GAME CONSOLE ORDER SYSTEM

LAPORAN PROYEK SEMESTER GANJIL

DISUSUN OLEH:

1. Bulan Syafitri : 25031554048
2. Faris Riky Pratama : 25031554176
3. Zahwa Aulia Savila : 25031554085

Dosen Pengampu:

Hasanuddin Al-Habib, S.Si., M.Si.

PROGRAM STUDI SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SURABAYA

2025

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dan komunikasi yang pesat telah membawa dampak signifikan dalam berbagai aspek kehidupan masyarakat, termasuk dalam bidang hiburan. Salah satu bentuk hiburan digital yang semakin diminati adalah game console, karena menyediakan pengalaman bermain yang interaktif dan menyenangkan, baik secara individu maupun bersama kelompok. Namun, harga game console yang relatif tinggi sering menjadi kendala bagi sebagian masyarakat, khususnya pelajar dan mahasiswa, sehingga tidak semua orang mampu membeli perangkat tersebut. Kondisi ini menciptakan peluang bagi penyedia layanan penyewaan game console sebagai alternatif hiburan yang lebih terjangkau, fleksibel, dan ekonomis.

Saat ini, pengelolaan penyewaan game console di banyak tempat masih dilakukan secara manual. Sistem manual ini biasanya melibatkan pencatatan data penyewa, transaksi sewa, serta pengelolaan pengembalian console. Meskipun sederhana, metode ini memiliki sejumlah kelemahan yang cukup signifikan. Kesalahan pencatatan data, kehilangan informasi, serta kesulitan dalam mencari data yang dibutuhkan merupakan masalah umum yang sering muncul. Selain itu, pengelolaan stok dan transaksi secara manual membutuhkan waktu yang relatif lama, sehingga mengurangi efisiensi operasional dan berpotensi menimbulkan ketidakakuratan data.

Permasalahan tersebut menunjukkan perlunya pengembangan sebuah sistem berbasis komputer yang mampu mendukung proses penyewaan game console secara lebih terstruktur, cepat, dan akurat. Dengan sistem ini, proses peminjaman, pengembalian, dan pengecekan ketersediaan console dapat dilakukan secara otomatis dan efisien. Selain itu, sistem juga memungkinkan pemilik usaha untuk mengelola data pelanggan, memantau stok console, dan mencatat transaksi dengan lebih rapi, sehingga operasional penyewaan menjadi lebih tertib dan terkontrol.

Dari sudut pandang pendidikan, proyek pengembangan sistem penyewaan game console menjadi sarana yang tepat bagi mahasiswa untuk menerapkan konsep dasar pemrograman, seperti variabel, fungsi, array, dan pengolahan file (file handling). Proyek ini juga menuntut mahasiswa untuk mengembangkan kemampuan logika, analisis, dan pemecahan masalah melalui penerapan algoritma dalam konteks dunia nyata. Dengan demikian, pembuatan sistem penyewaan game console tidak hanya bermanfaat secara praktis untuk mempermudah pengelolaan usaha, tetapi juga berperan sebagai media pembelajaran yang meningkatkan kompetensi teknis dan kemampuan berpikir kritis mahasiswa.

1.2 Rumusan Masalah

1. Bagaimana cara merancang sistem penyewaan game console yang efektif dan efesien?
2. Bagaimana sistem dapat mempermudah pencatatan dan pengelolaan data penyewaan console?
3. Bagaimana konsep dasar pemrograman diterapkan dalam pembuatan sistem penyewaan game console?

1.3 Tujuan

1. Mengembangkan sistem penyewaan game console yang mampu menegelola proses sewa secara otomatis.
2. Meningkatkan efisiensi dan akurasi pengelolaan data serta keterssediaan ruangan berdasarkan waktu.
3. Menerapkan konsep dasar pemrograman untuk solusi nyata.

BAB 2
ANALISIS DAN PERANCANGAN

2.1 Analisis Kebutuhan Aplikasi

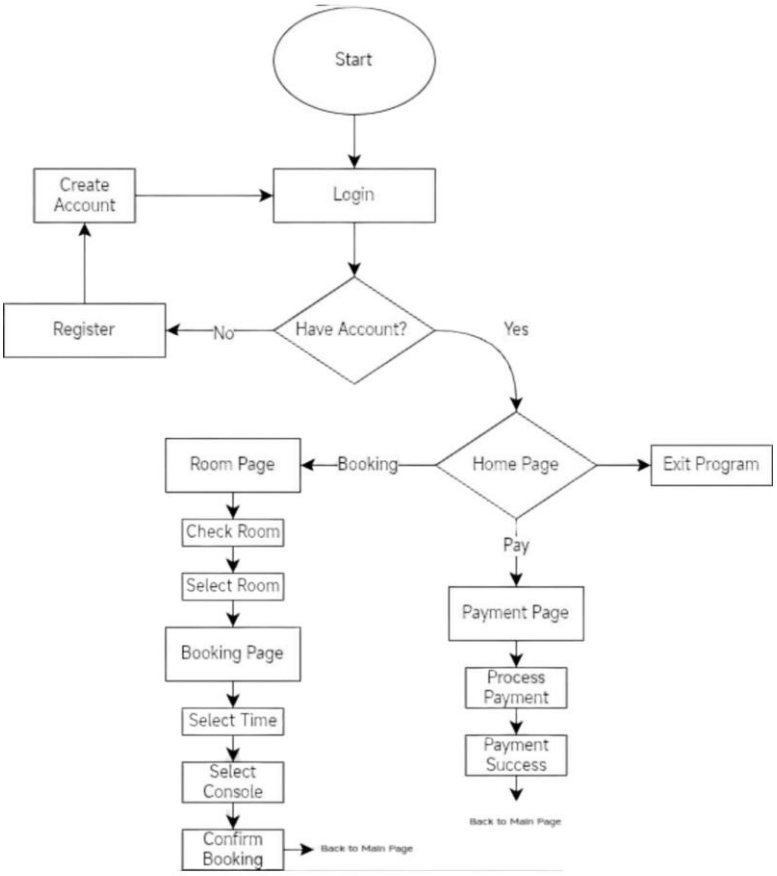
1. Gambaran Umum Aplikasi

Sistem Penyewaan Game Console merupakan aplikasi berbasis desktop yang dibangun menggunakan Python dengan antarmuka grafis (GUI) PyQt6. Sistem ini dirancang untuk membantu proses penyewaan game console agar lebih terstruktur, efisien, dan akurat dibandingkan sistem manual. Data disimpan dalam format JSON sehingga mudah diakses dan dikelola.

2. Kebutuhan Fungsional
   Kebutuhan fungsional sistem penyewaan game console ini meliputi:
   a. Login User
      Sistem menyediakan halaman login yang mewajibkan user memasukkan username dan password. Data login diverifikasi menggunakan file JSON.
   b. Menu Utama (Main Menu)
      Setelah login berhasil, user diarahkan ke menu utama yang berisi pilihan:
      - Pemesanan ruangan
      - Pembayaran
      - Keluar (exit)
   c. Pemesanan Ruangan
      User dapat melihat daftar ruangan yang tersedia serta memfilter ketersediaan berdasarkan waktu.
   d. Pemilihan Waktu dan Console
      User memilih durasi bermain (berdasarkan jam) dan satu jenis console untuk satu ruangan.
   e. Penyimpanan Data Pesanan
      Data pemesanan (user, ruangan, waktu, console, dan durasi) disimpan ke dalam file JSON.
   f. Sistem Pembayaran
      Sistem menampilkan data pesanan yang telah dibuat dan user memasukkan jumlah pembayaran.
   g. Cetak / Tampil Struk
      Sistem menampilkan struk berisi detail pesanan dan status pembayaran.
   h. Konfirmasi & Selesai
      Sistem menampilkan notifikasi bahwa pembayaran berhasil dan proses penyewaan selesai.

