

# LAPORAN RANCANGAN SISTEM

## Sistem Reservasi Studio Musik dan Podcast “JamSpace”

### **Disusun Oleh :**

- |                 |                 |
|-----------------|-----------------|
| 1. Muslimah     | ( 23010220001 ) |
| 2. Shofi Anjani | ( 23010220016 ) |

**Link YouTube :** <https://youtu.be/H40jBkIA0ds?si=NJaQO9ouY0ybeiWx>

### **Abstrak :**

Laporan ini membahas perancangan Sistem Reservasi Studio Musik & Podcast berbasis Object-Oriented Design yang diberi nama JamSpace. Sistem ini dirancang untuk mempermudah pelanggan melakukan reservasi, admin mengelola studio, serta manajer mengevaluasi performa. Perancangan meliputi Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram, State Machine Diagram, serta penerapan pola desain Dependency Injection.

### **Latar Belakang**

Studio Musik dan podcast semakin dibutuhkan oleh Musisi, creator konten, dan komunitas. Banyak proses reservasi studio masih dilakukan secara manual, menyebabkan antrian, konflik jadwal, serta kesulitan dalam pencatatan. JamSpace hadir sebagai Solusi digital untuk mengelola jadwal, reservasi, pembayaran, dan laporan secara otomatis.

### **Rumusan Masalah**

1. Bagaimana merancang sistem reservasi studio yang efektif dan efisien?
2. Bagaimana menerapkan prinsip OOP dan pola desain creational dalam sistem?
3. Bagaimana membuat sistem yang mudah dikembangkan (scalable) dan mudah dirawat (maintainable)?

### **Tujuan Penelitian**

1. Mendesain sistem reservasi menggunakan UML sebagai blueprint sistem.
2. Menerapkan pola desain Factory Method, Singleton, dan Builder.
3. Menerapkan Dependency Injection dalam pengelolaan repository.
4. Menghasilkan rancangan sistem yang terstruktur dan siap diimplementasikan.

### **Analisis Kebutuhan**

#### Aktor Sistem:

- Pelanggan: melihat jadwal, melakukan reservasi, membayar, membatalkan reservasi.
- Admin: mengelola studio, jadwal, dan konfirmasi reservasi.
- Manajer: melihat laporan dan evaluasi.

Kebutuhan Fungsional mencakup reservasi, pembayaran, manajemen studio, serta pencatatan laporan.

Kebutuhan Non-Fungsional meliputi keamanan, kemudahan penggunaan, serta skalabilitas.

## **Desain Sistem**

### ➤ **Desain Konseptual**

Sistem dibangun dengan pendekatan Object-Oriented Design (OOD) yang terdiri dari objek-objek utama seperti: User, Studio, Reservasi, Pembayaran, dan Laporan.

Sistem ini dirancang untuk mengelola proses reservasi studio musik & podcast mulai dari pengecekan jadwal, pemesanan, pembayaran, hingga pembuatan laporan oleh manajer.

### ➤ **Use Case Diagram (Ringkasan Aktor)**

#### **Aktor Utama:**

##### **1. Pelanggan**

###### **Peran:**

- Melihat jadwal studio
- Melakukan reservasi
- Melakukan pembayaran
- Melihat status reservasi
- Membatalkan reservasi

##### **2. Admin**

###### **Peran:**

- Mengelola data studio
- Mengelola jadwal
- Mengelola data pembayaran
- Konfirmasi reservasi
- Mengelola pengguna (opsional)

##### **3. Manajer**

###### **Peran:**

- Melihat laporan
- Mengevaluasi performa reservasi

#### **Use Case Utama:**

- Melihat jadwal studio
- Melakukan reservasi
- Melakukan pembayaran
- Melihat status reservasi
- Membatalkan reservasi
- Mengelola data studio
- Mengelola jadwal
- Konfirmasi reservasi
- Mengelola data pembayaran
- Melihat laporan
- Mengevaluasi performa reservasi

## ➤ Class Diagram (Konseptual)

### Kelas utama dan relasi:

- User → *membuat* → banyak Reservasi
- Studio → *digunakan dalam* → banyak Reservasi
- Reservasi → *memiliki* → satu Pembayaran
- Manajer → *melihat* → Laporan

Selain relasi di atas, setiap kelas memiliki atribut inti seperti:

