NGUYỄN ĐỖ TÚ MAI

N23DCPT091

D23CQPTTK01-N

CÂU HỎI CỦNG CỐ LÍ THUYẾT BÀI 6 – PHÂN TÍCH – ANALYSIS

(LECTURE 5&6)

I) CÂU HỎI TRẮC NGHIỆM

1. **C** – Lớp thực thể
2. **C** – Giao tiếp với người dùng/hệ thống ngoài (boundary)
3. **D** – Inheritance (kế thừa/tổng quát hóa)
4. **B** – Các lớp và quan hệ giữa các lớp
5. **B** – Include: UC phải gọi UC khác để hoàn thành
6. **B** – Scenario = kịch bản tương tác người dùng–hệ thống
7. **A** – Aggregation (chứa–thuộc, phần có thể tồn tại độc lập)
8. **B** – Thứ tự thông điệp giữa các đối tượng (Sequence)
9. **C** – Control (Controller trong MVC)
10. **C** – Sơ đồ lớp

II) CÂU HỎI NGẮN

1. **Lớp thực thể**: lớp chứa dữ liệu & trạng thái nghiệp vụ; thường ánh xạ DB.
2. **Lớp điều khiển**: điều phối luồng nghiệp vụ, nhận yêu cầu từ lớp biên, gọi entity/DAO.
3. **Scenario**: kịch bản mô tả *pre/post*, luồng chuẩn & ngoại lệ khi thực thi use case.
4. **Include**: UC A bao gồm UC B (bắt buộc gọi) để hoàn tất.
5. **Mục đích sơ đồ lớp**: mô hình cấu trúc tĩnh hệ thống (lớp/thuộc tính/phương thức/quan hệ).
6. **Aggregation vs Composition**: Aggregation — phần tử tồn tại độc lập; Composition — vòng đời phụ thuộc "toàn–phần".
7. **Sơ đồ tuần tự**: trình tự thông điệp giữa các đối tượng theo thời gian.
8. **Extend**: UC A mở rộng UC B, chỉ xảy ra khi điều kiện kích hoạt thỏa.

9. **Lớp biên**: giao tiếp UI/API, nhận/hiển thị dữ liệu, không chứa nghiệp vụ lõi.
10. **Sơ đồ cộng tác**: nhấn mạnh cấu trúc liên kết giữa đối tượng và đánh số thông điệp (thứ tự tương tác).

III) CÂU HỎI THẢO LUẬN NHÓM

1. **Thực thể–Biên–Điều khiển**: dữ liệu | giao tiếp | điều phối – tách trách nhiệm, giảm coupling.
2. **Agg vs Comp**: chọn Comp khi vòng đời phụ thuộc mạnh (Order–OrderItem).
3. **Tầm quan trọng sơ đồ lớp**: làm "hợp đồng" cấu trúc; nền ánh xạ DB & code; hỗ trợ estimate.
4. **Sequence vs Collaboration**: Sequence = *thời gian*; Collaboration = *liên kết + số thông điệp*.
5. **Controller trong MVC**: nối View–Model, gom rule, giữ UI mỏng.
6. **Vì sao viết scenario**: chống mơ hồ; đặt tiêu chí chấp nhận; bám test/QA.
7. **Trích đủ use case**: ma trận actor–mục tiêu, workshop, kiểm luồng ngoại lệ/biên.
8. **Use case & Scenario**: UC = mục tiêu/chức năng; Scenario = cách UC vận hành cụ thể.
9. **Ưu/nhược Sequence**: rõ flow & lỗi; nhưng dài, nên chia nhỏ theo ngữ cảnh.
10. **Nâng chất lượng scenario**: format chuẩn (Pre/Post/Main/Alt), từ ngữ rõ, số bước, review với stakeholder.

IV) CÂU HỎI TÌNH HUỐNG

1) Yêu cầu mới sau khi đã viết xong scenario (QL thư viện)

- Cách làm: mở quy trình Change Request → phân loại (CN/Phi CN) → phân tích tác động tới UC/Scenario/Lớp/CSDL → ước lượng & ưu tiên lại backlog/sprint → họp xác nhận với KH.
- Cập nhật: SRS, danh sách UC, Scenario V2 (main/alt), Sequence/Collaboration, Class diagram, test case truy vết theo yêu cầu.

2) Khó xác định lớp điều khiển

- Cách làm: bám Use Case: "mỗi UC → ≥1 Control" điều phối giữa Boundary và Entity. Dùng động từ trong UC để đặt tên (OrderController, EnrollController…). Vẽ nhanh Sequence để lộ Control; tách Control nếu flow quá dài.
- Cập nhật: Danh sách lớp BCE, Sequence minh họa 1–2 UC, quy tắc đặt trách nhiệm (không nhét nghiệp vụ vào Boundary/Entity).

3) Xong Class Diagram rồi phát sinh chức năng mới

- Cách làm: quay lại UC → thêm/sửa UC, rà lại quan hệ Include/Extend → từ UC mới trích lớp (BCE) → điều chỉnh quan hệ (association/aggregation/composition/inheritance) và multiplicity → xem ảnh hưởng DB.
- Cập nhật: Class diagram phiên bản mới, ERD/DDL nếu có, Sequence mới, danh mục thay đổi (changelog).

4) "Đăng ký khóa học" có nhiều ngoại lệ

- Cách làm: viết Scenario Chuẩn (đánh số bước) rồi liệt kê Alt/Exception dạng [A1], [A2] tham chiếu bước gốc (vd: A1 tại bước 3: "Môn đã đầy"; A2: "Không đủ điều kiện tiên quyết"…). Ngoại lệ lớn tách thành UC Extend hoặc Include ("Kiểm tra tiên quyết", "Thanh toán học phí").
- Cập nhật: Scenario V2 (Pre/Post/Main/Alt), ma trận bước ↔ thông điệp Sequence, UC Diagram có Include/Extend.

5) Sequence có đối tượng vai trò mờ

- Cách làm: gán stereotype BCE cho từng lifeline; hợp nhất đối tượng trùng vai; đổi tên theo trách nhiệm; thêm guard [điều kiện] cho nhánh; xóa đối tượng không gửi/nhận thông điệp.
- Cập nhật: Sequence chỉnh sửa + chú giải vai trò, bảng ánh xạ Lifeline ↔ Lớp BCE.

