).show();
            Intent resultIntent = new Intent();

resultIntent.putExtra("updatedUsername", username);
            setResult(RESULT_OK, resultIntent);
            finish(); // Close this activity
        })
        .addOnFailureListener(e -> {
            Log.e("ProfileActivity", "Error
saving data: " + e.getMessage());
            Toast.makeText(ProfileActivity.this,
"Error saving data.", Toast.LENGTH_SHORT).show();
        });
    }
});
}
}

```

### o **ChangePasswordActivity.java**

```

package com.example.cse441_project.ui.profileoverlay;

import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.R;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class ChangePasswordActivity extends AppCompatActivity {

    private EditText etCurrentPassword, etNewPassword, etConfirmPassword;
    private Button btnChangePassword;
    private ImageView btnBack;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_password);

        mAuth = FirebaseAuth.getInstance();

        etCurrentPassword = findViewById(R.id.etCurrentPassword);

```

```

        etNewPassword = findViewById(R.id.etNewPassword);
        etConfirmPassword = findViewById(R.id.etConfirmPassword);
        btnChangePassword = findViewById(R.id.btnChange);
        btnBack = findViewById(R.id.btnBack); // Nút quay lại

        btnChangePassword.setOnClickListener(v -> changePassword());

        btnBack.setOnClickListener(v -> onBackPressed());
    }

    private void changePassword() {
        String currentPassword =
etCurrentPassword.getText().toString().trim();
        String newPassword = etNewPassword.getText().toString().trim();
        String confirmPassword =
etConfirmPassword.getText().toString().trim();

        if (TextUtils.isEmpty(currentPassword)) {
            etCurrentPassword.setError("Current password is required");
            return;
        }
        if (TextUtils.isEmpty(newPassword)) {
            etNewPassword.setError("New password is required");
            return;
        }
        if (TextUtils.isEmpty(confirmPassword)) {
            etConfirmPassword.setError("Please confirm your new
password");
            return;
        }

        if (newPassword.length() < 8) {
            etNewPassword.setError("New password must be at least 8
characters");
            return;
        }

        if (!newPassword.equals(confirmPassword)) {
            etConfirmPassword.setError("Passwords do not match");
            return;
        }
        if (newPassword.equals(currentPassword)) {
            etNewPassword.setError("New password cannot be the same as
the current password");
            return;
        }

        FirebaseUser user = mAuth.getCurrentUser();
        if (user != null && user.getEmail() != null) {
            AuthCredential credential =
EmailAuthProvider.getCredential(user.getEmail(), currentPassword);

            user.reauthenticate(credential).addOnCompleteListener(task ->
{
                if (task.isSuccessful()) {
                    Log.d("ChangePassword", "Reauthentication
successful");
                }

                user.updatePassword(newPassword).addOnCompleteListener(updateTask -> {
                    if (updateTask.isSuccessful()) {
                        Toast.makeText(ChangePasswordActivity.this,
"Password changed successfully", Toast.LENGTH_SHORT).show();
                        finish();
                    }
                });
            });
        }
    }

```

```

        } else {
            Toast.makeText(ChangePasswordActivity.this,
                "Password change failed", Toast.LENGTH_SHORT).show();
            Log.e("ChangePassword", "Password update
failed", updateTask.getException());
        }
    });
} else {
    Toast.makeText(ChangePasswordActivity.this,
        "Authentication failed. Please check your current password.",
        Toast.LENGTH_SHORT).show();
    Log.e("ChangePassword", "Reauthentication failed",
        task.getException());
}
});
}
}
}
}

```

### o DeleteAccountActivity.java

```

package com.example.cse441_project.ui.profileoverlay;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.R;

public class DeleteAccountActivity extends AppCompatActivity {

    private Button btnDelete, btnCancel;
    private ImageView btnBack;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_delete_account);

        btnDelete = findViewById(R.id.btnDelete);
        btnCancel = findViewById(R.id.btnCancel);
        btnBack = findViewById(R.id.btnBack);

        btnBack.setOnClickListener(v -> onBackPressed());

        btnCancel.setOnClickListener(v -> onBackPressed());

        btnDelete.setOnClickListener(v -> {
            Intent intent = new Intent(DeleteAccountActivity.this,
                AcceptDeleteAccountActivity.class);
            startActivity(intent);
        });
    }
}

```

### o AcceptDeleteAccountActivity.java

```
package com.example.cse441_project.ui.profileoverlay;

import static android.content.ContentValues.TAG;

import static androidx.core.content.ContextCompat.startActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cse441_project.R;
import com.example.cse441_project.ui.auth.login.LoginActivity;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;

public class AcceptDeleteAccountActivity extends AppCompatActivity {

    private EditText edtPassword;
    private Button btnDelete;
    private ImageView btnBack;
    private FirebaseAuth mAuth;
    private FirebaseFirestore db;
    private static final String TAG = "DeleteAccount";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_accept_delete_account);