3. Kebutuhan Non-Fungsional
    a. Kemudahan Penggunan: Antarmuka sederhana dan mudah dipahami.
    b. Keamanan Data: Data login dan transaksi tersimpan dalam file JSON.
    c. Kinerja: Proses input, penyimpanan, dan pencarian data berjalan cepat.

2.2 Diagram Alur



Gambar 1 Diagram Alur

2.3 Sketsa Desain Antarmuka Grafis (Graphic User Interface)

1. Halaman Login
   Komponen UI:
   - Label judul aplikasi
   - Input username
   - Input password
   - Tombol Login
   - Label pesan error

   Deskripsi: Halaman awal aplikasi untuk validasi user sebelum mengakses sistem.
2. Halaman Menu Utama
   Komponen UI:
   - Tombol Pemesanan Ruangan
   - Tombol Pembayaran
   - Tombol Exit

   Deskripsi: Menjadi pusat navigasi utama setelah login berhasil.
3. Halaman Pemesanan Ruangan
   Komponen UI:
   - Dropdown / tabel daftar ruangan
   - Filter waktu (jam)
   - Tombol pilih ruangan

   Deskripsi: User dapat melihat dan memilih ruangan yang tersedia sesuai waktu.
4. Halaman Pemilihan Waktu & Console
   Komponen UI:
   - Input durasi bermain
   - Pilihan console (ComboBox / RadioButton)
   - Tombol Simpan

   Deskripsi: User menentukan detail bermain sebelum pesanan disimpan.
5. Halaman Pembayaran Struk
   Komponen UI:
   - Tabel/detail pesanan
   - Input jumlah pembayaran
   - Tombol Bayar
   - Label status pembayaran

   Deskripsi: Menampilkan detail transaksi dan konfirmasi pembayaran.

BAB 3
IMPLEMENTASI

3.1 Impor Modul

Dalam implementasi program *game console order*, digunakan beberapa modul yang berasal dari internal atau eksternal python untuk membangun kode yang terstruktur dan fungsional. Berikut modul-modul yang digunakan dalam membangun program *game console order*.

1. PyQt6: Modul yang digunakan untuk membuat antarmuka pengguna (*Graphical User Interface*), dengan tampilan yang dapat diatur menggunakan Qt Style Sheet berbasis CSS.
2. *JavaScript Object Notation* (JSON): Modul ini digunakan dengan tujuan untuk menyimpan dan mengelola data pengguna, data ketersedian ruangan, data pemesanan, dan data pembayaran dalam format JSON.
3. *Datetime*: Digunakan sebagai pengelola *User Id* berdasarkan waktu tanggal dan waktu serta mentransfer data dalam format waktu (Time).
4. *Operation System* (OS): Modul OS merupakan modul berfungsi untuk berinteraksi dengan operasi sistem.
5. Sys: Modul yang digunakan untuk berinteraksi langsung dengan *interpreter* Python dan lingkungan eksekusi program. Modul ini juga digunakan bersama PyQt6 untuk mengatur proses eksekusi dan penghentian aplikasi GUI.
6. ReportLab: Modul eksternal yang digunakan untuk menghasilkan dokumen PDF secara terprogram.

3.2 Penjelasan Kode

Pada program *game console order*, struktur kode dipisahkan menjadi beberapa bagian yang setiap bagian memiliki fungsi yang spesifik dan terorganisasi dengan baik. Program ini menyediakan beberapa fitur utama, meliputi manajemen pengguna (login dan registrasi), pengecekan ruangan berdasarkan waktu, pemesanan ruangan berdasarkan waktu dan jenis console, dan proses pembayaran serta fitur pendukung.

3.2.1 Fitur Manajemen Pengguna

Fitur ini memungkinkan pengguna dapat melakukan aktivitas registrasi dan login sebelum memasuki laman utama. Fitur ini terbagi menjadi beberapa bagian, meliputi *model*, *controller*, dan antarmuka pengguna (GUI). *User Model* digunakan untuk mengelola data *username* dan *password* pengguna, serta pembuatan *ID* pengguna berdasarkan tanggal, waktu, dan urutan pendaftaran. Selanjutnya, *User Controller* berfungsi sebagai penghubung antara model dan *User GUI*.

1. User Model

```python
from datetime import datetime

class User:
    sequence = 0

    def __init__(self, username, password):
        self.username = username
        self.password = password

        User.sequence += 1
        today = datetime.now().strftime('%Y%m%d')
        self.user_id = f"{today}{User.sequence:03d}"

    def get_data(self):
        return {
            "user_id": self.user_id,
            "username": self.username,
            "password": self.password
        }

    @classmethod
    def from_dict(cls, data):
        obj = cls(data["username"], data["password"])
        obj.user_id = data["user_id"]
        return obj
```

Gambar 2 User Model

Pada tahap inisialisasi terdapat dua parameter untuk menampung data username dan password pengguna serta menghasilkan ID pengguna berdasarkan tanggal, waktu, dan urutan pendaftaran. *Method* get_data digunakan untuk menyimpan data ke JSON dengan format yang terstruktur. Terakhir, method from_dict digunakan untuk mengembalikan data ke dalam bentuk *class* **User** sebagai objek baru setelah pengguna melakukan aktivitas login.

2. User Controller

```python
from utils.json_helpers import load_data, save_data
from models.user_model import User

class UserController:
    path = 'database/users.json'

    @staticmethod
    def validate_username(username):
        if not username:
            raise ValueError('Username tidak boleh kosong')
        if len(username) < 4:
            raise ValueError('Username harus terdiri minimal 4 huruf')

    @staticmethod
    def validate_password(password):
        if len(password) < 8:
            raise ValueError('Password minimal terdiri 8 karakter')
```

Gambar 3 User Controller Bagian 1

Di bagian ini terdapat method validate_username dan validate_password yang berfungsi sebagai mengembalikan error jika pengguna tidak menginput username dan password dengan benar.

```python
@classmethod
def load(data):
    all_data = load_data(data.path)
    User.sequence = len(all_data)
    return all_data

@classmethod
def save(data, data_user):
    return save_data(data.path, data_user)

@classmethod
def check_username(data, username):
    data_user = data.load()
    return any(u["username"] == username for u in data_user)

@classmethod
def search_user(data, username, password=None):
    data_user = data.load()
    for key in data_user:
        if key['username'] == username:
            if key['password'] == password or password is None:
                return key
    return None
```

Gambar 4 User Controller Bagian 2

Selanjutnya, terdapat 4 method lain yang digunakan di dalam class user controller. Method load dan save_data digunakan untuk mengambil data dari database dan menyimpan data baru dengan jalur file yang telah ditentukan sebelumnya serta memakai fungsi-fungsi pendukung yang diimpor dari folder bernama *utils*. Kemudian, terdapat check_username yang berfungsi untuk mengecek ketersediaan username, sedangkan method search_user digunakan untuk mencocokan username dan password berdasarkan input yang dilakukan oleh pengguna.

```python
@classmethod
def register(data, username, password):
    data.validate_username(username)
    data.validate_password(password)

    if data.check_username(username):
        raise ValueError('Username telah digunakan')

    new_data_user = User(username, password)
    all_data = data.load()
    all_data.append(new_data_user.get_data())
    data.save(all_data)

    return 'Registrasi Berhasil'

@classmethod
def login(data, username, password):
    data.validate_username(username)
    data.validate_password(password)

    data_user = data.search_user(username, password)
    if not data_user:
        raise ValueError('Username atau Password salah')

    return "Login Berhasil", User.from_dict(data_user)
```

Gambar 5 User Controller Bagian 2

Method yang terakhir yang ada dalam kelas user controller adalah register dan login. Method register digunakan untuk menangani proses pendaftaran pengguna. Metode login digunakan untuk menangani proses autentikasi pengguna. Kedua metode ini memakai 6 method yang telah dipaparkan sebelumnya.