6) Quan hệ lớp bị sai

- Cách làm: rà lại ngữ nghĩa quan hệ:
    - Aggregation (thoi rỗng) = "có–thuộc", phần tồn tại độc lập.
    - Composition (thoi đặc) = vòng đời phụ thuộc.
    - Inheritance = tổng quát hóa.
    - Kiểm multiplicity, tên quan hệ, chiều điều hướng. Dùng CRC cards/scenario để kiểm.

- Cập nhật: Class diagram sửa kí hiệu + multiplicity; checklist quy tắc chọn quan hệ.

7) Khó mô tả quan hệ giữa các Use Case

- Cách làm:
    - Include khi hành vi bắt buộc, tái dùng (Xác thực, Tính phí).
    - Extend khi tùy điều kiện (Khuyến mãi, Hủy đăng ký).
    - Generalization cho phân loại (Đăng nhập {SV/GV}).
      Lập bảng UC ↔ hành vi dùng chung để quyết định.
- Cập nhật: UC Diagram với Include/Extend/Generalization + ghi chú điều kiện kích hoạt.

8) Xác định lớp biên cho hệ thống bán hàng

- Đề xuất (Boundary): CustomerUI (đặt hàng/đăng ký/đăng nhập), CheckoutUI, AdminUI (QL sản phẩm/đơn), PaymentAPIAdapter, ShippingAPIAdapter.
- Nguyên tắc: Boundary nhận/hiển thị dữ liệu, không chứa logic nghiệp vụ; mọi xử lý qua Control.
- Cập nhật: Danh sách Boundary + giao tiếp vào/ra, sơ đồ gắn BCE (kèm 1 Sequence đặt hàng ngắn).

9) KH yêu cầu thêm chức năng sau khi hoàn thiện scenario

- Cách làm: mở CR, phân tích tác động, cập nhật UC/Scenario V2, sửa Class/Sequence/Collab, thêm test & tiêu chí chấp nhận; thương lượng phạm vi/thời gian/chi phí.
- Cập nhật: SRS vN+1, backlog/sprint plan, truy vết yêu cầu → artefact.

10) Collaboration diagram có tương tác sai

- Cách làm: tái dựng từ Scenario: liệt kê thông điệp theo thứ tự, đánh số 1, 1.1, 2… trên liên kết; bỏ liên kết không có thông điệp; đảm bảo mỗi thông điệp có người gửi–nhận hợp lệ; đối chiếu với Sequence để đồng bộ.
- Cập nhật: Collaboration sửa số thứ tự thông điệp + bảng đối chiếu với Sequence.

I. CÂU HỎI TRẮC NGHIỆM (Multiple Choice Questions)

1. **C** – Pha lấy yêu cầu
2. **C** – mô hình Agile
3. **C** – Quan sát
4. **C** – 3NF
5. **B** – Sơ đồ tuần tự
6. **B** – Sơ đồ tuần tự
7. **C** – Cải tiến quy trình (CMMI Level 5 – Optimizing)
8. **A** – Linh hoạt & thay đổi nhanh
9. **A** – DRY: hạn chế viết lặp
10. **B** – Đóng gói & tái sử dụng là then chốt khi thiết kế lớp
11. **C** – Kiểm thử có thể chạy song song với phát triển

II. CÂU HỎI TRẢ LỜI NGẮN (Short Answer Questions)

1. Phần mềm & công nghệ phần mềm
   Phần mềm = chương trình + dữ liệu + tài liệu vận hành. Công nghệ phần mềm là tập hợp quy trình/phương pháp/công cụ để xác định–mô hình hóa–đặc tả–thẩm định yêu cầu và phát triển, vận hành, bảo trì SP (đầu ra quan trọng là SRS).

2. Các mô hình vòng đời phổ biến (rất ngắn):
   Waterfall (tuần tự), V-Model (phát triển song hành kiểm thử), Spiral (lặp + quản rủi ro), Agile/Scrum (lặp ngắn, giao hàng sớm).

3. Ba loại yêu cầu chính:

- Chức năng: hệ thống làm gì (UC, luồng NV).
- Phi chức năng: hiệu năng, bảo mật, khả dụng, tuân thủ…
- Ràng buộc/miền: luật, nền tảng, chuẩn tích hợp.

4. Vai trò sơ đồ lớp UML: mô hình hóa cấu trúc tĩnh – lớp/thuộc tính/phương thức/quan hệ – làm nền cho thiết kế & ánh xạ CSDL/triển khai.

5. Vì sao kiểm thử quan trọng: phát hiện lỗi sớm, xác nhận yêu cầu được hiểu đúng, giảm chi phí sửa đổi về sau.

6. SOLID: S-Single Responsibility; O-Open/Closed; L-Liskov Substitution; I-Interface Segregation; D-Dependency Inversion.

7. Kiểm thử hộp đen vs hộp trắng:
   Hộp đen: dựa input/output, không cần biết code. Hộp trắng: dựa cấu trúc bên trong (nhánh/đường đi/điều kiện).

8. Thiết kế CSDL từ sơ đồ lớp (quy trình ngắn):
   Ánh xạ lớp→bảng, thuộc tính→cột, quan hệ→FK/bảng ghép; đặt khóa; chuẩn hóa đến 3NF để loại phụ thuộc bắc cầu.
9. Ba ưu điểm của Agile: phản hồi thay đổi nhanh, giao hàng liên tục/giá trị sớm, tăng hợp tác & minh bạch.
10. Chuẩn hóa CSDL – các giai đoạn chính: 1NF (nguyên tử) → 2NF (loại phụ thuộc bộ phận) → 3NF (loại phụ thuộc bắc cầu) → (BCNF nếu cần).

