**BÁO CÁO**  
Btl Môn Lập trình hướng đối tượng  
*(Gv Nguyễn Lê Thu)*  
  
**Sinh viên**  
Nguyễn Thanh Phong  
24100259  
  
**Nhóm 2**  
Hotel Manager  
  
  
  
  


**Nguyễn Thanh Phong - 24100259**  
Thiết kế cơ sở dữ liệu

Triển khai - hoàn thiện Model Room

Triển khai model Reservations

Thiết kế workflow.

Lên Project construction.

Thiết kế - hoàn thiện Front End.

Viết main docx - UML

**Đặng Đức Kiên - 24100323**

Triển khai - hoàn thiện Model Booking

Triển khai services Booking

Viết Unit test cho Model Room

Lên kế hoạch kiểm thử cho toàn bộ hệ thống

Thiết kế Frontend với Bootstrap  
  
**Copilot – Chatgpt 4.0**  
Hyper Carry NGL

# Public Links Repo (may contain web porting demo link in readme) : <https://github.com/nguyenthanhphong05/OOP_N02_T3_2_2025_Group2>Demo Video : [Group 2 Demo Hotel Manager](https://www.youtube.com/watch?v=a5xWk52TUK0&feature=youtu.be)

[](https://www.youtube.com/watch?v=a5xWk52TUK0&feature=youtu.be)

# App Main Screenshot

# 

# 

# 

# 

# 

# BÁO CÁO ĐỒ ÁN MÔN HỌC

## HOTEL MANAGEMENT SYSTEM

MÔN HỌC: Lập Trình Hướng Đối Tượng (OOP)  
MÃ MÔN: OOP\_N02\_T3\_2\_2025  
NHÓM: Group 2

THÀNH VIÊN NHÓM:

* Nguyễn Thanh Phong - 24100259 (Nhóm trưởng)
* Đặng Đức Kiên - 24100323
* Chatgpt 4.1
* Claude 4.0  
    
  (\*80% test unit covered by chatbot)

GIẢNG VIÊN HƯỚNG DẪN: Nguyễn Lệ Thu.

## MỤC LỤC

1. [GIỚI THIỆU DỰ ÁN](#X26fb732f3e6aa167e38a19dc25706afe9a77cf7)
2. [LỊCH SỬ VÀ PHÂN CHIA CÔNG VIỆC](#Xc83bc7b4690f27c023650427a57e65067282958)
3. [YÊU CẦU HỆ THỐNG](#Xe277ade764646c5034c59f34b26b7269c6668e7)
4. [THIẾT KẾ HỆ THỐNG](#X9eeea0e3bc49296243998e97450e821a17918a5)
5. [TRIỂN KHAI](#Xf19ac3510af7a1ee599853d59be197091ee962c)
6. [KIỂM THỬ](#X952200f14acb584e7cb29de6355491eb4ffe0fb)
7. [KẾT QUẢ ĐẠT ĐƯỢC](#Xff1f9ed956ee6d78a9e761f6a8db1fcc3c5e404)
8. [KHUYẾN NGHỊ VÀ PHÁT TRIỂN](#Xff1225743e2ce23b409789d822a793640a45107)
9. [KẾT LUẬN](#X0b936bf52c89bad0e760bef3dd053e02717a408)

## 1. GIỚI THIỆU DỰ ÁN

### 1.1 Tổng quan

Hotel Management System là hệ thống quản lý khách sạn được phát triển nhằm tự động hóa các quy trình nghiệp vụ cơ bản của một khách sạn. Dự án được thực hiện bởi nhóm 2 trong khuôn khổ môn học Lập Trình Hướng Đối Tượng (OOP).