3. User GUI

```python
from PyQt6.QtWidgets import QWidget, QFrame, QVBoxLayout, QPushButton, QLineEdit, QLabel
from PyQt6.QtCore import Qt
from utils.info_helper import show_info, show_error
from gui.main_menu import MainMenu

class Login(QWidget):
    def __init__(self, stack, controller):
        super().__init__()
        self.stack = stack
        self.controller = controller
        self.setFixedSize(420, 520)
        self.setStyleSheet(apply_style())

        title = QLabel('Login Account')
        title.setAlignment(Qt.AlignmentFlag.AlignCenter)
        title.setObjectName('title')

        self.username = QLineEdit()
        self.username.setPlaceholderText('Username')
        self.username.setObjectName('inputtext')

        self.password = QLineEdit()
        self.password.setPlaceholderText('Password')
        self.password.setEchoMode(QLineEdit.EchoMode.Password)
        self.password.setObjectName('inputtext')

        link = QLabel()
        link.setText(
            "<span style='color:white;'>Don't have an account?</span> "
            "<a href='signup' style='color:#4da3ff;'>Click here</a>"
        )
        link.setTextFormat(Qt.TextFormat.RichText)
        link.setOpenExternalLinks(False)
        link.linkActivated.connect(lambda: self.stack.setCurrentIndex(1))
        link.setObjectName('subtitle')
        link.setAlignment(Qt.AlignmentFlag.AlignRight)

        self.button = QPushButton("Login")
        self.button.clicked.connect(self.execute)
        self.button.setObjectName('button')

        frame = QFrame()
        frame.setObjectName('card')
        frameLayout = QVBoxLayout(frame)

        frameLayout.addWidget(title)
        frameLayout.addSpacing(30)
        frameLayout.addWidget(self.username)
        frameLayout.addWidget(self.password)
        frameLayout.addWidget(link)
        frameLayout.addSpacing(20)
        frameLayout.addWidget(self.button)

        mainLayout = QVBoxLayout(self)
        mainLayout.addStretch()
        mainLayout.addWidget(frame)
        mainLayout.addStretch()

    def execute(self):
        username = self.username.text().strip()
        password = self.password.text().strip()

        try:
            msg, data = self.controller.login(username, password)
            show_info(str(msg))

            main_menu = MainMenu(data)
            self.stack.addWidget(main_menu)
            self.stack.setCurrentIndex(2)

            self.stack.setFixedSize(main_menu.size())

        except Exception as msg:
            show_error(str(msg))
```

Gambar 6 User GUI Bagian Login

Kode ini merupakan implementasi antarmuka login menggunakan class yang diturunkan dari QWidget serta menerima parameter *stack* dan controller. Parameter stack digunakan sebagai wadah dari objek QtStackWidget agar halaman login dapat berpindah ke halaman lain tanpa membuat tanpa membuka *widget* baru. Sedangkan, paramter controller digunakan untuk menampung objek atau class **UserController**.

Pada tahap inisialisai, antarmuka diatur dengan ukuran tetap dan gaya tampilan menggunakan *stylesheet* berbasis CSS. Komponen antarmuka yang digunakan meliputi label judul, kolom input username dan password, tautan menuju halaman registrasi, serta tombol login yang akan menjalankan method execute jika ditekan. Method execute akan mengambil data input pengguna dan mengirimkannya ke controller untuk diproses. Jika berhasil akan menampilkan pesan informasi dan data user dikirimkan ke halaman utama untuk disimpan sementara. Sebaliknya jika terjadi kesalahan, sistem akan menampilkan pesan error.

```
def execute(self):
    username = self.username.text().strip()
    password = self.password.text().strip()

    try:
        message = self.controller.register(username, password)
        show_info(str(message))
        self.stack.setCurrentIndex(0)
        self.username.clear()
        self.password.clear()

    except Exception as msg:
        show_error(str(msg))
```

Gambar 7 User GUI Bagian Registrasi

Pada antarmuka registrasi, digunakan class yang diturunkan dari QWidget dengan parameter, ukuran layar, dan gaya tampilan yang sama seperti antarmuka login. Namun, terdapat perbedaan pada method executenya. Method execute pada halaman registrasi hanya menyimpan data input pengguna dan mengirimkannya ke controller untuk diproses. Jika proses berhasil akan menampilkan pesan informasi dan sebaliknya akan menampilkan pesan error kepada pengguna. Kemudian, terdapat fungsi style yang akan diterapkan baik di class login antarmuka atau registrasi antarmuka.

```
def apply_style():
    return """
        #card {
            background-color: #161b22;
            border: 1px solid #1f2937;
            border-radius: 18px;
            padding: 35px;
        }

        #title {
            font-size: 28px;
            font-weight: bold;
            color: #58a6ff;
            background: transparent;
            margin-bottom: 5px;
            border-radius: 18px;
            padding: 12px;
        }

        #subtitle {
            background-color: #161b22;
            font-size: 14px;
            color: #8b949e;
            margin-bottom: 20px;
        #inputtext {
            background-color: #0d1117;
            border: 2px solid #30363d;
            padding: 12px;
            border-radius: 10px;
            font-size: 15px;
            margin-bottom: 18px;
            color: #c9d1d9;
        }

        #inputtext:focus {
            border: 2px solid #58a6ff;
            background-color: #0c162d;
        }

        #button {
            background-color: #238636;
            color: white;
            padding: 12px;
            border-radius: 10px;
            font-size: 16px;
        }

        #button:hover {
            background-color: #2ea043;
        }

        #button:pressed {
            background-color: #196c2e;
        }
        """
```

Gambar 8 Gaya User GUI

3.2.2 Fitur Pengecekan Ruangan

Fitur pengecekan ruangan digunakan untuk menampilkan informasi ketersediaan ruangan game console sebelum proses pemesanan dilakukan. Melalui fitur ini, pengguna dapat mengetahui apakah suatu ruangan sedang tersedia atau telah digunakan pada waktu tertentu. Pengecekan ruangan dilakukan dengan dua kondisi, yaitu pengecekan umum

untuk seluruh waktu dan pengecekan berdasarkan jam tertentu. Fitur ini disusun berdasarkan tiga komponen utama, yaitu *Room Model*, *Room Controller*, dan *Room GUI*.

1. Room Model

```
from datetime import time

class Room:

    times_available = [time(t) for t in range(8, 18)]

    def __init__(self, room_id, name_room):
        self.room_id = room_id
        self.name_room = name_room
        self.time_each_rooms = Room.times_available.copy()
        self.booked_times = []
```

Gambar 9 Room GUI

Model room digunakan untuk merepresentasikan data dan kondisi setiap ruangan game console beserta waktu ketersediaannya. Pada kelas room, terdapat atribut class times_available yang berisi daftar waktu operasional ruangan, yaitu dari pukul 08.00 hingga 17.00. Daftar waktu ini dibuat menggunakan modul *datetime.time* dan digunakan sebagai acuan awal ketersediaan ruangan.

