

Stage 1



Project Name


Vehicle Rental Management System - **Group C**

Team Members

1. Baraza Brian
2. Tako Nellyvine Mizero

Problem Description

Many small vehicle rental companies, especially local car rental shops, still depend on manual paper-based records or simple spreadsheets to handle their daily operations. This manual approach often results in mistakes, such as accidentally booking the same vehicle to two different customers at the same time. Staff spend a lot of time manually checking and updating vehicle availability, which slows down the rental process. Calculating rental



fees based on the number of days and vehicle type is done by hand, leading to billing errors and customer complaints. Keeping track of past rentals and customer details is also disorganized, making it hard to find information quickly when needed. A basic computerized Vehicle Rental Management System would solve these issues by automating the key tasks. This system will make operations faster, more accurate, and easier for the rental staff to manage.

Main Objectives of the Programme

1. To automate the vehicle rental and return process
2. To maintain accurate records of vehicles, customers, and rentals
3. To ensure real-time tracking of vehicle availability
4. To calculate rental costs and track payments
5. To generate basic reports for management purposes

Main Features Of the Programme

6. Add, update, and remove vehicle records
7. Register and manage customer information
8. Rent out and return vehicles while tracking availability
9. Calculate rental costs based on duration and vehicle type
10. Store and retrieve rental history records
11. Generate simple reports for rentals and vehicles
12. Mark vehicles as available, rented, or in maintenance

Data to Be Stored in Files

- Vehicle details (vehicle ID, type, brand, availability status, rental rate)
- Customer information (customer ID, name, contact details, license number)
- Rental records (rental ID, customer ID, vehicle ID, rental date, return date, total cost)
- Payment records (payment ID, rental ID, amount paid, payment date)

Initial List of Planned Classes

- Vehicle – Stores vehicle details and availability status
- Customer – Manages customer information
- Rental – Handles rental transactions between customers and vehicles
- Payment – Manages payment details for rentals
- FileManager – Reads from and writes data to files
- MainSystem – Controls program flow and user interaction

Initial Program Flow

- User logs into the system
- User selects an action (manage vehicles, customers, or rentals)
- System checks vehicle availability
- Rental is created and stored in files
- Payment is processed and recorded
- Vehicle status is updated upon return

Facilitator Approval:

Approved ?

☒ YES

☐ NO

Feedback: ___ Good on my end please proceed!!_____

Date: _____

Stage 2/ Stage 4

Vehicle Rental Management System

Stage 2: Class Design and Test Plan

Project Name: Vehicle Rental Management System, Group C

Team Members:

- Baraza Brian
- Tako Nellyvine Mizero

1. System Overview

The Vehicle Rental Management System (VRMS) is designed to automate vehicle rental operations for small rental companies. The system replaces manual, paper-based processes with a computerized solution that improves accuracy, efficiency, and record management.

2. Class Design

2.1 Vehicle Class

Purpose: Stores and manages vehicle details and availability status.

Attributes:

- vehicleID: str
- type: str
- brand: str
- rentalRate: float
- status: str

Methods:

- get_availability_status()
- set_availability_status()
- to_string() / from_string()
- display_details()

2.2 Customer Class

Purpose: Manages customer information.

Attributes:

- customerID: str
- name: str
- contactDetails: str
- licenseNumber: str

Methods:

- get_customer_details()
- to_string() / from_string()
- display_details()

2.3 Rental Class

Purpose: Handles rental transactions between customers and vehicles.

Attributes:

- rentalID: str
- customerID: str
- vehicleID: str
- rentalDate: str
- returnDate: str
- totalCost: float

Methods:

- calculate_rental_cost()
- close_rental()
- get_rental_details()
- to_string() / from_string()
- display_details()

2.4 Payment Class

Purpose: Manages payment details for rentals.

Attributes:

- paymentID: str
- rentalID: str
- amountPaid: float
- paymentDate: str

Methods:

- validate_payment()
- process_payment()
- get_payment_details()
- to_string() / from_string()
- display_details()

2.5 FileManager Class

Purpose: Reads and writes system data to files.

Attributes:

- fileName: str

Methods:

- write_to_file()
- read_from_file()
- update_file()
- delete_from_file()
- search_by_id()

2.6 MainSystem Class

Purpose: Controls overall program flow and user interaction.