### 1.2 Mục tiêu dự án

Mục tiêu chính:

* Xây dựng hệ thống quản lý khách sạn hoàn chỉnh
* Áp dụng các nguyên lý lập trình hướng đối tượng
* Sử dụng Spring Boot framework cho phát triển ứng dụng web
* Thực hành thiết kế cơ sở dữ liệu và quản lý dữ liệu

Mục tiêu phụ:

* Tìm hiểu và áp dụng các design patterns
* Thực hành viết unit tests
* Học cách làm việc nhóm và quản lý source code với Git

### 1.3 Phạm vi dự án

Hệ thống bao gồm các chức năng chính:

* Quản lý phòng khách sạn
* Quản lý thông tin khách hàng
* Quản lý đặt phòng và hủy đặt phòng
* Tính toán giá phòng tự động
* Kiểm tra tình trạng phòng trống

## 2. LỊCH SỬ VÀ PHÂN CHIA CÔNG VIỆC

### 2.1 Lịch sử phát triển dự án

Quyết định làm lại project: Để phù hợp với yêu cầu đề bài, nhóm 2 đã quyết định làm lại toàn bộ project từ đầu. Toàn bộ lịch sử commit và changelog của phiên bản cũ vẫn được lưu trữ ở branch “old” để tham khảo.

Những thay đổi chính trong project:

2.1.1 Thay đổi về kiến trúc:

* Trước: Sử dụng 3 model phức tạp với nhiều dependency
* Sau: Đơn giản hóa thành 3 model chính: Room, Customer, và Reservation (thay vì Booking)
* Lý do: Tăng tính rõ ràng và dễ maintain của code

2.1.2 Thay đổi về công nghệ:

* Framework: Chuyển sang sử dụng Java Spring Boot framework thay vì Java thuần
* Database: Sử dụng H2 Database với JPA/Hibernate thay vì file-based storage
* Frontend: Tích hợp Thymeleaf template engine với Bootstrap 5
* Build Tool: Sử dụng Maven cho dependency management

2.1.3 Lý do thay đổi:

* Yêu cầu đề bài cần áp dụng framework hiện đại
* Cần demonstration về OOP principles một cách rõ ràng hơn
* Tăng tính professional của project
* Dễ dàng mở rộng và maintain trong tương lai

### 2.2 Phân chia công việc nhóm

Công việc được phân chia rõ ràng giữa 2 thành viên để đảm bảo hiệu quả và tránh conflict:

2.2.1 Nguyễn Thanh Phong - 24100259 (Nhóm trưởng)

Nhiệm vụ chính:

* Thiết kế cơ sở dữ liệu:
  + Thiết kế ERD và relationship mapping
  + Cấu hình H2 database và JPA entities
  + Tối ưu hóa database performance
* Triển khai Model Layer:
  + Hoàn thiện Room entity với business logic
  + Triển khai Reservation entity và calculations
  + Implement bidirectional relationships
* Thiết kế Workflow:
  + Xác định business processes
  + Thiết kế user journey và use cases
  + Tạo activity diagrams
* Project Construction:
  + Setup project structure và configuration
  + Maven dependencies management
  + Spring Boot configuration
* Frontend Development:
  + Thiết kế và hoàn thiện giao diện người dùng
  + Thymeleaf templates integration
  + Responsive design với Bootstrap
* Documentation:
  + Viết UML class diagrams
  + Tạo technical documentation
  + Báo cáo chính của dự án

2.2.2 Đặng Đức Kiên - 24100323

Nhiệm vụ chính:

* Model Implementation:
  + Triển khai và hoàn thiện Customer model
  + Implement business logic cho Customer entity
  + Data validation và error handling
* Service Layer Development:
  + Triển khai CustomerService với full CRUD operations
  + Business logic implementation
  + Service integration và dependency injection
* Testing:
  + Viết comprehensive unit tests cho Room model
  + Test coverage cho business logic
  + Integration testing cho service layer
* Quality Assurance:
  + Lên kế hoạch kiểm thử cho toàn bộ hệ thống
  + Bug tracking và fixing
  + Performance testing
* Frontend Support:
  + Bootstrap integration và styling
  + Form validation và user experience
  + Cross-browser compatibility testing