        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        edtPassword = findViewById(R.id.password);
        btnDelete = findViewById(R.id.btnDelete);

        btnDelete.setOnClickListener(v -> deleteAccount());

        btnBack = findViewById(R.id.btnBack);
        btnBack.setOnClickListener(v -> onBackPressed());

        Log.d(TAG, "onCreate: AcceptDeleteAccountActivity started");
    }

    private void deleteAccount() {
        String password = edtPassword.getText().toString().trim();

        if (TextUtils.isEmpty(password)) {
            edtPassword.setError("Password is required");
            Toast.makeText(this, "Please enter your password",
                Toast.LENGTH_SHORT).show();
            return;
        }
    }
}
```

```

        FirebaseAuth user = mAuth.getCurrentUser();
        if (user == null) {
            Log.e("DeleteAccount", "No user logged in");
            Toast.makeText(this, "No user logged in",
                Toast.LENGTH_SHORT).show();
            return;
        }

        Log.d("DeleteAccount", "Starting reauthentication process");
        AuthCredential credential =
            EmailAuthProvider.getCredential(user.getEmail(),
                edtPassword.getText().toString().trim());

        user.reauthenticate(credential).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Log.d("DeleteAccount", "Reauthentication successful");
                deleteUserFromFirestore(user.getId()); // Tiếp tục với
việc xóa Firestore
            } else {
                Log.e("DeleteAccount", "Reauthentication failed",
                    task.getException());
                Toast.makeText(DeleteAccountActivity.this,
                    "Incorrect password", Toast.LENGTH_SHORT).show();
            }
        });

        private void deleteUserFromFirestore(String uid) {
            Log.d("DeleteAccount", "Attempting to delete user data from
Firestore for UID: " + uid);

            db.collection("users").document(uid).get().addOnCompleteListener(userTask
            -> {
                if (userTask.isSuccessful() && userTask.getResult().exists())
                {
                    // Nếu tồn tại trong "users", tiến hành xóa
                    db.collection("users").document(uid).delete()
                        .addOnSuccessListener(aVoid -> {
                            Log.d("DeleteAccount", "User data deleted
from 'users' collection.");
                            deleteUserAuth(); // Tiếp tục xóa trong
Firestore Auth
                        })
                        .addOnFailureListener(e -> {
                            Log.e("DeleteAccount", "Failed to delete user
data from 'users' collection", e);
                            Toast.makeText(this, "Failed to delete
account data", Toast.LENGTH_SHORT).show();
                        });
                } else {
                    Log.d("DeleteAccount", "User not found in 'users',
checking 'admin' collection.");

                    db.collection("admin").document(uid).get().addOnCompleteListener(adminTask
                    -> {
                        if (adminTask.isSuccessful() &&
                            adminTask.getResult().exists()) {
                            // Nếu tồn tại trong "admin", tiến hành xóa
                            db.collection("admin").document(uid).delete()
                                .addOnSuccessListener(aVoid -> {
                                    Log.d("DeleteAccount", "User data

```

```

deleted from 'admin' collection.");
                                deleteUserAuth(); // Tiếp tục xóa
                                trong Firebase Auth
                                })
                                .addOnFailureListener(e -> {
                                    Log.e("DeleteAccount", "Failed to
delete account data from 'admin' collection", e);
                                    Toast.makeText(this, "Failed to
delete account data", Toast.LENGTH_SHORT).show();
                                });
                                } else {
                                    Log.e("DeleteAccount", "User not found in both
'users' and 'admin' collections.");
                                    Toast.makeText(this, "User data not found",
Toast.LENGTH_SHORT).show();
                                }
                            });
                        }
                    });
                }

private void deleteUserAuth() {
    FirebaseUser user = mAuth.getCurrentUser();
    if (user != null) {
        Log.d("DeleteAccount", "Deleting user from Firebase Auth");
        user.delete().addOnCompleteListener(deleteTask -> {
            if (deleteTask.isSuccessful()) {
                Log.d("DeleteAccount", "User deletion from Firebase
Auth successful");
                onAccountDeleted();
            } else {
                Log.e("DeleteAccount", "Account deletion from
Firebase Auth failed", deleteTask.getException());
                Toast.makeText(AcceptDeleteAccountActivity.this,
"Account deletion failed", Toast.LENGTH_SHORT).show();
            }
        });
    } else {
        Log.e("DeleteAccount", "User is null during deleteUserAuth");
    }
}

private void onAccountDeleted() {
    Log.d("DeleteAccount", "Account deleted successfully, navigating
to login screen.");
    Toast.makeText(AcceptDeleteAccountActivity.this, "Account deleted
successfully", Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(AcceptDeleteAccountActivity.this,
LoginActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
    finish();
}
}

```