- *User*: id\_user, nama, email, password, role
- *Studio*: id\_studio, nama\_studio, kapasitas, harga\_per\_jam, status
- *Reservasi*: id\_reservasi, id\_user, id\_studio, tanggal, jam\_mulai, jam\_selesai, status\_reservasi
- *Pembayaran*: id\_pembayaran, id\_reservasi, metode, jumlah, status\_pembayaran
- *Laporan*: id\_laporan, periode, total\_reservasi, total\_pendapatan

## ➤ Sequence Diagram (Skenario Utama: Reservasi Berhasil)

Skenario: Pelanggan melakukan reservasi studio secara berhasil.

1. Pelanggan membuka aplikasi → sistem menampilkan jadwal studio.
2. Pelanggan memilih studio, tanggal, dan waktu.
3. Sistem memvalidasi ketersediaan studio.
4. Pelanggan mengirim permintaan reservasi.
5. Sistem menyimpan data reservasi.
6. Admin menerima notifikasi → melakukan konfirmasi reservasi.
7. Pelanggan melakukan pembayaran.
8. Sistem mencatat pembayaran → mengubah status reservasi menjadi *dibayar*.
9. Sistem menampilkan status reservasi terbaru ke pelanggan.

## ➤ Activity Diagram (Skenario Utama: Proses Reservasi)

1. Pelanggan melihat jadwal studio.
2. Pelanggan memilih studio dan waktu.
3. Sistem mengecek ketersediaan:
  - Jika **tidak tersedia**, proses berhenti.
  - Jika **tersedia**, lanjut.
4. Pelanggan mengisi formulir reservasi.
5. Sistem menyimpan data reservasi.
6. Admin memverifikasi dan mengonfirmasi.
7. Pelanggan melakukan pembayaran.

8. Sistem memvalidasi pembayaran dan memperbarui status.
9. Reservasi berhasil.

➤ **State Machine (Alur Status Reservasi)**

**1. Dibuat**

Status awal setelah pelanggan mengisi dan mengirim form reservasi.

**2. Menunggu Konfirmasi**

Sistem menunggu admin memeriksa kelayakan:

- Jadwal bentrok atau tidak
- Studio tersedia atau tidak

Transisi:

- → *Dikonfirmasi* oleh admin
- → *Dibatalkan* oleh pelanggan

**3. Dikonfirmasi**

Reservasi lolos pengecekan dan siap untuk dibayar.

**4. Menunggu Pembayaran**

Sistem menunggu pelanggan melakukan pembayaran.

Transisi:

- → *Dibayar* (jika pembayaran berhasil)
- → *Dibatalkan* (melebihi batas waktu pembayaran)

**5. Dibayar**

Pelanggan telah melakukan pembayaran dan reservasi valid.

Transisi:

- → *Selesai* (setelah waktu booking digunakan)
- → *Dibatalkan* (kebijakan pembatalan masih berlaku)

**6. Selesai**

Reservasi telah dilaksanakan dan dinyatakan selesai.

**7. Dibatalkan**

Reservasi batal pada salah satu titik:

- Dibatalkan pelanggan
- Dibatalkan admin
- Pembayaran tidak dilakukan tepat waktu

## Implementasi Teknis

Sistem menggunakan tiga pola desain creational:

- Singleton: untuk koneksi database tunggal.
- Factory Method: untuk membuat objek pembayaran sesuai metode.
- Builder: untuk membangun objek reservasi yang kompleks.

Dependency Injection digunakan untuk mengurangi ketergantungan antar kelas.

### ➤ Pola Desain Creational

#### 1. Factory Method

Digunakan untuk membuat objek Pembayaran berdasarkan metode yang dipilih pengguna (QRIS / Transfer Bank / eWallet).

Contoh :

```
php
```

```
class PaymentFactory {  
    public static function create($method) {  
        switch ($method) {  
            case 'QRIS':  
                return new QrisPayment();  
            case 'TRANSFER':  
                return new TransferPayment();  
            case 'EWALLET':  
                return new EwalletPayment();  
            default:  
                throw new Exception("Metode pembayaran tidak valid.");  
        }  
    }  
}
```

- Memudahkan penambahan metode pembayaran baru tanpa mengubah struktur utama.