### 2.3 Quy trình làm việc nhóm

2.3.1 Version Control:

* Sử dụng Git với branch strategy rõ ràng
* Main branch cho production code
* Feature branches cho từng functionality
* Regular code review và merge process

2.3.2 Communication:

* Daily standup meetings
* Task tracking và progress reporting
* Code review sessions
* Documentation sharing

2.3.3 Development Workflow:

1. Planning Phase: Requirement analysis và task breakdown
2. Design Phase: Architecture design và database modeling
3. Implementation Phase: Parallel development với clear interfaces
4. Testing Phase: Unit testing và integration testing
5. Integration Phase: Code merge và system testing
6. Documentation Phase: Final documentation và report writing

## 3. YÊU CẦU HỆ THỐNG

### 3.1 Yêu cầu chức năng

3.1.1 Quản lý phòng:

* Thêm, sửa, xóa thông tin phòng
* Xem danh sách tất cả phòng
* Phân loại phòng theo loại (Single, Double, Suite)
* Cập nhật trạng thái phòng (Available/Occupied)

3.1.2 Quản lý khách hàng:

* Đăng ký thông tin khách hàng mới
* Cập nhật thông tin khách hàng
* Tìm kiếm khách hàng theo tên, email, số điện thoại
* Xóa thông tin khách hàng

3.1.3 Quản lý đặt phòng:

* Tạo đặt phòng mới
* Kiểm tra tình trạng phòng trống
* Tính toán giá phòng theo số ngày ở
* Hủy đặt phòng
* Xem lịch sử đặt phòng

### 3.2 Môi trường phát triển

Ngôn ngữ lập trình: Java 17  
Framework: Spring Boot 3.3.0  
Database: H2 Database  
Frontend: Thymeleaf + Bootstrap 5  
Build Tool: Maven  
IDE: Visual Studio Code  
Version Control: Git

## 4. THIẾT KẾ HỆ THỐNG

### 4.1 Kiến trúc tổng thể

Hệ thống được thiết kế theo mô hình MVC (Model-View-Controller) với kiến trúc layered:

Presentation Layer (Controllers):

* HomeController: Xử lý trang chủ
* RoomController: Quản lý phòng
* CustomerController: Quản lý khách hàng
* ReservationController: Quản lý đặt phòng

Business Layer (Services):

* RoomService: Logic nghiệp vụ phòng
* CustomerService: Logic nghiệp vụ khách hàng
* ReservationService: Logic nghiệp vụ đặt phòng

Data Access Layer (Repositories):

* RoomRepository: Truy cập dữ liệu phòng
* CustomerRepository: Truy cập dữ liệu khách hàng
* ReservationRepository: Truy cập dữ liệu đặt phòng

Domain Layer (Models):

* Room: Entity phòng
* Customer: Entity khách hàng
* Reservation: Entity đặt phòng

### 4.2 Thiết kế cơ sở dữ liệu

4.2.1 Bảng Room:

CREATE TABLE rooms (  
 id BIGINT PRIMARY KEY AUTO\_INCREMENT,  
 room\_id VARCHAR(255) UNIQUE NOT NULL,  
 type ENUM('SINGLE', 'DOUBLE', 'SUITE') NOT NULL,  
 price DOUBLE NOT NULL,  
 is\_available BOOLEAN NOT NULL,  
 description VARCHAR(255) NOT NULL  
);

4.2.2 Bảng Customer:

CREATE TABLE customers (  
 id BIGINT PRIMARY KEY AUTO\_INCREMENT,  
 full\_name VARCHAR(255) NOT NULL,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 phone VARCHAR(255),  
 id\_number VARCHAR(255)  
);