Pada tahap inisialisasi, terdapat dua parameter yang berfungsi sebagai penampung data nama ruangan dan ID ruangan. Kemudian, atribut time_each_rooms yang menyimpan salinan waktu tersedia untuk setiap ruangan, serta booked_times yang digunakan untuk menyimpan waktu-waktu yang telah dipesan.

2. Room Controller

```python
from models.room_model import Room
from utils.json_helpers import load_data, save_data
from datetime import time

class RoomManager:
    path = "database/rooms.json"

    def __init__(self, room_id, name_room):
        self.model = Room(room_id, name_room)
        self.load()

    def load(self):
        data = load_data(RoomManager.path)
        for value in data:
            if self.model.room_id == value["room_id"]:
                self.model.booked_times = [time(int(i.split(":")[0])) for i in value.get("booked_times"
, [])]
                self.model.time_each_rooms = [i for i in Room.times_available if i not in self.model.
booked_times]

                return value
        return None

    def check_room_general(self):
        return len(self.model.time_each_rooms) > 0

    def check_room_by_hour(self, selected_time):
        if not isinstance(selected_time, list):
            selected_time = [selected_time]

        return all(t in self.model.time_each_rooms for t in selected_time)

    def order(self, selected_time):
        for value in selected_time:
            if value in self.model.time_each_rooms:
                self.model.booked_times.append(value)
                self.model.time_each_rooms.remove(value)

        data = load_data(RoomManager.path)
        time_in_str = [f"{t.hour:02d}:00" for t in self.model.booked_times]
        for a in data:
            if self.model.room_id == a["room_id"]:
                a.update({
                    "room_id": self.model.room_id,
                    "name": self.model.name_room,
                    "status": "available" if len(self.model.time_each_rooms) > 0 else "unavailable",
                    "booked_times": time_in_str
                })
        save_data(RoomManager.path, data)
```

Gambar 10 Room Controller

Kode ini mendefinisikan **RoomManager** sebagai pengelola logika pemesanan ruangan berbasis data JSON. Saat inisialisai, objek **Room** dibuat dan data ruangan langsung dimuat dari database melalui method load, yang bertugas membaca data JSON, mengonversi jam booking ke tipe time, serta menentukan jam yang masih tersedia. Method check_room_general digunakan untuk mengecek apakah ruangan masih memiliki slot waktu kosong, sedangkan check_room_by_hour memvalidasi ketersediaan satu atau beberapa jam tertentu. Method order menangani proses pemesanan dengan memindahkan jam yang dipesan ke daftar booking, memperbarui status ruangan, lalu menyimpan perubahan tersebut kembali ke file JSON.

3. Room GUI

```python
from PyQt6.QtWidgets import QWidget, QPushButton, QGridLayout, QComboBox, QVBoxLayout, QLabel, QFrame
from PyQt6.QtCore import Qt
from datetime import time
from controller.room_controller import RoomManager
from gui.booking import BookingDialog

class RoomGui(QWidget):

    def __init__(self, stack, user):
        super().__init__()
        self.stack = stack
        self.user = user
        self.setFixedSize(650, 430)
        self.setStyleSheet(style())

        main_layout = QVBoxLayout()
        main_layout.setContentsMargins(20, 20, 20, 20)
        main_layout.setSpacing(0)
        self.setLayout(main_layout)

        frame = QFrame()
        frame.setObjectName("card")
        main_layout.addWidget(frame, alignment=Qt.AlignmentFlag.AlignCenter)

        self.grid = QGridLayout(frame)
        self.grid.setContentsMargins(20, 10, 20, 10)
        self.grid.setSpacing(25)

        title = QLabel("Order Room")
        title.setObjectName("title")
        self.grid.addWidget(title, 0, 0, 1, 2, alignment=Qt.AlignmentFlag.AlignCenter)

        self.combo_box = QComboBox()
        self.combo_box.setObjectName("selectbox")

        self.combo_box.addItem("Select Time", None)
        self.combo_box.addItem("All Time", "All")
        for t in range(8, 18):
            self.combo_box.addItem(f"{t}:00", time(t))
        self.grid.addWidget(self.combo_box, 1, 0, 1, 2)
        self.combo_box.setCurrentIndex(self.combo_box.findData("Select Time"))

        self.combo_box.currentIndexChanged.connect(self.update_data)

        self.rooms = [
            RoomManager(1, "Room A"),
            RoomManager(2, "Room B"),
            RoomManager(3, "Room C"),
            RoomManager(4, "Room D")
        ]

        self.button = []
        idx = 0
        for r in range(2, 4):
            for c in range(2):
                btn = QPushButton(self.rooms[idx].model.name_room)
                btn.setFixedSize(150, 90)
                btn.setObjectName("button")
                btn.clicked.connect(lambda checked=False, room=self.rooms[idx]: self.execute(room))
                self.button.append(btn)
                self.grid.addWidget(btn, r, c)
                idx += 1

        cancel_button = QPushButton("Back")
        cancel_button.setFixedSize(100, 35)
        cancel_button.setObjectName("nav_button")
        cancel_button.setStyleSheet("""
            QPushButton {
                background-color: #238636;
                color: white;
                border-radius: 10px;
                font-size: 16px;
            }
            QPushButton:hover {
                background-color: #2ea043;
            }
            QPushButton:pressed {
                background-color: #196c2e;
            }""")
        cancel_button.clicked.connect(self.exit_run)
        self.grid.addWidget(cancel_button, 4, 0, alignment=Qt.AlignmentFlag.AlignLeft)

        self.update_data()
        self.update_button()
```

```
    def update_data(self):
        selected_time = self.combo_box.currentData()

        for button, room in zip(self.button, self.rooms):
            if selected_time is None:
                is_available = room.check_room_general()
            elif selected_time == "All":
                is_available = room.check_room_general()
            else:
                is_available = room.check_room_by_hour(selected_time)

            if is_available:
                button.setEnabled(True)
                button.setStyleSheet("""
            #button {
                background: #0d1117;
                font-size: 16px;
                border-radius: 13px;
                border: 2px solid #58a6ff;
                }
            #button:hover {
                background-color: #2b7bff;
                border-color: #58a6ff;
                }
            #button:pressed {
                background-color: #165bb7;
                border-color: #3a8bff;
                }
                """)
            else:
                button.setEnabled(False)
                button.setStyleSheet("""
            #button {
                background-color: #30363d;
                color: #8b949e;
                font-size: 16px;
                border: 1px solid #21262d;
                border-radius: 13px;
                }
                """)

    def update_button(self):
        self.combo_box.setCurrentIndex(self.combo_box.findData(None))

    def exit_run(self):
        self.stack.setCurrentIndex(0)
        self.update_data()
        self.update_button()

    def execute(self, which_room):
        selected_time = self.combo_box.currentData()
        room_to_book = BookingDialog(callback=self.stack.parent(), user=self.user, room=which_room,
selected_times=selected_time)
        self.update_data()
        self.update_button()
        room_to_book.exec()

def style():
    return """
    #title {
        font-size: 26px;
        font-weight: bold;
        color: #58a6ff;
        background: transparent;
    }

    #selectbox {
        background-color: #0d1117;
        border: 2px solid #30363d;
        padding: 10px;
        border-radius: 10px;
        font-size: 15px;
        color: #c9d1d9;
    }

    #selectbox:hover {
        border: 2px solid #58a6ff;
    }

    #selectbox::drop-down {
        border: none;
        width: 30px;
    }

    #selectbox QAbstractItemView {
        background-color: #161b22;
        color: #c9d1d9;
        selection-background-color: #238636;
        border-radius: 8px;
    }
    """
```

Gambar 11 Room GUI

Kode ini mendefinisikan **RoomGUI**, antarmuka grafis berbasis PyQt6 yang digunakan untuk pemesanan ruangan. Pada tahap inisialisasi, tampilan utama dibangun dengan *layout*, *frame*, judul, *combo box* pemilihan waktu, serta tombol-tombol ruangan yang merepresentasikan setiap room. Di sini juga dibuat objek **RoomManager** untuk masing-masing ruangan dan dihubungkan dengan sinyal perubahan pilihan waktu agar status tombol ruangan selalu diperbarui sesuai ketersediaannya.