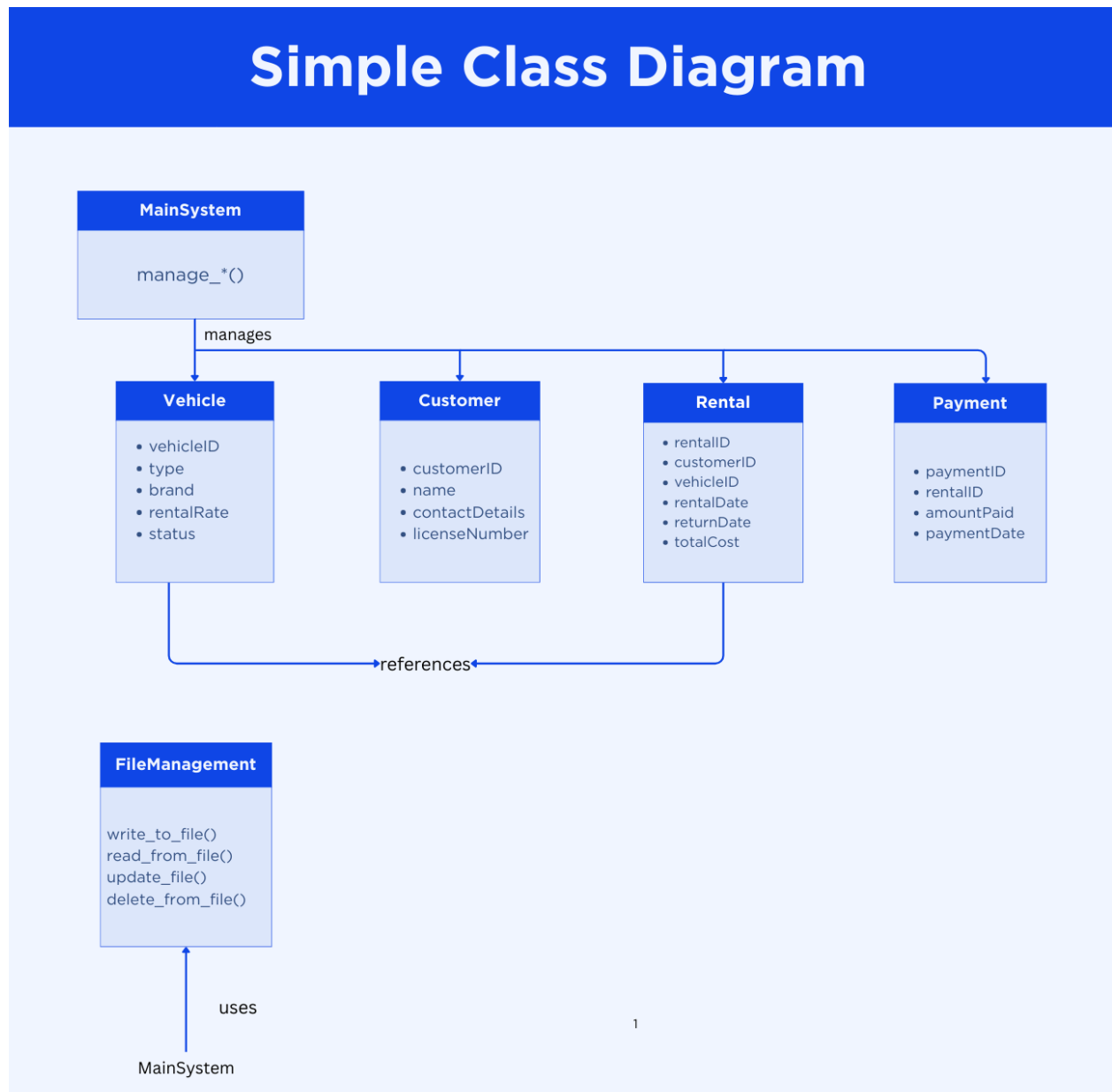
Attributes:

- currentUser: str
- vehicle_file:
- customer_file
- rental_file
- payment_file
- user_file

Methods:

- authentication_menu()
- display_menu()
- manage_vehicle()
- manage_customers()
- manage_rentals()
- manage_payment()
- generate_reports()

3. Simple Class Diagram (Textual Representation)



Drawn using canvas. This is the [link](#) to the drawing.

