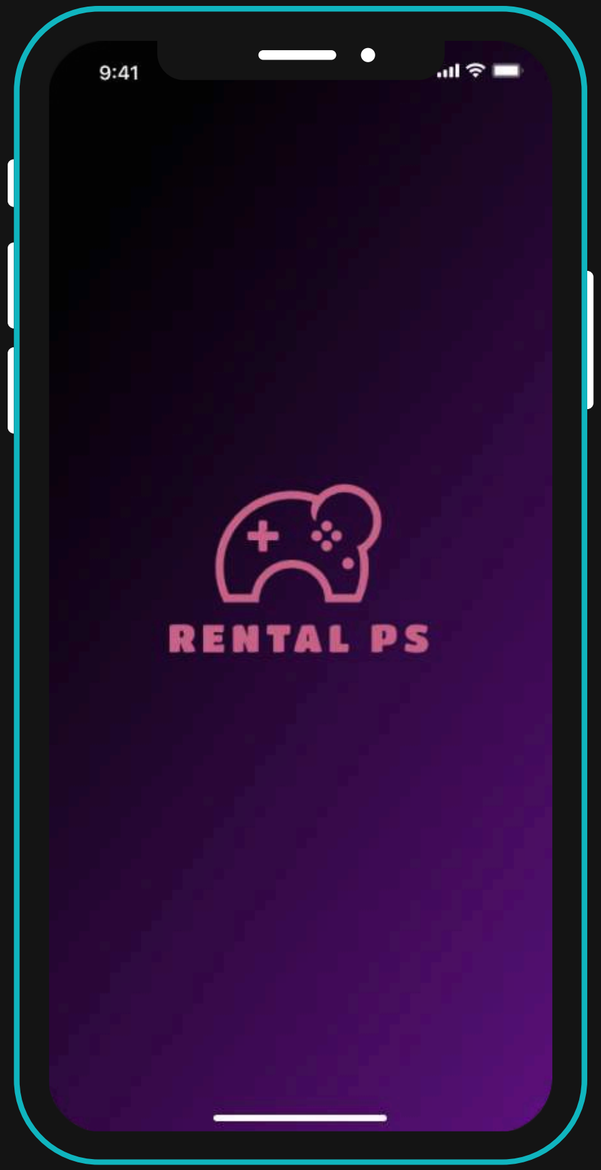
UTS Perangkat Lunak Berbasis Komponen

Playstation Rental System

by
Muhammad Bintang Indra Hidayat (2208107010023)
Shofia Nurul Huda (2208107010015)

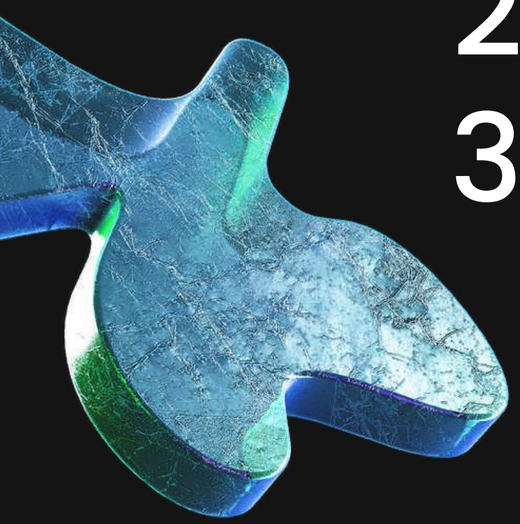


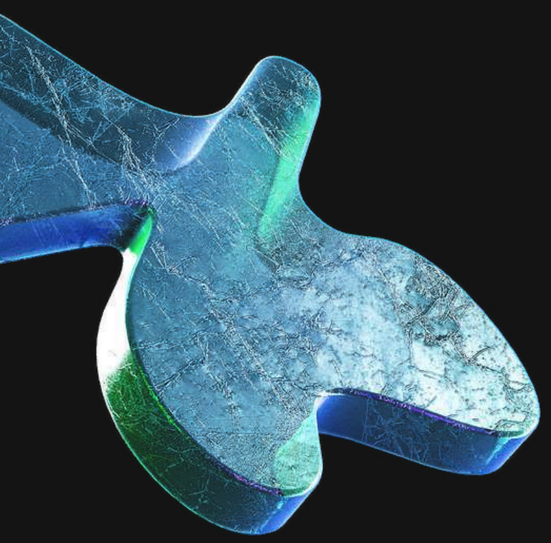
Apa itu PlayStation Rental System?

PlayStation Rental System adalah sistem yang mengelola penyewaan PlayStation, transaksi makanan/minuman, serta pencatatan keuangan. Cashier menangani transaksi rental dan penjualan, sementara Owner memantau keuangan dan mencatat pengeluaran, memastikan operasional berjalan efisien.

SPECIFICATION WORKFLOW

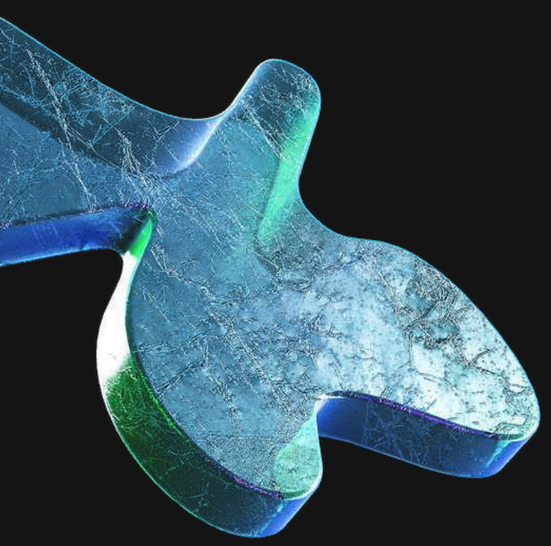
1. Component Identification
2. Component Interaction
3. Component Specification

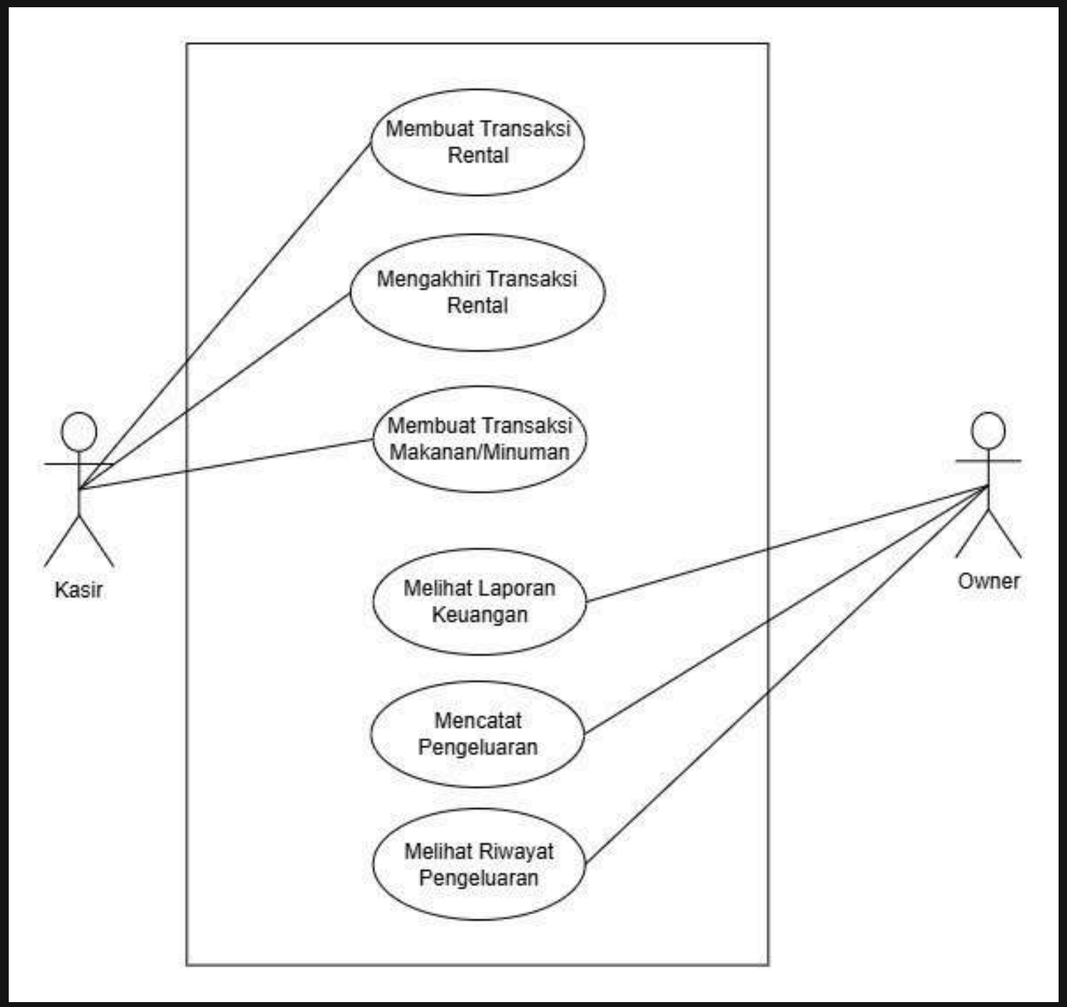


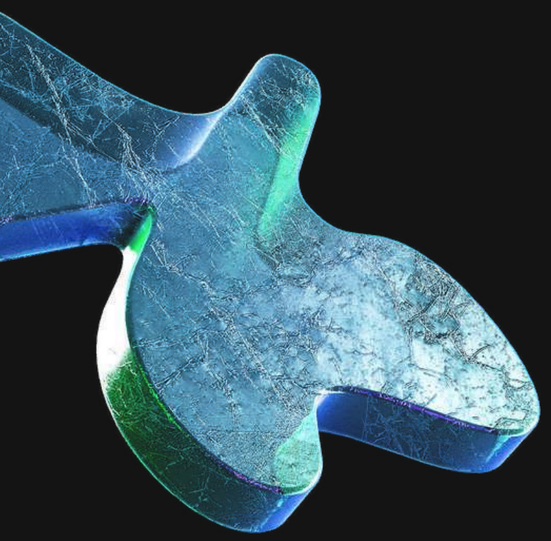


Component Identification

Use Case Diagram







Use Case Description

Membuat Transaksi Rental

Name: Membuat Transaksi Rental

Initiator: Kasir

Goal: Memulai sesi permainan pada unit PlayStation yang dipilih oleh pelanggan dan mencatat waktu mulai.