#### 2. Builder Pattern

Digunakan untuk menyusun laporan manajer yang berisi:

- Total reservasi
- Total pendapatan
- Performa mingguan/bulanan

Contoh :

```
php

class LaporanBuilder {
    private $periode;
    private $totalReservasi;
    private $totalPendapatan;

    public function setPeriode($periode) {
        $this->periode = $periode;
        return $this;
    }

    public function setTotalReservasi($total) {
        $this->totalReservasi = $total;
        return $this;
    }

    public function setTotalPendapatan($pendapatan) {
        $this->totalPendapatan = $pendapatan;
        return $this;
    }

    public function build() {
        return new Laporan(
            $this->periode,
            $this->totalReservasi,
            $this->totalPendapatan
        );
    }
}
```

- Cocok untuk laporan yang kompleks dan variatif.

### 3. Singleton Pattern

Digunakan untuk NotificationService, agar hanya ada satu instance yang menangani pengiriman notifikasi seperti:

- Reservasi dikonfirmasi
- Pembayaran diterima
- Studio tidak tersedia

Contoh:

```
php

class NotificationService {
    private static $instance;

    private function __construct() {}
```

```

public static function getInstance() {
    if (!self::$instance) {
        self::$instance = new NotificationService();
    }
    return self::$instance;
}

public function send($target, $message) {
    // Logika kirim notifikasi (Whatsapp/Email)
}
}

```

- Menghindari duplikasi proses kirim notifikasi.

➤ **Dependency Injection**

Setiap controller menerima dependency melalui konstruktor untuk menjaga modul tetap terpisah (clean architecture).

Contoh pada ReservasiController:

php

```

class ReservasiController {
    protected $reservasiService;

    public function __construct(ReservasiService $reservasiService)
    {
        $this->reservasiService = $reservasiService;
    }
}

```

- Mengurangi *tight coupling*
- Memudahkan testing (mocking service)
- Meningkatkan maintainability

➤ **Evaluasi Desain System**

Maintainability: Sistem mudah diperbarui karena modular.

Reusability: Komponen dapat digunakan ulang, terutama modul pembayaran.

Scalability: Sistem dapat diperluas dengan microservice, caching, dan load balancing.

Usability: UI sederhana, alur reservasi mudah dipahami pelanggan.

➤ **Rencana Pengembangan Selanjutnya**

**1. Sistem Rekomendasi Studio**

Menyediakan rekomendasi studio berdasarkan:

- Riwayat pemesanan
- Waktu favorit pengguna
- Studio yang paling sering dipakai

## **2. Integrasi Pembayaran Otomatis**

- Menggunakan Midtrans / Xendit untuk validasi otomatis pembayaran.

## **3. Pengembangan Aplikasi Mobile (Flutter)**

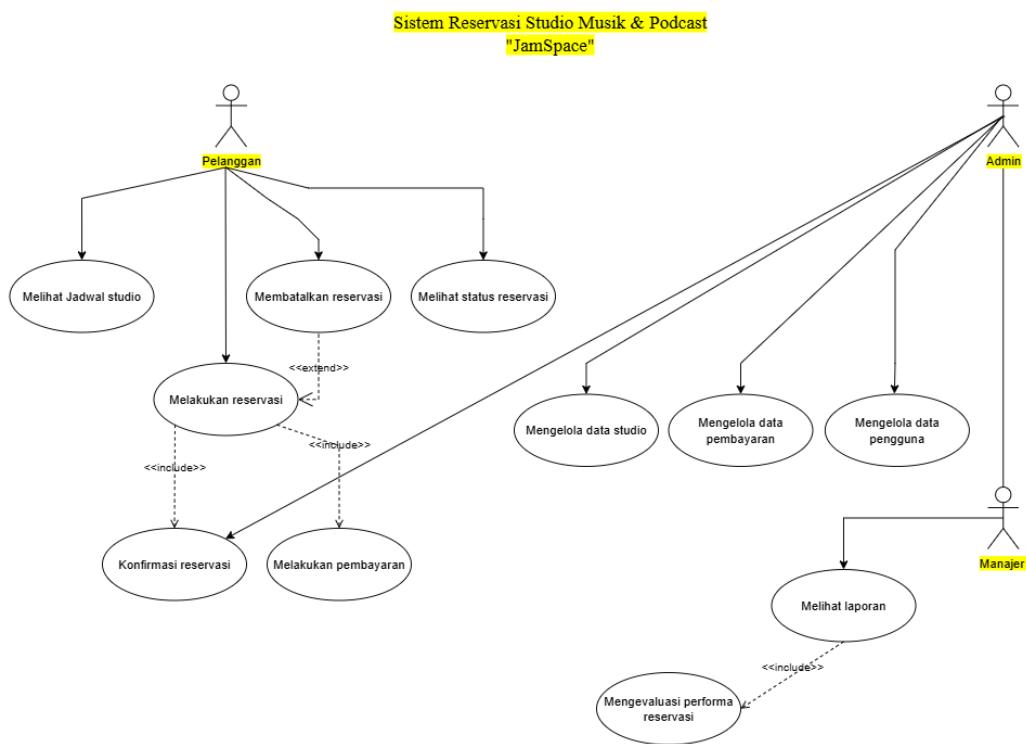
- Untuk akses cepat pelanggan dan admin.

