LAPORAN PROYEK MATA KULIAH PEMROGRAMAN DASAR

SEMESTER GASAL 2025/2026

RENTOR:
Aplikasi Manajemen Sewa Motor

Disusun oleh

21/478563/TK/52734 Farissa Fachrianur

Departemen Teknik Elektro dan Teknologi Informasi

Fakultas Teknik

Universitas Gadjah Mada

2025

1. DESKRIPSI APLIKASI

Pada tugas proyek akhir mata kuliah Pemrograman Dasar ini, dikembangkan sebuah aplikasi untuk mengelola penyewaan motor dengan menggunakan bahasa pemrograman C++ dan pendekatan pemrograman berorientasi objek atau *object-oriented programming* (OOP). Aplikasi ini dikembangkan untuk mendukung proses penyewaan motor secara online dengan menyederhanakan langkah-langkah mulai dari proses pendaftaran hingga pembayaran.

Aplikasi ini menyediakan antarmuka yang dapat secara mudah dipahami pengguna untuk berinteraksi dengan aplikasi. Sistem utamanya meliputi penyewa motor (*user*), pegawai rental motor (*employee*), dan manajer. Pengguna dapat melakukan pendaftaran untuk masuk sebagai penyewa motor, melihat motor tersedia, mencari motor berdasarkan merek, melakukan pemesanan, dan pembayaran. Selain itu, sistem ini juga menyediakan fitur untuk pegawai mengelola data kendaraan dan membuat invoice serta untuk manajer dapat melihat seluruh laporan transaksi.

2. ANALISIS KEBUTUHAN

Analisis kebutuhan merupakan proses yang melibatkan pemeriksaan mendalam terhadap aspek-aspek krusial yang mendukung proses pengembangan sistem perangkat lunak. Pada tahap ini, dilakukan pengidentifikasian kebutuhan pengguna dan sistem berdasarkan kebutuhan fungsional maupun non-fungsional.

2.1. Kebutuhan Fungsional

Bagian ini akan menjelaskan secara detail kebutuhan fungsional yang terkait dengan pelaksanaan proyek ini. Kebutuhan fungsional tersebut dijelaskan secara detail pada Tabel 1.

Tabel 1 Kebutuhan Fungsional

Req-ID	Kebutuhan Fungsional	Deskripsi
F-01	Registration	Fitur untuk pengguna membuat akun baru dan menambahkan data pengguna baru pada aplikasi.
F-02	Login	Fitur yang menmungkinkan pengguna, pegawai, dan manajer masuk ke sistem aplikasi dengan akun yang valid.
F-03	View Available Motor	Fitur untuk menampilkan daftar motor tersedia.
F-04	Search Motor by Brand	Fitur untuk menampilkan motor berdasarkan merek.
F-05	Select Motor	Fitur yang memungkinkan penyewa motor untuk memilih motor yang ingin disewa.

F-06	Booking	Fitur untuk membuat pemesanan motor.
F-06	Payment	Fitur untuk melakukan pembayaran terhadap motor yang disewa.
F-06	Manage Motor Data	Fitur yang memungkinkan pegawai untuk mengelola data motor.
F-06	Generate Invoice	Fitur yang memungkinkan pegawai dapat mencetak invoice untuk diberikan ke penyewa motor sebagai surat tagihan dan bukti pembayaran.
F-06	View Transaction Reports	Fitur yang memungkinkan manajer melihat laporan transaksi.

2.2. Kebutuhan Non Fungsional

Bagian ini akan menjelaskan secara detail kebutuhan non-fungsional yang terkait dengan pelaksanaan proyek ini. Kebutuhan non-fungsional tersebut terdiri dari penggunaan alat dan bahasa pemrograman yang digunakan pada pemodelan aplikasi manajemen sewa motor. Lalu, terdapat batasan-batasan serta pengerjaan secara individual dalam pengerjaan proyek ini.

a. Alat

1) Laptop HP 15 Intel Core i5 dengan spesifikasi sistem operasi Windows 11 sebagai perangkat keras untuk menjalankan perangkat lunak yang terlibat dalam pengerjaan pemodelan use case diagram, class diagram, pembuatan user interface, implementasi program dalam C++.
2) Aplikasi perangkat lunak Enterprise Architect Academic sebagai pendukung pemodelan *use case diagram* dan *class diagram*.
3) Ekstensi *code generation* C++ versi 0.9.4.
4) *Visual Studio Code* (VS code) g++.

b. Bahasa Pemrograman

Pada proyek ini digunakan bahasa pemrograman C++ karena mendukung metode pemrograman berorientasi objek atau object-oriented programming (OOP) yang sesuai dengan pemodelan menggunakan unified modeling language (UML) khususnya seperti use case diagram dan class diagram.

c. Batasan Pengerjaan Proyek

Beberapa batasan pada pengerjaan proyek ini terdiri dari objek pengerjaan proyek, metode pengerjaan proyek, waktu pengerjaan proyek, sumber daya manusia, dan perangkat lunak. Berikut merupakan detail dari batasan-batasan tersebut:

1) Objek Pengerjaan Proyek: Aktor penyewa motor (*motor renter*), aktor pegawai (*employee*), dan aktor manajer.
2) Metode Pengerjaan Proyek: Pembuatan use case diagram dan class diagram pada aplikasi manajemen sewa motor.
3) Waktu Pengerjaan Proyek: 24 November 2025 hingga 2 Desember 2025.
4) Sumber Daya Manusia: Pengerjaan seluruh proyek ini hanya dilakukan secara individual.
5) Perangkat Lunak: Pengerjaan proyek ini hanya dilakukan menggunakan alat-alat perangkat lunak seperti VS code dan EA academic versi gratis sehingga tidak ada keterlibatan fitur berbayar pada pengerjaan proyek ini.

d. Pengerjaan Proyek

Pengerjaan pada proyek ini dilakukan secara individual dengan detail pekerjaan sebagai berikut:

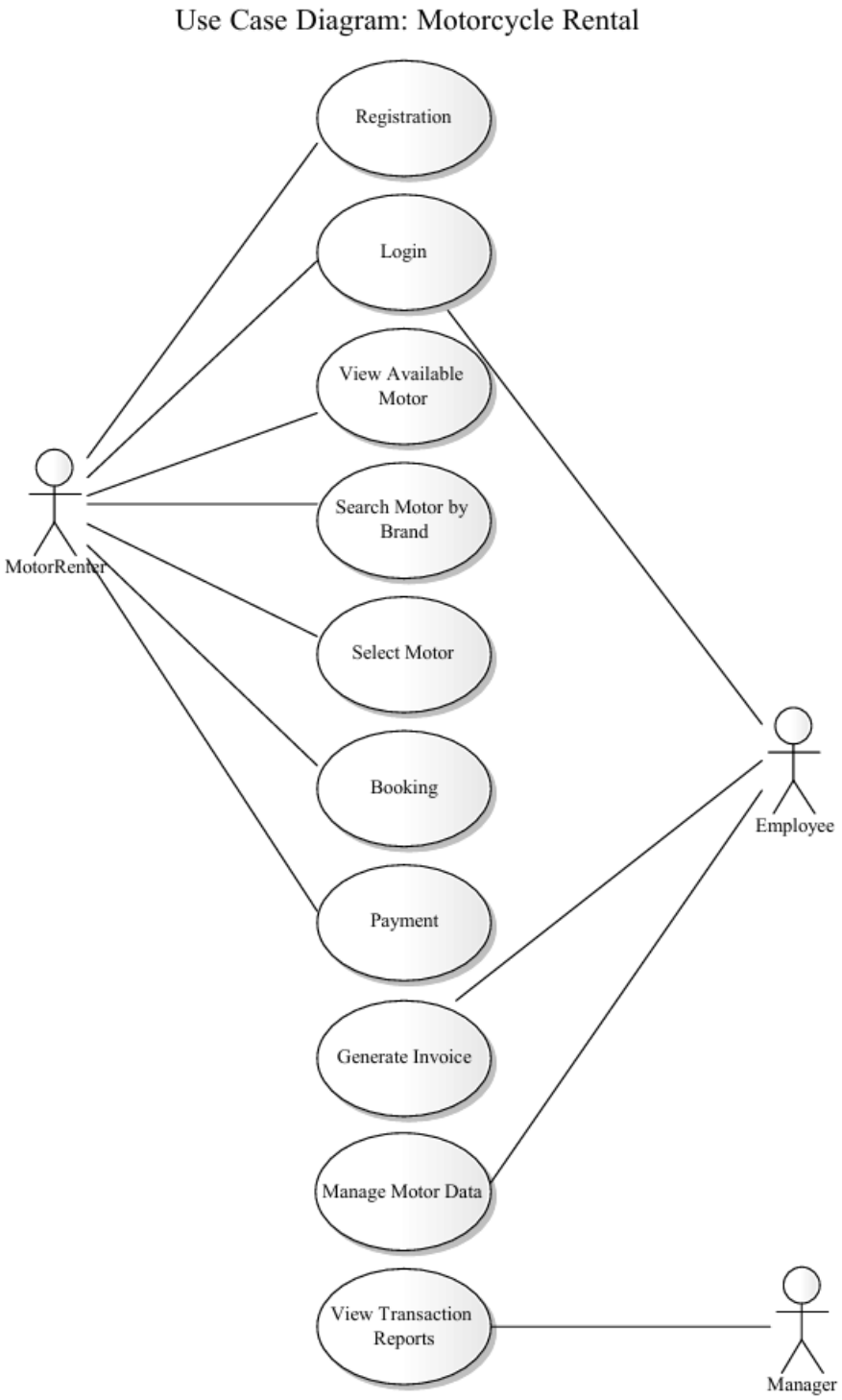
Tabel 2. Pengerjaan Proyek

No.	Jenis Pekerjaan	Nama
		Farissa Fachrianur
1	Pembuatan *use case diagram*	✓
2	Pembuatan *class diagram*	✓
3	Pembuatan *user interface*	✓
4	Implementasi Program dalam C++	✓
5	Penulisan laporan	✓

3. PEMODELAN DAN PERANCANGAN SISTEM

3.1. Use Case Diagram

Use case diagram adalah salah satu jenis diagram dalam Unified Modeling Language (UML) yang digunakan untuk menggambarkan hubungan dan interaksi antara aktor (pengguna) dengan sistem yang sedang dikembangkan seperti aplikasi manajemen sewa motor ini. Diagram-diagram ini menunjukkan layanan-layanan atau fitur yang dapat diakses oleh actor.



Gambar 1. *Use case diagram* pada aplikasi sewa motor

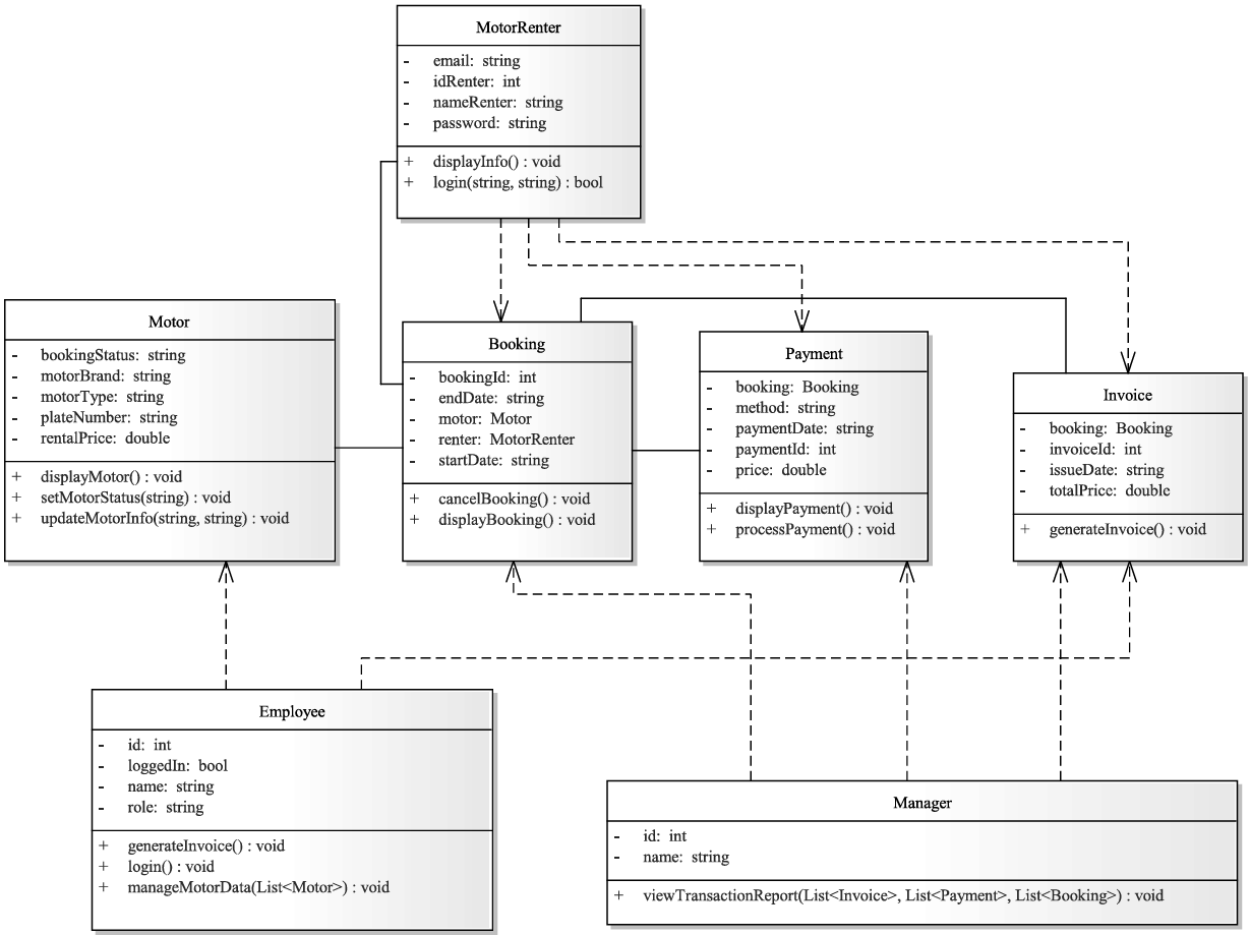
Use case diagram untuk aplikasi ini dimodelkan seperti pada Gambar 1 yang melibatkan 3 actor dan 10 use case. Aktor tersebut terdiri dari MotorRenter (penyewa motor), Employee (pegawai), dan Manager (manajer). Kemudian, 10 use case tersebut dijelaskan secara detail pada Tabel 3.

Tabel 3. Skenario *Use case diagram* untuk setiap kebutuhan fungsional

Req-ID	Kebutuhan Fungsional	Skenario use case
F-01	*Register*	1. Pengguna (penyewa motor) masuk ke menu *register* 2. Aplikasi meminta data pengguna baru 3. Pengguna baru memasukkan data pribadi 4. Sistem menyimpan data pengguna baru
F-02	*Login*	1. Pengguna masuk ke sistem aplikasi sewa motor untuk mengakses fitur-fitur lain 2. Pegawai masuk ke sistem aplikasi sewa motor untuk mengakses fitur-fitur lain
F-03	*View Available Motor*	1. Sistem akan menampilkan daftar motor beserta status dan harga sewa
F-04	*Search Motor by Brand*	1. Pengguna memasukkan nama merek dari motor yang ingin disewa 2. Sistem menampilkan daftar motor sesuai merek yang dicari oleh pengguna
F-05	*Select Motor*	1. Pengguna memilih motor dari daftar tersebut 2. Sistem akan menampilkan detail dari motor yang dipilih pengguna
F-06	*Booking*	1. Pengguna memasukkan tanggal mulai hingga selesai sewa 2. Sistem akan membuat objek *booking* untuk motor yang dipilih untuk disewa oleh pengguna 3. Motor berubah status menjadi "*booked*"
F-07	*Payment*	1. Pengguna memilih metode pembayaran yang akan dilakukan 2. Sistem akan memverifikasi data dan menyimpan hasil transaksi yang dilakukan oleh pengguna
F-08	*Generate Invoice*	1. Pegawai memilih *booking* yang sudah dibayarkan oleh pengguna untuk menyewa motor 2. Sistem akan mengeluarkan invoice digital sebagai bukti transaksi
F-09	*Manage Motor Data*	1. Pegawai memilih menu manajemen data motor 2. Pegawai menambah, menghapus, dan mengubah data motor 3. Pegawai memperbarui informasi detail motor 4. Sistem menyimpan pembaruan informasi motor pada basis data
F-10	*View Transaction Reports*	1. Manajer memilih menu laporan transaksi 2. Sistem menampilkan daftar transaksi 3. Manajer melihat detail transaksi berisi pembayaran dan invoice

3.2. Class Diagram

Class diagram merupakan diagram unified modeling language (UML) untuk memodelkan sistem perangkat lunak dalam konteks atribut dan operasi pada setiap kelasnya, serta hubungan antar-kelas yang terjadi di dalam sistem tersebut. Pada proyek ini, class diagram dimodelkan seperti pada Gambar 2 yang terdiri dari tujuh kelas: Motor, User, Employee, Manager, Booking, Payment, dan Invoice serta relasi *dependency* dan *association* di antara objek-objek tersebut.