Method update_data berfungsi mengecek ketersediaan setiap ruangan berdasarkan waktu yang dipilih (semua waktu, satu jam tertentu, atau belum memilih waktu) dengan memanggil method pengecekan pada **RoomManager**, lalu mengaktifkan atau menonaktifkan tombol ruangan sekaligus mengubah tampilannya. Method update_button mengatur ulang kondisi awal combo box, sedangkan exit_run menangani navigasi kembali ke tampilan sebelumnya dan menyegarkan data. Method execute dijalankan saat tombol ruangan ditekan untuk membuka dialog booking dan meneruskan data ruangan, pengguna, serta waktu yang dipilih. Terakhir, method style berisi pengaturan tampilan (stylesheet) agar komponen GUI memiliki desain yang konsisten dan menarik.

3.2.3 Fitur Pemesanan

Fitur pemesanan berfungsi sebagai mekanisme utama yang mengatur proses pemilihan waktu bermain dan pemesanan console oleh pengguna berdasarkan ruangan yang telah dipilih. Secara umum, fitur ini memungkinkan pengguna untuk menentukan waktu bermain yang tersedia dan memilih jenis console yang diinginkan. Sama seperti fitur pengecekan ruangan, fitur ini juga terbagi menjadi 3 bagian, yaitu *Booking Model*, *Booking Controller*, dan *Booking GUI*.

1. Booking Model

```python
class Booking:
    sequence = 0

    def __init__(self, username, room_id, times, console, price):
        Booking.sequence += 1
        self.booking_id = str(Booking.sequence)

        self.username = username
        self.room_id = room_id
        self.times = times
        self.console = console
        self.price = price
        self.status = "pending"

    def get_dict(self):
        return {
            "booking_id": self.booking_id,
            "username": self.username,
            "room_id": self.room_id,
            "times": self.times,
            "console": self.console,
            "price": self.price,
            "status": self.status
        }
```

Gambar 12 Booking Model

Pada booking model, terdapat class **Booking** yang berfungsi sebagai model data untuk mempresentasikan satu transaksi pemesanan. Pada tahap inisialisasi, sistem akan menginisialisasi data utama seperti nama pengguna, ID ruangan, waktu yang dipesan, jenis console, harga, serta menetapkan status awal pemesanan. Method get_dict digunakan agar data dapat disimpan dalam bentuk *dictionary*.

2. Booking Controller

```python
from models.booking import Booking
from utils.json_helpers import load_data, save_data
from utils.pdf_helper import save_booking_to_pdf

class BookingController:
    console = {
        "PS5": 15000,
        "PS4": 10000,
        "Xbox Series X": 20000,
        "Nintendo Switch": 18000
    }

    def __init__(self, user, room):
        self.user = user
        self.room = room

    def price(self, console, time_selected):
        price = BookingController.console[console]
        return price * len(time_selected)

    def create_booking(self, console, time_selected):
        if not console:
            raise ValueError("Pilih Jenis Console")
        if not time_selected:
            raise ValueError("Pilih Waktu Bermain")

        path = "database/booking.json"
        data = load_data(path)
        Booking.sequence = len(data)

        time_in_str = [f"{t.hour:02d}:00" for t in time_selected]
        price = self.price(console, time_selected)

        data_booking = Booking(
            username=self.user.username,
            room_id=self.room.model.room_id,
            times=time_in_str,
            console=console,
            price=price
        )

        data.append(data_booking.get_dict())
        save_data(path, data)

        save_booking_to_pdf(data_booking)

        return data_booking
```

Gambar 13 Booking Controller

Pada class **BookingController**, terdapat atribut daftar harga console yang digunakan sebagai acuan perhitungan biaya sewa per jam. Pada tahap inisialisasi, **BookingController** menyimpan user model dan room model yang sedang diproses. Method price digunakan untuk menghitung total biaya pemesanaan berdasarkan jenis console yang dipilih dan jumlah waktu bermain, sedangkan method create_booking merupakan method utama yang menangani proses pembuatan pemesanan, meliputi validasi

input, pembuatan objek **Booking**, penyimpanan data ke database JSON, serta pembuatan bukti pemesanan dalam bentuk pdf.

3. Booking GUI

```python
import json
from PyQt6.QtWidgets import (
    QApplication, QDialog, QLabel, QComboBox, QPushButton,
    QVBoxLayout, QHBoxLayout, QMessageBox, QGridLayout, QFrame
)
from PyQt6.QtGui import QFont
from PyQt6.QtCore import Qt
from controller.booking_controller import BookingController
from datetime import time
from utils.info_helper import show_info, show_error


class BookingDialog(QDialog):
    def __init__(self, callback, user, room, selected_times):
        super().__init__()
        self.callback = callback
        self.book = BookingController(user=user, room=room)
        self.selected_times = selected_times
        self.setWindowTitle("Sistem Booking Console")

        self.resize(650, 430)

        font_besar = QFont("Arial", 14)
        self.setFont(font_besar)

        layout = QVBoxLayout()
        layout.setSpacing(20)     # Jarak antar elemen lebih lega
        layout.setContentsMargins(40, 40, 40, 40)

        title_label = QLabel(f"Booking {self.book.room.model.name_room}")
        layout.addWidget(title_label, alignment=Qt.AlignmentFlag.AlignCenter)
        title_label.setFont(QFont("Arial", 20, QFont.Weight.Bold))

        frame = QFrame()

        h1 = QGridLayout(frame)
        h1.addWidget(QLabel("Pilih Waktu Mulai:"), 0, 0, 1, 5)

        self.buttons = []
        for row in range(1, 3):
            for col in range(5):
                waktu = f"{8 + (row - 1) * 5 + col}:00"
                self.btn = QPushButton(waktu)
                self.btn.setObjectName("button")
                self.btn.setCheckable(True)
                self.btn.setStyleSheet("""
            #button {
                background: #0d1117;
                border-radius: 13px;
                border: 2px solid #58a6ff
                }
            #button:hover {
                background-color: #2b7bff;
                border-color: #58a6ff;
                }
            #button:pressed {
                background-color: #165bb7;
                border-color: #3a8bff;
                }
            #button:checked {
                background-color: #1158d6;
                }
                """)
                self.btn.setFixedSize(100, 40)
                self.btn.clicked.connect(self.click_to_update)
                self.buttons.append(self.btn)
                h1.addWidget(self.btn, row, col)
        layout.addWidget(frame)

        h2 = QHBoxLayout()
        h2.addWidget(QLabel("Pilih Console:"))

        self.combo_console = QComboBox()
        self.combo_console.setObjectName("box")
        self.combo_console.addItems(["PS5", "PS4", "Xbox Series X", "Nintendo Switch"])
        self.combo_console.setMinimumWidth(300)
        self.combo_console.currentIndexChanged.connect(self.update_price)
        h2.addWidget(self.combo_console)
        layout.addLayout(h2)

        h3 = QVBoxLayout()
        self.price_label = QLabel("Harga Total: Rp0")
        h3.addWidget(self.price_label, alignment=Qt.AlignmentFlag.AlignRight)
        layout.addLayout(h3)

        self.btn_simpan = QPushButton("Simpan Booking")
        self.btn_simpan.setMinimumHeight(50)
        self.btn_simpan.setStyleSheet("font-size: 18px;")
        self.btn_simpan.clicked.connect(self.simpan_booking)
        layout.addWidget(self.btn_simpan)

        self.setLayout(layout)
        self.update_buttons()
        self.update_price()
```

```
    def click_to_update(self):
        self.update_price()

    def update_buttons(self):
        booked_times = self.book.room.model.booked_times
        if self.selected_times is None or self.selected_times == "All":
            preselected_hours = []
        else:
            preselected_hours = [self.selected_times]

        for btn in self.buttons:
            hour = time(int(btn.text().split(":")[0]))
            if hour in booked_times:
                btn.setEnabled(False)
                btn.setChecked(False)
                btn.setStyleSheet("""
            #button {
                background-color: #30363d;
                color: #8b949e;
                border: 1px solid #21262d;
                border-radius: 13px;
                }
                """)
                btn.setToolTip("Sudah dipesan")
            else:
                btn.setEnabled(True)
                if hour in preselected_hours:
                    btn.setChecked(True)
                else:
                    btn.setChecked(False)

    def update_price(self):
        selected_console = self.combo_console.currentText()
        selected_times = [time(int(btn.text().split(":")[0])) for btn in self.buttons if btn.isChecked()
and btn.isEnabled()]

        if not selected_times:
            self.price_label.setText("Harga Total: Rp0")
            return

        total_price = self.book.price(selected_console, selected_times)
        self.price_label.setText(f"Harga Total: Rp{total_price}")

    def simpan_booking(self):
        selected_console = self.combo_console.currentText()
        selected_times = [time(int(btn.text().split(":")[0])) for btn in self.buttons if btn.isChecked()
and btn.isEnabled()]

        if not selected_times:
            show_error("Silakan pilih minimal satu waktu booking.")
            return

        try:
            booking = self.book.create_booking(selected_console, selected_times)
            self.book.room.order(selected_times)
            show_info(f"Booking berhasil!\nID Booking: {booking.booking_id}\nTotal Harga: Rp{booking.
price}")
            self.callback.calldata_to_MainGUI(booking)
            self.accept()
        except ValueError as e:
            show_error(str(e))
        except Exception as e:
            show_error(f"Terjadi kesalahan: {str(e)}")
```