Main Success Scenario:

1. Sistem menampilkan daftar playstation yang tersedia.
2. Kasir memilih unit PlayStation yang tersedia dari daftar unit di sistem (misalnya, PS1, PS2).
3. Sistem mencatat waktu mulai sesi (timestamp) ke dalam database.
4. Sistem mengubah status unit PlayStation menjadi "InUse".
5. Sistem menampilkan pesan konfirmasi bahwa sesi rental telah dimulai.

Extensions:

1. Tidak ada unit PlayStation yang tersedia
    a. Sistem menampilkan pesan error: "Tidak ada unit tersedia saat ini."
    b. Kembali ke menu utama
2. ID unit yang dimasukkan tidak valid atau tidak ada dalam daftar tersedia
    a. Sistem menampilkan pesan error: "ID unit tidak valid atau unit tidak tersedia."
    b. Sistem menampilkan ulang daftar unit tersedia dan meminta Kasir memasukkan ID unit lain.
    c. Jika Kasir membatalkan (input: 0), kembali ke menu utama.

Mengakhiri Transaksi Rental

Name: Mengakhiri Transaksi Rental

Initiator: Kasir

Goal: Menghentikan sesi permainan, menghitung durasi dan biaya sewa, serta menampilkan total biaya kepada kasir.

Main Success Scenario:

1. Sistem menampilkan daftar unit aktif.
2. Kasir memilih unit PlayStation yang sedang digunakan dari daftar unit aktif di sistem.
3. Kasir mengkonfirmasi untuk mengakhiri sesi rental (ya/tidak).
4. Sistem mencatat waktu selesai sesi (timestamp) ke dalam database.
5. Sistem menghitung durasi permainan (dalam menit) berdasarkan selisih waktu mulai dan waktu selesai.
6. Sistem menghitung biaya sewa (Rp9.000/jam, prorata per menit, misalnya Rp150/menit).
7. Sistem menyimpan transaksi rental (ID transaksi, ID unit, waktu mulai, waktu selesai, durasi, total biaya) ke database.
8. Sistem mengubah status unit PlayStation menjadi "Available".
9. Sistem menampilkan total biaya sewa kepada kasir untuk diberikan kepada pelanggan.

Extensions:

1. Tidak ada unit PlayStation yang aktif (tidak ada sesi rental berjalan)
    a. Sistem menampilkan pesan: "Tidak ada unit aktif saat ini."
    b. Kembali ke menu utama
2. ID unit yang dimasukkan tidak valid atau tidak ada dalam daftar unit aktif
    a. Sistem menampilkan pesan error: "ID unit tidak valid atau unit tidak aktif."
    b. Sistem menampilkan ulang daftar unit aktif dan meminta Kasir memasukkan ID unit lain.
    c. Jika Kasir membatalkan (input: 0), proses berhenti dan kembali ke menu utama.

3a. Kasir mengkonfirmasi “ya”
    a. Sistem menampilkan pesan: “Pengakhiran sesi selesai”
    b. Proses selesai dan kembali ke menu utama

3b. Kasir mengkonfirmasi “tidak”
    a. Sistem menampilkan pesan: “Pengakhiran sesi dibatalkan”
    b. Proses selesai dan kembali ke menu utama

Membuat Pembelian makanan/minuman

Name: Membuat Pembelian makanan/minuman

Initiator: Kasir

Goal: Membuat pembelian makanan atau minuman oleh pelanggan dan menyimpan transaksi ke database

Main Success Scenario:

1. Sistem menampilkan daftar item makanan/minuman.
2. Kasir memasukkan ID item untuk memesan atau "s" untuk selesai.
3. Kasir memasukkan jumlah item.
4. Sistem menyimpan item dan jumlah ke daftar sementara.
5. Langkah 2–4 diulangi hingga kasir memasukkan "s".
6. Sistem menghitung total biaya dan menampilkan ringkasan transaksi.
7. Kasir mengkonfirmasi transaksi.
8. Sistem menyimpan transaksi ke database dan menampilkan total biaya ke kasir.
9. Sistem menampilkan total biaya kepada kasir untuk diberikan kepada pelanggan.

Extensions:

2. Kasir memasukkan ID item yang tidak valid
   a. Sistem menampilkan pesan error: "ID item tidak valid."
   b. Sistem menampilkan ulang daftar item dan meminta Kasir memasukkan ID item lain atau "s" untuk selesai.
   c. Jika Kasir memasukkan "s", lanjut ke langkah 6 (jika daftar sementara kosong, proses berhenti).
3. Jumlah item tidak valid (misalnya, negatif atau nol)
   a. Sistem menampilkan pesan error: "Jumlah item tidak valid, masukkan jumlah positif."
   b. Sistem meminta Kasir memasukkan jumlah yang valid untuk ID item yang sama.
   c. Jika Kasir membatalkan (input: 0), item tersebut tidak ditambahkan ke daftar sementara, kembali ke langkah 2.
7. Kasir tidak mengkonfirmasi transaksi (input: "N" atau tidak)
   a. Sistem menampilkan pesan: "Transaksi dibatalkan."
   b. Daftar sementara dibersihkan, proses berhenti dan kembali ke menu utama.

Melihat Laporan Keuangan

Name: Melihat Laporan Keuangan

Initiator: Owner

Goal: Menampilkan ringkasan keuangan, termasuk total pemasukan, pengeluaran, dan laba/rugi untuk periode tertentu.

Main Success Scenario:

1. Owner memilih periode laporan (misalnya, harian, mingguan, bulanan) di sistem.
2. Sistem mengambil data transaksi rental dan makanan/minuman untuk periode tersebut.
3. Sistem menghitung total pemasukan dari transaksi rental dan makanan/minuman.
4. Sistem mengambil data pengeluaran untuk periode tersebut.
5. Sistem menghitung total pengeluaran.
6. Sistem menghitung laba/rugi (total pemasukan - total pengeluaran).
7. Sistem menampilkan laporan keuangan (periode, total pemasukan, total pengeluaran, laba/rugi).

Extensions:

1. Tidak ada data untuk periode yang dipilih
   a. Sistem menampilkan pesan : “Belum ada data untuk periode ini”.
   b. Proses selesai dan kembali ke menu utama.