Gambar 2. *Class diagram* pada aplikasi manajemen sewa motor

a. Kelas Motor

Kelas Motor merepresentasikan objek kendaraan roda dua yang disewakan kepada pengguna. Kelas ini menyimpan atribut penting seperti motorBrand, motorType, plateNumber, rentalPrice, dan bookingStatus. Atribut-atribut tersebut digunakan untuk menyimpan identitas motor, status ketersediaan, serta harga sewanya.

Kelas ini menyediakan beberapa operasi, antara lain displayMotor() untuk menampilkan detail motor, setMotorStatus() untuk mengubah status motor (misalnya "available" atau "booked"), serta updateMotorInfo() untuk memperbarui informasi merek maupun tipe motor. Dengan demikian, kelas Motor menjadi pusat informasi objek rental dalam sistem.

b. Kelas MotorRenter

Kelas MotorRenter merepresentasikan penyewa motor. Atribut yang disimpan mencakup idRenter, nameRenter, email, dan password. Seluruhnya bersifat privat untuk menjaga kerahasiaan data pribadi. Kelas ini menyediakan beberapa operasi seperti login() untuk proses autentikasi berdasarkan email dan password, serta displayInfo() yang menampilkan identitas penyewa. Objek MotorRenter akan terlibat dalam proses pemesanan, pembayaran, hingga penerimaan invoice.

c. Kelas Booking

Kelas Booking bertindak sebagai representasi transaksi pemesanan motor. Atribut yang disimpan meliputi bookingId, motor (objek Motor yang disewa), renter (objek MotorRenter), startDate, dan endDate. Atribut tersebut menggabungkan data motor dan penyewa dalam satu unit transaksi. Kelas Booking memiliki operasi seperti displayBooking() untuk menampilkan detail pemesanan dan cancelBooking() untuk melakukan pembatalan pemesanan sekaligus mengembalikan status motor menjadi tersedia. Dengan demikian, kelas ini menjadi penghubung utama antara penyewa dan unit motor yang disewa.

d. Kelas Payment

Kelas Payment menyimpan informasi pembayaran dari sebuah transaksi pemesanan. Atribut yang dimiliki antara lain paymentId, booking (yang dibayar), method, paymentDate, dan price. Struktur ini memungkinkan sistem mencatat metode pembayaran, nominal transaksi, serta kaitannya terhadap pemesanan tertentu. Operasi yang tersedia meliputi processPayment() untuk menampilkan proses pembayaran secara terinci serta displayPayment() untuk merangkum hasil pembayaran. Dalam konteks sistem, kelas Payment menjamin bahwa setiap booking memiliki jejak pembayaran yang terdokumentasi.

e. Kelas Invoice

Kelas Invoice merepresentasikan bukti tagihan resmi yang dikeluarkan setelah pembayaran berhasil dilakukan. Atributnya meliputi invoiceId, booking, issueDate, dan totalPrice. Kelas ini berfungsi sebagai dokumen digital yang menyatukan data penyewa, motor, periode sewa, serta total biaya. Operasi utama kelas ini adalah generateInvoice() yang menghasilkan tampilan invoice secara lengkap dan terstruktur. Dengan adanya kelas ini, bukti transaksi dapat dicetak atau disimpan sebagai arsip.

f. Kelas Employee

Kelas Employee mewakili pegawai yang mengelola operasional sistem, termasuk pengelolaan data motor dan pembuatan invoice. Atribut utama yang disimpan mencakup id, name, role, dan status loggedIn. Kelas ini menyediakan operasi administratif seperti login(), manageMotorData() untuk menambah, menghapus, atau melihat data motor, serta generateInvoice() untuk menginisiasi pembuatan invoice berdasarkan transaksi. Selain itu, Employee dapat menyampaikan klaim asuransi jika fitur tersebut diaktifkan dalam sistem.

g. Kelas Manager

Kelas Manager merepresentasikan aktor dengan otoritas tertinggi dalam aplikasi. Atribut kelas ini sederhana, meliputi id dan name. Operasi utamanya adalah viewTransactionReport() yang menampilkan laporan lengkap mengenai seluruh Booking, Payment, dan Invoice. Melalui kelas ini, manajer dapat melakukan monitoring terhadap aktivitas penyewaan dalam rentang waktu tertentu.

h. Hubungan Antar-Kelas

Terdapat beberapa hubungan antar-kelas dalam *class diagram* yang menggambarkan bahwa setiap antar-kelas memiliki hubungan dengan arti tertentu. Umumnya, koneksi dibangun dengan *dependency* yang berarti satu kelas hubungan yang bergantung secara sementara pada kelas lain. Selain itu, terdapat hubungan antar-kelas yang dibangun dengan koneksi *association* dan *directed association* yang berarti hubungan bersifat satu arah. Hal ini berarti kelas asal mengenal langsung pada kelas yang dituju, tetapi tidak berlaku untuk sebaliknya, seperti yang dijelaskan pada Tabel 4.

Tabel 4. Hubungan antar-kelas pada *class diagram*

Hubungan Antar-Kelas	Koneksi	Penjelasan
MotorRenter - Booking	*Dependency*	MotorRenter menggunakan Booking ketika melakukan pemesanan. Penyewa memicu pembuatan objek Booking, sehingga

		terdapat hubungan ketergantungan sementara.
MotorRenter - Payment	*Dependency*	Penyewa memproses pembayaran melalui kelas Payment. Hal ini menunjukkan bahwa objek Payment bergantung pada aksi pengguna.
MotorRenter - Invoice	*Dependency*	Penyewa menjadi pihak yang menerima invoice yang dihasilkan, sehingga ada ketergantungan informasi antara kedua kelas.
Booking - Motor	*Association*	Kelas Booking memiliki objek Motor, yang menunjukkan hubungan "pemesanan terhadap motor tertentu".
Booking - MotorRenter	*Association*	Kelas Booking menyimpan objek MotorRenter, menunjukkan siapa penyewa motor tersebut.
Payment - Booking	*Directed Association*	Setiap pembayaran selalu terkait dengan satu Booking. Payment mengenal Booking, tetapi Booking tidak mengenal Payment.
Invoice - Booking	*Directed Association*	Invoice merujuk pada satu Booking dan mengambil data dari kelas tersebut untuk membentuk detail invoice.
Employee - Motor	*Dependency*	Employee mengelola data motor, sehingga tindakan manajemen bergantung pada objek Motor.
Employee - Invoice	*Dependency*	Employee memiliki kewenangan membuat invoice, sehingga terdapat hubungan ketergantungan.
Manager - Booking, Payment, Invoice	*Dependency*	Manager meninjau laporan transaksi, sehingga bergantung pada ketiga kelas tersebut untuk menampilkan laporan menyeluruh.

