Course: BSD 214 – Object-Oriented Programming 2
Project Title: Car Rental and Login System
Student Name: Meek Ngugi
Registration No: BSE-01-0238/2025
Date: 03rd-october, 2025

GitHub Repository:
https://github.com/meekngugi/OOP-assignment-one-.git

Table of Contents

1. Car Rental System

Introduction

The **Car Rental System** is an Object-Oriented Java application designed to manage a rental agency's cars, customers, and rental records. It allows staff to register customers, add cars, rent cars, return cars, and view all transactions.

CarRentalSystem

- Entry point of the application.
- Handles user interaction via console menu.
- Provides operations: rent, return, add car, register customer, view cars/customers/rentals.

Car

- Represents a single rental car.
- Attributes:
  - carId, make, model, year, dailyRate, isAvailable.
- Methods:
  - Getters/Setters.
  - displayInfo() – shows car details.

Customer

- Represents a customer.
- Attributes:
  - customerId, name, email, phone.
- Methods:
  - Getters.
  - displayInfo() – shows customer details.

RentalRecord

- Represents a rental transaction.
- Attributes:
  - carId, customerId, rentalDate, returnDate, rentalDays, totalCost, isActive.
- Methods:
  - Constructor – calculates return date & cost.
  - completeRental() – marks rental as finished.
  - displayInfo() – shows rental details.

RentalAgency

- Manages collections of cars, customers, and rentals.
- Key Methods:
  - addCar(Car c)
  - addCustomer(Customer c)
  - rentCar(carId, customerId, days)
  - returnCar(carId)
  - displayAvailableCars(), displayAllCustomers(), displayAllRentals()

CAR RENTAL SYSTEM SCREENSHOTS

1. **Car system codes**

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.time.LocalDate;

public class CarRentalSystem {
    public static void main(String[] args) {
        RentalAgency agency = new RentalAgency();
        Scanner scanner = new Scanner( source: System.in);

        // Initialize with sample data
        initializeSampleData(agency);

        boolean running = true;

        System.out.println( x: "=== CAR RENTAL MANAGEMENT SYSTEM ===");

        while (running) {
            displayMenu();
            int choice = getIntInput(scanner, prompt: "Enter your choice: ");

            switch (choice) {
                case 1:
                    rentCar(agency, scanner);
                    break;
                case 2:
                    returnCar(agency, scanner);
                    break;
                case 3:
                    addNewCar(agency, scanner);
                    break;
                case 4:
                    registerCustomer(agency, scanner);
                    break;
                case 5:
                    agency.displayAvailableCars();
                    break;
                case 6:
                    agency.displayAllCustomers();
                    break;
                case 7:
                    agency.displayAllRentals();
                    break;
                case 8:
                    running = false;
                    System.out.println( x: "Thank you for using Car Rental System!");
```

Figure 1: login code

```
                System.out.println(x: "Thank you for using Car Rental System!");
                break;
            default:
                System.out.println(x: "Invalid choice! Please try again.");
        }
    }
    scanner.close();
}

private static void displayMenu() {
    System.out.println(x: "\n===== MAIN MENU =====");
    System.out.println(x: "1. Rent a Car");
    System.out.println(x: "2. Return a Car");
    System.out.println(x: "3. Add New Car");
    System.out.println(x: "4. Register New Customer");
    System.out.println(x: "5. View Available Cars");
    System.out.println(x: "6. View All Customers");
    System.out.println(x: "7. View All Rentals");
    System.out.println(x: "8. Exit");
    System.out.println(x: "====================");
}

private static void rentCar(RentalAgency agency, Scanner scanner) {
    System.out.println(x: "\n--- Rent a Car ---");
    agency.displayAvailableCars();