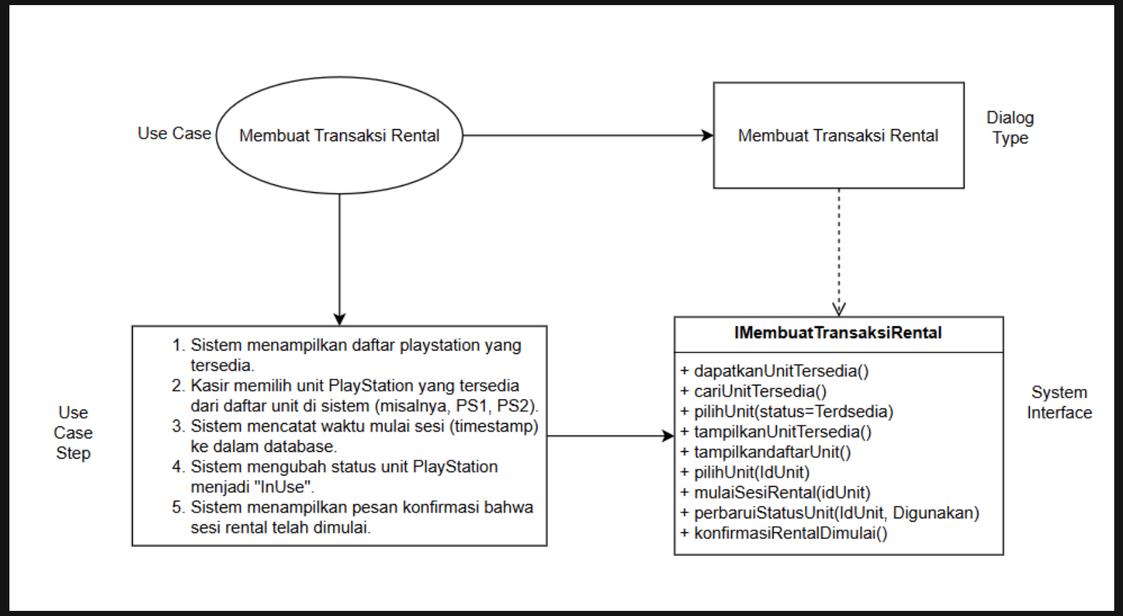
Mencatat Pengeluaran

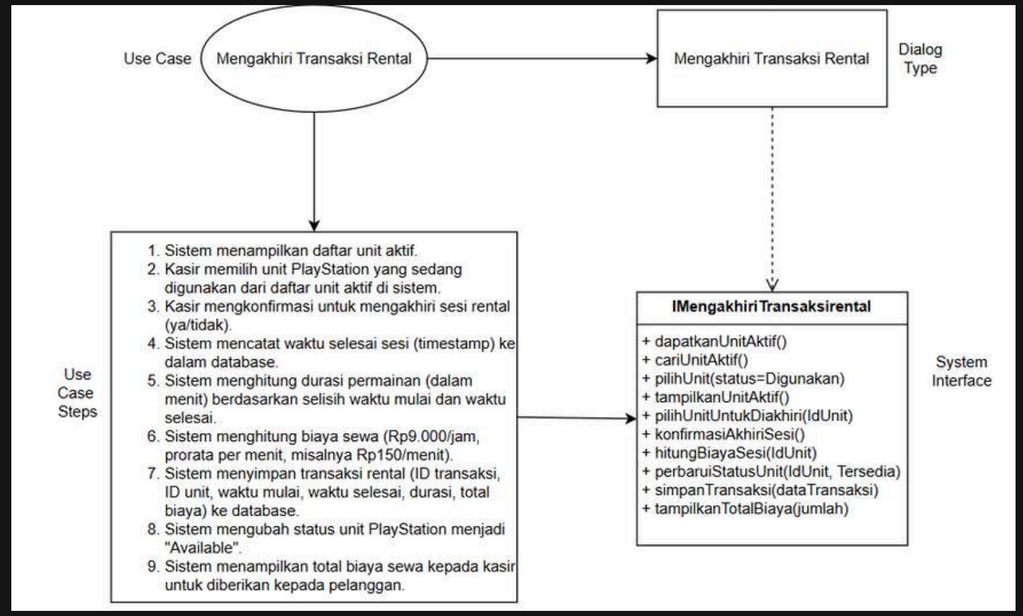
Name: Mencatat Pengeluaran
Initiator: Owner
Goal: Mencatat pengeluaran operasional (misalnya, token listrik, perawatan unit, stok makanan/minuman) ke dalam database.
Main Success Scenario: 1. Owner memilih opsi untuk mencatat pengeluaran di sistem. 2. Owner memasukkan jenis pengeluaran (misalnya, TokenListrik, Maintenance, StockPurchase). 3. Owner memasukkan jumlah pengeluaran (dalam Rupiah). 4. Owner memasukkan tanggal pengeluaran. 5. Owner memasukkan deskripsi pengeluaran (opsional). 6. Sistem memvalidasi input (misalnya, jumlah tidak negatif, tanggal valid). 7. Sistem menyimpan data pengeluaran (ID pengeluaran, jenis, jumlah, tanggal, deskripsi) ke database. 8. Sistem menampilkan pesan konfirmasi: "Pengeluaran berhasil dicatat."
Extensions:

Melihat Riwayat Pengeluaran

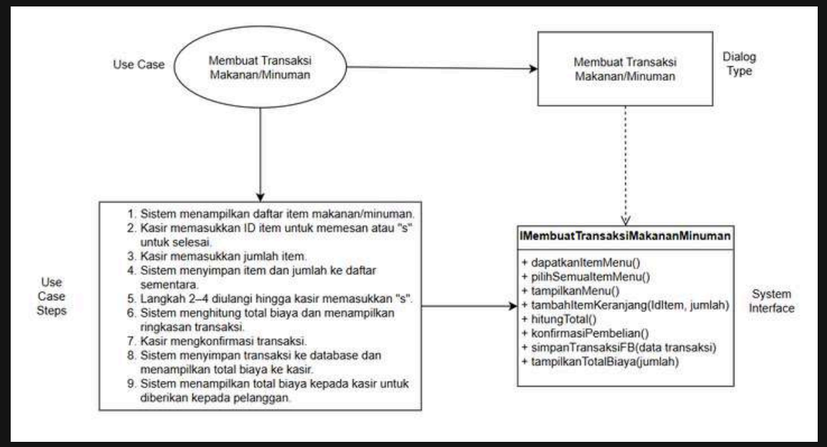
Name: Melihat Riwayat Pengeluaran
Initiator: Owner
Goal: Menampilkan daftar pengeluaran yang telah dicatat, termasuk tanggal, jenis, jumlah, dan deskripsi.
Main Success Scenario: 1. Owner memilih opsi untuk melihat riwayat pengeluaran di sistem. 2. Sistem menampilkan daftar pengeluaran (ID pengeluaran, tanggal, jenis, jumlah, deskripsi).
Extensions:

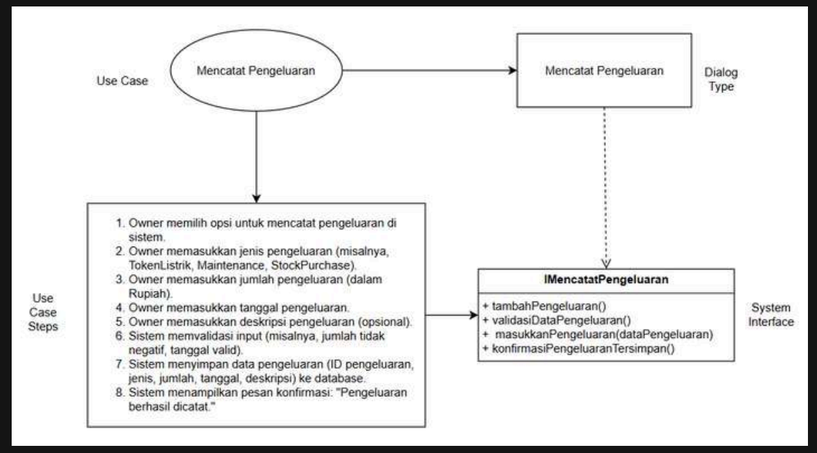
System Interface

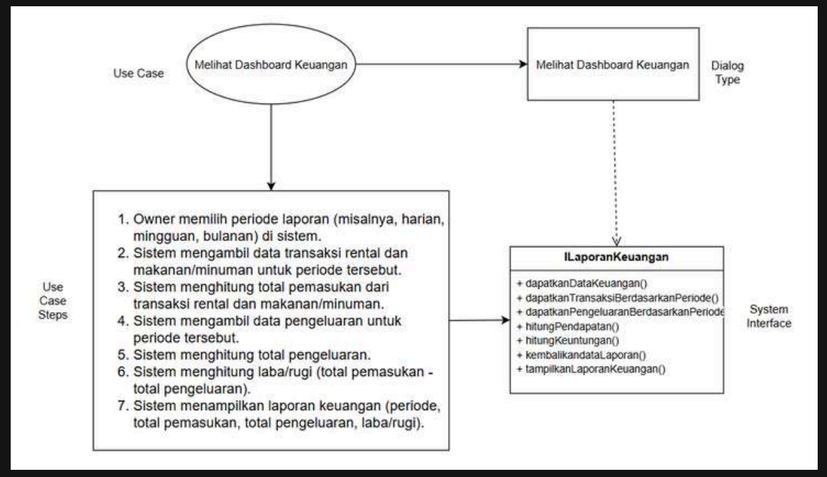


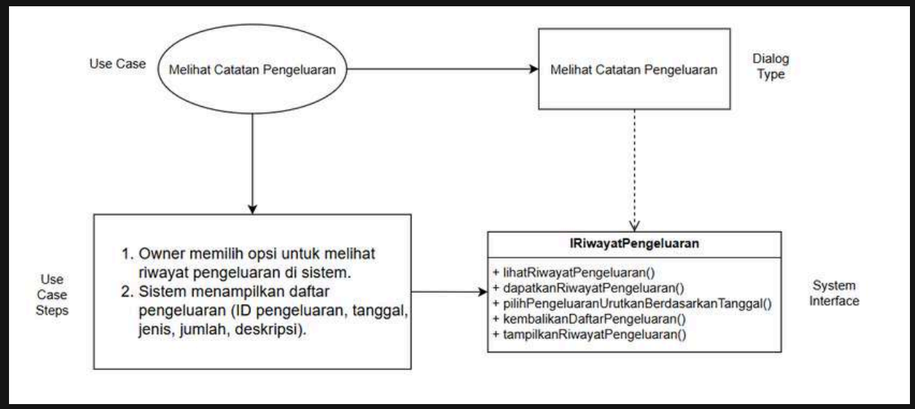


System Interface

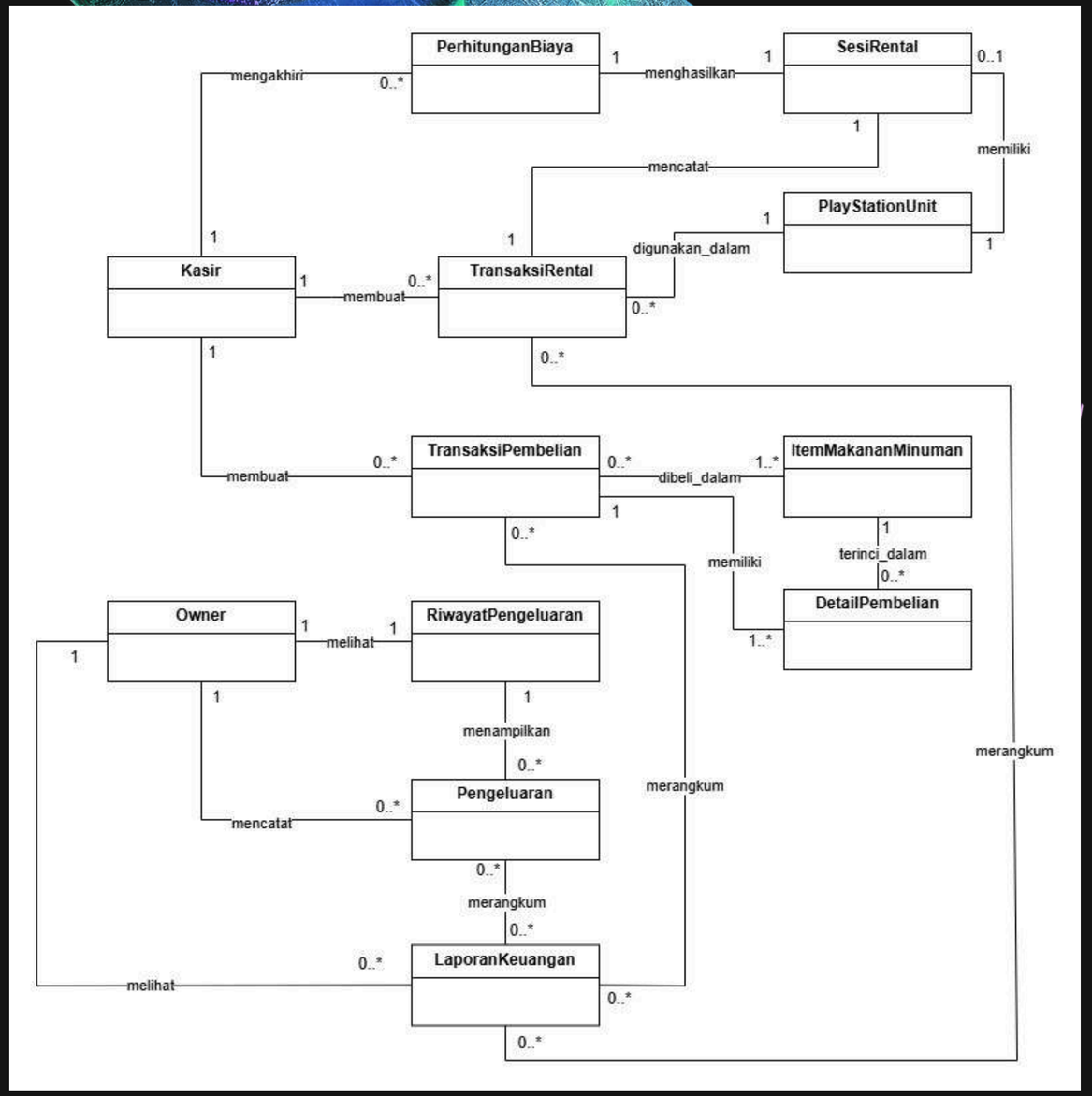




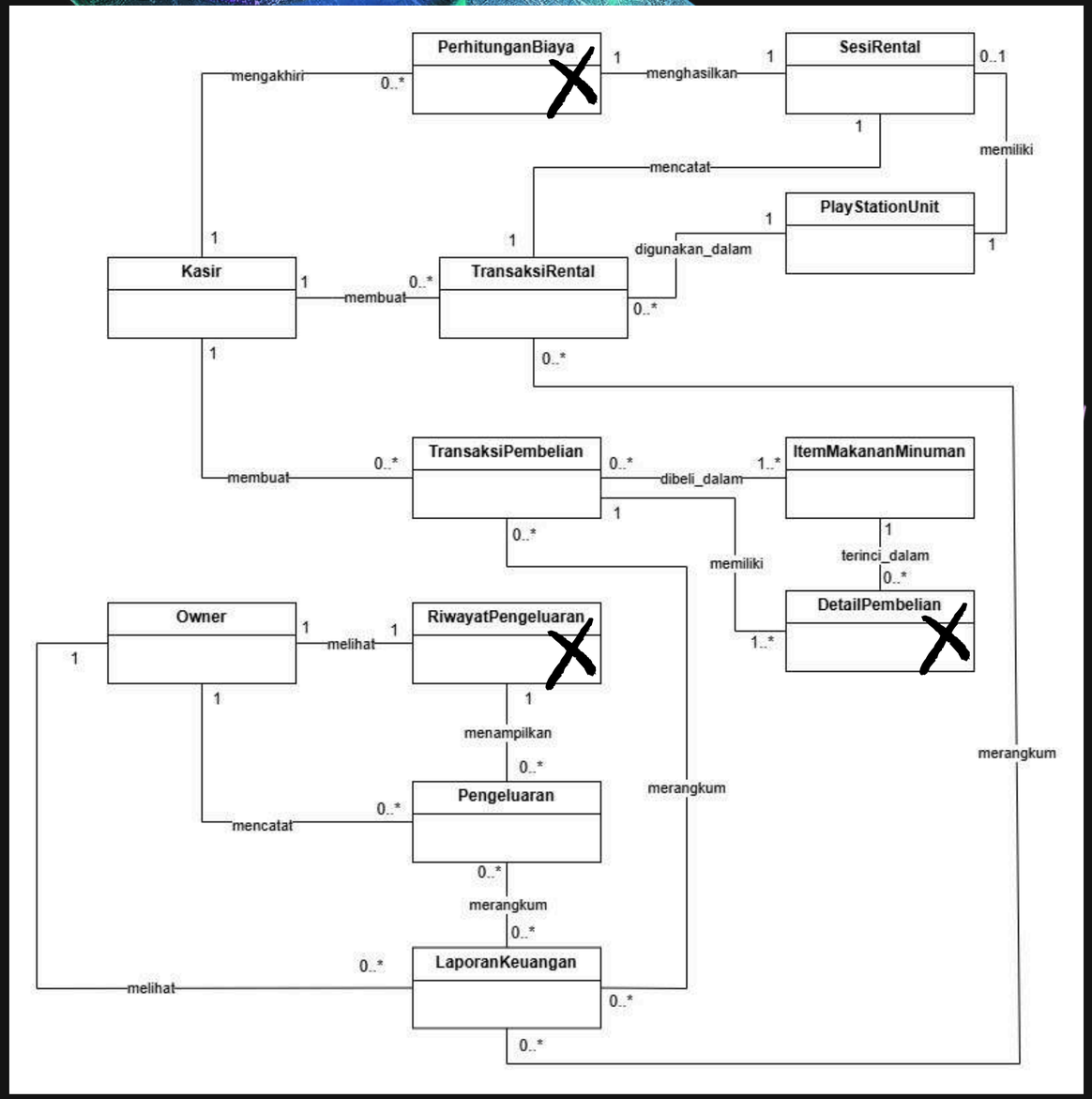




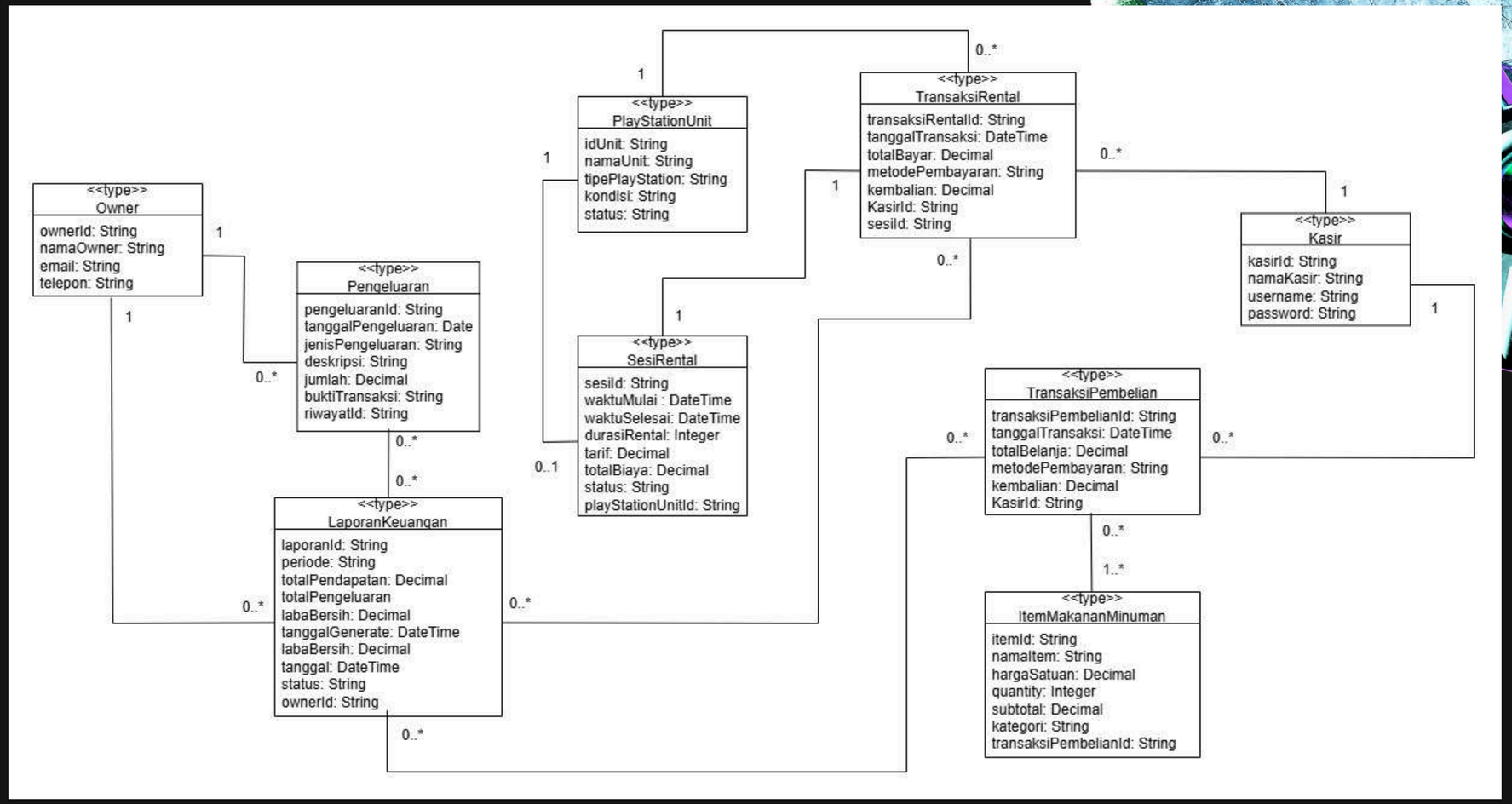
Business Concept Model