3.3. Perancangan User Interface

Pengembangan sistem aplikasi manajemen sewa motor ini dilakukan dengan menggunakan *Console-Based User Interface*. *User interface* berbasis *console* ini dipilih karena ringan dan mudah diimplementasikan pada *command prompt* tanpa memerlukan sumber daya yang kompleks. Pada proyek ini, *user interface* yang dikembangkan terdiri dari beberapa menu utama sebagai berikut:

a. *Main Menu*

Terdiri dari tiga menu utama yaitu: Registrasi, Login, dan Exit.

```
cout << "\n=== APLIKASI SEWA MOTOR ===";
cout << "\n*** Selamat Datang! ***";
cout << "\nSilakan Pilih Menu di bawah ini:\n";
cout << "1. Registrasi\n";
cout << "2. Login\n";
cout << "0. Exit\n";
cout << "Choice: ";
cin >> menu; cin.ignore();
```

b. Registrasi

User interface terdiri dari empat form input sederhana di *console.* Pengguna akan diminta untuk memasukkan data pribadi berupa ID (nomor identitas unik pengguna), Name (nama lengkap), Email, dan Password (kata sandi untuk keamanan akun). Data-data ini diperlukan untuk membuat akun penyewa motor baru.

```
if (menu == 1) {
    int id;
    string name, email, pass;
    cout << "ID: "; cin >> id; cin.ignore();
    cout << "Name: "; getline(cin, name);
    cout << "Email: "; getline(cin, email);
    cout << "Password: "; getline(cin, pass);

    renters.push_back(MotorRenter(id, name, email, pass));
    cout << "Registration successful.\n";
```

c. *Login*

Pada menu Login ini, pengguna dapat memilih peran yang akan digunakan untuk masuk ke dalam aplikasi. Terdapat tiga pilihan utama, yaitu Renter (penyewa motor), Employee (pegawai), dan Manager (manajer).

```
cout << "\nLogin as:\n1. Renter\n2. Employee\n3. Manager\nChoice: ";
int type; cin >> type; cin.ignore();

/* RENTER LOGIN */
if (type == 1) {
    string e, p;
    cout << "Email: "; getline(cin, e);
    cout << "Password: "; getline(cin, p);

    bool found = false;
    for (auto &r : renters) {
        if (r.login(e, p)) {
            loggedRenter = &r;
            found = true;
            cout << "Login successful.\n";
            break;
```

d. User Menu

Setelah login sebagai penyewa, pengguna dapat melihat daftar motor yang tersedia, mencari motor berdasarkan merek, dan melakukan pemesanan.

```
/* USER MENU */
int opt;
do {
    cout << "\n=== Renter Menu ===\n";
    cout << "1. View Available Motors\n";
    cout << "2. Search Motor by Brand\n";
    cout << "0. Logout\n";
    cout << "Choice: ";
    cin >> opt; cin.ignore();
```

e. Employee Menu

Setelah login sebagai *employee*, pegawai dapat mengelola data motor yang tersedia dan membuat invoice.

```
cout << "\n=== Employee Menu ===\n";
cout << "1. Manage Motor Data\n";
cout << "2. Generate Invoice \n";
cout << "0. Logout\nChoice: ";
cin >> opt; cin.ignore();
```

f. Manager Menu

Digunakan untuk manajer melihat laporan transaksi yang sudah dilakukan dalam aplikasi.

```
for (auto &m : managers) {
    if (m.getId()==id && m.getName()==name) {
        m.viewTransactionReport(invoices, payments, bookings);
        ok = true;
        break;
```

3.4. Transformasi Class Diagram Menjadi Kode Program

MotorRenter.cpp

```
///////////////////////////////////////////////////////////
//  MotorRenter.cpp
//  Implementation of the Class MotorRenter
//  Created on:      02-Dec-2025 12:13:58
//  Original author: HP
///////////////////////////////////////////////////////////

#include "MotorRenter.h"

MotorRenter::MotorRenter(){

}
MotorRenter::~MotorRenter(){

}

void MotorRenter::displayInfo(){

}

bool MotorRenter::login(string password, string email){

       return false;
}
```

MotorRenter.h

```
///////////////////////////////////////////////////////////
//  MotorRenter.h
//  Implementation of the Class MotorRenter
//  Created on:      02-Dec-2025 12:13:58
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_)
#define EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_

class MotorRenter
{

public:
	MotorRenter();
	virtual ~MotorRenter();

	void displayInfo();
	bool login(string password, string email);

private:
	string email;
	int idRenter;
	string nameRenter;
	string password;

};
#endif // !defined(EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_)
```

Motor.cpp

```
///////////////////////////////////////////////////////////
//  Motor.cpp
//  Implementation of the Class Motor
//  Created on:      02-Dec-2025 12:14:00
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Motor.h"

Motor::Motor(){
}

Motor::~Motor(){
}

void Motor::displayMotor(){
}

void Motor::setMotorStatus(string status){
}

void Motor::updateMotorInfo(string newType, string newBrand){
}
```

Motor.h

```
///////////////////////////////////////////////////////////
//  Motor.h
//  Implementation of the Class Motor
//  Created on:      02-Dec-2025 12:14:00
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_)
#define EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_

class Motor
{
public:
	Motor();
	virtual ~Motor();

	void displayMotor();
	void setMotorStatus(string status);
	void updateMotorInfo(string newType, string newBrand);

private:
	/**
	 * Possible values: "available", "booked"
	 */
	string bookingStatus;
	string motorBrand;
	string motorType;
	string plateNumber;
	double rentalPrice;

};
#endif // !defined(EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_)
```

Booking.cpp

```
///////////////////////////////////////////////////////////
//  Booking.cpp
//  Implementation of the Class Booking
//  Created on:      02-Dec-2025 12:14:01
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Booking.h"

Booking::Booking(){
}

Booking::~Booking(){
}

void Booking::cancelBooking(){
}

void Booking::displayBooking(){
}}
```

Booking.h

```
///////////////////////////////////////////////////////////
//  Booking.h
//  Implementation of the Class Booking
//  Created on:      02-Dec-2025 12:14:01
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_)
#define EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_

#include "Motor.h"
#include "MotorRenter.h"

class Booking
{
public:
	Booking();
	virtual ~Booking();
	Motor *m_Motor;
	MotorRenter *m_MotorRenter;

	void cancelBooking();
	void displayBooking();

private:
	int bookingId;
	string endDate;
	Motor motor;
	MotorRenter renter;
	string startDate;

};
#endif // !defined(EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_)
```

Payment.cpp

```
///////////////////////////////////////////////////////////
//  Payment.cpp
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Payment.h"

Payment::Payment(){
}

Payment::~Payment(){
}

void Payment::displayPayment(){
}
void Payment::processPayment(){
}
```

Payment.h

```
///////////////////////////////////////////////////////////
//  Payment.h
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
#define EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_

#include "Booking.h"

class Payment
{

public:
	Payment();
	virtual ~Payment();
	Booking *m_Booking;

	void displayPayment();
	void processPayment();

private:
	Booking booking;
	string method;
	string paymentDate;
	int paymentId;
	double price;

};
#endif // !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
```

Invoice.cpp

```
///////////////////////////////////////////////////////////
//  Invoice.cpp
//  Implementation of the Class Invoice
//  Created on:      02-Dec-2025 12:14:03
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Invoice.h"

Invoice::Invoice(){
}

Invoice::~Invoice(){
}

void Invoice::generateInvoice(){
}
```

Invoice.h

```
///////////////////////////////////////////////////////////
//  Invoice.h
//  Implementation of the Class Invoice
//  Created on:      02-Dec-2025 12:14:03
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_)
#define EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_

#include "Booking.h"

class Invoice
{

public:
	Invoice();
	virtual ~Invoice();
	Booking *m_Booking;

	void generateInvoice();

private:
	Booking booking;
	int invoiceId;
	string issueDate;
	double totalPrice;

};
#endif // !defined(EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_)
```

Employee.cpp

```
///////////////////////////////////////////////////////////
//  Employee.cpp
//  Implementation of the Class Employee
//  Created on:      02-Dec-2025 12:14:04
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Employee.h"

Employee::Employee(){
}
Employee::~Employee(){
}
void Employee::generateInvoice(){
}
void Employee::login(){
}
void Employee::manageMotorData(List<Motor> motors){
}
```

Employee.h

```
///////////////////////////////////////////////////////////
//  Payment.h
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
#define EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_

#include "Booking.h"

class Payment
{

public:
	Payment();
	virtual ~Payment();
	Booking *m_Booking;

	void displayPayment();
	void processPayment();

private:
	Booking booking;
	string method;
	string paymentDate;
	int paymentId;
	double price;

};
#endif // !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
```