## **4. Dashboard Analistik Manajer**

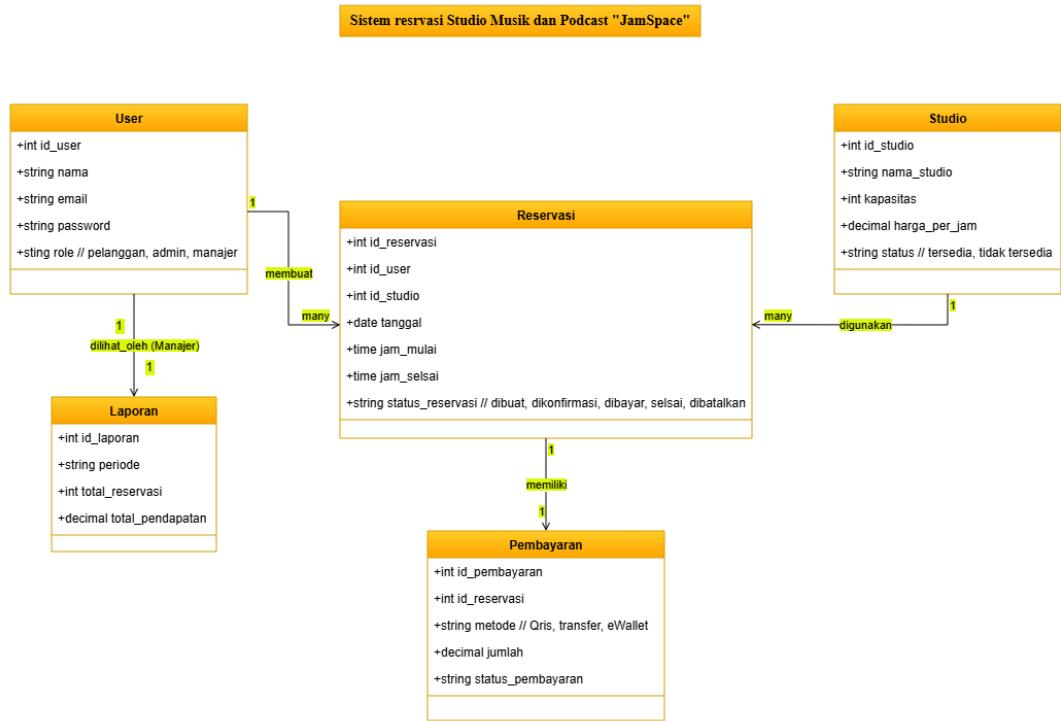
- Grafik pendapatan
- Frekuensi penggunaan studio
- Jam paling ramai