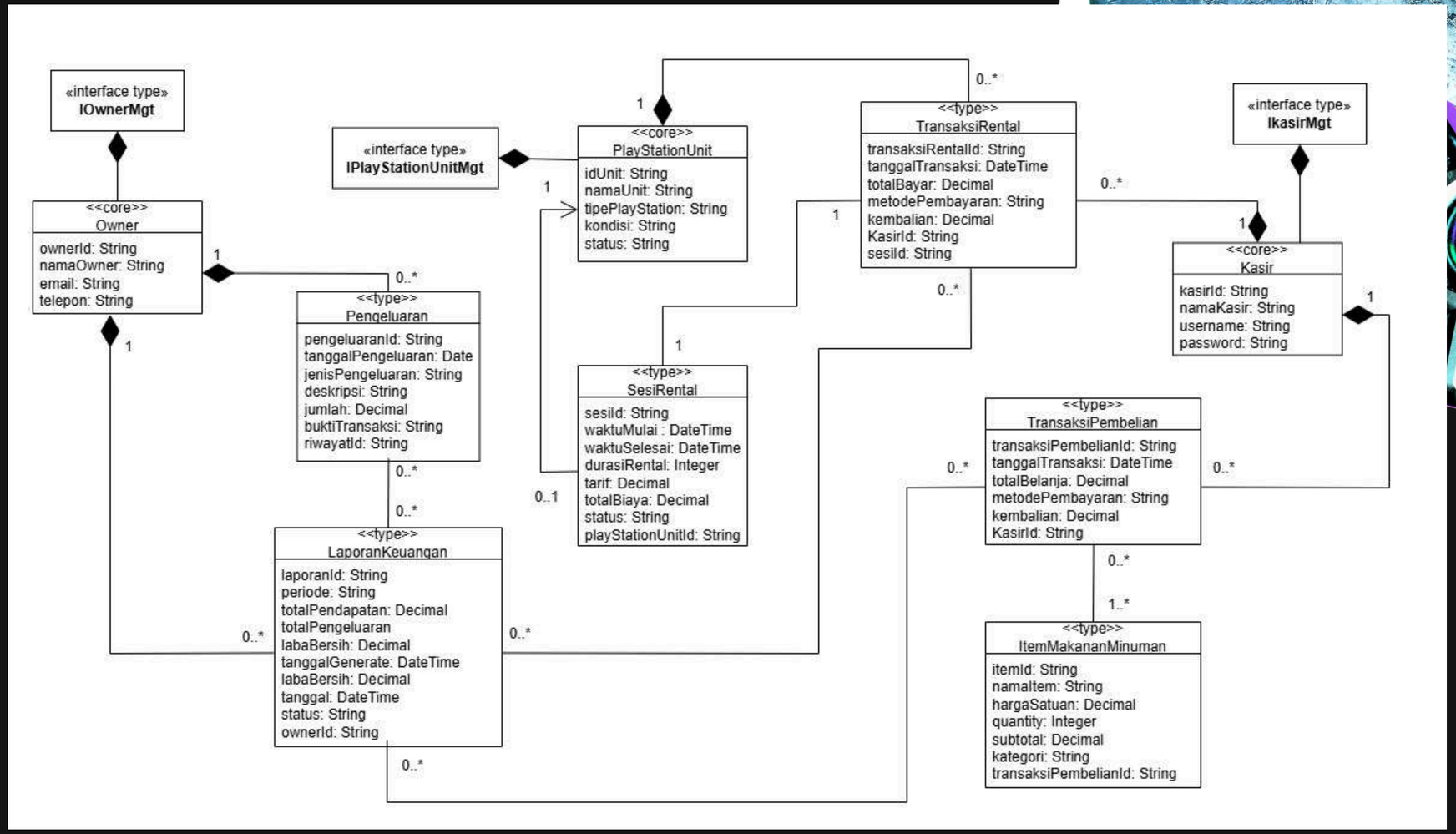
- PerhitunganBiaya dieliminasi karena hanya merupakan proses kalkulasi tanpa nilai sebagai data permanen.
- DetailPembelian dieliminasi karena informasinya sudah tercakup dalam entitas ItemMakananMinuman.
- RiwayatPengeluaran dieliminasi karena hanya merupakan hasil agregasi dari entitas Pengeluaran