Manager.cpp

```
///////////////////////////////////////////////////////////
//  Manager.cpp
//  Implementation of the Class Manager
//  Created on:      02-Dec-2025 12:14:05
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Manager.h"

Manager::Manager(){
}

Manager::~Manager(){
}

void Manager::viewTransactionReport(List<Invoice> invoices,
List<Payment> payments, List<Booking> bookings){

}
```

Manager.h

```
///////////////////////////////////////////////////////////
//  Manager.h
//  Implementation of the Class Manager
//  Created on:      02-Dec-2025 12:14:05
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_)
#define EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_

class Manager
{

public:
	Manager();
	virtual ~Manager();

	void viewTransactionReport(List<Invoice> invoices, List<Payment>
payments, List<Booking> bookings);

private:
	int id;
	string name;

};
#endif // !defined(EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_)
```

4. IMPLEMENTASI APLIKASI

4.1. Fitur F-01 Register

Sistem akan menampilkan menu utama dari aplikasi manajemen sewa motor. Pengguna baru akan memilih menu 1 untuk Registrasi terlebih dahulu karena sistem belum memiliki data pengguna tersebut sebagai penyewa motor dalam aplikasi. Saat pengguna memilih menu registrasi, aplikasi meminta beberapa data identitas berupa ID penyewa, nama lengkap, email, dan kata sandi.

```
=== APLIKASI SEWA MOTOR ===
*** Selamat Datang! ***
Silakan Pilih Menu di bawah ini:
1. Registrasi
2. Login
0. Exit
Choice: 1
```

Pengguna memasukkan data tersebut melalui antarmuka berbasis teks. Setelah seluruh data diisi, sistem membuat sebuah objek penyewa baru dan menyimpannya ke dalam daftar penyewa (vector renters).

```
ID: 882
Name: Farissa Fachrianur
Email: farissafachrianur@gmail.com
Password: 123farissa
Registration successful.
```

Sistem kemudian menampilkan pesan bahwa proses registrasi telah berhasil. Pada tahap ini, data penyewa belum digunakan untuk proses lainnya sampai pengguna melakukan login. Fitur ini berfungsi sebagai pintu masuk awal sebelum penyewa dapat mengakses fasilitas pemesanan motor.

4.2. Fitur F-02 Login

Setelah registrasi, pengguna dapat memilih menu Login dan menentukan jenis akun yang digunakan: Penyewa, Pegawai, atau Manajer. Pada fitur ini dilakukan autentikasi untuk tiga jenis akun tersebut untuk memastikan hanya pengguna tervalidasi yang dapat mengakses fungsi tertentu sesuai perannya.

a. Login Penyewa

```
Login as:
1. Renter
2. Employee
3. Manager
Choice: 1
Email: farissafachrianur@gmail.com
Password: 123farissa
Login successful.
```

Sistem meminta email dan kata sandi. Jika keduanya sesuai dengan salah satu akun yang telah terdaftar, penyewa berhasil masuk dan diarahkan ke menu utama penyewa. Jika tidak sesuai, sistem menolak proses login dan kembali ke menu awal.

b. Login Pegawai

```
Login as:
1. Renter
2. Employee
3. Manager
Choice: 2
Employee ID: 102
Name: Sari
Employee Sari logged in.
```

Pegawai memasukkan ID dan nama. Sistem mencocokkan keduanya dengan daftar pegawai yang telah tersedia. Setelah berhasil, pegawai diarahkan ke menu pengelolaan data motor.

c. Login Manajer

```
Login as:
1. Renter
2. Employee
3. Manager
Choice: 3
Manager ID: 9001
Name: Agus

=== Transaction Report ===

-- Bookings --

-- Payments --

-- Invoices --
```

Manajer juga masuk menggunakan ID dan nama. Jika valid, sistem langsung menampilkan laporan transaksi.

4.3. Fitur F-03 Melihat Daftar Motor yang Tersedia

Setelah login sebagai penyewa berhasil dilakukan, pengguna dapat memilih menu *View Available Motors*. Sistem menampilkan seluruh motor yang berstatus "*available*". Informasi yang ditampilkan meliputi: merek motor, tipe motor, nomor plat, harga sewa per hari, dan status motor.

```
=== Renter Menu ===
1. View Available Motors
2. Search Motor by Brand
0. Logout
Choice: 1
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: available
Yamaha Mio | Plate: AB5678CD | Price: Rp55000 | Status: available
Suzuki Nex | Plate: AB4321EF | Price: Rp52000 | Status: available
```

Daftar motor yang ditampilkan bersifat dinamis dan hanya menunjukkan motor yang belum pesan oleh penyewa lain. Fitur ini membantu penyewa menilai pilihan motor yang dapat disewa sebelum memasuki tahap pencarian lanjutan maupun proses booking.

4.4. Fitur F-04 Pencarian Motor Berdasarkan Merek

Penyewa dapat melakukan pencarian berdasarkan merek motor, misalnya “Yamaha”. Sistem meminta input berupa nama merek dan kemudian menampilkan semua motor dengan merek tersebut yang masih berstatus available. fitur pencarian brand memanfaatkan pemeriksaan atribut motorBrand melalui perbandingan sederhana if (m.getBrand() == brand).

```
=== Renter Menu ===
1. View Available Motors
2. Search Motor by Brand
0. Logout
Choice: 2
Enter brand: Yamaha
Yamaha Mio | Plate: AB5678CD | Price: Rp55000 | Status: available
Book one? (y/n):
```

Motor yang memenuhi kriteria dan masih berstatus “available” ditampilkan pada layar. Implementasi ini memberikan kemudahan bagi penyewa untuk menemukan motor dengan merek tertentu tanpa harus melihat seluruh daftar.

4.5. Fitur F-05 Pemilihan Motor

Setelah daftar motor muncul (baik dari fitur lihat semua maupun pencarian), penyewa diberi kesempatan untuk memilih motor tertentu agar dapat melanjutkan proses booking. Implementasi dilakukan dengan menyimpan daftar motor yang ditampilkan ke dalam vector pointer sehingga pemilihan dapat dilakukan berdasarkan indeks pilihan pengguna. Fitur ini menjadi penghubung langsung antara menu pencarian dan pembuatan objek booking. Sistem akan menanyakan apakah pengguna ingin melakukan booking.

```
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: available
Yamaha Mio | Plate: AB5678CD | Price: Rp55000 | Status: available
Suzuki Nex | Plate: AB4321EF | Price: Rp52000 | Status: available
Book one? (y/n): y
Enter number (1-3): 1
```

Jika pengguna memilih “yes”, sistem meminta pengguna memilih salah satu motor berdasarkan nomor urut daftar. Motor yang dipilih kemudian dianggap sebagai motor yang akan disewa.

4.6. Fitur F-06 Booking Motor

Setelah motor dipilih, proses *booking* dilakukan dengan pembuatan objek *booking* yang mencatat penyewa, motor, tanggal mulai dan selesai sewa, serta ID booking otomatis. Motor yang dipilih akan diubah statusnya menjadi "booked". Sistem akan menambahkan booking tersebut ke daftar riwayat pemesanan.

```
Book one? (y/n): y
Enter number (1-3): 1
Start date (yyyy-mm-dd): 2025-11-10
End date (yyyy-mm-dd): 2025-11-15
```

Objek booking kemudian ditambahkan ke vector bookings. Fitur ini mengatur reservasi motor dan memastikan bahwa motor yang sudah dibooking tidak kembali muncul dalam daftar motor tersedia.