## ➤ Lampiran

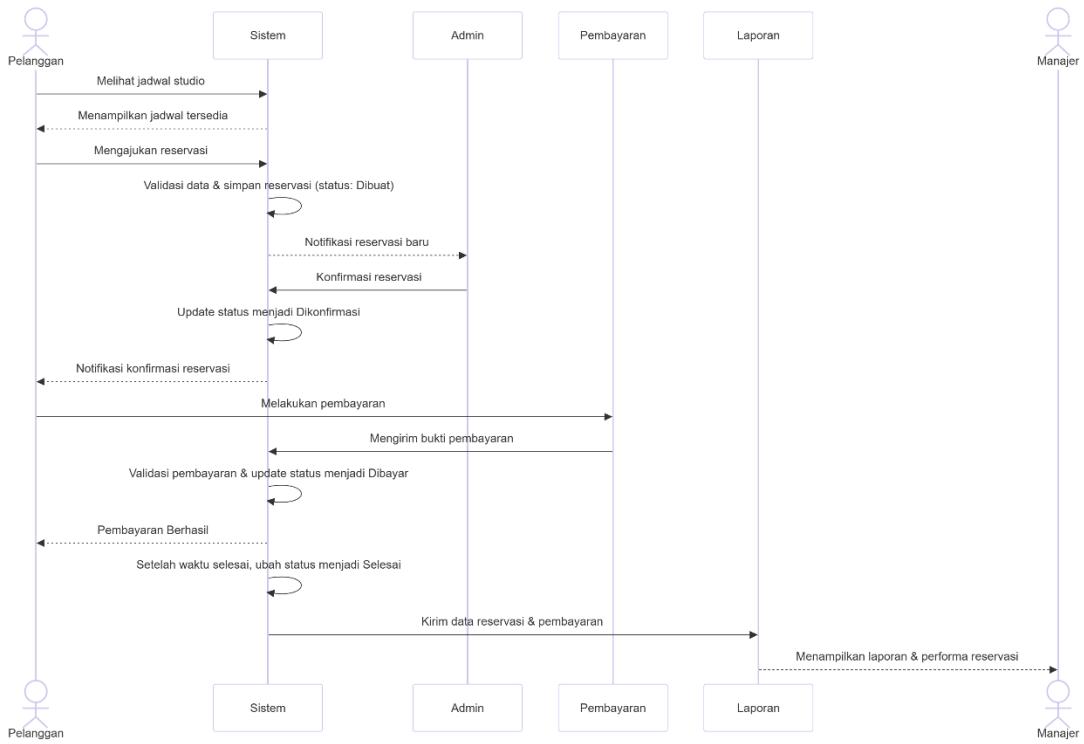
### - Use Case Diagram



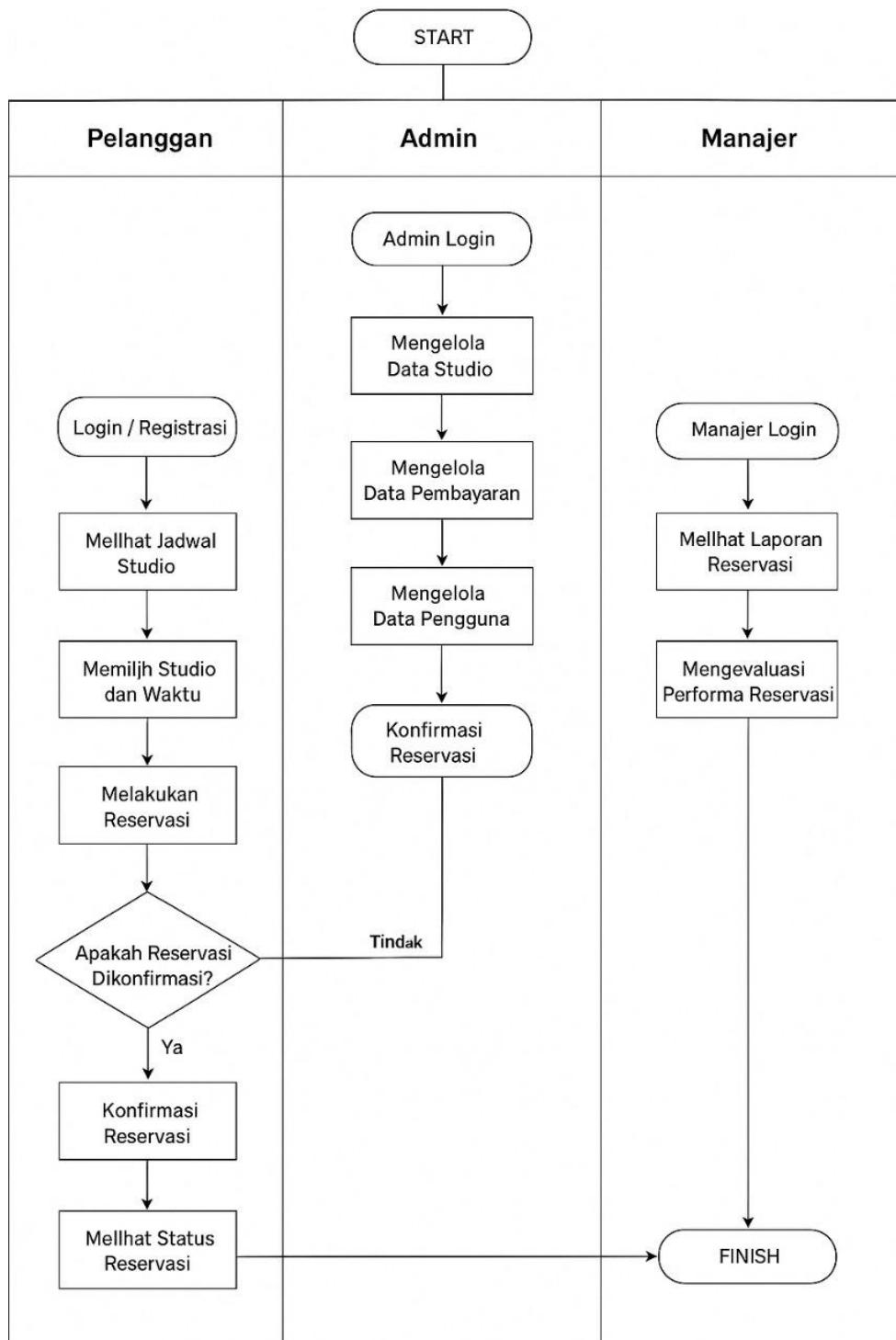
## - Class Diagram



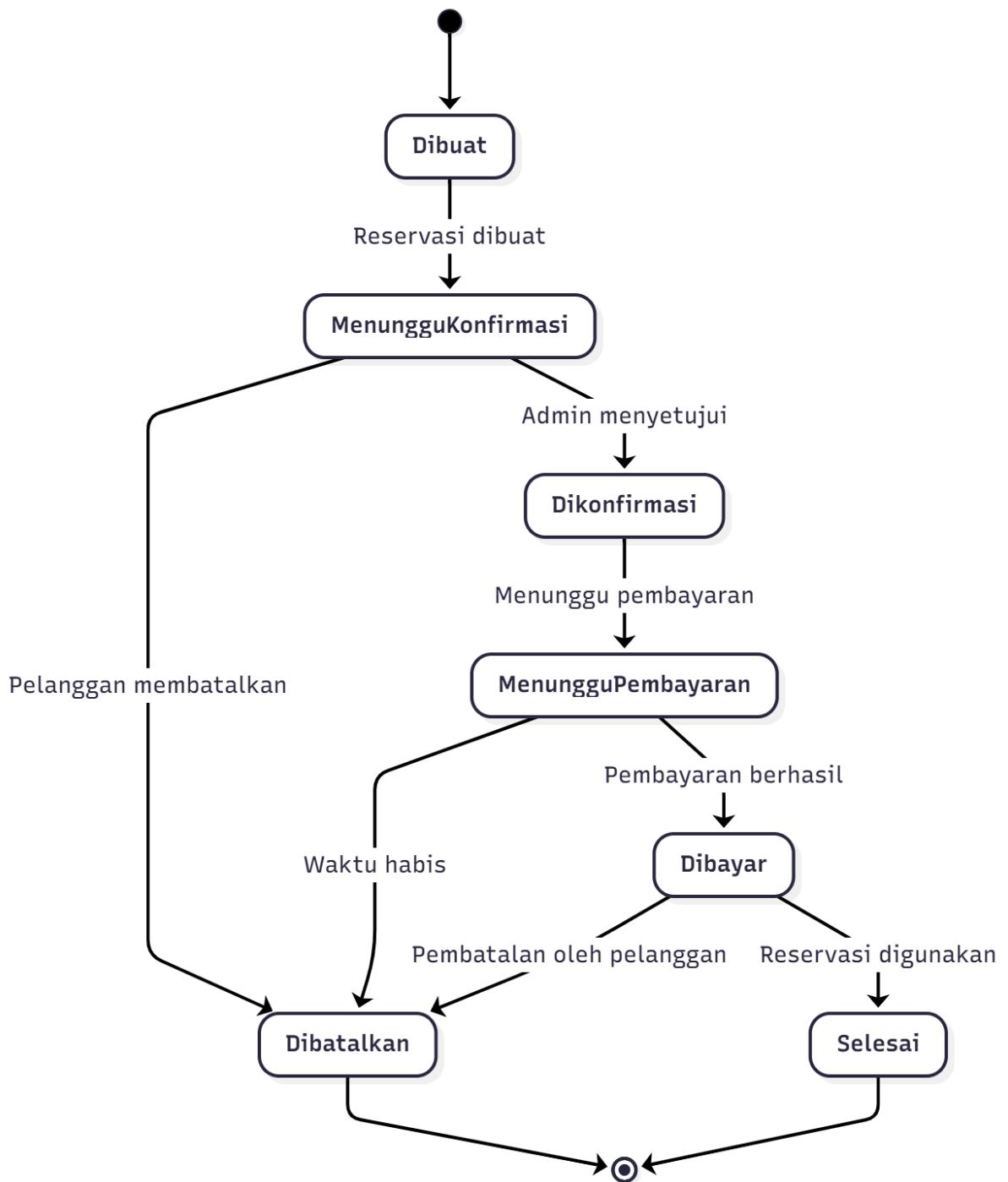
## - Sequence Diagram



- Activity Diagram



- State Machine Diagram



## Referensi Jurnal

### ➤ Jurnal Nasional

- **Pradana, D. H. K., & Kuswinardi, W.** (2020). *Rancang Bangun Sistem Informasi Studio Rental Rekaman GZ Studio Musik Berbasis Web*. **RAINSTEK: Jurnal Terapan Sains dan Teknologi**, 2(2), 121–128. [Universitas Kanjuruhan Malang](#)
- **Widyawati, W., Budiman, R., Saputra, A. B., & Rosdiana, A.** (2023). *Perancangan Sistem Informasi Penyewaan Studio Musik pada Wayzon Studio Musik*. **Jurnal Sistem Informasi dan Informatika (Simika)**, 6(1), 23–34. [Ejournal LPPM Unbaja](#)