III. CÂU HỎI THẢO LUẬN NHÓM (Discussion Questions)

1. **Agile vs Waterfall:** Agile lặp ngắn, ưu tiên giá trị/feedback; Waterfall tuần tự, hợp yêu cầu ổn định & tuân thủ.
2. **Lợi ích UML:** ngôn ngữ chung; giảm mơ hồ; làm tài liệu thiết kế/trao đổi; hỗ trợ kiểm tra & bảo trì.
3. **Bảo trì tốt:** tách BCE/MVC, áp dụng SOLID/DRY, test tự động, tài liệu sống, module hóa.
4. **Vì sao phát triển lặp:** giảm rủi ro, nhận phản hồi sớm, ưu tiên tính năng giá trị cao.
5. **Vai trò kiến trúc:** định nghĩa cấu trúc & chất lượng (scalability/security), cho phép phát triển song song.
6. **Đáp ứng bảo mật:** threat modeling, tiêu chuẩn OWASP, kiểm thử bảo mật, quản lý secret, log & giám sát.
7. **Unit vs Integration:** Unit cô lập 1 lớp/hàm; Integration kiểm tương tác giữa module/DB/API.
8. **Thách thức thu thập yêu cầu:** nhiều stakeholder, thuật ngữ mơ hồ, thay đổi liên tục → giải pháp: phỏng vấn/quan sát, danh mục từ khóa & thống nhất thuật ngữ.
9. **Áp dụng Scrum thực tế:** backlog ưu tiên, sprint 1-2 tuần, review+retro, DoD rõ ràng.
10. **Vì sao tối ưu mã nguồn quan trọng:** hiệu năng, chi phí hạ tầng, trải nghiệm người dùng, dễ bảo trì/mở rộng.

IV. CÂU HỎI TÌNH HUỐNG

1. **KH đổi yêu cầu liên tục: dùng quy trình Change Request;** phân tích tác động, ước lượng, tái ưu tiên backlog/scope theo sprint; xác nhận lại bằng tài liệu.

2. **Bug nghiêm trọng nhưng lead không sửa:** nêu rủi ro/ảnh hưởng nghiệp vụ & bảo mật; đề xuất fix/mitigation; ghi issue; cần thì *escalate*; thống nhất lại tiêu chí chấp nhận.
3. **Chậm tiến độ do đổi yêu cầu:** chốt scope iteration, giảm phạm vi giao hàng, bổ sung buffer, kiểm soát CR, minh bạch với KH.
4. **Thiết kế CSDL TMĐT không dư thừa:** mô hình ER đúng, khóa & quan hệ FK, bảng ghép n-n, chuẩn hóa đến 3NF/BCNF; index hợp lý.
5. **Chọn mô hình cho startup:** Agile (biến động cao, cần time-to-market nhanh & feedback sớm).
6. **Tối ưu hiệu suất:** đo (profiling/APM) → tìm nút thắt → cache; tối ưu truy vấn/index; batch/async; scale ngang; load test.
7. **Nhiều lỗi bảo mật nhưng không muốn gián đoạn:** vá theo mức rủi ro; bật WAF/rate-limit; TLS/secret mgmt; log & alert; triển khai dần bằng feature toggle.
8. **Nhiều ý kiến trái chiều về triển khai:** đặt tiêu chí quyết định; làm POC/A-B test; quyết định bởi PO/board kiến trúc; dựa dữ liệu.
9. **Bảo đảm UI thân thiện:** guideline UX & accessibility; test người dùng; tiêu chí chấp nhận UI; thiết kế phản hồi rõ ràng.
10. **Dự án cũ không tài liệu:** đọc mã/DB/log, phỏng vấn vận hành, dựng tài liệu tối thiểu (SRS rút gọn/ADR), viết testcase hồi quy rồi tiếp tục phát triển.