4. Test Plan

4.1 Vehicle Class – Test Plan

Class Being Tested: Vehicle

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|---------|-----------------------|------------------------------|--|---------------------------------------|
| TC1 | addVehicle | Add a new valid vehicle | vehicleID="V001", type="Sedan", brand="Toyota", rentalRate=45.0 | Vehicle successfully added |
| TC2 | updateVehicle | Update existing vehicle rate | vehicleID="V001", new rentalRate=50.0 | Vehicle rental rate updated to 50.0 |
| TC3 | setAvailabilityStatus | Mark the vehicle as rented | vehicleID="V001", status="Rented" | Status changed to "Rented" |
| TC4 | addVehicle | Add vehicle with missing ID | vehicleID=None or empty string | Error message displayed |
| TC5 | removeVehicle | Remove non-existent vehicle | vehicleID="V999" (doesn't exist) | Operation rejected with error message |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|---------|---|-----------|--|
| TC1 | Vehicle object created and saved to vehicles.txt file | Pass | Vehicle details displayed correctly with status "Available." |
| TC2 | Vehicle record in file updated with new rate | Pass | Changes reflected in the vehicle file |

| | | | |
|-----|--|------|------------------------------------|
| TC3 | Vehicle status changed to Rented | Pass | Status updated correctly |
| TC4 | System displayed "Error: Invalid or duplicate Vehicle ID." | Pass | Missing ID handled |
| TC5 | System displayed "Vehicle not found." | Pass | Handles missing records gracefully |

4.2 Customer Class – Test Plan

Class Being Tested: Customer

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|----------------|----------------------------|-----------------------------|--|------------------------------------|
| TC1 | addCustomer | Register a valid customer | customerID="C001", name="John Doe", contact="+250788123456", license="DL123456" | Customer added successfully |
| TC2 | updateCustomer | Update contact details | customerID="C001", new contact="+250788999999" | Details updated |
| TC3 | getCustomerDetails | Retrieve existing customer | customerID="C001" | Customer info displayed |

| | | | | |
|-----|----------------|-----------------------------------|--|-------------------------|
| TC4 | addCustomer | Add customer with missing license | customerID="C002", license=None or empty | Error message displayed |
| TC5 | removeCustomer | Remove invalid customer | customerID="C999" (doesn't exist) | Operation rejected |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|---------|--|-----------|---------------------------------------|
| TC1 | Customer object created and saved to customers.txt | Pass | All customer details stored correctly |
| TC2 | Customer record updated in file with new contact | Pass | Data updated successfully |
| TC3 | Dictionary returned with all customer attributes | Pass | Correct customer returned |
| TC4 | System showed "Error: License number is required" | Pass | License required |
| TC5 | Message displayed "Customer not found or could not be removed" | Pass | Invalid ID handled |

4.3 Rental Class – Test Plan

Class Being Tested: Rental

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|----------------|----------------------------|--|--|--|
| TC1 | createRental | Create valid rental | rentalID="R001", customerID="C001", vehicleID="V001", rentalDate="2026-01-01" | Rental created, vehicle status changed to "Rented" |
| TC2 | calculateRentalCost | Calculate cost for normal period | rentalDate="2026-01-01", returnDate="2026-01-06", rate=45.0 | Correct total cost, Total cost = 225.0 (5 days × 45) |
| TC3 | closeRental | Return vehicle and close rental | rentalID="R001", returnDate="2026-01-06" | Vehicle marked Available, rental closed |
| TC4 | createRental | Attempt to rent an unavailable vehicle | vehicleID="V002" (status="Rented") | Rental denied with error message |
| TC5 | calculateRentalCost | Negative rental duration | rentalDate="2026-01-05", returnDate="2026-01-03" | Error message for invalid duration |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|----------------|--|------------------|--|
| TC1 | Rental saved to file, vehicle marked as | Pass | Complete rental workflow successful |

| | | | |
|-----|---|------|--|
| | Rented | | |
| TC2 | Method returned 225.0, totalCost updated | Pass | Calculation is accurate for a 5-day rental |
| TC3 | Vehicle status updated, returnDate set | Pass | Return process completed successfully |
| TC4 | System displayed "Error: Vehicle is not available. Current status: Rented." | Pass | Prevents double booking |
| TC5 | ValueError raised: "Rental duration must be at least 1 day." | Pass | Date validation prevents logical errors |