4.2.3 Bảng Reservation:

CREATE TABLE reservations (  
 id BIGINT PRIMARY KEY AUTO\_INCREMENT,  
 room\_id BIGINT,  
 customer\_id BIGINT,  
 check\_in\_date DATE NOT NULL,  
 check\_out\_date DATE NOT NULL,  
 total\_price DOUBLE NOT NULL,  
 FOREIGN KEY (room\_id) REFERENCES rooms(id),  
 FOREIGN KEY (customer\_id) REFERENCES customers(id)  
);

### 4.3 Thiết kế UML

4.3.1 Class Diagram:

Hệ thống bao gồm các class chính:

* Room: Quản lý thông tin phòng với các thuộc tính id, roomId, type, price, isAvailable, description
* Customer: Quản lý thông tin khách hàng với fullName, email, phone, idNumber
* Reservation: Quản lý đặt phòng với checkInDate, checkOutDate, totalPrice

Mối quan hệ:

* Room (1) - (n) Reservation: Một phòng có thể có nhiều lượt đặt
* Customer (1) - (n) Reservation: Một khách hàng có thể đặt nhiều phòng

4.3.2 Activity Diagrams:

Hệ thống có 5 activity diagrams chính:

1. Customer Management Process
2. Room Management Process
3. Reservation Booking Process
4. Room Availability Check Process
5. Price Calculation Process

## 5. TRIỂN KHAI

### 5.1 Cấu trúc project

complete/  
├── src/  
│ ├── main/  
│ │ ├── java/com/g2/hotelm/  
│ │ │ ├── model/  
│ │ │ │ ├── Room.java  
│ │ │ │ ├── Customer.java  
│ │ │ │ └── Reservation.java  
│ │ │ ├── repository/  
│ │ │ │ ├── RoomRepository.java  
│ │ │ │ ├── CustomerRepository.java  
│ │ │ │ └── ReservationRepository.java  
│ │ │ ├── service/  
│ │ │ │ ├── RoomService.java  
│ │ │ │ ├── CustomerService.java  
│ │ │ │ └── ReservationService.java  
│ │ │ ├── controller/  
│ │ │ │ ├── HomeController.java  
│ │ │ │ ├── RoomController.java  
│ │ │ │ ├── CustomerController.java  
│ │ │ │ └── ReservationController.java  
│ │ │ └── HotelManagementApplication.java  
│ │ └── resources/  
│ │ ├── templates/  
│ │ │ ├── index.html  
│ │ │ ├── rooms.html  
│ │ │ ├── customers.html  
│ │ │ ├── reservations.html  
│ │ │ └── customer-form.html  
│ │ └── application.properties  
│ └── test/  
└── pom.xml

### 5.2 Triển khai Model Layer

5.2.1 Room Entity:

@Entity  
@Table(name = "rooms")  
public class Room {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
   
 @Column(unique = true, nullable = false)  
 private String roomId;  
   
 @Enumerated(EnumType.STRING)  
 private RoomType type;  
   
 private Double price;  
 private Boolean isAvailable;  
 private String description;  
   
 @OneToMany(mappedBy = "room", cascade = CascadeType.ALL)  
 private List<Reservation> reservations = new ArrayList<>();  
   
 // Business methods  
 public boolean isAvailableForPeriod(LocalDate checkIn, LocalDate checkOut) {  
 return reservations.stream()  
 .noneMatch(reservation -> {  
 LocalDate resCheckIn = reservation.getCheckInDate();  
 LocalDate resCheckOut = reservation.getCheckOutDate();  
 return !(checkOut.isBefore(resCheckIn) || checkIn.isAfter(resCheckOut));  
 });  
 }  
}

5.2.2 Customer Entity:

@Entity  
public class Customer {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
   
 @Column(nullable = false)  
 private String fullName;  
   
 @Column(nullable = false, unique = true)  
 private String email;  
   
 private String phone;  
 private String idNumber;  
   
 @OneToMany(mappedBy = "customer", cascade = CascadeType.ALL)  
 private List<Reservation> reservations = new ArrayList<>();  
   
 public int getTotalReservations() {  
 return reservations.size();  
 }  
}

5.2.3 Reservation Entity:

@Entity  
@Table(name = "reservations")  
public class Reservation {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
   