Gambar 14 Booking GUI

Kode ini merupakan GUI sistem booking konsol game berbasis PyQt6. Class **BookingDialog** diturunkan dari QDialog yang berfungsi sebagai dialog pemesanan, tempat pengguna memilih jam bermain, jenis konsol, melihat total harga, lalu menyimpan booking. Method __init__ menginisialisasi seluruh komponen antarmuka dan koneksi untuk menjalankan method. Method update_button berfungsi untuk memperbarui status tombol waktu (tersedia, terpilih, atau sudah dibooking), sedangkan method update_price untuk menghitung dan menampilkan total harga secara otomatis. Terakhir, simpan_booking memvalidasi input, menyimpan data booking melalui booking controller, dan menampilkan pesan hasil proses kepada pengguna serta memindahkan pengguna ke laman utama.

3.2.4 Fitur Pembayaran

Fitur pembayaran berfungsi sebagai tahap akhir dari proses pemesanan pada program *game console order*. Setelah data pemesanan (booking model) berhasil disimpan ke dalam database JSON melalui booking controller, data tersebut kemudian diteruskan ke *payment controller* untuk diproses lebih lanjut. Pada tahap ini, sistem melakukan validasi data pemesanan, menghitung total biaya berdasarkan durasi penggunaan dan jenis console yang dipilih, serta mencatat status pembayaran. Dengan mekanisme ini, fitur pembayaran memastikan bahwa setiap pesanan dapat dikonfirmasi secara resmi, tercatat dengan baik, dan siap untuk diproses atau ditindaklanjuti oleh sistem.

1. Payment Model

```python
from datetime import datetime
class Payment:
    sequence = 0

    def __init__(self, booking_id, username, room_id, times_book, console, price, status):
        Payment.sequence += 1
        self.payment_id = str(Payment.sequence)

        self.booking_id = booking_id
        self.username = username
        self.room_id = room_id
        self.times_book = times_book
        self.console = console
        self.price = price
        self.time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        self.status = status

    def get_dict(self):
        return {
            "payment_id": self.payment_id,
            "booking_id": self.booking_id,
            "username": self.username,
            "room_id": self.room_id,
            "time booked": self.times_book,
            "console": self.console,
            "price": self.price,
            "time": self.time,
            "status": self.status
        }
```

Gambar 15 Payment Model

Payment model dibuat dengan menggunakan class **Payment** yang berfungsi sebagai model data untuk menyimpan informasi pembayaran. Model ini secara otomatis menghasilkan ID pembayaran yang unik serta mencatat waktu transaksi saat objek dibuat. Data yang diinisialisasi meliputi ID pembayaran, ID booking, nama pengguna, ruangan, waktu yang dipesan, jenis konsol, harga, waktu transaksi, dan status pembayaran. Method get_dict berfungsi mengubah objek pembayaran menjadi *dictionary* agar disimpan secara terstruktur.

2. Payment Controller

Pada bagian ini, class **PayController** menerima parameter berupa objek **Booking** yang diterima setelah proses pemesanan dilakukan. Method pay berfungsi untuk memvalidasi input pembayaran dari pengguna, menghitung dan mengembalikan uang

kembalian, membuat objek **Payment**, serta menyimpan data pembayaran ke dalam database berbasis JSON. Setelah seluruh proses berhasil, metode ini mengembalikan pesan yang menandakan bahwa pembayaran telah berhasil dilakukan.

```python
from utils.json_helpers import load_data, save_data
from models.payment import Payment

class PayController:
    path = "database/payment.json"

    def __init__(self, booking):
        self.booking = booking
        self.price = booking.price
        self.payment = None

    def pay(self, amount):
        if amount <= 0:
            raise ValueError("Input Bayar harus lebih dari nol")
        if amount < self.price:
            raise ValueError("Jumlah Bayar kurang dari Harga Bayar")

        cashback = amount - self.price
        self.booking.status = "paid"

        data = load_data(PayController.path)
        Payment.sequence = len(data)

        self.payment = Payment(
            booking_id=self.booking.booking_id,
            username=self.booking.username,
            room_id=self.booking.room_id,
            times_book=self.booking.times,
            console=self.booking.console,
            price=self.price,
            status=self.booking.status
        )

        data.append(self.payment.get_dict())
        save_data(PayController.path, data)

        return f"Transaksi Berhasil\n{cashback} adalah kembalian anda"
```

Gambar 16 Payment Controller

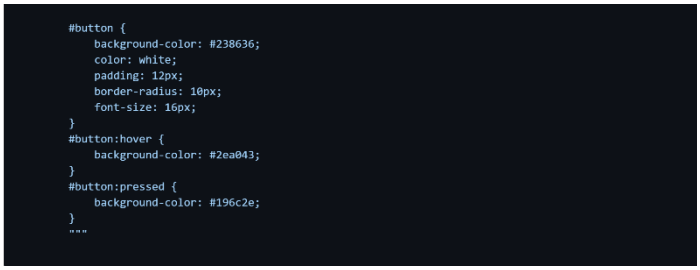
3. Payment GUI

Kode ini merupakan antarmuka grafis pembayaran yang dibangun menggunakan PyQt6 yang berfungsi sebagai dialog untuk memproses pembayaran setelah booking dilakukan. GUI ini menampilkan detail booking seperti ID booking, nama pengguna, berapa banyak (panjang) waktu yang dipesan, dan jenis konsol, serta menyediakan input nominal pembayaran dan tombol untuk melakukan atau membatalkan pembayaran.