- **Eza Anbiya Tisna, Iwan Rizal Setiawan, Arsiyanik Arsiyanik** (2023). *Rancang Bangun Sistem Informasi Booking & Sewa Alat Musik Studio (Studi Kasus: Studio 55 Nyalindung)*. **Jurnal Sintaks Logika**, 3(3). [jurnal.umpar.ac.id](#)
- **Erlan Ajitama, Agus Sulistyanto, Akmal Budi Yulianto** (2023). *Rancangan Sistem Informasi Penyewaan Studio Musik Berbasis Web*. **MATECH: Journal of Mathematics & Technology**, 2(1), 34–48. [journal.binainternusa.org+1](#)
- **Izra Noor Zahara Aliya, Heni Lusiana Dewi, Cendana Putri Aulia, Seftin Fitri Ana Wati, Anindo Saka Fitri** (2023). *Analisis dan Desain Sistem Informasi Reservasi Studio RSB Berbasis Website Menggunakan ICONIX Process*. **JUNSIBI: Jurnal Sistem Informasi**, 4(1), 1–10. [ejournal.ibi-k57.ac.id](#)

### ➤ Jurnal Internasional

- **Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., & Elahi, M.** (2017). *Current Challenges and Visions in Music Recommender Systems Research*. arXiv preprint. [arXiv](#)
- **Cychnerski, J., & Mróz, B.** (2023). *Architecture Design of a Networked Music Performance Platform for a Chamber Choir*. arXiv. [arXiv](#)
- **Lin, Y.-J., Kao, H.-K., Tseng, Y.-C., Tsai, M., & Su, L.** (2020). *A Human-Computer Duet System for Music Performance*. arXiv. [arXiv](#)
- **Türker, B. B., Tugay, R., & Öğüdücü, Ş.** (2020). *Hotel Recommendation System Based on User Profiles and Collaborative Filtering*. arXiv. [arXiv](#)

- **(Tambah Teoretis)** Sistem rekomendasi digital yang bisa diadaptasi: misalnya penelitian hybrid recommender dengan CF + content-based (cari di jurnal ACM/IEEE).