    String carId = getStringInput(scanner, prompt: "Enter Car ID to rent: ");
    int customerId = getIntInput(scanner, prompt: "Enter Customer ID: ");
    int days = getIntInput(scanner, prompt: "Enter rental days: ");

    boolean success = agency.rentCar(carId, customerId, days);
    if (success) {
        System.out.println(x: "Car rented successfully!");
    } else {
        System.out.println(x: "Failed to rent car. Please check availability.");
    }
}

private static void returnCar(RentalAgency agency, Scanner scanner) {
    System.out.println(x: "\n--- Return a Car ---");
    String carId = getStringInput(scanner, prompt: "Enter Car ID to return: ");

    boolean success = agency.returnCar(carId);
    if (success) {
        System.out.println(x: "Car returned successfully!");
    } else {
```

```
            System.out.println("Car returned successfully!");
        } else {
            System.out.println("Failed to return car. Car not found or not rented.");
        }
    }

    private static void addNewCar(RentalAgency agency, Scanner scanner) {
        System.out.println("\n--- Add New Car ---");
        String id = getStringInput(scanner, "Enter Car ID: ");
        String make = getStringInput(scanner, "Enter Car Make: ");
        String model = getStringInput(scanner, "Enter Car Model: ");
        int year = getIntInput(scanner, "Enter Year: ");
        double dailyRate = getDoubleInput(scanner, "Enter Daily Rate: ");

        Car car = new Car(carId: id, make, model, year, dailyRate);
        agency.addCar(car);
        System.out.println("✅ New car added successfully!");
    }

    private static void registerCustomer(RentalAgency agency, Scanner scanner) {
        System.out.println("\n--- Register New Customer ---");
        int id = getIntInput(scanner, "Enter Customer ID: ");
        String name = getStringInput(scanner, "Enter Customer Name: ");
        String email = getStringInput(scanner, "Enter Email: ");
        String phone = getStringInput(scanner, "Enter Phone: ");

        Customer customer = new Customer(customerId:id, name, email, phone);
        agency.addCustomer(customer);
        System.out.println("✅ New customer registered successfully!");
    }

    private static void initializeSampleData(RentalAgency agency) {
        // Add sample cars
        agency.addCar(new Car(carId: "C001", make: "Toyota", model: "Camry", year: 2022, dailyRate: 45.00));
        agency.addCar(new Car(carId: "C002", make: "Honda", model: "Civic", year: 2023, dailyRate: 40.00));
        agency.addCar(new Car(carId: "C003", make: "Ford", model: "Mustang", year: 2021, dailyRate: 65.00));

        // Add sample customers
        agency.addCustomer(new Customer(customerId:101, name: "John Doe", email: "john@email.com", phone: "123-456-7890"));
        agency.addCustomer(new Customer(customerId:102, name: "Jane Smith", email: "jane@email.com", phone: "123-456-7891"));
    }

    // Utility methods for input validation
    private static int getIntInput(Scanner scanner, String prompt) {
        System.out.print(s: prompt);
        while (!scanner.hasNextInt()) {
```

```
            System.out.print(x: "Invalid input! Please enter a number: ");
            scanner.next();
        }
        int value = scanner.nextInt();
        scanner.nextLine();
        return value;
    }

    private static double getDoubleInput(Scanner scanner, String prompt) {
        System.out.print(x: prompt);
        while (!scanner.hasNextDouble()) {
            System.out.print(x: "Invalid input! Please enter a decimal number: ");
            scanner.next();
        }
        double value = scanner.nextDouble();
        scanner.nextLine();
        return value;
    }

    private static String getStringInput(Scanner scanner, String prompt) {
        System.out.print(x: prompt);
        return scanner.nextLine();
    }
}
class Car {
    private String carId;
    private String make;
    private String model;
    private int year;
    private double dailyRate;
    private boolean isAvailable;
    public Car(String carId, String make, String model, int year, double dailyRate) {
        this.carId = carId;
        this.make = make;
        this.model = model;
        this.year = year;
        this.dailyRate = dailyRate;
        this.isAvailable = true; // Cars are available by default when added
    }
    public String getCarId() { return carId; }
    public String getMake() { return make; }
    public String getModel() { return model; }
    public int getYear() { return year; }
    public double getDailyRate() { return dailyRate; }
    public boolean isAvailable() { return isAvailable; }
```

