**Tunku Abdul Rahman**

**University of Management and Technology**

**BACS2023**

**Object-Oriented Programming**

**Assignment**

**2022/2023**

|  |  |  |
| --- | --- | --- |
| Programme | : | Bachelor of Software Engineering |
| Tutorial Group | : | 3 |

**Team Members:**

|  |  |
| --- | --- |
| **No** | **Student Name** |
| 1. | Chan Kien Yew |
| 2. |  |
| 3. |  |
| 4. |  |

|  |  |  |
| --- | --- | --- |
| No. | Team Member | Task(s) Allocated |
| 1. | Chan Kien Yew | all |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |

**Coursework Declaration**

We confirm that we have read and shall comply with all the terms and conditions of TAR University College’s plagiarism policy.

We declare that this assignment is free from all forms of plagiarism and for all intents and purposes is our own properly derived work.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Signature | : | KY |  |  |  |
| Name | : | Chan Kien Yew |  |  |  |

UML

TaxiInterface

-------------------------------------------------------

-------------------------------------------------------

+calculateTotalFee(distance: double, taxiFee: double)

+setTaxiId(taxiId: int): void

+setIsAvailable(\_isAvailable: boolean): void

+setTotalFee(\_totalFee: double): void

+setDepartTime(\_departTime: String): void

+setDepartDate(\_departDate: String): void

+setSeatType(\_seatType: String): void

+setNoSeats(\_noSeats: int): void

+getTaxiId(): int

+getIsAvailable(): boolean

+getTotalFee(): double

+getDepartTime(): String

+getDepartDate(): String

+getSeatType(): String

+getNoSeats(): int

+toString(): String

+equals(obj: Object): boolean

Time

---------------------

~departTime: String

~departDate: String

---------------------

+Time()

+toString(): String

Taxi

-----------------------------------------------------------

#taxiId: int

#isAvailable: boolean

#totalFee: double

#time: Time

#seat: Seat

------------------------------------------------------------

+Taxi()

+calculateTotalFee(distance: double, taxiFee: double): double

+setTaxiId(\_taxiId: int): void

+setIsAvailable(\_isAvailable: boolean): void

+setTotalFee(\_totalFee: double): void

+setDepartTime(\_departTime: String): void

+setDepartDate(\_departDate: String): void

+setSeatType(\_seatType: String): void

+setNoSeats(\_noSeats: int): void

+getTaxiId(): int

+getIsAvailable(): boolean

+getTotalFee(): double

+getDepartTime(): String

+getDepartDate(): String

+getSeatType(): String

+getNoSeats(): int

+toString(): String

+equals(obj: Object): boolean

Seat

--------------------

~seatType: String

~noSeats: int

--------------------

+Seat()

+toString(): String

Limousine

------------------------

--------------------------

+Limousine(\_taxiId: int, \_isAvailable: boolean, \_departDate: String, \_departTime: String, \_seatType: String, \_noSeats: int)

+equals (o: Object): boolean

+toString(): String

+calculateTotalFee(distance: double, taxiFee: double): double

SUV

------------------------

--------------------------

+SUV(\_taxiId: int, \_isAvailable: boolean, \_departDate: String, \_departTime: String, \_seatType: String, \_noSeats: int)

+equals (o: Object): boolean

+toString(): String

+calculateTotalFee(distance: double, taxiFee: double): double

Sedan

---------------------------------

----------------------------------

+Sedan(\_taxiId: int, \_isAvailable: boolean, \_departDate: String, \_departTime: String, \_seatType: String, \_noSeats: int)

+equals (o: Object): boolean

+toString(): String

+calculateTotalFee(distance: double, taxiFee: double): double

*Credentials*

------------------------------------------------------------------------------

#email: String

#password: String

------------------------------------------------------------------------------

#Credentials()