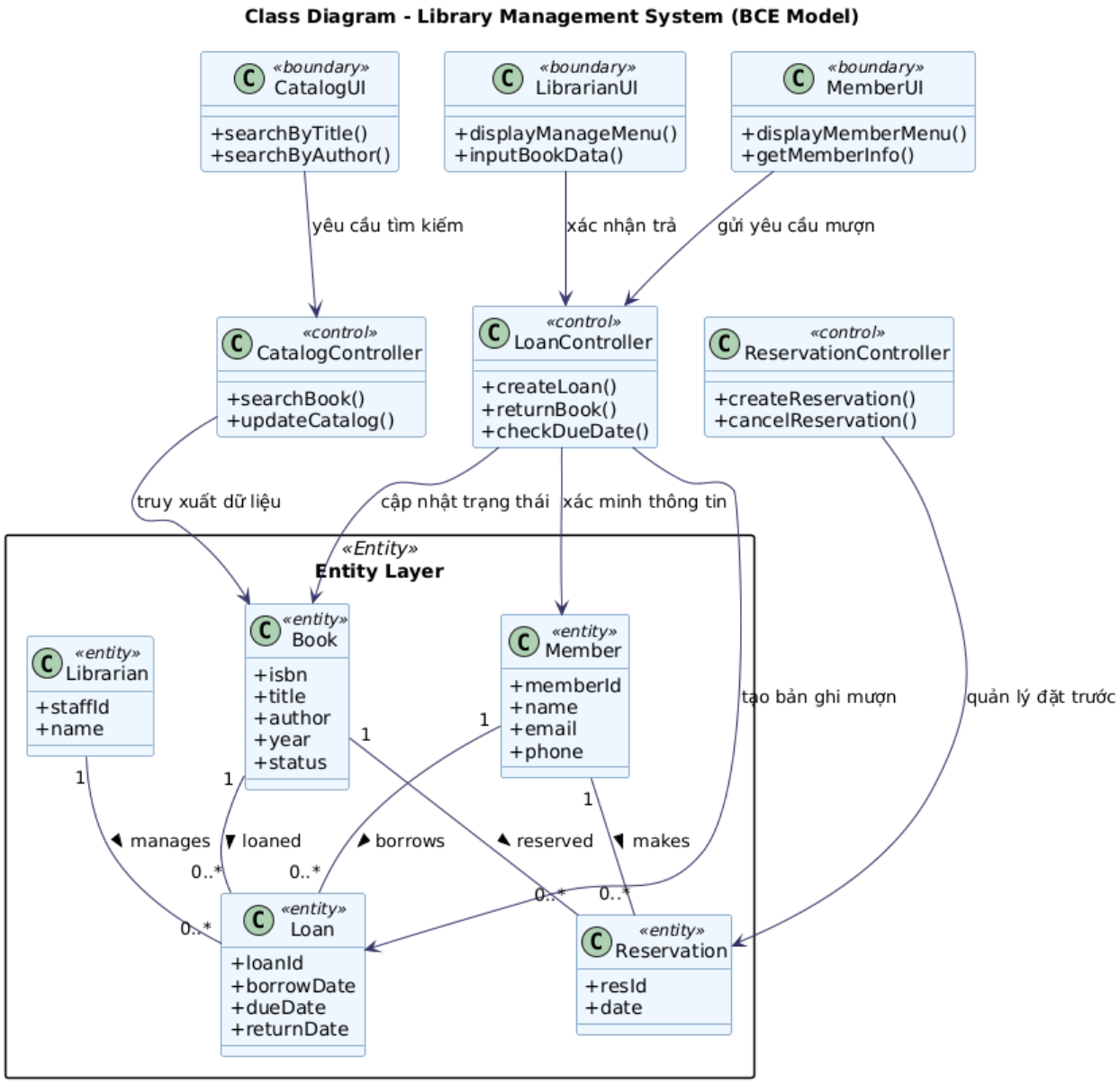
V. BÀI TẬP THỰC HÀNH

1) Java OOP – Quản lý sinh viên

```
// Student.java
public class Student {
  private String id, name; private int year;
  public Student(String id,String name,int
year){this.id=id;this.name=name;this.year=year;}
  public String getId(){return id;} public String getName(){return name;}
  public int getYear(){return year;} public void setYear(int y){year=y;}
}
// StudentService.java
import java.util.*;
```

```
public class StudentService{
  private final Map<String,Student> store=new HashMap<>();
  public void add(Student s){store.put(s.getId(),s);}
  public Student find(String id){return store.get(id);}
  public List<Student> all(){return new ArrayList<>(store.values());}
  public void remove(String id){store.remove(id);}
}
// Main.java
public class Main{
  public static void main(String[] args){
    StudentService svc=new StudentService();
    svc.add(new Student("SV01","Mai",2023));
    System.out.println("Total: "+svc.all().size());
  }
}
```

2) Sơ đồ lớp UML – Hệ thống quản lý thư viện (PlantUML)



Sơ đồ lớp hệ thống Quản lý Thư viện (Mô hình BCE)

3) Test case – Đăng ký tài khoản

ID	Tiền điều kiện	Bước	Kỳ vọng
TC01	–	Nhập hợp lệ → Submit	Thành công + email xác nhận
TC02	–	Bỏ trống email	Báo “Email bắt buộc”
TC03	–	Email sai định dạng	Báo “Email không hợp lệ”
TC04	–	Mật khẩu yếu	Báo “Mật khẩu yếu”

TC05	Tài khoản tồn tại	Dùng email đã đăng ký	Báo “Email đã tồn tại”

4) CSDL bán hàng (đến 3NF) – lược đồ + DDL

Bảng: Category, Product(cat_id FK), Customer, "Order"(cust_id FK), OrderItem(order_id, prod_id PK), Payment(order_id FK), Shipment(order_id FK).

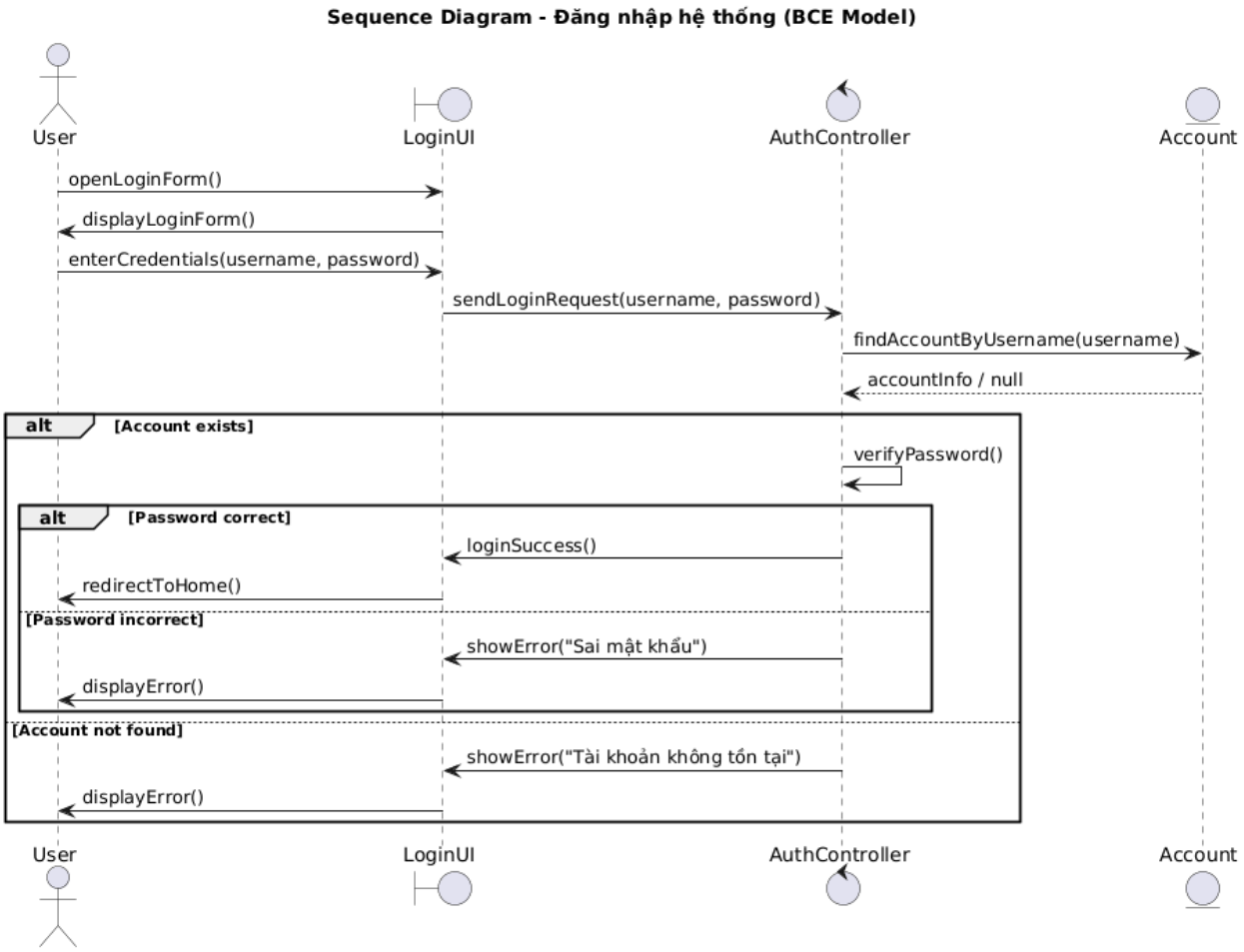
```
CREATE TABLE Category(cat_id INT PRIMARY KEY, name VARCHAR(100));

CREATE TABLE Product(prod_id INT PRIMARY KEY, cat_id INT REFERENCES
Category(cat_id),

  name VARCHAR(150), price DECIMAL(12,2), stock INT);

CREATE TABLE Customer(cust_id INT PRIMARY KEY, name VARCHAR(120), email
VARCHAR(120) UNIQUE, phone VARCHAR(20));

CREATE TABLE "Order"(order_id INT PRIMARY KEY, cust_id INT REFERENCES
Customer(cust_id),

  order_date TIMESTAMP, status VARCHAR(20), total DECIMAL(12,2));

CREATE TABLE OrderItem(order_id INT REFERENCES "Order"(order_id),

  prod_id INT REFERENCES Product(prod_id), qty INT, unit_price DECIMAL(12,2),

  PRIMARY KEY(order_id, prod_id));

CREATE TABLE Payment(pay_id INT PRIMARY KEY, order_id INT REFERENCES
"Order"(order_id),

  method VARCHAR(30), status VARCHAR(20), amount DECIMAL(12,2));

CREATE TABLE Shipment(ship_id INT PRIMARY KEY, order_id INT REFERENCES
"Order"(order_id),

  address TEXT, carrier VARCHAR(50), fee DECIMAL(12,2));
```