Method update_info berfungsi memperbarui informasi booking berdasarkan data dari controller. Method pay menangani validasi nominal pembayaran, memproses pembayaran melalui controller, dan menampilkan pesan hasil pembayaran. Method cancel_payment digunakan untuk membatalkan proses pembayaran, sedangkan method style mengatur tampilan visual agar antarmuka konsisten dan mudah digunakan.

```python
from PyQt6.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QGridLayout, QPushButton, QLineEdit,
QLabel, QFrame
from PyQt6.QtCore import Qt
from utils.info_helper import show_error, show_info

class PaymentGui(QWidget):
    def __init__(self, stack, controller):
        super().__init__()
        self.stack = stack
        self.controller = controller
        self.setFixedSize(650, 430)
        self.setStyleSheet(self.style())

        main_layout = QHBoxLayout(self)
        main_layout.setContentsMargins(20, 20, 20, 30)

        left_bar = QFrame()
        left_bar.setObjectName("sidebar")
        left_bar.setFixedWidth(220)

        left_layout = QVBoxLayout(left_bar)
        left_layout.setContentsMargins(30, 40, 30, 40)
        left_layout.setSpacing(5)

        title_left = QLabel("To Pay")
        title_left.setObjectName("title_left")
        title_left.setAlignment(Qt.AlignmentFlag.AlignCenter)

        self.price = QLabel()
        self.price.setObjectName("price")
        self.price.setAlignment(Qt.AlignmentFlag.AlignCenter)

        cancel_button = QPushButton("Cancel Payment")
        cancel_button.setObjectName("button")
        cancel_button.clicked.connect(self.cancel_payment)

        left_layout.addStretch()
        left_layout.addWidget(title_left)
        left_layout.addWidget(self.price)
        left_layout.addStretch()
        left_layout.addWidget(cancel_button)

        right_bar = QFrame()
        right_layout = QVBoxLayout(right_bar)
        right_layout.setSpacing(10)
        right_layout.setContentsMargins(40, 15, 40, 15)

        title = QLabel("Payment")
        title.setObjectName("title")

        frame = QFrame()
        frame.setObjectName("card")

        layout = QGridLayout(frame)
        layout.setVerticalSpacing(12)

        text = QLabel("Booking Id:")
        text.setObjectName("text")
        self.booking_id = QLabel()
        self.booking_id.setObjectName("text")
        layout.addWidget(text, 0, 0)
        layout.addWidget(self.booking_id, 0, 1, alignment=Qt.AlignmentFlag.AlignRight)

        text1 = QLabel("Username:")
        text1.setObjectName("text")
        self.username = QLabel()
        self.username.setObjectName("text")
        layout.addWidget(text1, 1, 0)
        layout.addWidget(self.username, 1, 1, alignment=Qt.AlignmentFlag.AlignRight)

        text2 = QLabel("Total Waktu Booking:")
        text2.setObjectName("text")
        self.time_booked = QLabel()
        self.time_booked.setObjectName("text")
        layout.addWidget(text2, 2, 0)
        layout.addWidget(self.time_booked, 2, 1, alignment=Qt.AlignmentFlag.AlignRight)

        text3 = QLabel("Console:")
        text3.setObjectName("text")
        self.console = QLabel()
        self.console.setObjectName("text")
        layout.addWidget(text3, 3, 0)
        layout.addWidget(self.console, 3, 1, alignment=Qt.AlignmentFlag.AlignRight)

        self.input = QLineEdit()
        self.input.setObjectName("inputtext")
        self.input.setPlaceholderText("Fill Your Money")

        button1 = QPushButton("Pay")
        button1.setObjectName("button")
        button1.clicked.connect(self.Pay)

        right_layout.addWidget(title)
        right_layout.addWidget(frame)
        right_layout.addWidget(self.input)
        right_layout.addWidget(button1, Qt.AlignmentFlag.AlignCenter)
        right_layout.addStretch()

        main_layout.addWidget(left_bar)
        main_layout.addWidget(right_bar)
```

```
    def cancel_payment(self):
        self.stack.setCurrentIndex(0)
        self.input.clear()

    def update_info(self, controller):
        self.controller = controller
        if controller is None:
            self.price.setText("Rp0")
            self.booking_id.setText("-")
            self.username.setText("-")
            self.time_booked.setText("-")
            self.console.setText("-")
        else:
            self.price.setText(f"Rp{controller.booking.price}")
            self.booking_id.setText(str(controller.booking.booking_id))
            self.username.setText(controller.booking.username)
            self.time_booked.setText(
                f"{len(controller.booking.times)} Jam"
            )
            self.console.setText(controller.booking.console)

    def Pay(self):
        amount = self.input.text().strip()

        if amount == "":
            show_error("Input tidak boleh kosong")
            return

        if not amount.isdigit():
            show_error("Input hanya boleh angka")
            return

        amount = int(amount)
        try:
            msg = self.controller.pay(amount)
            show_info(msg)
            self.input.clear()
            self.stack.parent().data_booking = None
            self.stack.setCurrentIndex(0)
        except Exception as e:
            show_error(str(e))

    def style(self):
        return """
        QWidget {
            background-color: #0d1117;
            color: #c9d1d9;
            font-family: 'Segoe UI';
        }
        #sidebar {
            background-color: #0d1117;
            border: 1px solid #58a6ff;
            border-radius: 16px;
        }
        #price {
            font-size: 32px;
            font-weight: bold;
            color: #58a6ff;
            background: transparent;
        }

        #card {
            background-color: #161b22;
            border: 1px solid #30363d;
            border-radius: 16px;
            padding: 25px;
        }

        #title {
            font-size: 26px;
            font-weight: bold;
            color: #58a6ff;
        }

        #title_left {
            font-size: 20px;
            font-weight: bold;
            color: #58a6ff;
            background: transparent;
        }

        #text {
            font-size: 13px;
            background: transparent;
            color: white;
        }

        #inputtext {
            background-color: #0d1117;
            border: 2px solid #30363d;
            padding: 12px;
            border-radius: 10px;
            font-size: 15px;
            margin-bottom: 18px;
            color: #c9d1d9;
        }
        #inputtext:focus {
            border: 2px solid #58a6ff;
            background-color: #0c162d;
        }

        #button {
            background-color: #238636;
```



Gambar 17 Payment GUI

3.2.5 Fitur Pendukung

Gambar 18 Fitur Pendukung Bagian 1

Fitur ini berfungsi untuk membantu menampilkan pesan kepada pengguna melalui jendela pop-up berbasis PyQt6. Fungsi show_info digunakan untuk menampilkan pesan informasi, sedangkan fungsi show_error digunakan untuk menampilkan pesan kesalahan. Kode ini membantu menyederhanakan proses pemberian umpan balik kepada pengguna dan menjaga konsistensi tampilan pesan dalam aplikasi.

Gambar 19 Fitur Pendukung Bagian 2

Bagian ini berfungsi untuk menangani proses pembacaan dan penyimpanan data ke dalam file JSON. Fungsi load_data digunakan untuk memuat data dari file JSON dengan penanganan kesalahan jika file tidak ditemukan atau format tidak valid. Fungsi save_data digunakan untuk menyimpan data ke file JSON dengan memastikan direktori penyimpanan tersedia. Modul ini berperan sebagai lapisan penyimpanan data sederhana bagi aplikasi.