```
    public boolean isAvailable() { return isAvailable; }
    public void setAvailable(boolean available) { isAvailable = available; }
    public void displayInfo() {
        System.out.printf(format: "ID: %s | %s %s %s | Rate: $%.2f/day | Status: %s%n",
                args: carId, args: year, args: make, args: model, args: dailyRate,
                isAvailable ? "Available" : "Rented");
    }
}
class Customer {
    private int customerId;
    private String name;
    private String email;
    private String phone;
    public Customer(int customerId, String name, String email, String phone) {
        this.customerId = customerId;
        this.name = name;
        this.email = email;
        this.phone = phone;
    }
    public int getCustomerId() { return customerId; }
    public String getName() { return name; }
    public String getEmail() { return email; }
    public String getPhone() { return phone; }
    public void displayInfo() {
        System.out.printf(format: "ID: %d | Name: %s | Email: %s | Phone: %s%n",
                args: customerId, args: name, args: email, args: phone);
    }
}
class RentalRecord {
    private String carId;
    private int customerId;
    private LocalDate rentalDate;
    private LocalDate returnDate;
    private int rentalDays;
    private double totalCost;
    private boolean isActive;
    public RentalRecord(String carId, int customerId, int rentalDays, double dailyRate) {
        this.carId = carId;
        this.customerId = customerId;
        this.rentalDate = LocalDate.now();
        this.rentalDays = rentalDays;
        this.returnDate = this.rentalDate.plusDays(daysToAdd: rentalDays);
        this.totalCost = dailyRate * rentalDays;
        this.isActive = true;
    }
```

```
    }
    public void completeRental() {
        this.isActive = false;
    }
    public String getCarId() { return carId; }
    public int getCustomerId() { return customerId; }
    public boolean isActive() { return isActive; }
    public void displayInfo() {
        System.out.printf(format: "Car ID: %s | Customer ID: %d | Rental Date: %s | Return Date: %s | Cost: $%.2f |
                arga: carId, arga: customerId, arga: rentalDate, arga: returnDate, arga: totalCost,
                isActive ? "Active" : "Completed");
    }
}
class RentalAgency {
    private List<Car> cars;
    private List<Customer> customers;
    private List<RentalRecord> rentals;
    public RentalAgency() {
        this.cars = new ArrayList<>();
        this.customers = new ArrayList<>();
        this.rentals = new ArrayList<>();
    }
    public void addCar(Car car) {
        cars.add(e: car);
    }
    public void addCustomer(Customer customer) {
        customers.add(e: customer);
    }
    public boolean rentCar(String carId, int customerId, int days) {
        Car car = findCarById(carId);
        if (car == null || !car.isAvailable()) {
            return false;
        }
        Customer customer = findCustomerById(customerId);
        if (customer == null) {
            return false;
        }
        RentalRecord rental = new RentalRecord(carId, customerId, rentalDays:days, dailyRate: car.getDailyRate());
        rentals.add(e: rental);
        car.setAvailable(available: false);

        return true;
    }

    // Return a rented car
    public boolean returnCar(String carId) {
```

```
        RentalRecord rental = findActiveRentalByCarId(carId);
        if (rental == null) {
            return false;
        }
        rental.completeRental();
        Car car = findCarById(carId);
        if (car != null) {
            car.setAvailable(available: true);
        }

        return true;
    }
    private Car findCarById(String carId) {
        for (Car car : cars) {
            if (car.getCarId().equals(anObject: carId)) {
                return car;
            }
        }
        return null;
    }