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "room\_id")  
 private Room room;  
   
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "customer\_id")  
 private Customer customer;  
   
 private LocalDate checkInDate;  
 private LocalDate checkOutDate;  
 private Double totalPrice;  
   
 public Double calculateTotalPrice() {  
 if (room != null && checkInDate != null && checkOutDate != null) {  
 long nights = ChronoUnit.DAYS.between(checkInDate, checkOutDate);  
 return room.getPrice() \* nights;  
 }  
 return 0.0;  
 }  
}

### 5.3 Triển khai Service Layer

5.3.1 CustomerService:

@Service  
public class CustomerService {  
 @Autowired  
 private CustomerRepository customerRepository;  
   
 public List<Customer> findAllCustomers() {  
 return customerRepository.findAll();  
 }  
   
 public Customer saveCustomer(Customer customer) {  
 validateCustomer(customer);  
 return customerRepository.save(customer);  
 }  
   
 private void validateCustomer(Customer customer) {  
 if (customer.getFullName() == null || customer.getFullName().trim().isEmpty()) {  
 throw new IllegalArgumentException("Customer full name is required");  
 }  
   
 if (customer.getEmail() == null || customer.getEmail().trim().isEmpty()) {  
 throw new IllegalArgumentException("Customer email is required");  
 }  
   
 // Email format validation  
 String email = customer.getEmail().trim();  
 if (!email.contains("@") || !email.contains(".")) {  
 throw new IllegalArgumentException("Invalid email format");  
 }  
 }  
}

### 5.4 Triển khai Controller Layer

5.4.1 ReservationController:

@Controller  
@RequestMapping("/reservations")  
public class ReservationController {  
 @Autowired  
 private ReservationService reservationService;  
   
 @Autowired  
 private RoomService roomService;  
   
 @Autowired  
 private CustomerService customerService;  
   
 @PostMapping("/save")  
 public String saveReservation(@ModelAttribute Reservation reservation,  
 @RequestParam String roomId,  
 @RequestParam(required = false) Long customerId,  
 RedirectAttributes redirectAttributes) {  
 try {  
 // Get room by roomId  
 Optional<Room> roomOpt = roomService.findRoomByRoomId(roomId);  
 if (!roomOpt.isPresent()) {  
 redirectAttributes.addFlashAttribute("errorMessage", "Room not found!");  
 return "redirect:/reservations";  
 }  
   
 reservation.setRoom(roomOpt.get());  
   
 // Handle customer selection or creation  
 if (customerId != null && customerId > 0) {  
 Optional<Customer> customerOpt = customerService.findCustomerById(customerId);  
 if (customerOpt.isPresent()) {  
 reservation.setCustomer(customerOpt.get());  
 }  
 }  
   
 // Calculate total price  
 if (reservation.getCheckInDate() != null && reservation.getCheckOutDate() != null) {  
 reservation.setTotalPrice(reservation.calculateTotalPrice());  
 }  
   
 reservationService.saveReservation(reservation);  
 redirectAttributes.addFlashAttribute("successMessage", "Reservation created successfully!");  
 } catch (Exception e) {  
 redirectAttributes.addFlashAttribute("errorMessage", "Error creating reservation: " + e.getMessage());  
 }  
 return "redirect:/reservations";  
 }  
}

## 

## 6. KIỂM THỬ

### 6.1 Kế hoạch kiểm thử

6.1.1 Unit Testing:

* Test các method trong model classes
* Test business logic trong service classes
* Test validation functions

6.1.2 Integration Testing:

* Test controller endpoints
* Test database operations
* Test service integrations

### 6.2 Kết quả kiểm thử

6.2.1 Unit Tests cho Room Model:

@Test  
public void testIsAvailableForPeriod() {  
 Room room = new Room();  
 room.setIsAvailable(true);  
   