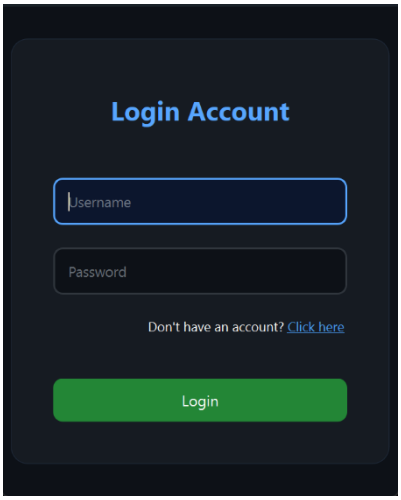
```python
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from datetime import datetime
import os


def save_booking_to_pdf(booking):
    os.makedirs("pdf", exist_ok=True)

    file_path = f"pdf/booking_{booking.booking_id}.pdf"
    c = canvas.Canvas(file_path, pagesize=A4)
    width, height = A4

    y = height - 50

    c.setFont("Helvetica-Bold", 18)
    c.drawCentredString(width / 2, y, "BUKTI BOOKING CONSOLE")
    y -= 40

    c.setFont("Helvetica", 12)
    data = [
        ("ID Booking", booking.booking_id),
        ("Username", booking.username),
        ("Room ID", booking.room_id),
        ("Console", booking.console),
        ("Waktu", ", ".join(booking.times)),
        ("Total Harga", f"Rp{booking.price}"),
        ("Tanggal", datetime.now().strftime("%d-%m-%Y %H:%M"))
    ]

    for label, value in data:
        c.drawString(50, y, f"{label} : {value}")
        y -= 20

    y -= 20
    c.setFont("Helvetica-Oblique", 10)
    c.drawString(50, y, "Terima kasih telah melakukan booking.")

    c.save()

    return file_path
```

Gambar 20 Fitur Pendukung Bagian 3

Fungsi pendukung terakhir memiliki fungsi untuk menghasilkan file PDF sebagai bukti booking. Fungsi save_booking_to_pdf menerima objek booking dan membuat dokumen PDF yang berisi informasi pemesanan, seperti ID booking, nama pengguna, ruangan, konsol, waktu, total harga, dan tanggal transaksi. Modul ini digunakan untuk menyediakan output dokumen resmi yang dapat disimpan atau dicetak oleh pengguna.

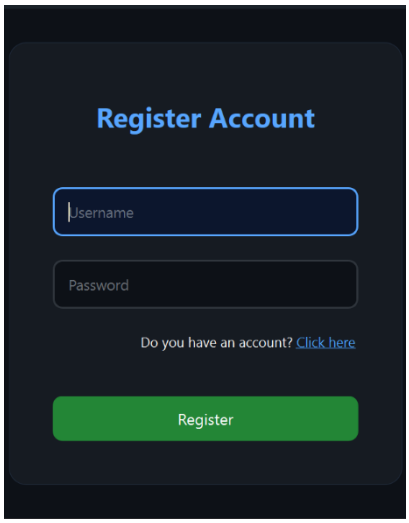
3.3 Dokumentasi Tampilan Aplikasi

1. Halaman Login



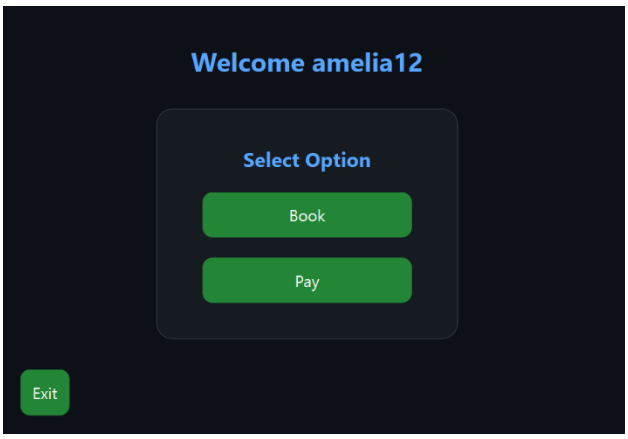
Gambar 21 Halaman Login

2. Halaman Registrasi



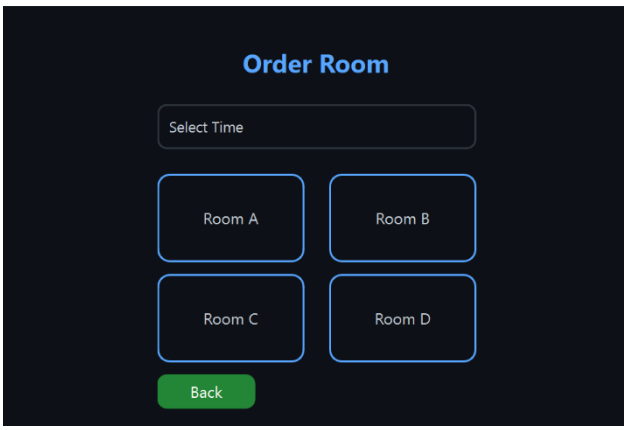
Gambar 22 Halaman Registrasi

3. Halaman Utama



Gambar 23 Halaman Utama

4. Halaman Pengecekan Ruangan

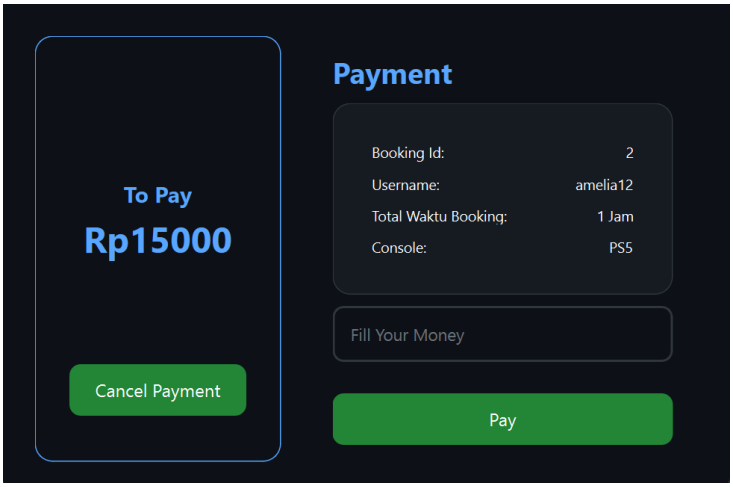


Gambar 24 Halaman Pengecekan Ruangan

5. Halaman Pemesanan

Gambar 25 Halaman Pemesanan

6. Halaman Pembayaran



Gambar 26 Halaman Pembayaran

LAMPIRAN

Repository GitHub Source Code Aplikasi Game Console Order

Source code aplikasi Game Console Order disimpan dalam sebuah repository GitHub yang dapat diakses melalui tautan berikut:

Link Source Code

Repository tersebut berisi seluruh kode program, struktur direktori, serta dokumentasi pendukung yang digunakan dalam pengembangan dan implementasi aplikasi ini.

DAFTAR PUSTAKA

ECMA International. (2017). *The JSON Data Interchange Syntax (ECMA-404)*.

Python Software Foundation. (2024). *Python Documentation*.

ReportLab Inc. (2024). *ReportLab PDF Library User Guide*.

Riverbank Computing. (2023). *PyQt6 Documentation*.