5) Java – Minh họa đóng gói (Encapsulation)

```
public class BankAccount {

  private String accNo; private double balance;

  public BankAccount(String accNo){ this.accNo = accNo; }

  public double getBalance(){ return balance; }

  public void deposit(double amt){ if(amt>0) balance += amt; }
```

```
  public boolean withdraw(double amt){ if(amt>0 && amt<=balance){ balance-=amt;
return true; } return false; }

}
```

6) Sơ đồ tuần tự – Đăng nhập hệ thống



Sơ đồ tuần tự Use Case "Đăng nhập hệ thống"

7) Unit Test (JUnit) – tính tổng hai số

```
// SumUtil.java

public class SumUtil { public static int add(int a, int b){ return a + b; } }

// SumUtilTest.java (JUnit 5)

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;
```

```
class SumUtilTest {
  @Test void add_basic(){ assertEquals(7, SumUtil.add(3,4)); }
  @Test void add_negative(){ assertEquals(-1, SumUtil.add(3,-4)); }
}
```

8) Kịch bản kiểm thử chức năng Thanh toán (Scenario V2)

Use case gốc: Thanh toán đơn hàng
Actor chính: Khách hàng; **Hệ thống ngoài:** Cổng thanh toán

Tiền điều kiện (Pre):

- Có đơn hàng "Pending", tổng tiền đã tính; phương thức thanh toán hợp lệ.

Hậu điều kiện (Post):

- Nếu thanh toán thành công: đơn hàng ở **trạng thái Paid**, tạo hóa đơn, gửi email.
- Nếu thất bại: đơn hàng **Unpaid** hoặc **Failed**, log lý do, cho thử lại.

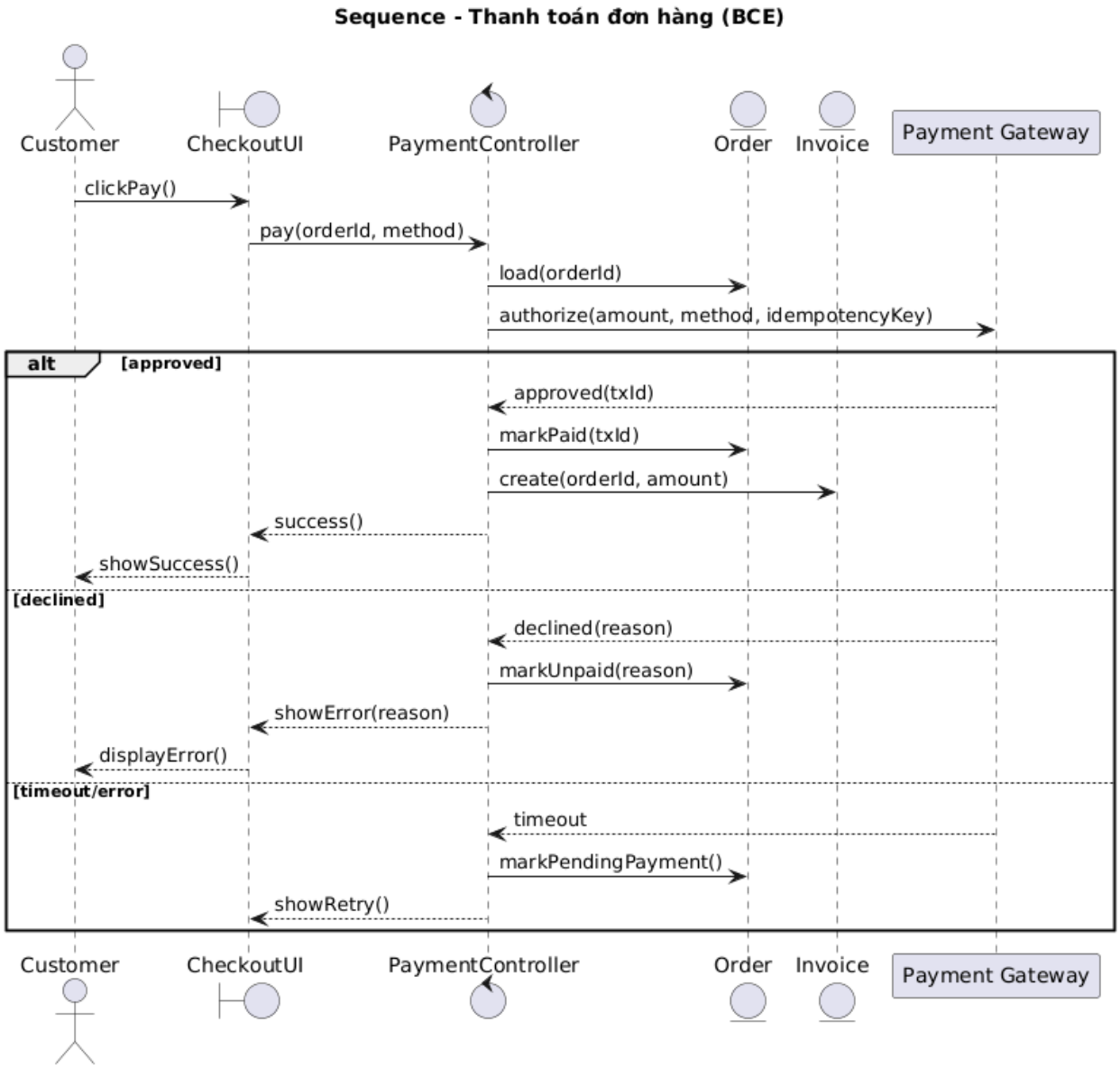
Scenario chuẩn (Main Success)