4.4 Payment Class – Test Plan

Class Being Tested: Payment

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|----------------|----------------------------|-----------------------------|--|---|
| TC1 | processPayment | Process valid payment | paymentID="P001", rentalID="R001", amount=225.0, date="2026-01-06" | Payment recorded successfully |
| TC2 | validatePayment | Validate exact payment | amountPaid=225.0, expectedAmount=225.0 | Payment accepted, validation returns True |
| TC3 | validatePayment | Detect underpayment | amountPaid=200.0, expectedAmount=225.0 | Payment rejected, returns False |

| | | | | |
|-----|-------------------|--------------------------|------------------|--------------------|
| TC4 | processPayment | Attempt negative payment | amount=-100.0 | Error message |
| TC5 | getPaymentDetails | Retrieve payment | paymentID="P001" | Payment info shown |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|---------|--|-----------|---|
| TC1 | Payment saved to payments.txt file | Pass | Payment processing workflow complete |
| TC2 | Method returned True | Pass | Exact amount validated |
| TC3 | Method returned False | Pass | Underpayment properly detected |
| TC4 | ValueError raised: "Payment amount cannot be negative." | Pass | Input validation prevents negative values |
| TC5 | Dictionary with all payment details returned | Pass | Data retrieval accurate |

4.5 FileManager Class – Test Plan

Class Being Tested: FileManager

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|---------|---------------------|----------------------|---|-----------------------------|
| TC1 | writeToFile | Save valid data | data="V001 Sedan Toyota 45.0 Available" | Data saved to file |

| | | | | |
|-----|----------------|------------------------|--|---|
| TC2 | readFromFile | Read existing file | fileName="vehicles.txt" | List of all lines returned |
| TC3 | updateFile | Update existing record | recordID="V001", updatedData="V001 Sedan Toyota 50.0 Available" | Record updated in file |
| TC4 | deleteFromFile | Delete record | recordID="V001" | Record removed from file |
| TC5 | readFromFile | Read non-existent file | fileName="missing.txt" | Error handled gracefully, empty list returned |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|---------|---|-----------|--|
| TC1 | Line added to file, method returned True | Pass | Write operation successful |
| TC2 | All records returned as list of strings | Pass | Read operation retrieves all data |
| TC3 | Specific line replaced with new data | Pass | Update locates and modifies correct record |
| TC4 | Line deleted, remaining records intact | Pass | Delete operation preserves other records |
| TC5 | Message displayed: "File not found. Creating new file." Returns [] | Pass | FileNotFoundException handled properly |

4.6 MainSystem Class – Test Plan

Class Being Tested: MainSystem

Test Case Table (Before Coding)

| Test ID | Method Being Tested | Scenario Description | Input(s) | Expected Output / Behaviour |
|----------------|----------------------------|-----------------------------|--|------------------------------------|
| TC1 | login | Valid login | username="admin", password="admin123" | Access granted, user logged in |
| TC2 | login | Invalid login | username="wrong", password="wrong" | Access denied, error message shown |
| TC3 | displayMenu | Show main menu | None (system state) | Menu with 6 options displayed |
| TC4 | handleUserChoice | Invalid menu option | choice="9" (out of range) | Error message for invalid option |
| TC5 | exitSystem | Exit program gracefully | choice="6" | System closes with goodbye message |

Test Execution Table (After Coding)

| Test ID | Actual Output | Pass/Fail | Comments |
|----------------|---|------------------|--------------------------------|
| TC1 | Login successful, main menu displayed | Pass | Authentication works correctly |
| TC2 | Message: "Invalid credentials. Please try again." | Pass | Rejects incorrect credentials |
| TC3 | All menu options | Pass | Menu formatting correct |

| | | | |
|-----|---|------|-----------------------------------|
| | printed clearly | | |
| TC4 | Message: "Invalid option. Please select 1-6." | Pass | Input validation prevents crashes |
| TC5 | Program exited | Pass | System closed correctly |