4.7. Fitur F-07 Pembayaran

Setelah booking dibuat, penyewa dapat melanjutkan ke proses pembayaran. Implementasi pembayaran menggunakan objek Payment, di mana sistem mencatat:

- Metode pembayaran yang dipilih (misalnya Cash atau Transfer)
- Sistem meminta tanggal pembayaran
- Sistem menghitung total pembayaran berdasarkan harga sewa per hari (pada program ini, harga belum dikalikan durasi, sehingga harga motor digunakan langsung)
- Sistem menampilkan payment receipt yang berisi: ID pembayaran, metode pembayaran, detail booking, total biaya, dan tanggal pembayaran

Hasil pembayaran disimpan sebagai objek Payment dan ditambahkan ke dalam daftar transaksi pembayaran.

```
Continue to payment? (y/n): y
Payment method (Cash / Transfer): Transfer
Payment date: 2025-11-10
--- Payment Receipt ---
Payment ID: 201
Method: Transfer
Booking ID: 101
Renter: Farissa Fachrianur | Email: farissafachrianur@gmail.com
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: booked
From: 2025-11-10 To: 2025-11-15
Total: Rp50000
Payment Date: 2025-11-10
Invoice date: 2025-11-10
```

4.8. Fitur F-08 Generate Invoice

Setelah pembayaran berhasil, sistem akan meminta tanggal penerbitan invoice. Sistem kemudian menghasilkan dokumen invoice dalam bentuk tampilan teks pada layar, berisi:

- ID invoice
- rincian booking
- tanggal penerbitan
- total harga

```
--- Invoice ---
Invoice ID: 301
Booking ID: 101
Renter: Farissa Fachrianur | Email: farissafachrianur@gmail.com
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: booked
From: 2025-11-10 To: 2025-11-15
Issue Date: 2025-11-10
Total Price: Rp50000
```

Invoice ini secara otomatis disimpan ke dalam daftar riwayat invoice menggunakan *vector* invoices untuk keperluan pelaporan oleh manajer. Fitur ini memberikan keluaran berupa bukti sewa resmi yang dapat menjadi arsip transaksi.

4.9. Fitur F-09 Manajemen Data Motor

Fitur ini hanya dapat diakses oleh pegawai setelah log in. Melalui menu khusus, pegawai dapat:

- **Menambahkan motor baru**

```
=== Motor Management ===
1. Add Motor
2. Delete Motor
3. View All Motors
0. Back
Choice: 1
Brand: Honda
Type: ZY
Plate: 12345
Price: 56000
Motor added successfully.
```

Sistem meminta input merek, tipe, nomor plat, dan harga sewa. Motor baru kemudian ditambahkan ke daftar motor.

- **Menghapus motor**

```
Choice: 2
1. Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: available
2. Yamaha Mio | Plate: AB5678CD | Price: Rp55000 | Status: available
3. Suzuki Nex | Plate: AB4321EF | Price: Rp52000 | Status: available
4. Honda ZY | Plate: 12345 | Price: Rp56000 | Status: available
Delete number: 4
Deleted.
```

Sistem menampilkan seluruh motor berikut nomor urutnya. Pegawai dapat memilih motor yang ingin dihapus.

- **Melihat seluruh data motor**

```
Choice: 3
1. Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: available
2. Yamaha Mio | Plate: AB5678CD | Price: Rp55000 | Status: available
3. Suzuki Nex | Plate: AB4321EF | Price: Rp52000 | Status: available
```

Sistem menampilkan seluruh motor tanpa memandang statusnya.

Implementasi fitur ini memberikan pegawai kontrol penuh atas ketersediaan data kendaraan yang menjadi inti operasional aplikasi.

4.10. Fitur F-10 Melihat Laporan Transaksi

Fitur ini hanya dapat diakses oleh manajer. Program menggabungkan seluruh data booking, pembayaran, dan invoice, kemudian menampilkannya melalui m.viewTransactionReport(invoices, payments, bookings);

Setelah login berhasil, sistem langsung menampilkan tiga output laporan meliputi:

- Daftar Booking: Menampilkan seluruh booking yang pernah dilakukan.

```
Login as:
1. Renter
2. Employee
3. Manager
Choice: 3
Manager ID: 9001
Name: Agus

=== Transaction Report ===

-- Bookings --
Booking ID: 101
Renter: Farissa Fachrianur | Email: farissafachrianur@gmail.com
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: booked
From: 2025-11-10 To: 2025-11-15
```

- Daftar Pembayaran: Menampilkan seluruh pembayaran yang terekam.

```
-- Payments --
Payment ID: 201 | Amount: Rp50000 | Method: Transfer | Date: 2025-11-10
```

- Daftar Invoice: Menampilkan semua invoice yang telah diterbitkan.

```
-- Invoices --

--- Invoice ---
Invoice ID: 301
Booking ID: 101
Renter: Farissa Fachrianur | Email: farissafachrianur@gmail.com
Honda Beat | Plate: AB1234XY | Price: Rp50000 | Status: booked
From: 2025-11-10 To: 2025-11-15
Issue Date: 2025-11-10
Total Price: Rp50000
```

Laporan ditampilkan secara menyeluruh tanpa filter, sehingga manajer dapat melakukan evaluasi terhadap aktivitas sewa motor dalam periode tertentu.

LAMPIRAN

- Source code

Main.cpp

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

/* ============================================================
   CLASS MotorRenter
   ============================================================ */
class MotorRenter {
private:
    int idRenter;
    string nameRenter;
    string email;
    string password;

public:
    MotorRenter(int id, string name, string email, string password)
        : idRenter(id), nameRenter(name), email(email),
password(password) {}

    bool login(string e, string p) {
        return (email == e && password == p);
    }

    void displayInfo() const {
        cout << "Renter: " << nameRenter << " | Email: " << email <<
endl;
    }

    int getId() const { return idRenter; }
    string getName() const { return nameRenter; }
};

/* ============================================================
   CLASS Motor
   ============================================================ */
class Motor {
private:
    string motorBrand;
    string motorType;
    string plateNumber;
    double rentalPrice;
    string bookingStatus;
```

```
public:
    Motor(string brand, string type, string plate, double price)
        : motorBrand(brand), motorType(type), plateNumber(plate),
          rentalPrice(price), bookingStatus("available") {}

    void displayMotor() const {
        cout << motorBrand << " " << motorType
             << " | Plate: " << plateNumber
             << " | Price: Rp" << rentalPrice
             << " | Status: " << bookingStatus << endl;
    }

    void setMotorStatus(string status) { bookingStatus = status; }

    void updateMotorInfo(string brand, string type) {
        motorBrand = brand;
        motorType = type;
    }

    string getBrand() const { return motorBrand; }
    string getType() const { return motorType; }
    string getPlate() const { return plateNumber; }
    double getPrice() const { return rentalPrice; }
    string getStatus() const { return bookingStatus; }
};

/* ============================================================
   CLASS Booking
   ============================================================ */
class Booking {
private:
    int bookingId;
    string startDate;
    string endDate;
    Motor motor;
    MotorRenter renter;

public:
    Booking(int id, Motor m, MotorRenter r, string start, string end)
        : bookingId(id), motor(m), renter(r), startDate(start),
endDate(end) {}

    void displayBooking() const {
        cout << "Booking ID: " << bookingId << endl;
        renter.displayInfo();
        motor.displayMotor();
        cout << "From: " << startDate << " To: " << endDate << endl;
    }
```

```
    void cancelBooking() {
        cout << "Booking canceled.\n";
    }

    int getId() const { return bookingId; }
    Motor getMotor() const { return motor; }
    MotorRenter getRenter() const { return renter; }
};

/* ============================================================
   CLASS Payment
   ============================================================ */
class Payment {
private:
    Booking booking;
    string method;
    string paymentDate;
    int paymentId;
    double price;

public:
    Payment(int id, Booking b, double p, string date, string method)
        : paymentId(id), booking(b), price(p),
          paymentDate(date), method(method) {}

    void processPayment() const {
        cout << "--- Payment Receipt ---\n";
        cout << "Payment ID: " << paymentId << endl;
        cout << "Method: " << method << endl;
        booking.displayBooking();
        cout << "Total: Rp" << price << endl;
        cout << "Payment Date: " << paymentDate << endl;
    }

    void displayPayment() const {
        cout << "Payment ID: " << paymentId
             << " | Amount: Rp" << price
             << " | Method: " << method
             << " | Date: " << paymentDate << endl;
    }
};

/* ============================================================
   CLASS Invoice
   ============================================================ */
class Invoice {
private:
```

```
    Booking booking;
    int invoiceId;
    string issueDate;
    double totalPrice;

public:
    Invoice(int id, Booking b, string date, double price)
        : invoiceId(id), booking(b), issueDate(date),
totalPrice(price) {}

    void generateInvoice() const {
        cout << "\n--- Invoice ---\n";
        cout << "Invoice ID: " << invoiceId << endl;
        booking.displayBooking();
        cout << "Issue Date: " << issueDate << endl;
        cout << "Total Price: Rp" << totalPrice << endl;
        cout << "----------------\n";
    }
};