+setEmail(\_email: String): void

+setPassword(\_password: String): void

+getEmail(): String

+getPassword(): String

+toString(): String

+equals (obj: Object): boolean

*+register(): void*

+login(): void

UserCredentials

-----------------------------------------

-paidTaxiId: int

-totalPrice: double

-----------------------------------------

+setPaidTaxiId(\_paidTaxiId: int): void

+setTotalPrice(\_totalPrice: double): void

+getPaidTaxiId(): int

+getTotalPrice(): double

+register(): void

+login(): void

+toString(): String

+equals(obj: Object): boolean

AdminCredentials

------------------------------

------------------------------

+register(): void

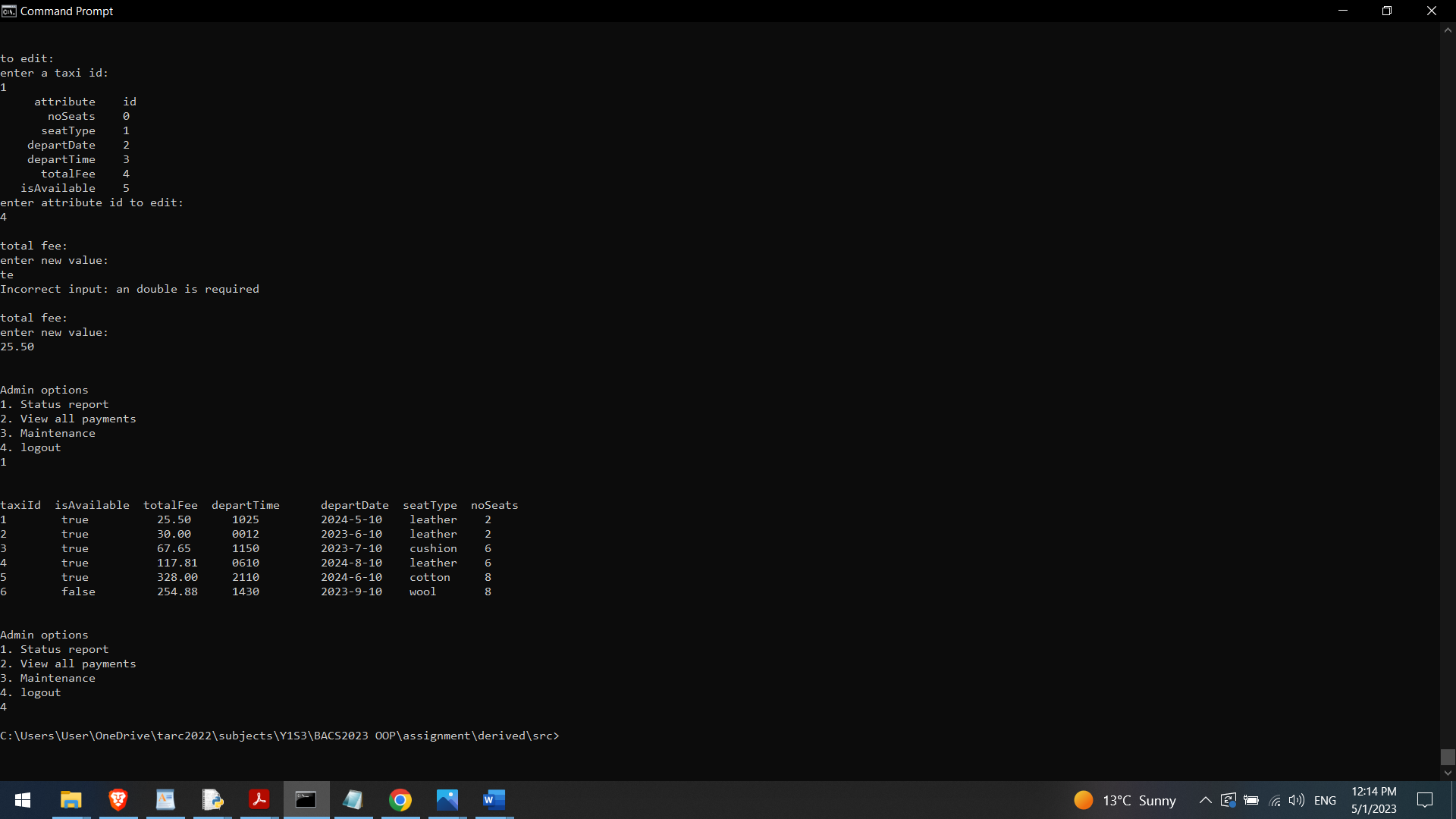
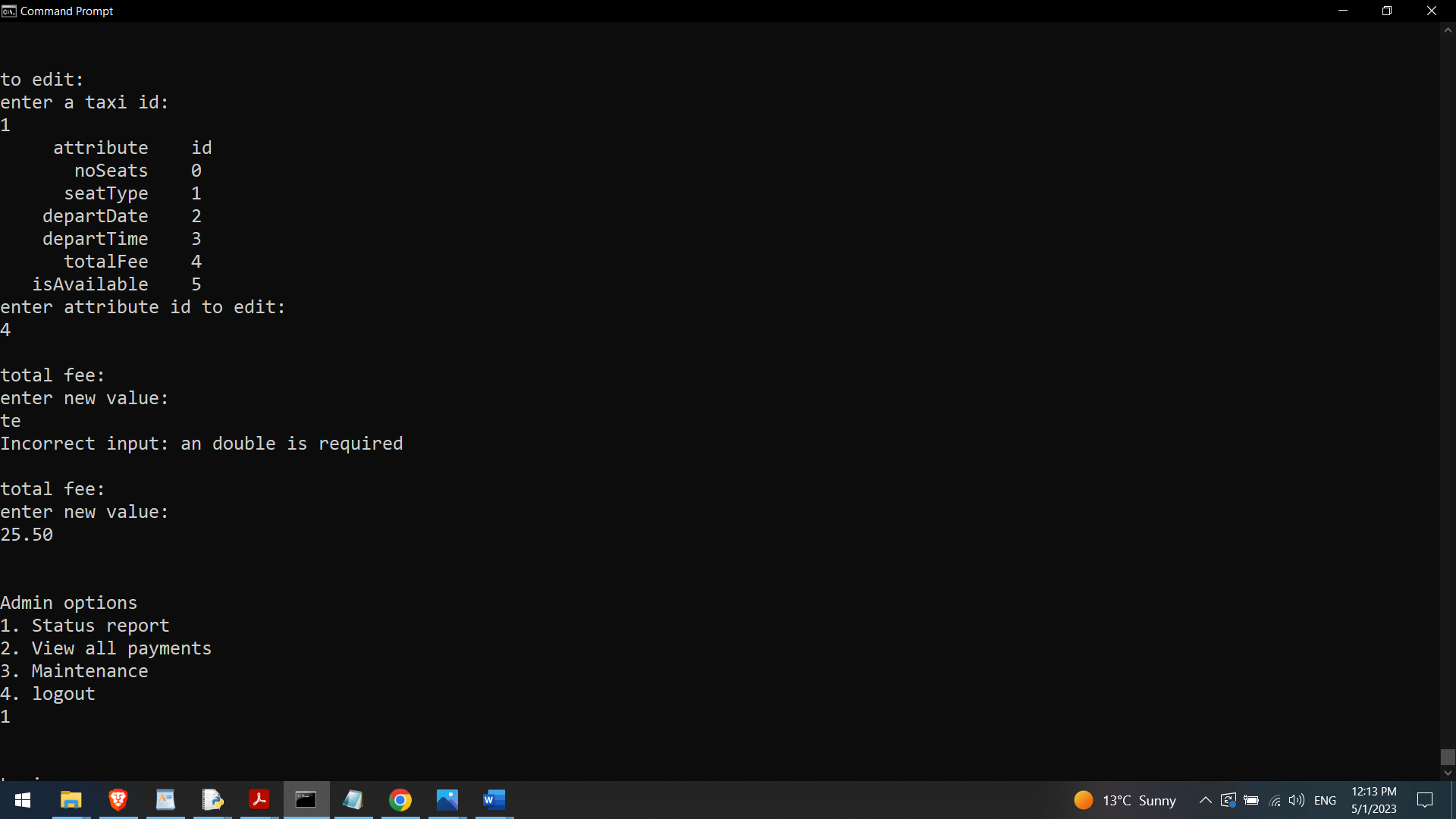
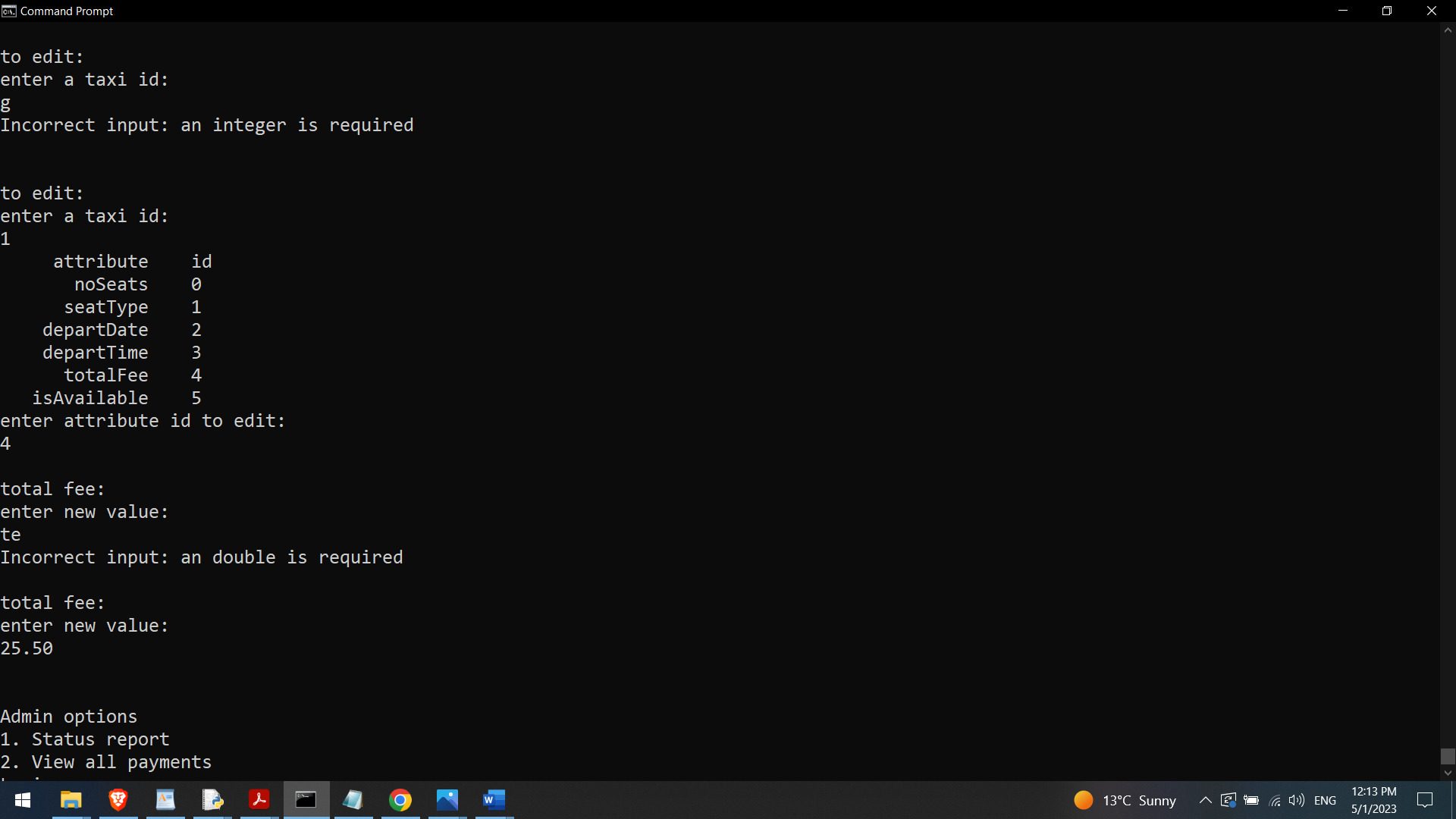
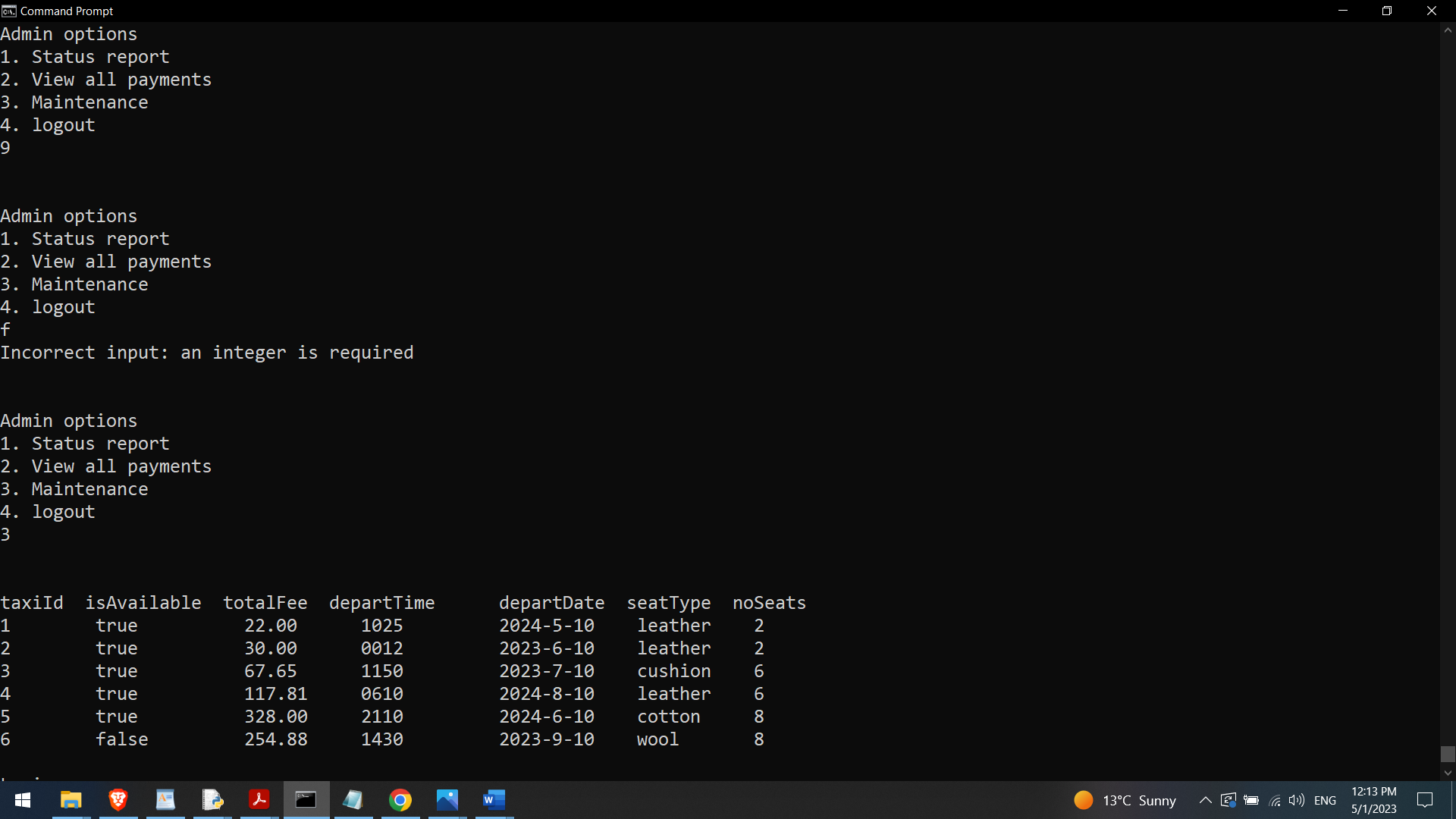