    // Helper method to find customer by ID
    private Customer findCustomerById(int customerId) {
        for (Customer customer : customers) {
            if (customer.getCustomerId() == customerId) {
                return customer;
            }
        }
        return null;
    }
    private RentalRecord findActiveRentalByCarId(String carId) {
        for (RentalRecord rental : rentals) {
            if (rental.getCarId().equals(anObject: carId) && rental.isActive()) {
                return rental;
            }
        }
        return null;
    }
    public void displayAvailableCars() {
        System.out.println(x: "\n--- AVAILABLE CARS ---");
        boolean found = false;
        for (Car car : cars) {
            if (car.isAvailable()) {
                car.displayInfo();
                found = true;
```

```
            if (car.isAvailable()) {
                car.displayInfo();
                found = true;
            }
        }
        if (!found) {
            System.out.println( x: "No cars available at the moment.");
        }
    }
    public void displayAllCustomers() {
        System.out.println( x: "\n--- REGISTERED CUSTOMERS ---");
        if (customers.isEmpty()) {
            System.out.println( x: "No customers registered.");
        } else {
            for (Customer customer : customers) {
                customer.displayInfo();
            }
        }
    }
    public void displayAllRentals() {
        System.out.println( x: "\n--- RENTAL HISTORY ---");
        if (rentals.isEmpty()) {
            System.out.println( x: "No rental records found.");
        } else {
            for (RentalRecord rental : rentals) {
                rental.displayInfo();
            }
        }
    }
}
```

2. Rent a car and view rental history

```
====================
Enter your choice: 1

--- Rent a Car ---

--- AVAILABLE CARS ---
ID: C001 | 2022 Toyota Camry | Rate: $45.00/day | Status: Available
ID: C002 | 2023 Honda Civic | Rate: $40.00/day | Status: Available
ID: C003 | 2021 Ford Mustang | Rate: $65.00/day | Status: Available
Enter Car ID to rent: C001
Enter Customer ID: 101
Enter rental days: 10
? Car rented successfully!

===== MAIN MENU =====
1. Rent a Car
2. Return a Car
3. Add New Car
4. Register New Customer
5. View Available Cars
6. View All Customers
7. View All Rentals
8. Exit
====================
Enter your choice: 7

--- RENTAL HISTORY ---
Car ID: C001 | Customer ID: 101 | Rental Date: 2025-10-03 | Return Date: 2025-10-13 | Cost: $450.00 | Status: Active
```

Figure 2:rent car/view rental

3. Return a car

```
===== MAIN MENU =====
1. Rent a Car
2. Return a Car
3. Add New Car
4. Register New Customer
5. View Available Cars
6. View All Customers
7. View All Rentals
8. Exit
====================
Enter your choice:
C001
Invalid input! Please enter a number: 2

--- Return a Car ---
Enter Car ID to return: C001
? Car returned successfully!
```

Figure 3: return a car

4. Add new cars and Display available cars

```
--- Add New Car ---
Enter Car ID: 104
Enter Car Make: Kenya
Enter Car Model: Mobius
Enter Year: 2023
Enter Daily Rate: 20
? New car added successfully!

===== MAIN MENU =====
1. Rent a Car
2. Return a Car
3. Add New Car
4. Register New Customer
5. View Available Cars
6. View All Customers
7. View All Rentals
8. Exit
====================
Enter your choice: 5

--- AVAILABLE CARS ---
ID: C001 | 2022 Toyota Camry | Rate: $45.00/day | Status: Available
ID: C002 | 2023 Honda Civic | Rate: $40.00/day | Status: Available
ID: C003 | 2021 Ford Mustang | Rate: $65.00/day | Status: Available
ID: 104 | 2023 Kenya Mobius | Rate: $20.00/day | Status: Available
```

Figure 4: add a new car and display them

5. **Register customers and view all customers**

```
====================
Enter your choice: 4

--- Register New Customer ---
Enter Customer ID: C007
Invalid input! Please enter a number: 102
Enter Customer Name: Mike
Enter Email: email@example.com
Enter Phone: 0192738623
? New customer registered successfully!