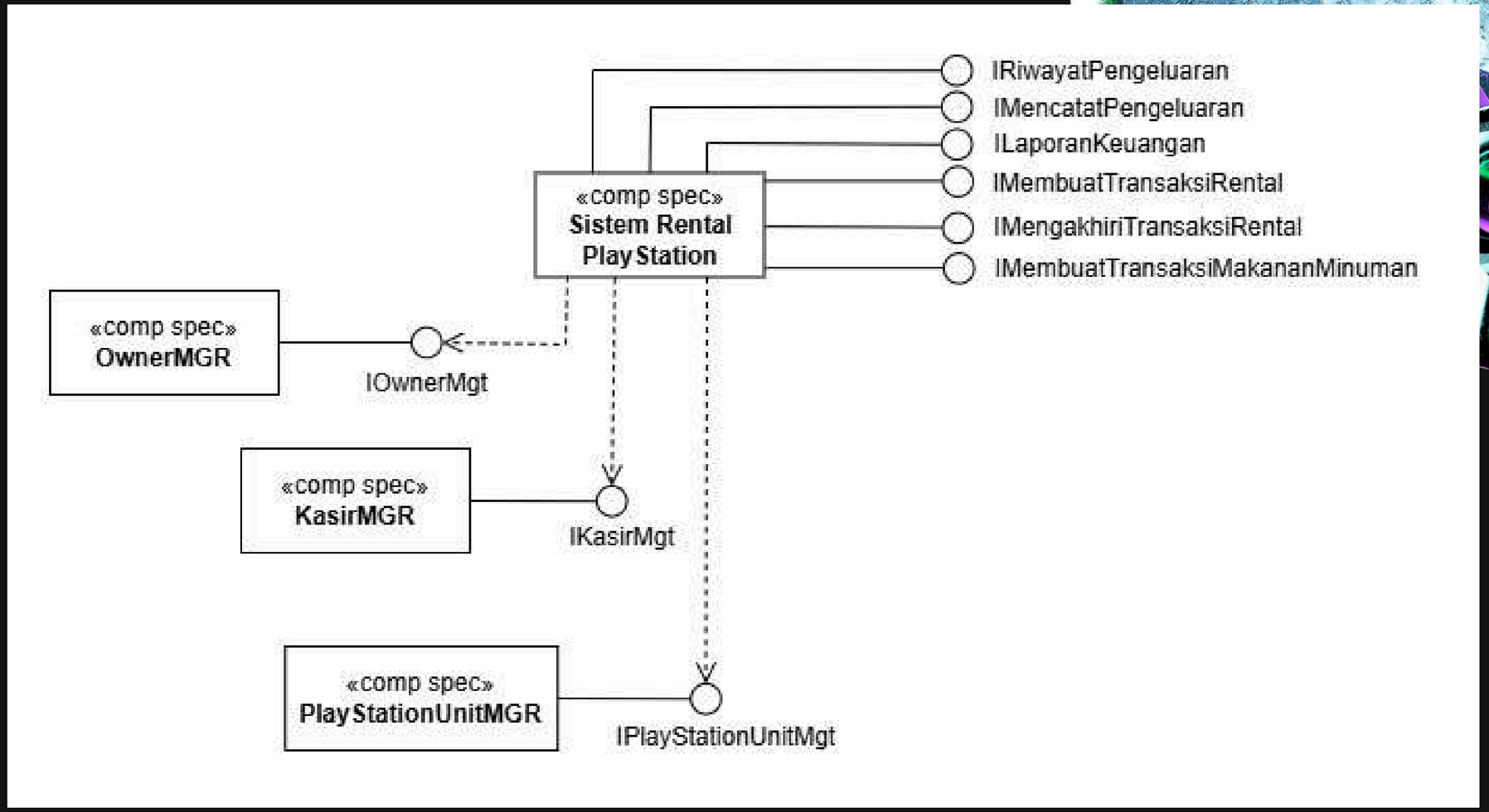
Business Type Model

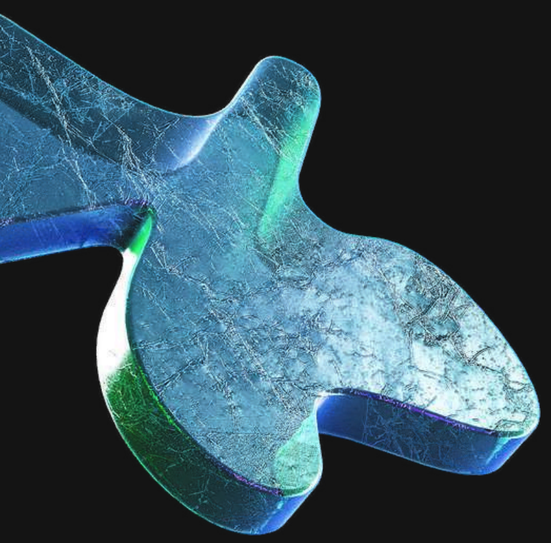


Business Interface



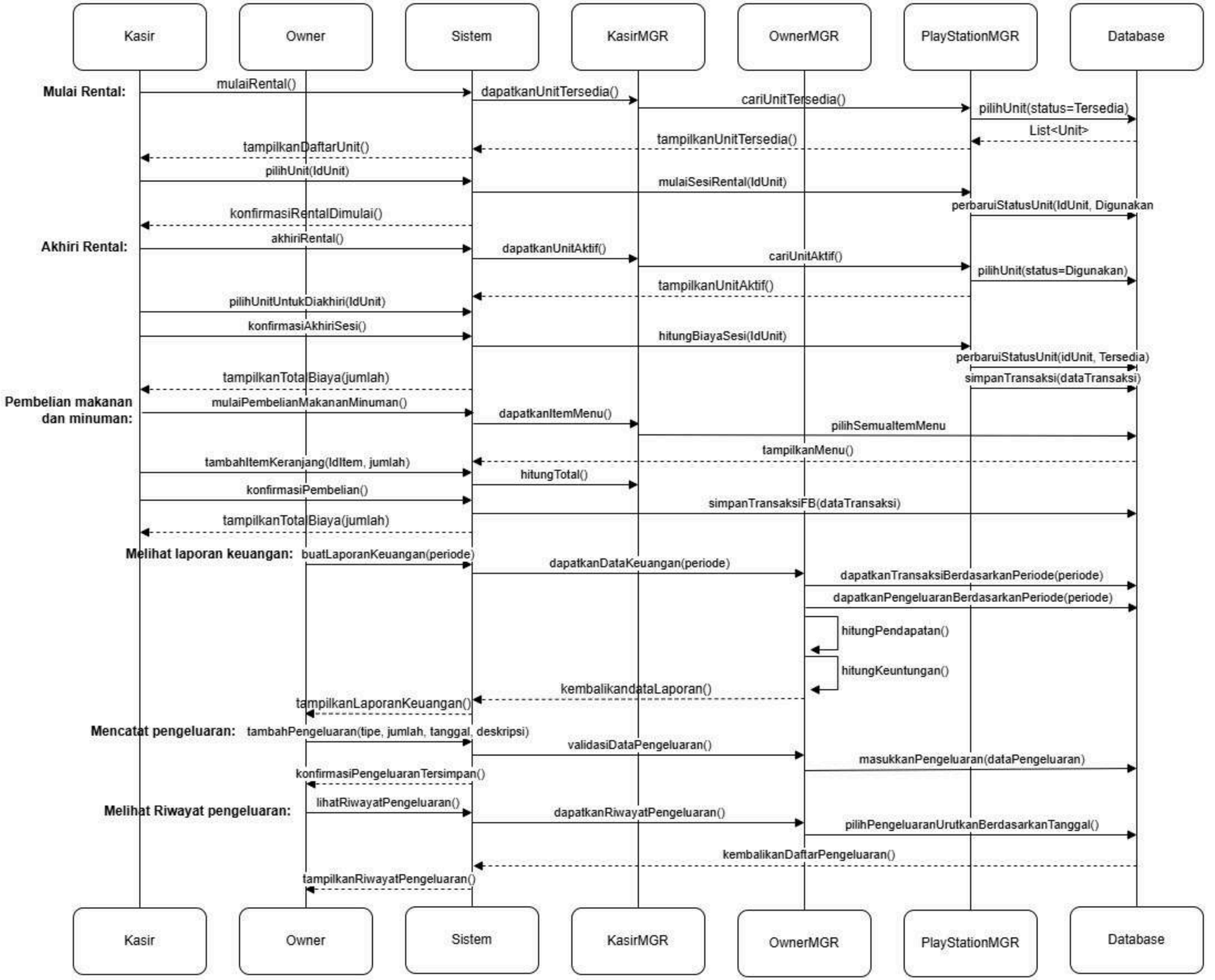
Initial Component Diagram

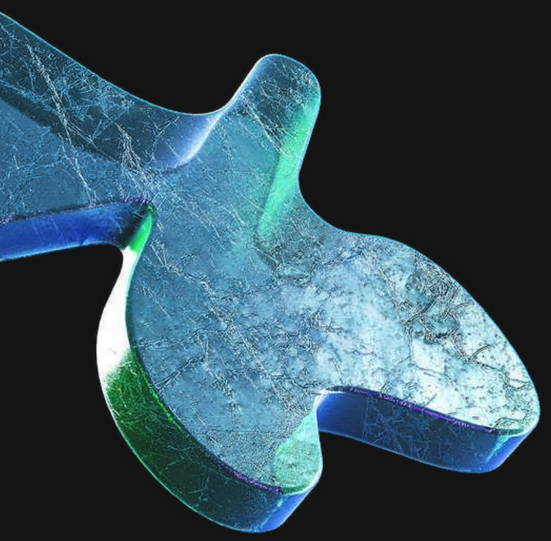




Component Interaction

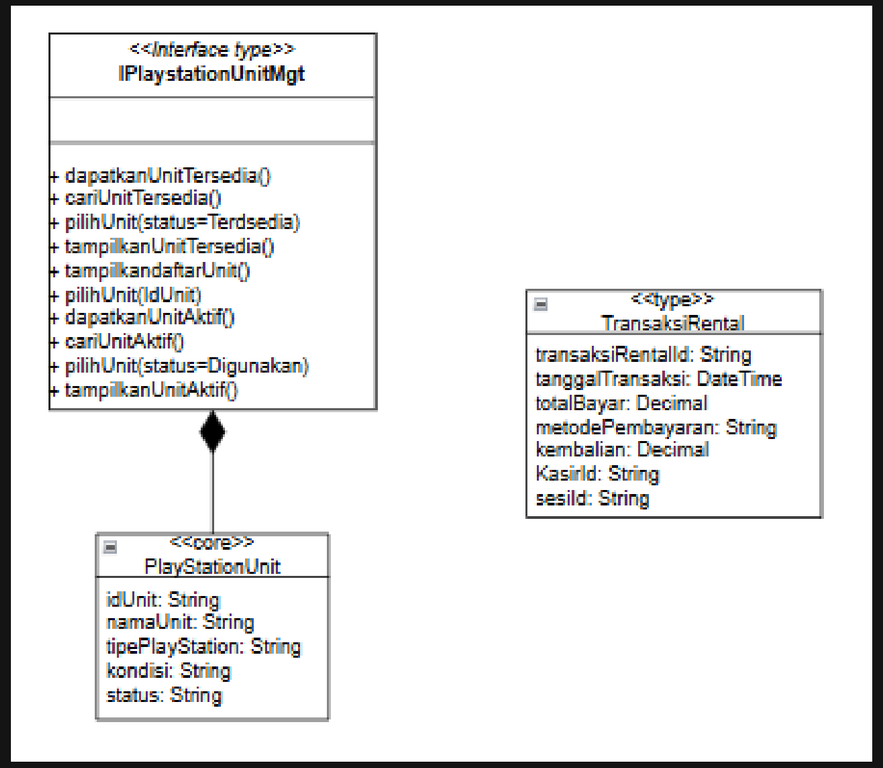
Component Interaction Diagram/ Sequence Diagram





Component Specification

Interface Specification Diagram Untuk IPlayStationMgt & OCL IPlayStationMgt



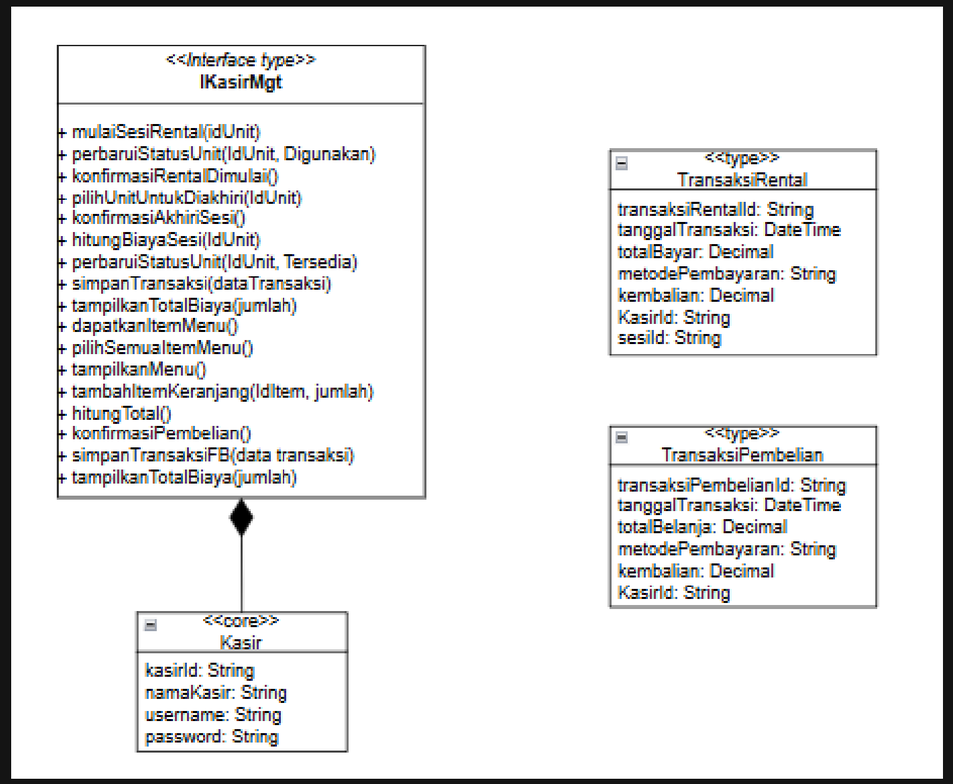

```
-- Invariant: dapatkanUnitTersedia mengembalikan semua unit dengan status Available
context IPlayStationUnitMgt
inv dapatkanUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Invariant: cariUnitTersedia mengembalikan unit dengan status Available
context IPlayStationUnitMgt
inv cariUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Konteks: IPlayStationUnitMgt::pilihUnit(status: String): PlayStationUnit
context IPlayStationUnitMgt::pilihUnit(status: String): PlayStationUnit
pre:
  -- Status harus valid dan unit harus ada
  (status = 'Available' or status = 'InUse') and PlayStationUnit.allInstances()->exists(u | u.status = status)
post:
  -- Returns a unit with the specified status
  PlayStationUnit.allInstances()->exists(u | u = result and u.status = status)

-- Invariant: tampilkanUnitTersedia mengembalikan semua unit dengan status Available
context IPlayStationUnitMgt
inv tampilkanUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Invariant: tampilkandaftarUnit mengembalikan semua unit
context IPlayStationUnitMgt
inv tampilkandaftarUnit: PlayStationUnit.allInstances()->asSet()->notEmpty()

-- Konteks: IPlayStationUnitMgt::pilihUnit(idUnit: String): PlayStationUnit
context IPlayStationUnitMgt::pilihUnit(idUnit: String): PlayStationUnit
pre:
  -- Unit harus ada
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit)
post:
  -- Returns the selected unit
  PlayStationUnit.allInstances()->exists(u | u = result and u.idUnit = idUnit)

-- Konteks: IPlayStationUnitMgt::mulaiSesiRental(idUnit: String): RentalTransaction
context IPlayStationUnitMgt::mulaiSesiRental(idUnit: String): RentalTransaction
pre:
  -- Unit harus ada dan statusnya Available
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = 'Available')
post:
  -- Creates a new rental transaction
  RentalTransaction.allInstances()->exists(t | t.idUnit = idUnit and t.startTime = DateTime(2025, 5, 30, 16, 29) and t = result)