/* ============================================================
   CLASS Employee
   ============================================================ */
class Employee {
private:
    int id;
    string name;
    string role;
    bool loggedIn;

public:
    Employee(int id, string name, string role)
        : id(id), name(name), role(role), loggedIn(false) {}

    int getId() const { return id; }
    string getName()  const { return name; }

    void login() {
        loggedIn = true;
        cout << "Employee " << name << " logged in.\n";
    }

    void manageMotorData(vector<Motor>& motors) {
        int choice;
        do {
            cout << "\n=== Motor Management ===\n";
            cout << "1. Add Motor\n";
            cout << "2. Delete Motor\n";
```

```
                cout << "3. View All Motors\n";
                cout << "0. Back\n";
                cout << "Choice: ";
                cin >> choice;
                cin.ignore();

                if (choice == 1) {
                    string brand, type, plate;
                    double price;

                    cout << "Brand: "; getline(cin, brand);
                    cout << "Type: "; getline(cin, type);
                    cout << "Plate: "; getline(cin, plate);
                    cout << "Price: "; cin >> price; cin.ignore();

                    motors.push_back(Motor(brand, type, plate, price));
                    cout << "Motor added successfully.\n";
                }
                else if (choice == 2) {
                    for (size_t i=0; i<motors.size(); i++){
                        cout << i+1 << ". "; motors[i].displayMotor();
                    }
                    int x;
                    cout << "Delete number: ";
                    cin >> x; cin.ignore();
                    if (x >= 1 && x <= motors.size()) {
                        motors.erase(motors.begin() + (x-1));
                        cout << "Deleted.\n";
                    }
                }
                else if (choice == 3) {
                    for (size_t i=0; i<motors.size(); i++){
                        cout << i+1 << ". ";
                        motors[i].displayMotor();
                    }
                }
            } while (choice != 0);
        }

        void generateInvoice() {
            if (!loggedIn) {
                cout << "Please login first.\n";
                return;
            }
            cout << "Generating invoice...\n";
        }
    };
```

```
/* ============================================================
   CLASS Manager
   ============================================================ */
class Manager {
private:
    int id;
    string name;

public:
    Manager(int id, string name)
        : id(id), name(name) {}

    void viewTransactionReport(
        const vector<Invoice>& invoices,
        const vector<Payment>& payments,
        const vector<Booking>& bookings
    ) const {
        cout << "\n=== Transaction Report ===\n";

        cout << "\n-- Bookings --\n";
        for (auto &b : bookings) b.displayBooking();

        cout << "\n-- Payments --\n";
        for (auto &p : payments) p.displayPayment();

        cout << "\n-- Invoices --\n";
        for (auto &i : invoices) i.generateInvoice();
    }

    int getId() const { return id; }
    string getName() const { return name; }
};

/* ============================================================
   MAIN PROGRAM
   ============================================================ */
int main() {
    vector<Motor> motors = {
        Motor("Honda", "Beat", "AB1234XY", 50000),
        Motor("Yamaha", "Mio", "AB5678CD", 55000),
        Motor("Suzuki", "Nex", "AB4321EF", 52000)
    };

    vector<MotorRenter> renters;
    vector<Employee> employees = {
        Employee(101, "Budi", "Admin"),
        Employee(102, "Sari", "Staff")
    };
```

```
        vector<Manager> managers = {
            Manager(9001, "Agus"),
            Manager(9002, "Ratna")
        };

        vector<Booking> bookings;
        vector<Payment> payments;
        vector<Invoice> invoices;

        MotorRenter* loggedRenter = nullptr;
        Employee* loggedEmployee = nullptr;

        int menu;

        do {
            cout << "\n=== APLIKASI SEWA MOTOR ===";
            cout << "\n*** Selamat Datang! ***";
            cout << "\nSilakan Pilih Menu di bawah ini:\n";
            cout << "1. Registrasi\n";
            cout << "2. Login\n";
            cout << "0. Exit\n";
            cout << "Choice: ";
            cin >> menu; cin.ignore();

            /* --------------------- REGISTER --------------------- */
            if (menu == 1) {
                int id;
                string name, email, pass;
                cout << "ID: "; cin >> id; cin.ignore();
                cout << "Name: "; getline(cin, name);
                cout << "Email: "; getline(cin, email);
                cout << "Password: "; getline(cin, pass);

                renters.push_back(MotorRenter(id, name, email, pass));
                cout << "Registration successful.\n";
            }

            /* --------------------- LOGIN ------------------------ */
            else if (menu == 2) {
                cout << "\nLogin as:\n1. Renter\n2. Employee\n3.
    Manager\nChoice: ";
                int type; cin >> type; cin.ignore();

                /* RENTER LOGIN */
                if (type == 1) {
                    string e, p;
                    cout << "Email: "; getline(cin, e);
```

```
                        cout << "Password: "; getline(cin, p);

                        bool found = false;
                        for (auto &r : renters) {
                            if (r.login(e, p)) {
                                loggedRenter = &r;
                                found = true;
                                cout << "Login successful.\n";
                                break;
                            }
                        }
                        if (!found) continue;

                        /* USER MENU */
                        int opt;
                        do {
                            cout << "\n=== Renter Menu ===\n";
                            cout << "1. View Available Motors\n";
                            cout << "2. Search Motor by Brand\n";
                            cout << "0. Logout\n";
                            cout << "Choice: ";
                            cin >> opt; cin.ignore();

                            vector<Motor*> listMotor;

                            if (opt == 1) {
                                for (auto &m : motors) {
                                    if (m.getStatus() == "available") {
                                        m.displayMotor();
                                        listMotor.push_back(&m);
                                    }
                                }
                            }
                            else if (opt == 2) {
                                string brand;
                                cout << "Enter brand: ";
                                getline(cin, brand);

                                for (auto &m : motors) {
                                    if (m.getStatus()=="available" &&
    m.getBrand()==brand) {
                                        m.displayMotor();
                                        listMotor.push_back(&m);
                                    }
                                }
                            }

                            if (!listMotor.empty()) {
```

```
                            char choose;
                            cout << "Book one? (y/n): ";
                            cin >> choose; cin.ignore();
                            if (choose == 'y') {
                                int pick;
                                cout << "Enter number (1-" <<
    listMotor.size() << "): ";
                                cin >> pick; cin.ignore();
                                if (pick < 1 || pick > listMotor.size())
    continue;

                                Motor &selected = *listMotor[pick-1];
                                selected.setMotorStatus("booked");

                                string s, e;
                                cout << "Start date (yyyy-mm-dd): ";
                                getline(cin, s);
                                cout << "End date (yyyy-mm-dd): ";
                                getline(cin, e);

                                Booking b(101 + bookings.size(),
    selected, *loggedRenter, s, e);
                                bookings.push_back(b);

                                cout << "\nContinue to payment? (y/n): ";
                                cin >> choose; cin.ignore();
                                if (choose == 'y') {
                                    string date, method;
                                    cout << "Payment method (Cash /
    Transfer): ";
                                    getline(cin, method);
                                    cout << "Payment date: ";
                                    getline(cin, date);

                                    Payment p(201 + payments.size(), b,
    selected.getPrice(), date, method);
                                    p.processPayment();
                                    payments.push_back(p);

                                    string invoiceDate;
                                    cout << "Invoice date: ";
                                    getline(cin, invoiceDate);

                                    Invoice inv(301 + invoices.size(), b,
    invoiceDate, selected.getPrice());
                                    inv.generateInvoice();
                                    invoices.push_back(inv);
                                }
```

```
                }
            }

        } while (opt != 0);

        loggedRenter = nullptr;
    }

    /* EMPLOYEE LOGIN */
    else if (type == 2) {
        int id;
        string name;
        cout << "Employee ID: "; cin >> id; cin.ignore();
        cout << "Name: "; getline(cin, name);

        bool ok = false;
        for (auto &emp : employees) {
            if (emp.getId()==id && emp.getName()==name) {
                loggedEmployee = &emp;
                emp.login();
                ok = true;
                break;
            }
        }
        if (!ok) continue;

        int opt;
        do {
            cout << "\n=== Employee Menu ===\n";
            cout << "1. Manage Motor Data\n";
            cout << "2. Generate Invoice \n";
            cout << "0. Logout\nChoice: ";
            cin >> opt; cin.ignore();

            if (opt == 1)
                loggedEmployee->manageMotorData(motors);