===== MAIN MENU =====
1. Rent a Car
2. Return a Car
3. Add New Car
4. Register New Customer
5. View Available Cars
6. View All Customers
7. View All Rentals
8. Exit
====================
Enter your choice: 6

--- REGISTERED CUSTOMERS ---
ID: 101 | Name: John Doe | Email: john@email.com | Phone: 123-456-7890
ID: 102 | Name: Jane Smith | Email: jane@email.com | Phone: 123-456-7891
ID: 102 | Name: Mike | Email: email@example.com | Phone: 0192738623
```

Figure 5: Register customers and view all customers

---

2. Login System

The **Login System** is a simple authentication module that verifies user credentials before allowing access to the Car Rental System. It has built-in password masking and a maximum of 3 login attempts.

LoginSystem

- Handles login process.
- Predefined credentials:
  - Username → admin
  - Password → password123

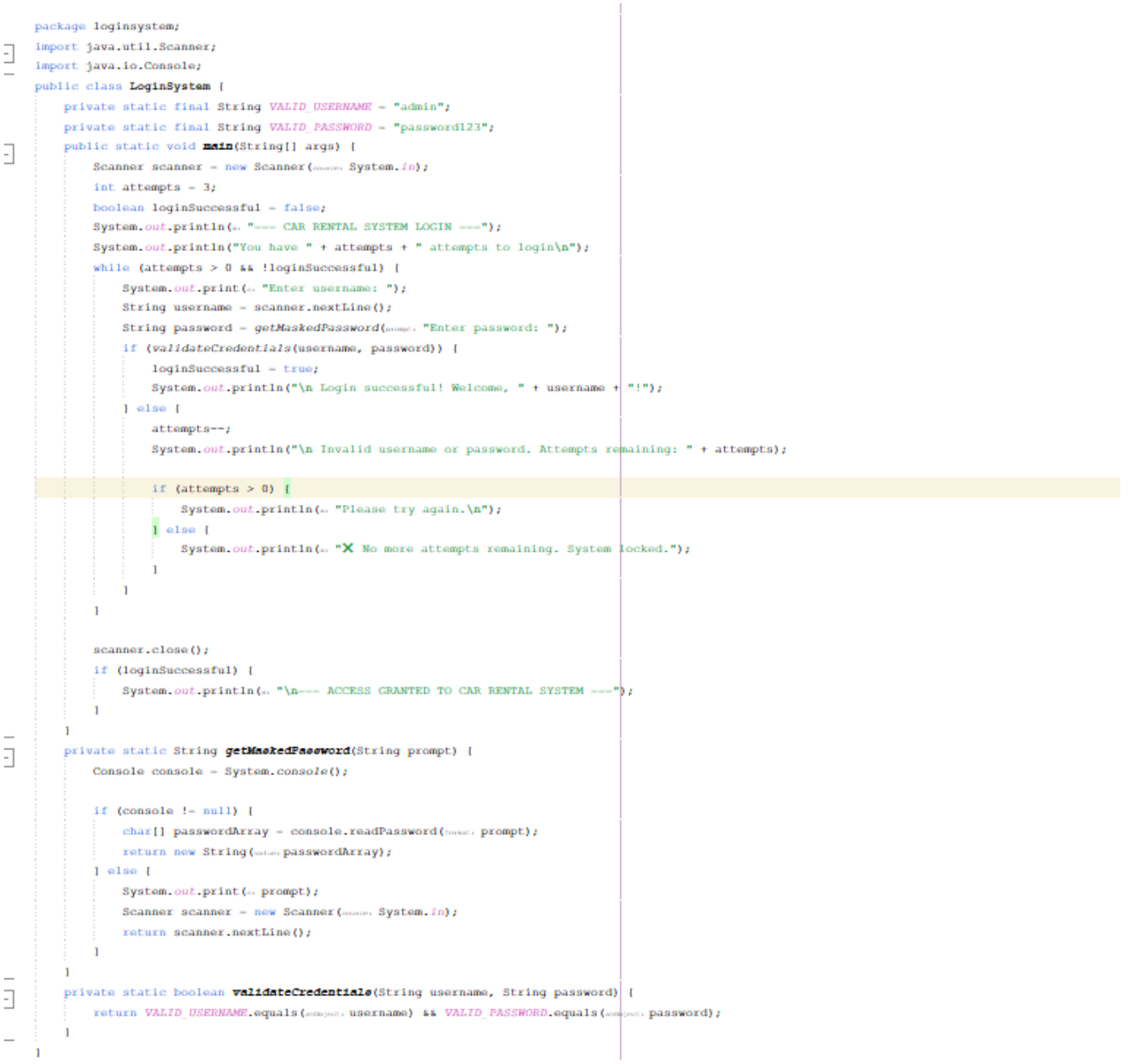
- Methods:
  - main() → runs authentication loop.
  - getMaskedPassword(prompt) ⟶ hides password input if possible.
  - validateCredentials(username, password) → checks validity.

Features

Username & password validation
Password masking support ()
3 attempts allowed before lockout
On success → access granted to Car Rental System

Login screenshots

Login code