 LocalDate checkIn = LocalDate.of(2025, 1, 1);  
 LocalDate checkOut = LocalDate.of(2025, 1, 5);  
   
 assertTrue(room.isAvailableForPeriod(checkIn, checkOut));  
}  
  
@Test  
public void testCalculateTotalPrice() {  
 Room room = new Room();  
 room.setPrice(100.0);  
   
 Customer customer = new Customer();  
   
 Reservation reservation = new Reservation();  
 reservation.setRoom(room);  
 reservation.setCustomer(customer);  
 reservation.setCheckInDate(LocalDate.of(2025, 1, 1));  
 reservation.setCheckOutDate(LocalDate.of(2025, 1, 3));  
   
 assertEquals(200.0, reservation.calculateTotalPrice(), 0.01);  
}

### 6.3 Các lỗi đã phát hiện và khắc phục

6.3.1 Database Lock Issue:

* Vấn đề: Database bị lock khi chuyển từ in-memory sang file-based
* Nguyên nhân: File database đã tồn tại và đang được sử dụng
* Khắc phục: Cấu hình đúng connection string và clean shutdown

6.3.2 Validation Errors:

* Vấn đề: Lỗi validation khi submit form
* Nguyên nhân: Thiếu validation annotations và error handling
* Khắc phục: Thêm @Valid annotations và BindingResult processing

## 7. KẾT QUẢ ĐẠT ĐƯỢC

### 7.1 Chức năng đã hoàn thành

✅ Quản lý phòng:

* Thêm, sửa, xóa phòng
* Hiển thị danh sách phòng
* Validation dữ liệu đầu vào

✅ Quản lý khách hàng:

* Đăng ký khách hàng mới
* Tìm kiếm khách hàng
* Cập nhật thông tin khách hàng

✅ Quản lý đặt phòng:

* Tạo đặt phòng mới
* Tính toán giá tự động
* Hiển thị lịch sử đặt phòng

✅ Giao diện người dùng:

* Responsive design với Bootstrap
* Form validation
* Success/Error messages

✅ Cơ sở dữ liệu:

* H2 database integration
* JPA/Hibernate configuration
* Relationship mapping

## 8. KẾT LUẬN

### 8.1 Tóm tắt dự án

Hotel Management System đã được phát triển thành công với đầy đủ các chức năng cơ bản của một hệ thống quản lý khách sạn. Dự án đã áp dụng thành công các nguyên lý lập trình hướng đối tượng và sử dụng Spring Boot framework để xây dựng một ứng dụng web hoàn chỉnh.

Việc quyết định làm lại toàn bộ project đã tỏ ra đúng đắn, giúp nhóm tạo ra một hệ thống có kiến trúc rõ ràng hơn, sử dụng công nghệ hiện đại và tuân thủ các best practices trong phát triển phần mềm.

### 8.2 Đánh giá kết quả

Điểm mạnh:

* Kiến trúc hệ thống rõ ràng, dễ maintain
* User interface thân thiện và responsive
* Database design hợp lý và scalable
* Áp dụng thành công Spring Boot framework
* Phân chia công việc nhóm hiệu quả

Điểm yếu:

* Thiếu một số chức năng advanced
* Performance chưa được optimize tối đa
* Chưa có comprehensive error handling

* Responsive web design
* Unit testing coverage 80%

### Lời cảm ơn

Nhóm xin chân thành cảm ơn giảng viên hướng dẫn đã tạo điều kiện và hỗ trợ nhóm trong quá trình thực hiện dự án. Đặc biệt cảm ơn về sự hướng dẫn trong việc áp dụng các nguyên lý OOP và best practices trong phát triển phần mềm.

Dự án này đã giúp nhóm:

* Hiểu sâu hơn về lập trình hướng đối tượng
* Nắm vững các kỹ năng phát triển phần mềm thực tế
* Học cách làm việc nhóm hiệu quả
* Áp dụng được các công nghệ hiện đại trong thực tế