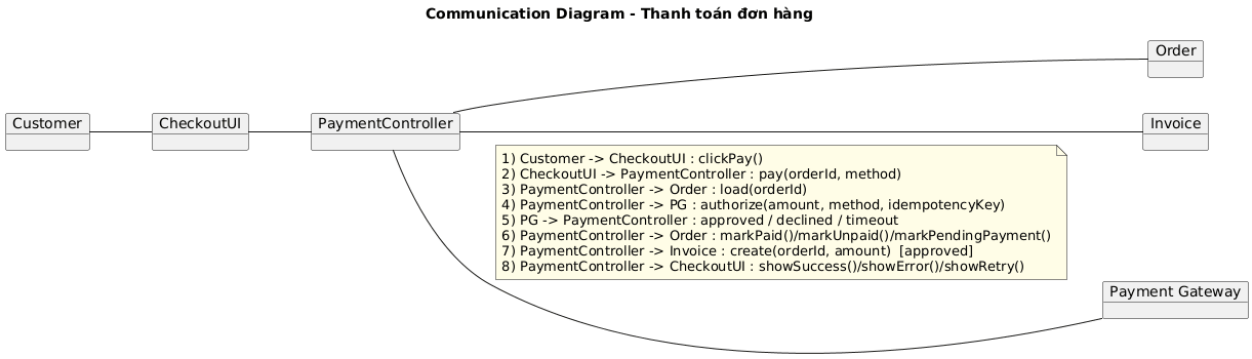
1. KH nhấn "Thanh toán" tại CheckoutUI.
2. CheckoutUI gửi yêu cầu đến PaymentController (PC).
3. PC tạo phiên thanh toán, gửi yêu cầu **authorize** tới PaymentGateway.
4. Gateway trả về **approved**.
5. PC đánh dấu Order=**Paid**, tạo Invoice, gửi mail xác nhận.
6. Hệ thống hiển thị "Thanh toán thành công".

Ngoại lệ/nhánh (Alt/Exception)

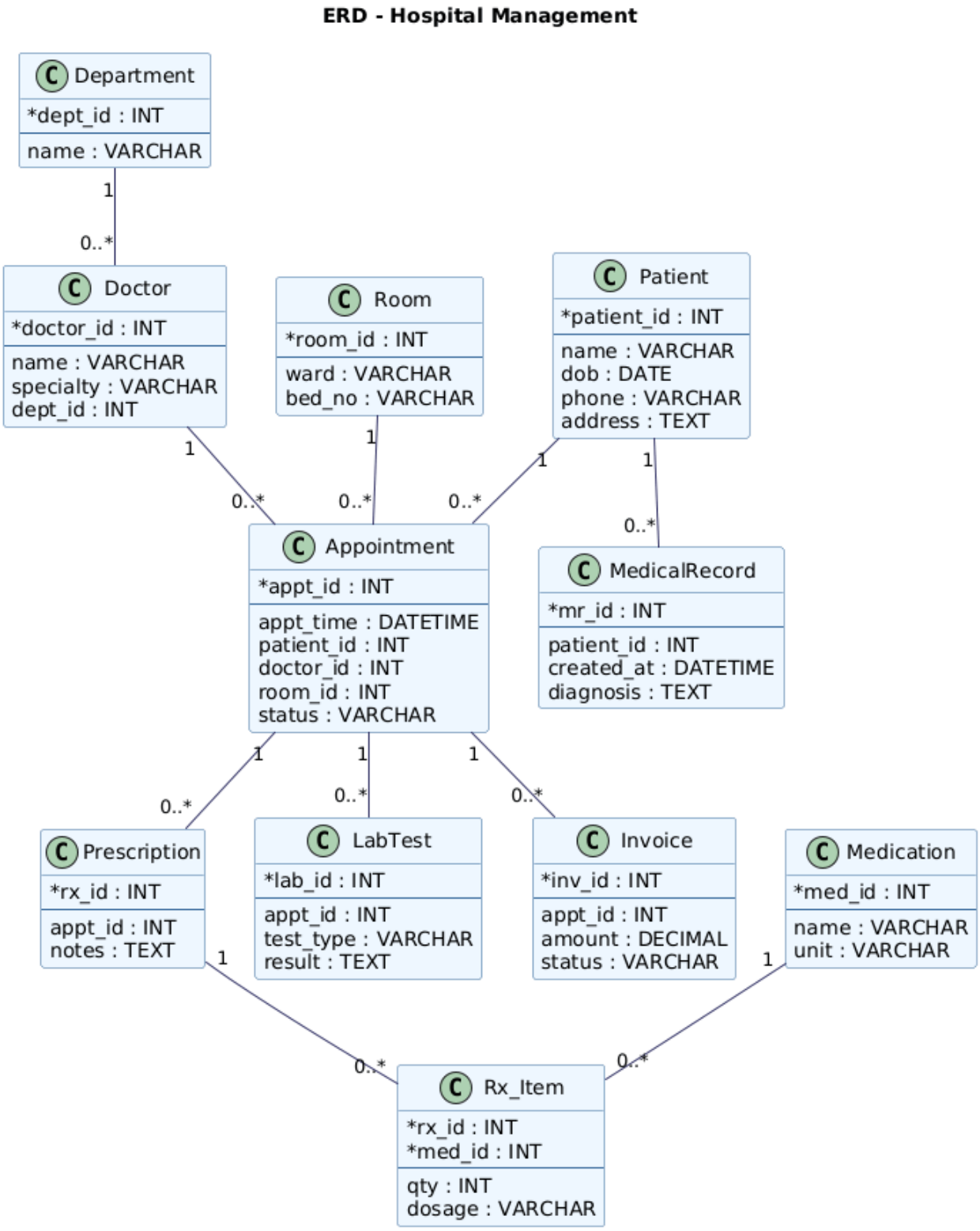
- **[A1] Thẻ bị từ chối** (bước 4): gateway trả **declined** → PC cập nhật Order=**Unpaid**, trả thông báo "Thẻ bị từ chối", cho phép chọn phương thức khác.
- **[A2] Timeout/Network** (bước 3/4): retry tối đa 2 lần; nếu vẫn lỗi → Order=**PendingPayment**, hiển thị "Kết nối gián đoạn, thử lại sau".
- **[A3] Số tiền lệch** (bước 4): số tiền capture ≠ order.total → PC rollback/cancel, log cảnh báo, thông báo "Số tiền không khớp".