```

Interface Specification Diagram Untuk IPlayStationMgt & OCL IPlayStationMgt

```
-- Konteks: IPlayStationUnitMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
context IPlayStationUnitMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
pre:
  -- Unit harus ada dan status baru harus valid
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit) and (status = 'Available' or status = 'InUse')
post:
  -- Status unit diperbarui
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = status) and result = 0

-- Invariant: konfirmasiRentalDimulai mengembalikan pesan konfirmasi
context IPlayStationUnitMgt
inv konfirmasiRentalDimulai: result = 'Sesi rental telah dimulai'

-- Invariant: dapatkanUnitAktif mengembalikan semua unit dengan status InUse
context IPlayStationUnitMgt
inv dapatkanUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Invariant: cariUnitAktif mengembalikan unit dengan status InUse
context IPlayStationUnitMgt
inv cariUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Konteks: IPlayStationUnitMgt::pilihUnit(status: String): PlayStationUnit
context IPlayStationUnitMgt::pilihUnit(status: String): PlayStationUnit
pre:
  -- Status harus valid dan unit harus ada
  (status = 'Available' or status = 'InUse') and PlayStationUnit.allInstances()->exists(u | u.status = status)
post:
  -- Returns a unit with the specified status
  PlayStationUnit.allInstances()->exists(u | u = result and u.status = status)

-- Invariant: tampilkanUnitAktif mengembalikan semua unit dengan status InUse
context IPlayStationUnitMgt
inv tampilkanUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Konteks: IPlayStationUnitMgt::pilihUnitUntukDiakhiri(idUnit: String): PlayStationUnit
context IPlayStationUnitMgt::pilihUnitUntukDiakhiri(idUnit: String): PlayStationUnit
pre:
  -- Unit harus ada dan statusnya InUse
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = 'InUse')
post:
  -- Returns the selected unit
  PlayStationUnit.allInstances()->exists(u | u = result and u.idUnit = idUnit and u.status = 'InUse')
```

```
-- Invariant: konfirmasiAkhiriSesi mengembalikan nilai konfirmasi
context IPlayStationUnitMgt
inv konfirmasiAkhiriSesi: result = true or result = false

-- Konteks: IPlayStationUnitMgt::hitungBiayaSesi(idUnit: String): Real
context IPlayStationUnitMgt::hitungBiayaSesi(idUnit: String): Real
pre:
  -- Unit harus memiliki transaksi rental yang aktif
  RentalTransaction.allInstances()->exists(t | t.idUnit = idUnit and t.endTime = null)
post:
  -- Biaya dihitung berdasarkan durasi (Rp150/m | t.endTime = null)->asSequence()->first() in
  transaction.duration = (DateTime(2025, 5, 30, 16, 29) - transaction.startTime).minutes() and
  result = transaction.duration * 150

-- Konteks: IPlayStationUnitMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
context IPlayStationUnitMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
pre:
  -- Unit harus ada dan status baru harus valid
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit) and (status = 'Available' or status = 'InUse')
post:
  -- Status unit diperbarui
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = status) and result = 0

-- Konteks: IPlayStationUnitMgt::simpanTransaksi(dataTransaksi: RentalTransaction): Integer
context IPlayStationUnitMgt::simpanTransaksi(dataTransaksi: RentalTransaction): Integer
pre:
  -- Data transaksi harus valid
  dataTransaksi.idUnit <> '' and dataTransaksi.totalCost >= 0
post:
  -- Transaksi disimpan
  RentalTransaction.allInstances()->exists(t | t = dataTransaksi) and result = 0

-- Invariant: tampilkanTotalBiaya mengembalikan total cost sebagai string
context IPlayStationUnitMgt
inv tampilkanTotalBiaya: result = jumlah.toString()
```

Interface Specification Diagram Untuk IKasirMgt & OCL IKasirMgt



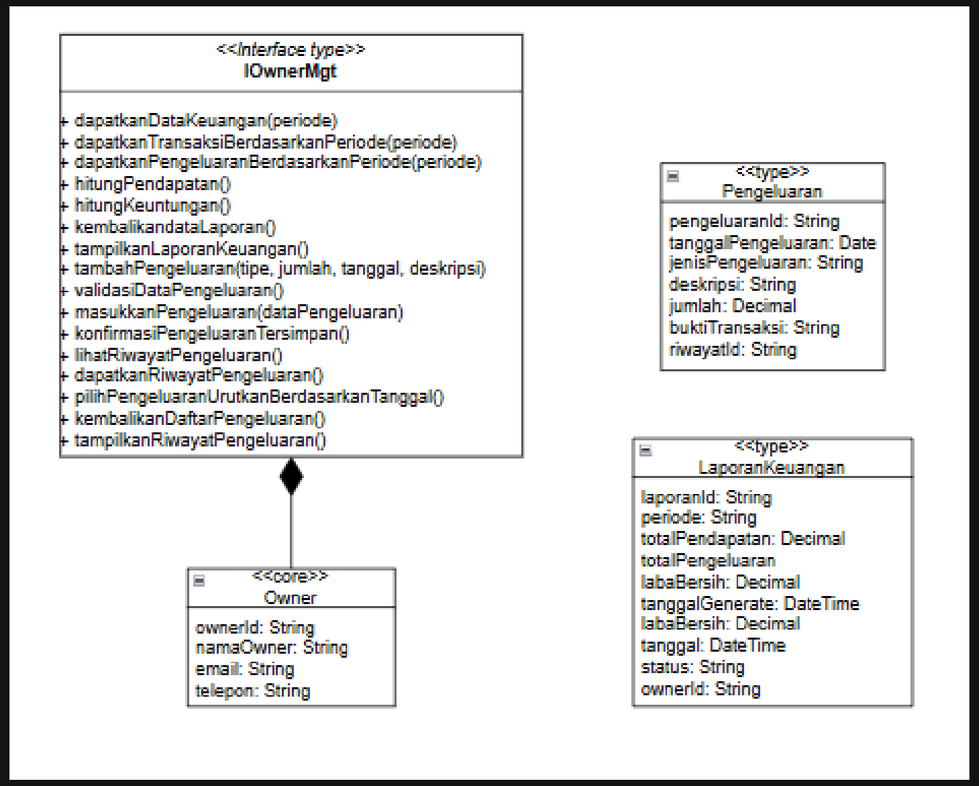

```
-- Invariant: dapatkanUnitTersedia mengembalikan semua unit dengan status Available
context IKasirMgt
inv dapatkanUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Invariant: cariUnitTersedia mengembalikan unit dengan status Available
context IKasirMgt
inv cariUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Konteks: IKasirMgt::pilihUnit(status: String): PlayStationUnit
context IKasirMgt::pilihUnit(status: String): PlayStationUnit
pre:
  -- Status harus Available
  status = 'Available' and PlayStationUnit.allInstances()->exists(u | u.status = 'Available')
post:
  -- Returns a unit with status Available
  PlayStationUnit.allInstances()->exists(u | u = result and u.status = 'Available')

-- Invariant: tampilkanUnitTersedia mengembalikan semua unit dengan status Available
context IKasirMgt
inv tampilkanUnitTersedia: PlayStationUnit.allInstances()->select(u | u.status = 'Available')->asSet()->notEmpty()

-- Invariant: tampilkandaftarUnit mengembalikan semua unit
context IKasirMgt
inv tampilkandaftarUnit: PlayStationUnit.allInstances()->asSet()->notEmpty()

-- Konteks: IKasirMgt::pilihUnit(idUnit: String): PlayStationUnit
context IKasirMgt::pilihUnit(idUnit: String): PlayStationUnit
pre:
  -- Unit harus ada dan statusnya Available
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = 'Available')
post:
  -- Returns the selected unit
  PlayStationUnit.allInstances()->exists(u | u = result and u.idUnit = idUnit and u.status = 'Available')

-- Konteks: IKasirMgt::mulaiSesiRental(idUnit: String): RentalTransaction
context IKasirMgt::mulaiSesiRental(idUnit: String): RentalTransaction
pre:
  -- Unit harus ada dan statusnya Available
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = 'Available')
post:
  -- Creates a new rental transaction with start time
  RentalTransaction.allInstances()->exists(t | t.idUnit = idUnit and t.startTime = DateTime(2025, 5, 30, 16, 24) and t = result)

-- Konteks: IKasirMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
context IKasirMgt::perbaruiStatusUnit(idUnit: String, status: String): Integer
pre:
  -- Unit harus ada dan status baru harus valid
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit) and (status = 'Available' or status = 'InUse')
post:
  -- Status unit diperbarui
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = status) and result = 0

-- Invariant: konfirmasiRentalDimulai mengembalikan pesan konfirmasi
context IKasirMgt
inv konfirmasiRentalDimulai: result = 'Sesi rental telah dimulai'
```