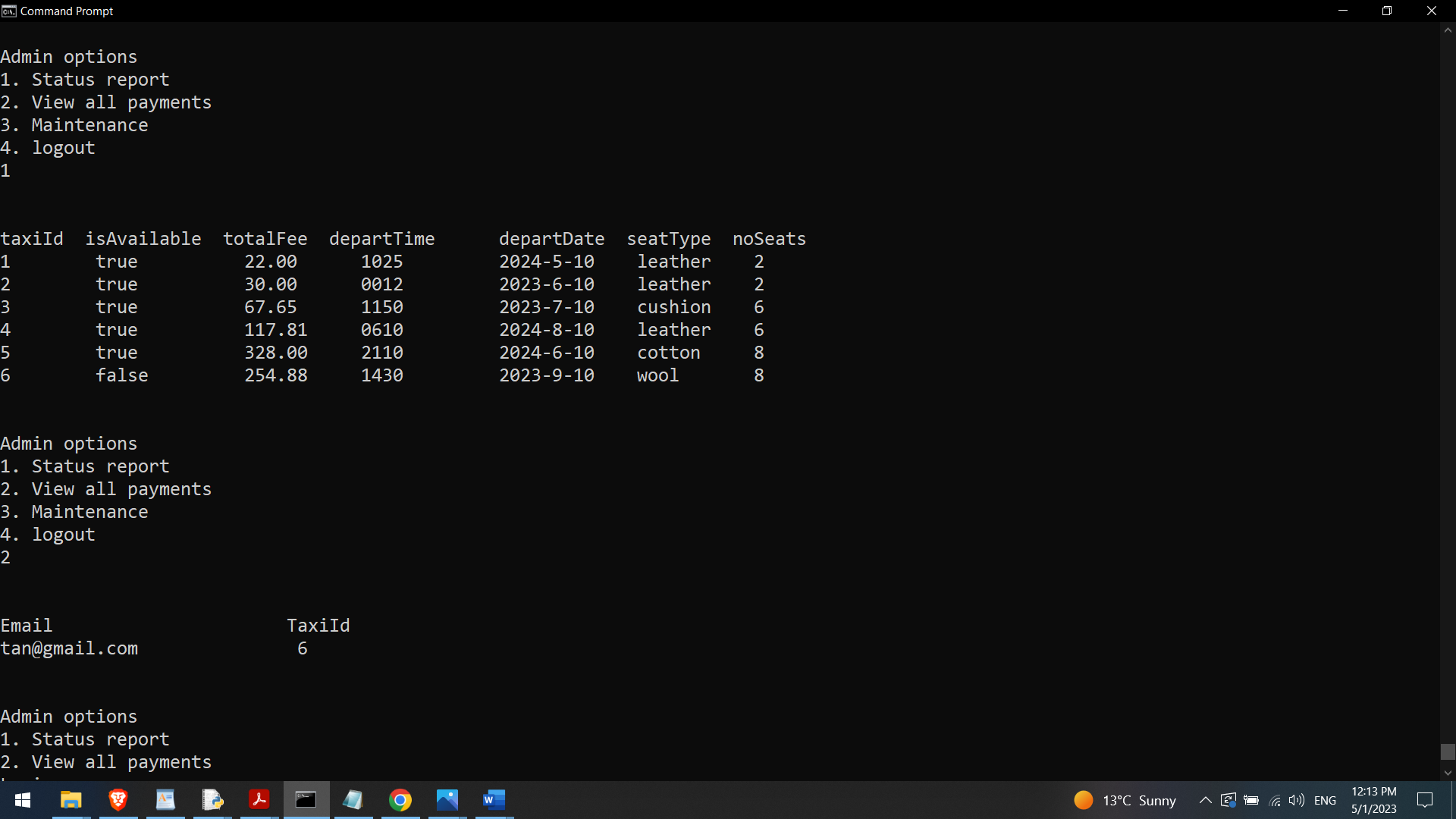
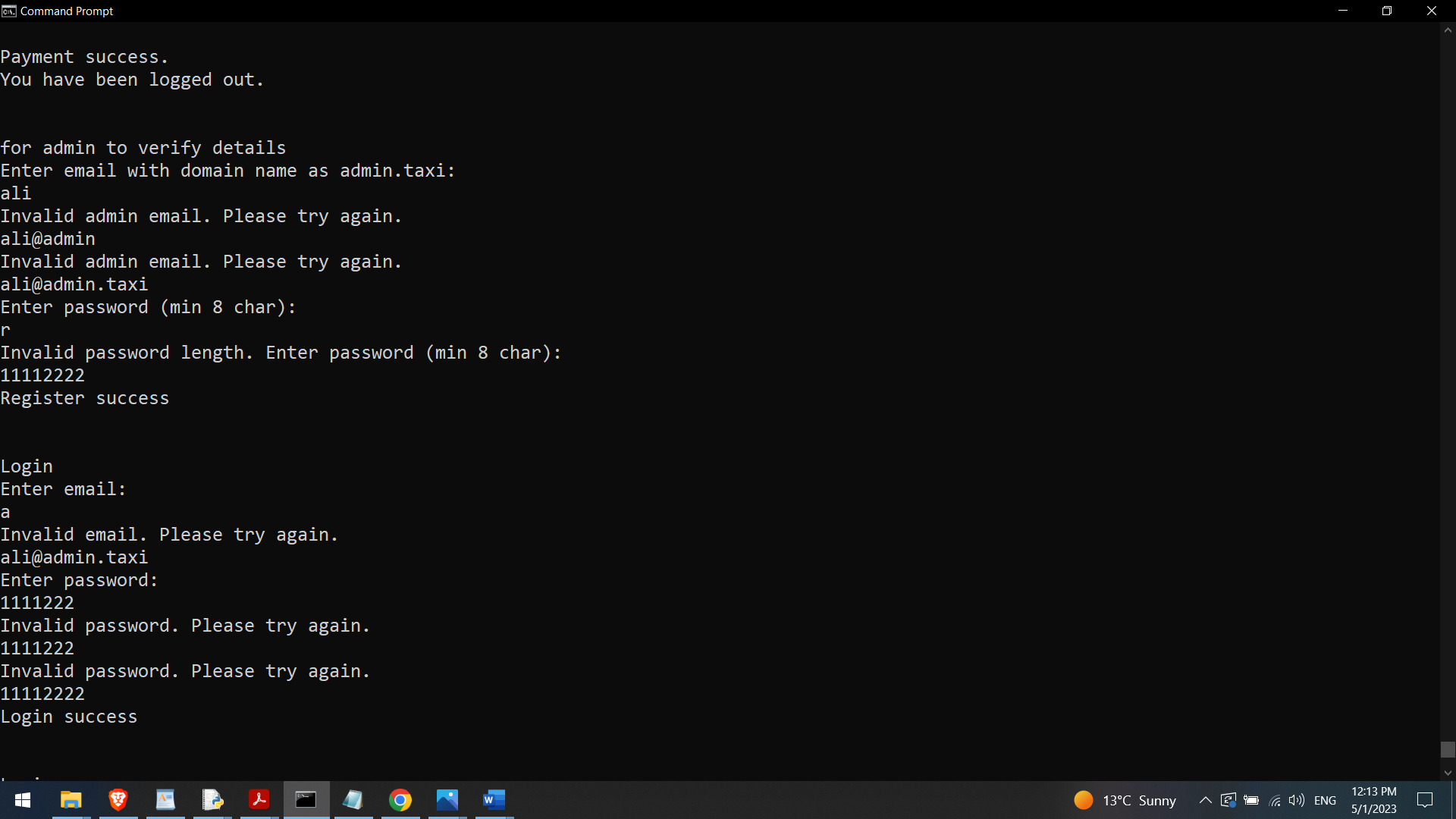
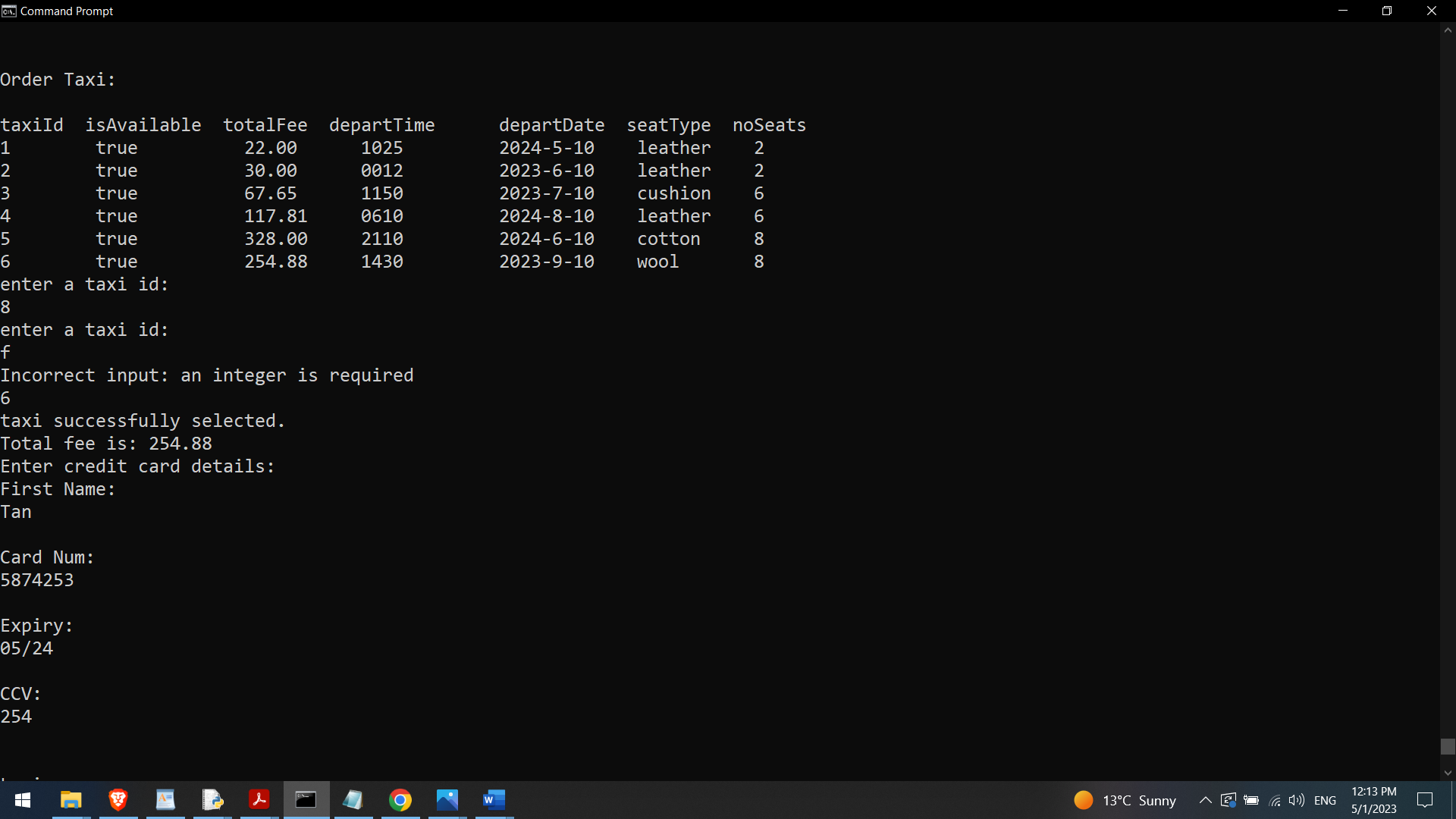
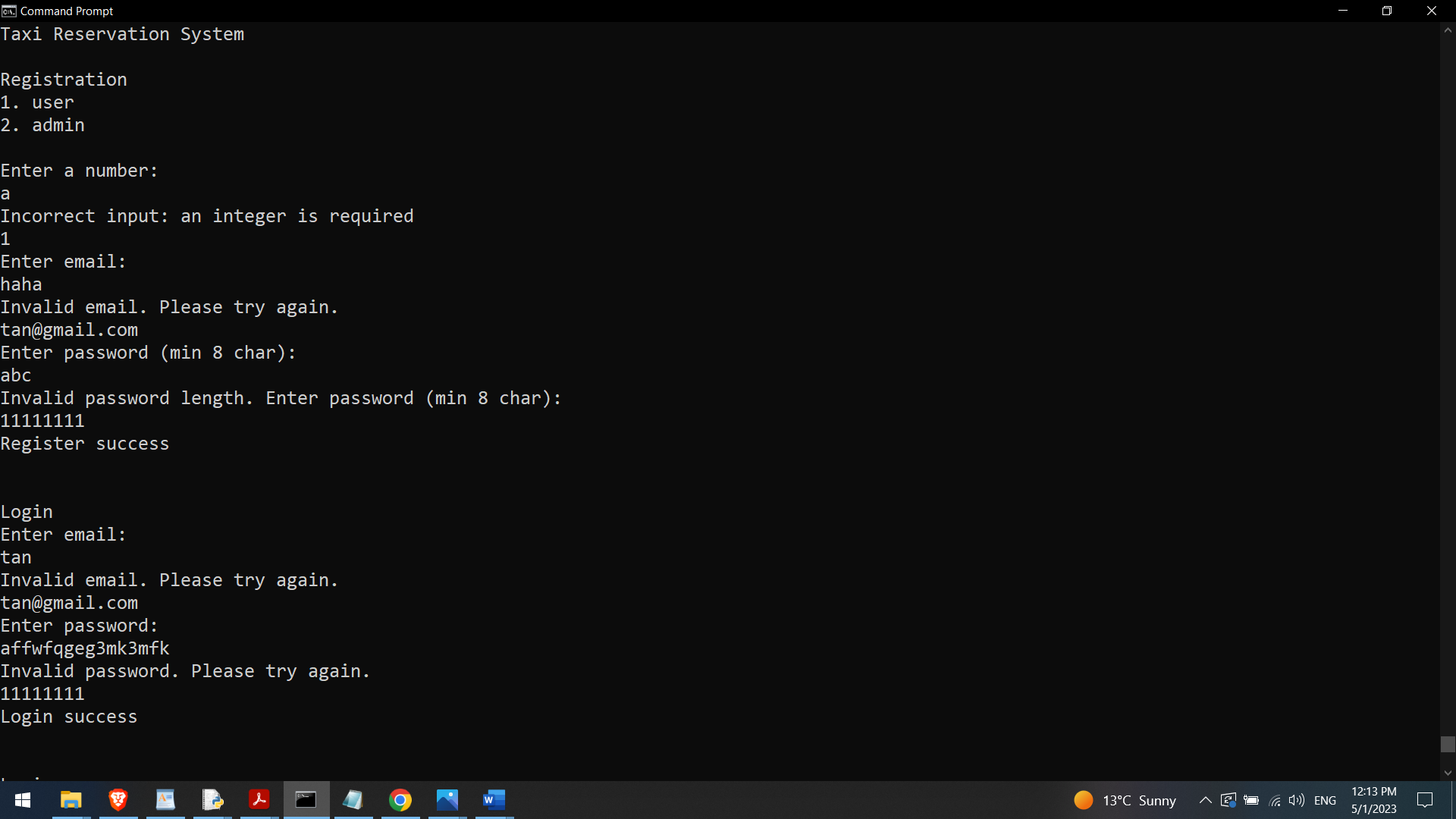
+login(): void