        } while (opt != 0);

        loggedEmployee = nullptr;
    }

    /* MANAGER LOGIN */
    else if (type == 3) {
        int id;
        string name;
        cout << "Manager ID: "; cin >> id; cin.ignore();
        cout << "Name: "; getline(cin, name);
```

```
                        bool ok = false;
                        for (auto &m : managers) {
                            if (m.getId()==id && m.getName()==name) {
                                m.viewTransactionReport(invoices, payments,
      bookings);
                                ok = true;
                                break;
                            }
                        }
                        if (!ok) cout << "Login failed.\n";
                    }
                }

            } while (menu != 0);

            cout << "Program selesai.\n";
            return 0;
}
```

MotorRenter.cpp

```
////////////////////////////////////////////////////////////////
//  MotorRenter.cpp
//  Implementation of the Class MotorRenter
//  Created on:      02-Dec-2025 12:13:58
//  Original author: HP
////////////////////////////////////////////////////////////////

#include "MotorRenter.h"

MotorRenter::MotorRenter(){

}
MotorRenter::~MotorRenter(){

}

void MotorRenter::displayInfo(){

}

bool MotorRenter::login(string password, string email){

return false;
}
```

MotorRenter.h

```
///////////////////////////////////////////////////////////
//  MotorRenter.h
//  Implementation of the Class MotorRenter
//  Created on:      02-Dec-2025 12:13:58
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_)
#define EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_

class MotorRenter
{

public:
	MotorRenter();
	virtual ~MotorRenter();

	void displayInfo();
	bool login(string password, string email);

private:
	string email;
	int idRenter;
	string nameRenter;
	string password;

};
#endif // !defined(EA_D72D18A7_0B25_458c_B46C_29F33FC8281A__INCLUDED_)
```

Motor.cpp

```
///////////////////////////////////////////////////////////
//  Motor.cpp
//  Implementation of the Class Motor
//  Created on:      02-Dec-2025 12:14:00
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Motor.h"

Motor::Motor(){
}

Motor::~Motor(){
}

void Motor::displayMotor(){
}

void Motor::setMotorStatus(string status){
}

void Motor::updateMotorInfo(string newType, string newBrand){
}
```

Motor.h

```
///////////////////////////////////////////////////////////
//  Motor.h
//  Implementation of the Class Motor
//  Created on:      02-Dec-2025 12:14:00
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_)
#define EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_

class Motor
{
public:
	Motor();
	virtual ~Motor();

	void displayMotor();
	void setMotorStatus(string status);
	void updateMotorInfo(string newType, string newBrand);

private:
	/**
	 * Possible values: "available", "booked"
	 */
	string bookingStatus;
	string motorBrand;
	string motorType;
	string plateNumber;
	double rentalPrice;

};
#endif // !defined(EA_9B7108F9_D99B_4cdb_ABA8_AA4B6EAA6247__INCLUDED_)
```

Booking.cpp

```
///////////////////////////////////////////////////////////
//  Booking.cpp
//  Implementation of the Class Booking
//  Created on:      02-Dec-2025 12:14:01
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Booking.h"

Booking::Booking(){
}

Booking::~Booking(){
}

void Booking::cancelBooking(){
}

void Booking::displayBooking(){
}}
```

Booking.h

```
///////////////////////////////////////////////////////////
//  Booking.h
//  Implementation of the Class Booking
//  Created on:      02-Dec-2025 12:14:01
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_)
#define EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_

#include "Motor.h"
#include "MotorRenter.h"

class Booking
{
public:
	Booking();
	virtual ~Booking();
	Motor *m_Motor;
	MotorRenter *m_MotorRenter;

	void cancelBooking();
	void displayBooking();

private:
	int bookingId;
	string endDate;
	Motor motor;
	MotorRenter renter;
	string startDate;

};
#endif // !defined(EA_D859DE25_A113_4457_9E76_83D76BA08414__INCLUDED_)
```

Payment.cpp

```
///////////////////////////////////////////////////////////
//  Payment.cpp
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Payment.h"

Payment::Payment(){
}

Payment::~Payment(){
}

void Payment::displayPayment(){
}
void Payment::processPayment(){
}
```

Payment.h

```
///////////////////////////////////////////////////////////
//  Payment.h
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
#define EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_

#include "Booking.h"

class Payment
{

public:
	Payment();
	virtual ~Payment();
	Booking *m_Booking;

	void displayPayment();
	void processPayment();

private:
	Booking booking;
	string method;
	string paymentDate;
	int paymentId;
	double price;

};
#endif // !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
```

Invoice.cpp

```
///////////////////////////////////////////////////////////
//  Invoice.cpp
//  Implementation of the Class Invoice
//  Created on:      02-Dec-2025 12:14:03
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Invoice.h"

Invoice::Invoice(){
}

Invoice::~Invoice(){
}

void Invoice::generateInvoice(){
}
```

Invoice.h

```
///////////////////////////////////////////////////////////
//  Invoice.h
//  Implementation of the Class Invoice
//  Created on:      02-Dec-2025 12:14:03
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_)
#define EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_

#include "Booking.h"

class Invoice
{

public:
	Invoice();
	virtual ~Invoice();
	Booking *m_Booking;

	void generateInvoice();

private:
	Booking booking;
	int invoiceId;
	string issueDate;
	double totalPrice;

};
#endif // !defined(EA_9011EFC0_9D0A_43ec_B337_3D2A0CD3C828__INCLUDED_)
```

Employee.cpp

```
///////////////////////////////////////////////////////////
//  Employee.cpp
//  Implementation of the Class Employee
//  Created on:      02-Dec-2025 12:14:04
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Employee.h"

Employee::Employee(){
}
Employee::~Employee(){
}
void Employee::generateInvoice(){
}
void Employee::login(){
}
void Employee::manageMotorData(List<Motor> motors){
}
```

Employee.h

```
///////////////////////////////////////////////////////////
//  Payment.h
//  Implementation of the Class Payment
//  Created on:      02-Dec-2025 12:14:02
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
#define EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_

#include "Booking.h"

class Payment
{

public:
	Payment();
	virtual ~Payment();
	Booking *m_Booking;

	void displayPayment();
	void processPayment();

private:
	Booking booking;
	string method;
	string paymentDate;
	int paymentId;
	double price;

};
#endif // !defined(EA_B1965507_BF1A_4261_84FC_5960A99DCDE6__INCLUDED_)
```

Manager.cpp

```
///////////////////////////////////////////////////////////
//  Manager.cpp
//  Implementation of the Class Manager
//  Created on:      02-Dec-2025 12:14:05
//  Original author: HP
///////////////////////////////////////////////////////////

#include "Manager.h"

Manager::Manager(){
}

Manager::~Manager(){
}

void  Manager::viewTransactionReport(List<Invoice>  invoices,  List<Payment>
payments, List<Booking> bookings){

}
```

Manager.h

```
///////////////////////////////////////////////////////////
//  Manager.h
//  Implementation of the Class Manager
//  Created on:      02-Dec-2025 12:14:05
//  Original author: HP
///////////////////////////////////////////////////////////

#if !defined(EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_)
#define EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_

class Manager
{

public:
	Manager();
	virtual ~Manager();

	void viewTransactionReport(List<Invoice> invoices, List<Payment> payments,
List<Booking> bookings);

private:
	int id;
	string name;

};
#endif // !defined(EA_D1AF2607_9532_422d_97FC_89B466FC8E28__INCLUDED_)
```

- Link Github
  https://github.com/farissaf/Tugas_Pemrograman_Dasar_2025/tree/main/PROYEK%20AKHIR%20PEMROGRAMAN%20DASAR
  Folder terdiri dari:
  1. Main_code.cpp
  2. Class Diagram Code Generate
  3. File eap *Use Case Diagram*
  4. File eap *Class Diagram*
  5. File pdf untuk *Use Case* dan *Class Diagram*
  6. File pdf Laporan