- **[A4] 3-D Secure/OTP thất bại**: gateway trả **auth_failed** → Order=**Unpaid**, hướng dẫn xác minh lại.
- **[A5] Trùng giao dịch**: nếu phát hiện idempotency key đã dùng → không tạo giao dịch mới, hiển thị kết quả của giao dịch trước.





9) ERD hệ thống quản lý bệnh viện



10) Use case cụ thể – Đặt vé máy bay (khứ hồi)

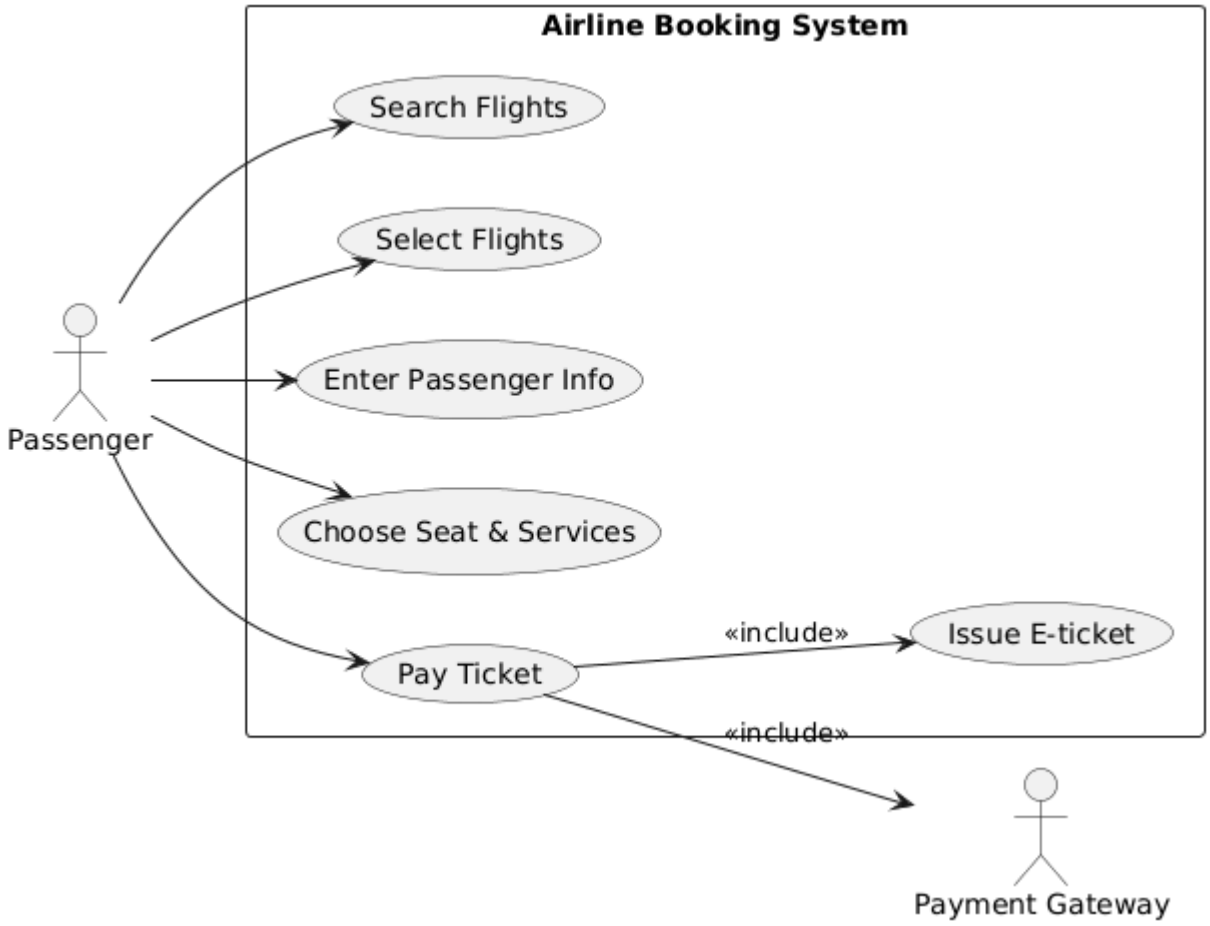
Tên/ID: UC-FLIGHT-BOOK-RT – Đặt vé khứ hồi
Mức: Mục tiêu người dùng (user goal)
Actor chính: Hành khách
Stakeholders: Hãng bay (xuất vé), Cổng thanh toán, Email/SMS service
Tiền điều kiện: Có chuyến còn chỗ & giá; người dùng có tài khoản hoặc đặt nhanh.
Kích hoạt: Hành khách chọn điểm đi/đến, ngày đi & về, nhấn Tìm chuyến.
Hậu điều kiện thành công: Tạo PNR/BookingCode, gửi e-ticket; ghế được giữ/issued.
Hậu điều kiện thất bại: Không tạo đặt chỗ; giữ tạm nếu chờ thanh toán.