+toString(): String

+equals(obj: Object): boolean

1. Description of your team’s assignment idea.

Taxi reservation system allow users to view, book taxi rides and pay using credit card. Admin can verify who booked tickets and is given access to status reports and can maintain system by updating ride details.



**Part 2**

With reference to your own tasks / modules, write an individual report that is not more than 3 pages (including code segments of your assignment) to describe the concepts of Object-Oriented Programming that you applied in your work.

Taxi class inherits an interface (TaxiInterface) to establish weak is-a relationship between Taxi and TaxiInterface. Aggregation is used to represent ownership relationship between 2 classes. For example, Taxi is the aggregating class while Seat is the aggregated class. Seat is not exclusively owned by Taxi as different taxis can share the same seat type and number. Composition is used to represent exclusive ownership relationship between 2 classes: Taxi and Time. Time is exclusively owned by Taxi as each taxi has unique depart time and date to avoid traffic and schedule clash.

Taxi is inherited by Sedan, SUV, and Limousine to allow polymorphism as the calculateTotalFee method is overriden in each inherited class so each inherited class display different behavior. Polymorphism allows calculateTotalFee method name to be used throughout the class family, with each class implementing the method in a way that is appropriate to its own purpose (different formula to calculate fee for each class).

Limousine

public double calculateTotalFee(double distance, double taxiFee)

{

return distance\*taxiFee + taxiFee\*3;

}

SUV

public double calculateTotalFee(double distance, double taxiFee)

{

return distance\*taxiFee + taxiFee\*2;

}

Sedan

public double calculateTotalFee(double distance, double taxiFee)

{

return distance\*taxiFee + taxiFee;

}

Credentials is an abstract class containing abstract method like

public abstract void register();

as do not know how to implement register yet. Register implementation is specific to the inherited classes: AdminCredentials and UserCredentials. During registration input, admin email must end with @admin.taxi while user email can end with any domain.

AdminCredentials

public void register()

{

System.out.println("Enter email with domain name as admin.taxi: ");

Scanner scan = new Scanner(System.in);

this.email = scan.nextLine();

while (!this.email.matches("\\b[A-Za-z0-9.\_%+-]+@admin\\.taxi\\b"))

{

System.out.println("Invalid admin email. Please try again.");

this.email = scan.nextLine();

}

System.out.println("Enter password (min 8 char): ");

this.password = scan.nextLine();

while (this.password.length()<8)

{

System.out.println("Invalid password length. Enter password (min 8 char): ");

this.password = scan.nextLine();

}

System.out.println("Register success");

}

UserCredentials

{

System.out.println("Enter email: ");

Scanner scan = new Scanner(System.in);

this.email = scan.nextLine();

while (!this.email.contains(Character.toString('@')))

{

System.out.println("Invalid email. Please try again.");

this.email = scan.nextLine();

}

System.out.println("Enter password (min 8 char): ");

this.password = scan.nextLine();

while (this.password.length()<8)

{

System.out.println("Invalid password length. Enter password (min 8 char): ");

this.password = scan.nextLine();

}

System.out.println("Register success");

}