```java
package loginsystem;
import java.util.Scanner;
import java.io.Console;
public class LoginSystem {
    private static final String VALID_USERNAME = "admin";
    private static final String VALID_PASSWORD = "password123";
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int attempts = 3;
        boolean loginSuccessful = false;
        System.out.println("=== CAR RENTAL SYSTEM LOGIN ===");
        System.out.println("You have " + attempts + " attempts to login\n");
        while (attempts > 0 && !loginSuccessful) {
            System.out.print("Enter username: ");
            String username = scanner.nextLine();
            String password = getMaskedPassword("Enter password: ");
            if (validateCredentials(username, password)) {
                loginSuccessful = true;
                System.out.println("\n Login successful! Welcome, " + username + "!");
            } else {
                attempts--;
                System.out.println("\n Invalid username or password. Attempts remaining: " + attempts);

                if (attempts > 0) {
                    System.out.println("Please try again.\n");
                } else {
                    System.out.println("✗ No more attempts remaining. System locked.");
                }
            }
        }

        scanner.close();
        if (loginSuccessful) {
            System.out.println("\n=== ACCESS GRANTED TO CAR RENTAL SYSTEM ===");
        }
    }
    private static String getMaskedPassword(String prompt) {
        Console console = System.console();

        if (console != null) {
            char[] passwordArray = console.readPassword(prompt);
            return new String(passwordArray);
        } else {
            System.out.print(prompt);
            Scanner scanner = new Scanner(System.in);
            return scanner.nextLine();
        }
    }
    private static boolean validateCredentials(String username, String password) {
        return VALID_USERNAME.equals(username) && VALID_PASSWORD.equals(password);
    }
}
```

Figure 6: Login code

Login errors

```
run:
=== CAR RENTAL SYSTEM LOGIN ===
You have 3 attempts to login

Enter username: adm
Enter password: pass\

? Invalid username or password. Attempts remaining: 2
Please try again.

Enter username: admin
Enter password: pasww

? Invalid username or password. Attempts remaining: 1
Please try again.

Enter username: admin
Enter password: 123

? Invalid username or password. Attempts remaining: 0
? No more attempts remaining. System locked.
BUILD SUCCESSFUL (total time: 41 seconds)
```

Figure 7: Logging errors

Login success

```
run:
=== CAR RENTAL SYSTEM LOGIN ===
You have 3 attempts to login

Enter username: admin
Enter password: password123

? Login successful! Welcome, admin!

=== ACCESS GRANTED TO CAR RENTAL SYSTEM ===
BUILD SUCCESSFUL (total time: 58 seconds)
```

Figure 8: Login success

GitHub Repository

You can access the source code here:

GitHub Link: Meek Github assignment