Interface Specification Diagram Untuk IKasirMgt & OCL IKasirMgt

```
-- Invariant: dapatkanUnitAktif mengembalikan semua unit dengan status InUse
context IKasirMgt
inv dapatkanUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Invariant: cariUnitAktif mengembalikan unit dengan status InUse
context IKasirMgt
inv cariUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Konteks: IKasirMgt::pilihUnit(status: String): PlayStationUnit
context IKasirMgt::pilihUnit(status: String): PlayStationUnit
pre:
  -- Status harus InUse
  status = 'InUse' and PlayStationUnit.allInstances()->exists(u | u.status = 'InUse')
post:
  -- Returns a unit with status InUse
  PlayStationUnit.allInstances()->exists(u | u = result and u.status = 'InUse')

-- Invariant: tampilkanUnitAktif mengembalikan semua unit dengan status InUse
context IKasirMgt
inv tampilkanUnitAktif: PlayStationUnit.allInstances()->select(u | u.status = 'InUse')->asSet()->notEmpty()

-- Konteks: IKasirMgt::pilihUnitUntukDiakhiri(idUnit: String): PlayStationUnit
context IKasirMgt::pilihUnitUntukDiakhiri(idUnit: String): PlayStationUnit
pre:
  -- Unit harus ada dan statusnya InUse
  PlayStationUnit.allInstances()->exists(u | u.idUnit = idUnit and u.status = 'InUse')
post:
  -- Returns the selected unit
  PlayStationUnit.allInstances()->exists(u | u = result and u.idUnit = idUnit and u.status = 'InUse')

-- Invariant: konfirmasiAkhiriSesi mengembalikan nilai konfirmasi
context IKasirMgt
inv konfirmasiAkhiriSesi: result = true or result = false

-- Konteks: IKasirMgt::hitungBiayaSesi(idUnit: String): Real
context IKasirMgt::hitungBiayaSesi(idUnit: String): Real
pre:
  -- Unit harus memiliki transaksi rental yang aktif
  RentalTransaction.allInstances()->exists(t | t.idUnit = idUnit and t.endTime = null)
post:
  -- Biaya dihitung berdasarkan durasi (Rp150/menit)
  let transaction: RentalTransaction = RentalTransaction.allInstances()->select(t | t.idUnit = idUnit and t.endTime = null)->asSequence()->first()
  transaction.duration = (DateTime(2025, 5, 30, 16, 24) - transaction.startTime).minutes() and
  result = transaction.duration * 150

-- Konteks: IKasirMgt::simpanTransaksi(dataTransaksi: RentalTransaction): Integer
context IKasirMgt::simpanTransaksi(dataTransaksi: RentalTransaction): Integer
pre:
  -- Data transaksi harus valid
  dataTransaksi.idUnit <> '' and dataTransaksi.totalCost >= 0
post:
  -- Transaksi disimpan
  RentalTransaction.allInstances()->exists(t | t = dataTransaksi) and result = 0
```

```
-- Invariant: tampilkanTotalBiaya mengembalikan total cost sebagai string
context IKasirMgt
inv tampilkanTotalBiaya: result = jumlah.toString()

-- Invariant: dapatkanItemMenu mengembalikan semua item menu
context IKasirMgt
inv dapatkanItemMenu: Item.allInstances()->asSet()->notEmpty()

-- Invariant: pilihSemuaItemMenu mengembalikan semua item menu
context IKasirMgt
inv pilihSemuaItemMenu: Item.allInstances()->asSet()->notEmpty()

-- Invariant: tampilkanMenu mengembalikan semua item menu
context IKasirMgt
inv tampilkanMenu: Item.allInstances()->asSet()->notEmpty()

-- Konteks: IKasirMgt::tambahItemKeranjang(idItem: String, jumlah: Integer): Integer
context IKasirMgt::tambahItemKeranjang(idItem: String, jumlah: Integer): Integer
pre:
  -- ID item harus valid dan jumlah harus positif
  Item.allInstances()->exists(i | i.idItem = idItem) and jumlah > 0
post:
  -- Item ditambahkan ke keranjang
  let transaction: FoodBeverageTransaction = FoodBeverageTransaction.allInstances()->asSequence()->last() in
  transaction.items->exists(i | i.idItem = idItem and i.quantity = jumlah) and result = 0

-- Invariant: hitungTotal mengembalikan total biaya transaksi makanan/minuman
context IKasirMgt
inv hitungTotal: let transaction: FoodBeverageTransaction = FoodBeverageTransaction.allInstances()->asSequence()->last() in
                 transaction.totalCost = transaction.items->collect(i | i.price * i.quantity)->sum()

-- Invariant: konfirmasiPembelian mengembalikan nilai konfirmasi
context IKasirMgt
inv konfirmasiPembelian: result = true or result = false

-- Konteks: IKasirMgt::simpanTransaksiFB(dataTransaksi: FoodBeverageTransaction): Integer
context IKasirMgt::simpanTransaksiFB(dataTransaksi: FoodBeverageTransaction): Integer
pre:
  -- Data transaksi harus valid
  dataTransaksi.items->notEmpty() and dataTransaksi.totalCost >= 0
post:
  -- Transaksi disimpan
  FoodBeverageTransaction.allInstances()->exists(t | t = dataTransaksi) and result = 0

-- Invariant: tampilkanTotalBiaya mengembalikan total cost sebagai string
context IKasirMgt
inv tampilkanTotalBiaya: result = jumlah.toString()
```

Interface Specification Diagram Untuk IOwnerMgt & OCL IOwnerMgt




```
-- Invariant: dapatkanDataKeuangan mengembalikan data keuangan terakhir
context IOwnerMgt
inv dapatkanDataKeuangan: FinancialReport.allInstances()->asSequence()-
>last()->notEmpty()

-- Invariant: dapatkanTransaksiBerdasarkanPeriode mengembalikan transaksi
dalam periode
context IOwnerMgt
inv dapatkanTransaksiBerdasarkanPeriode: let report: FinancialReport =
FinancialReport.allInstances()->asSequence()->last() in
                                        RentalTransaction.allInstances()-
>select(t | t.startTime >= report.periodStart and t.startTime <=
report.periodEnd)->asSet()
                                        -
>union(FoodBeverageTransaction.allInstances()->select(t | t.transactionTime
>= report.periodStart and t.transactionTime <= report.periodEnd)->asSet())-
>notEmpty()

-- Invariant: dapatkanPengeluaranBerdasarkanPeriode mengembalikan
pengeluaran dalam periode
context IOwnerMgt
inv dapatkanPengeluaranBerdasarkanPeriode: let report: FinancialReport =
FinancialReport.allInstances()->asSequence()->last() in
                                        Expense.allInstances()->select(e
| e.date >= report.periodStart and e.date <= report.periodEnd)->asSet()-
>notEmpty()

-- Invariant: hitungPendapatan mengembalikan total pendapatan
context IOwnerMgt
inv hitungPendapatan: (RentalTransaction->collect(t: RentalTransaction |
t.totalCost)->sum() +
                        FoodBeverageTransaction->collect(t:
FoodBeverageTransaction | t.totalCost)->sum()) >= 0

-- Invariant: hitungKeuntungan mengembalikan keuntungan
context IOwnerMgt
inv hitungKeuntungan: let revenue: Real = RentalTransaction->collect(t:
RentalTransaction | t.totalCost)->sum() +
                                        FoodBeverageTransaction->collect(t:
FoodBeverageTransaction | t.totalCost)->sum() in
                        let expenses: Real = Expense->collect(e: Expense |
e.amount)->sum() in
                        revenue - expenses >= 0
```

Interface Specification Diagram Untuk IPlayStationMgt & OCL IOwnerMgt

```
-- Invariant: kembalikandataLaporan mengembalikan laporan keuangan terakhir
context IOwnerMgt
inv kembalikandataLaporan: FinancialReport.allInstances()->asSequence()->last()->notEmpty()

-- Invariant: tampilkanLaporanKeuangan mengembalikan laporan keuangan yang valid
context IOwnerMgt
inv tampilkanLaporanKeuangan: let report: FinancialReport = FinancialReport.allInstances()->asSequence()->last() in
                              report.reportId = FinancialReport.reportId and
                              report.period = FinancialReport.period and
                              report.totalIncome = RentalTransaction->select(t: RentalTransaction | t.startTime >= report.periodStart and t.startTime <= report.periodEnd)->
                                                   collect(t: RentalTransaction | t.totalCost)->sum() +
                                                   FoodBeverageTransaction->select(t: FoodBeverageTransaction | t.transactionTime >= report.periodStart and t.transactionTime <= report.periodEnd)->
                                                   collect(t: FoodBeverageTransaction | t.totalCost)->sum() and
                              report.totalExpense = Expense->select(e: Expense | e.date >= report.periodStart and e.date <= report.periodEnd)->
                                                    collect(e: Expense | e.amount)->sum() and
                              report.profit = report.totalIncome - report.totalExpense and
                              report.createdDate = DateTime(2025, 5, 30, 16, 34) and
                              report.status = 'completed' and
                              report.ownerId = Owner.ownerId
```

```
-- Konteks: IOwnerMgt::tambahPengeluaran(dataPengeluaran: Expense, out pengeluaranId: String): Integer
context IOwnerMgt::tambahPengeluaran(dataPengeluaran: Expense, out pengeluaranId: String): Integer
pre:
  -- Data pengeluaran harus valid
  dataPengeluaran.amount > 0 and dataPengeluaran.description <> ''
post:
  -- A new Expense is created
  Expense->exists(e: Expense | e.idExpense = pengeluaranId and e.amount = dataPengeluaran.amount and e.description = dataPengeluaran.description)
  and result = 0 -- Success

-- Invariant: validasiDataPengeluaran memvalidasi data pengeluaran
context IOwnerMgt
inv validasiDataPengeluaran: result = (dataPengeluaran.amount > 0 and dataPengeluaran.date <= DateTime(2025, 5, 30, 16, 34))

-- Konteks: IOwnerMgt::masukkanPengeluaran(dataPengeluaran: Expense): Integer
context IOwnerMgt::masukkanPengeluaran(dataPengeluaran: Expense): Integer
pre:
  -- Data pengeluaran harus valid
  dataPengeluaran.amount > 0 and dataPengeluaran.date <= DateTime(2025, 5, 30, 16, 34)
post:
  -- Expense is added
  Expense->exists(e: Expense | e.amount = dataPengeluaran.amount and e.date = dataPengeluaran.date and e.description = dataPengeluaran.description)
  and result = 0 -- Success

-- Invariant: konfirmasiPengeluaranTersimpan mengembalikan pesan konfirmasi
context IOwnerMgt
inv konfirmasiPengeluaranTersimpan: result = 'Pengeluaran berhasil dicatat'

-- Invariant: lihatRiwayatPengeluaran mengembalikan semua catatan pengeluaran
context IOwnerMgt
inv lihatRiwayatPengeluaran: Expense->asSet()->notEmpty()

-- Invariant: dapatkanRiwayatPengeluaran mengembalikan semua catatan pengeluaran
context IOwnerMgt
inv dapatkanRiwayatPengeluaran: Expense->asSet()->notEmpty()

-- Invariant: pilihPengeluaranUrutkanBerdasarkanTanggal mengembalikan pengeluaran yang diurutkan berdasarkan tanggal
context IOwnerMgt
inv pilihPengeluaranUrutkanBerdasarkanTanggal: Expense.allInstances()->asSequence()->sortedBy(e | e.date)->notEmpty()

-- Invariant: kembalikanDaftarPengeluaran mengembalikan semua catatan pengeluaran
context IOwnerMgt
inv kembalikanDaftarPengeluaran: Expense->asSet()->notEmpty()

-- Invariant: tampilkanRiwayatPengeluaran mengembalikan semua catatan pengeluaran
context IOwnerMgt
inv tampilkanRiwayatPengeluaran: Expense->asSet()->notEmpty()
```

Terima Kasih

Semoga Anda mempelajari sesuatu yang baru.