Scooter Rental Management System

Title Page

Scooter Rental Management System
A C Programming Project
Submitted By: [Your Name]
Date: November 2025

Abstract

This report presents the design, implementation, and testing of a comprehensive **Scooter Rental Management System** using the C programming language. The objective is to build a console-based application that supports real-time fleet management, customer registration, rental processing, cost calculation, and reporting. The system is designed for scalability, reliability, and clarity, demonstrating best practices in modular programming and data handling in C.

Problem Definition

Urban mobility challenges have increased the demand for flexible and affordable transportation solutions. Scooter rental services address this need by providing easy access to short-term personal vehicles. However, managing a rental fleet, customer records, and rental transactions efficiently requires an organized and robust software system. The problem tackled in this project is to design and implement a software solution for:

- Managing scooter inventory with up-to-date status
- Registering customers and validating their information
- Facilitating rental processes with time tracking and billing
- Ensuring proper data management and user interface for operators

---

System Design

Flowchart

Figure 1: System Flowchart: Main Program Loop and Rental Operations

Algorithms

1. Start Rental Algorithm

1. Prompt user for customer ID and scooter ID.
2. Verify customer exists.
3. Verify scooter exists and is available.
4. Set rental start time to current time.
5. Update scooter status to RENTED.

6. Add rental entry with active status.
7. Display confirmation and rental details.

2. End Rental Algorithm

1. Prompt user for rental ID.
2. Verify active rental exists.
3. Set rental end time to current time.
4. Calculate duration and cost: $cost = duration \times hourly\_rate$
5. Update rental entry and scooter status.
6. Display rental summary and charges.

Implementation Details

The project is organized into three main source files:

- rental_manager.h: Structure definitions and function prototypes
- rental_manager.c: Implementation of management functions
- main.c: User interface and main loop

Key Data Structures

Struct	Purpose
Scooter	Stores scooter info and status
Customer	Tracks customer details and rentals
Rental	Links scooters to customers, timing, cost
RentalManager	Root container for all arrays

Table 1: Main Data Structures in the System

Code Snippets

Scooter Structure

```
typedef struct {
int scooter_id;
char model[MAX_MODEL_LENGTH];
char registration_number[20];
float hourly_rate;
int battery_level;
ScooterStatus status;
} Scooter;
```

Add Scooter Function

```
int add_scooter(RentalManager *manager, const char *model, const char *reg_number,
float hourly_rate) {
Scooter *new_scooter = &manager->scooters[manager->scooter_count];
new_scooter->scooter_id = manager->scooter_count + 1;
strncpy(new_scooter->model, model, MAX_MODEL_LENGTH - 1);
```

```
new_scooter->hourly_rate = hourly_rate;
new_scooter->status = AVAILABLE;
manager->scooter_count++;
printf("Scooter added! ID: %d\n", new_scooter->scooter_id);
return new_scooter->scooter_id;
}
```

Rental Cost Calculation

```
float calculate_rental_cost(Rental *rental, Scooter *scooter) {
double hours = (double)(rental->end_time - rental->start_time) / 3600.0;
return (float)(hours * scooter->hourly_rate);
}
```

Testing & Results

Test Cases

- Add multiple scooters to the fleet and verify correct display
- Register customers and attempt rental with invalid and valid IDs
- Start and end rentals, observing cost calculations for different durations
- Attempt removing scooters during an active rental (validation check)
- Generate reports of scooters, customers, and rental history

Sample Result Output

Operation	Input	Output	Status
Add Scooter	Model: Activa	ID assigned, listed	Pass
Start Rental	Valid IDs	Rental started, status updated	Pass
End Rental	Valid rental	Cost calculated, scooter available	Pass
Remove Scooter	Rented scooter	Error message	Pass

Table 2: Sample Testing Results for Core Features

Conclusion & Future Work

The **Scooter Rental Management System** fulfills essential requirements for managing urban scooter fleets, customer records, and transactions. Its modular C implementation demonstrates structure, error validation, and clear user interface practices. Future enhancements could include:

- Persistent storage (file I/O/database integration)
- Dynamic data structures for scalability
- Advanced search/filter capabilities
- Multi-location and multi-user support
- Payment processing

- Integration with GPS and IoT telemetry

References

[1] Booqable. (2025). Scooter Rental Software - Features and Best Practices. https://booqable.com/industries/scooter-rental-software/
[2] Reservety. (2012). All-in-One Scooter Rental Software. https://reservety.com/scooter-rental-software/
[3] QoreUps. (2025). Best Scooter Rental Software Solutions. https://www.qoreups.com/academy/best-scooter-rental-software-solutions/
[4] Squillion. (2024). Everything You Need To Know About Scooter Sharing Software. https://www.squillion.tech/everything-need-know-scooter-sharing-software/
[5] Luca Vall. (2024). How to Structure C Projects: Best Practices. https://www.lucavall.in/blog/how-to-structure-c-projects-my-experience-best-practices
[6] Mohammad Shadab Abedin. (2025). Best Practices for C/C++ Project Structure. https://www.linkedin.com/pulse/best-practices-cc-project-structure-mohammad-shadab-abedin-chizc
[7] Studocu. (2021). SRS Document for Rental Property Management System. https://www.studocu.com/in/document/university-of-mumbai/computer-engineering/srs-example/16730518

Appendix

Sample Menu from Application

- Add Scooter
- Display All Scooters
- Display Available Scooters
- Remove Scooter
- Add Customer
- Display All Customers
- Start Rental
- End Rental
- Display Active Rentals
- Display Rental History
- Update Scooter Status
- Exit

End of Report