Main Success Scenario (Scenario chuẩn)

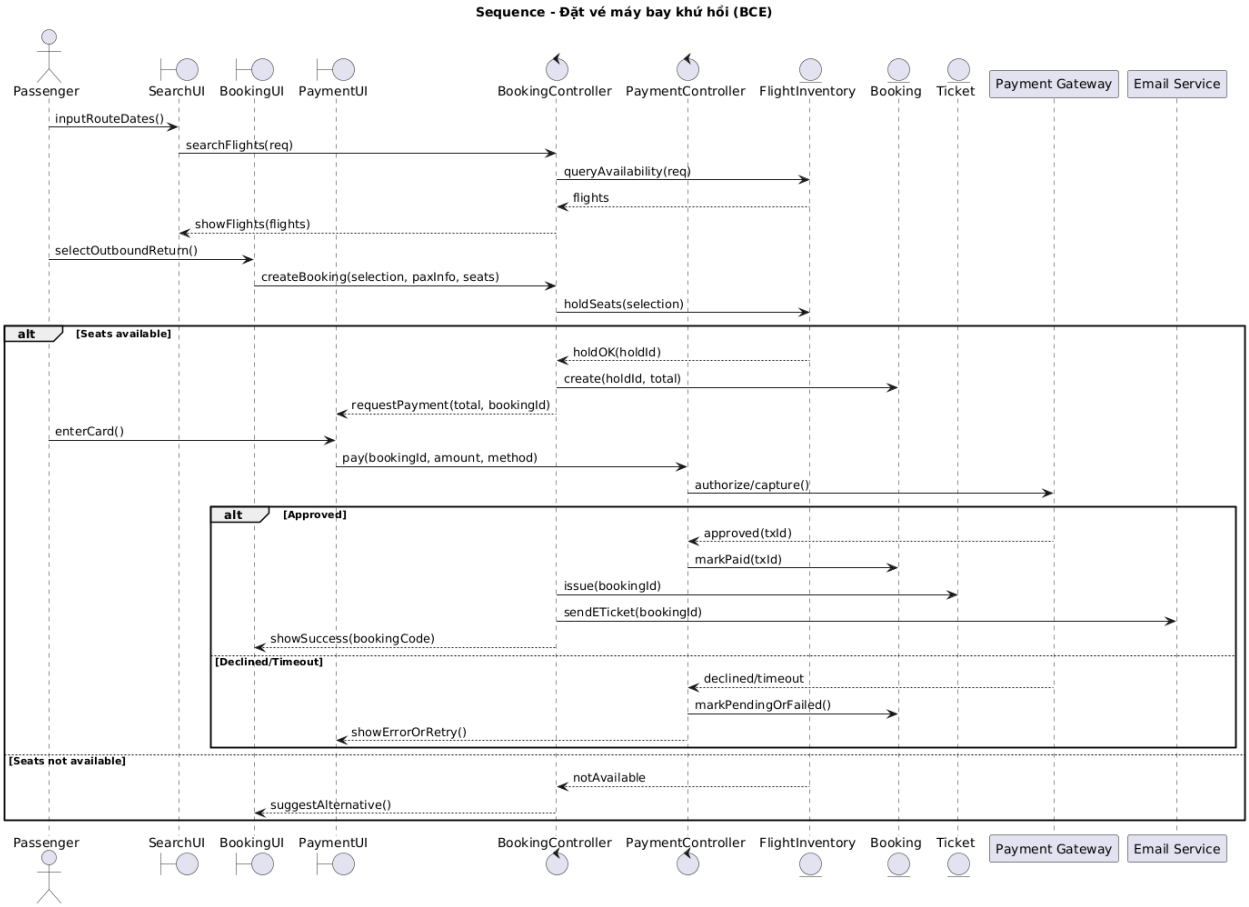
1. Hành khách nhập hành trình đi–về và tìm chuyến.
2. Hệ thống hiển thị danh sách chuyến & giá theo hạng.
3. Hành khách chọn cặp chuyến (đi & về).
4. Nhập thông tin hành khách (họ tên, giấy tờ), chọn chỗ/dịch vụ.
5. Xác nhận điều lệ & tổng giá.
6. Chọn phương thức thanh toán và thực hiện thanh toán.
7. Thanh toán **approved**; hệ thống phát hành vé (issue), tạo mã đặt chỗ.
8. Gửi e-ticket qua email/SMS, hiển thị "Đặt vé thành công".

Extensions / Ngoại lệ

- **[E1] Hết chỗ/Giá thay đổi (bước 2–3):** đề xuất chuyến/giá thay thế; cho phép đặt danh sách chờ.
- **[E2] Thông tin hành khách không hợp lệ (bước 4):** báo lỗi trường cụ thể; không cho tiếp tục.
- **[E3] Chọn chỗ thất bại:** tự động gán chỗ khả dụng hoặc cho chọn lại.
- **[E4] Thanh toán bị từ chối/timeout (bước 6):** giữ PNR ở trạng thái *On Hold/Pending* một thời gian, cho đổi phương thức hoặc thanh toán lại.
- **[E5] Trùng giao dịch:** dùng idempotency để trả lại kết quả giao dịch trước.



USE CASE DIAGRAM ĐẶT VÉ



Sequence Diagram đặt vé máy bay khứ hồi