Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS			
Programme title	Higher National Diploma in Computing		
Assessor	Mr.Sachintha Sampath	**Internal Verifier**	
Unit(s)			
Assignment title			
Student's name	K.A Bhashitha Maduwantha		
List which assessment criteria the Assessor has awarded.	**Pass**	**Merit**	**Distinction**

INTERNAL VERIFIER CHECKLIST		
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N	
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N	
Has the work been assessed accurately?	Y/N	
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	Y/N Y/N Y/N Y/N	
Does the assessment decision need amending?	Y/N	

Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)			

		Date	

Confirm action completed			
Remedial action taken Give details:			
Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

BTEC

ESOFT
Shaping Lives, Creating Futures.

Higher Nationals - Summative Assignment Feedback Form

Student Name/ID	K.A Bhashitha Maduwantha		
Unit Title			
Assignment Number		**Assessor**	
Submission Date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback:

LO1. Define basic algorithms to carry out an operation and outline the proce s of programming an application.

Pass, Merit & Distinction Descripts **P1 M1** ☐ ☐ D1 ☐

LO2. Explain the characteristics of procedural, object-orientated and event-d iven ramming, conduct ar analysis Integrated Development Environment (IDE).

Pass, Merit & Distinction Descripts **P2** ☐ **M2** ☐ **D2** ☐

LO3. Implement basic algorithms in code using an IDE.

Pass, Merit & Distinction Descripts **P3 M3** ☐ ☐ **D3** ☐

LO4. Determine the debugging process and explain the importance of a codi g standard.

Pass, Merit & Distinction Descripts **P4** ☐ **P5** ☐ **M4** ☐ **D4** ☐

Grade:	**Assessor Signature:**	**Date:**

Resubmission Feedback:

Grade: Assessor	**Signature: Date:**	

Internal Verifier's Comments:

Signature & Date:

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Assignment Feedback

Formative Feedback: Assessor to Student
Action Plan
Summative feedback

Feedback: Student to Assessor			
Assessor signature		Date	
Student signature		Date	

Pearson Higher Nationals in

Computing

Unit 01: Programming
Assignment 01

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing**. Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each pag**e. This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.

3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.

10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct way. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of the Edexcel UK.
3. I know what the consequences will be if I plagiaries or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspects of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Edexcel UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the attached.

Student's Signature:
(***Provide E-mail ID***)

Date:
(***Provide Submission Date***)

Higher National Diploma in Computing

Assignment Brief

Student Name /ID Number	K.A Bhashitha Maduwantha
Unit Number and Title	**Unit 01: Programming**
Academic Year	2021/22
Unit Tutor	Mr.Sachitha Sampath
Assignment Title	**Design &Implement a GUI based system using a suitable Integrated Development Environment**
Issue Date	04/09/2022
Submission Date	20/11/2022
IV Name & Date	

Submission Format
This submission will have 3 components 1. Written Report This submission is in the form of an individual written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system. (**The recommended word count is 1,500–2,000 words for the report excluding annexures)** 2. Implemented System (Software) The student should submit a GUI based system developed using an IDE. The system should connect with a backend database and should have at least 5 different forms and suitable functionality including insert, edit and delete of main entities and transaction processing. 3. Presentation With the submitted system student should do a presentation to demonstrate the system that was developed. Time allocated is 10 to 15 min. Student may use 5 to 10 PowerPoint slides while doing the presentation, but live demonstration of the system is required. Evaluator will also check the ability to modify and debug the system using the IDE.

Unit Learning Outcomes:	
	LO1. Define basic algorithms to carry out an operation and outline the process of programming an application. **LO2.** Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE). **LO3.** Implement basic algorithms in code using an IDE. **LO4.** Determine the debugging process and explain the importance of a coding standard

Assignment Brief and Guidance:

Activity 1

A. The Fibonacci numbers are the numbers in the following integer sequence. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation.

$$F_n = F_{n-1} + F_{n-2}$$

B. Factorial of a non-negative integer, is multiplication of all integers smaller than or equal to n. For example, factorial of 6 is 6*5*4*3*2*1 which is 720.

$$n! = n * (n - 1) * \ldots\ldots 1$$

Define what an algorithm is and outline the characteristics of a good algorithm. Write the algorithms to display the Fibonacci series and the factorial value for a given number using Pseudo code. Determine the steps involved in the process of writing and executing a program.
Take a sample number and dry run the above two algorithms. Show the outputs at the end of each iteration and the final output. Examine what Big-O notation is and explain its role in evaluating efficiencies of algorithms. Write the Python program code for the above two algorithms and critically evaluate their efficiencies using Big-O notation.

Activity 2

2.1 Explain what is meant by a Programming Paradigm and the main characteristics of Procedural, Object oriented and Event-driven paradigms and the relationships among them. Write small snippets of code as example for the above three programming paradigms using a suitable programming language(s). you also need to critically evaluate the code samples that you have given above in relation to their structure and the unique characteristics.

Activity 3 and Activity 4 are based on the following Scenario.

Ayubo Drive is the transport arm of Ayubo Leisure (Pvt) Ltd, an emerging travel & tour company in Sri Lanka. It owns a fleet of vehicles ranging from cars, SUVs to vans.

The vehicles that it owns are hired or rented with or without a driver. The tariffs are based on the vehicle type. Some of the vehicle types that it operates are, small car, sedan car, SVUs, Jeep (WD), 7seater van and Commuter van. New vehicle types are to be added in the future.

Vehicle rent and hire options are described below.

1. Rent (With or without driver) – For each type of vehicle rates are given per day, per week and per month. Rate for a driver also given per day. Depending on the rent period the total rent amount needs to be calculated. For example: if a vehicle is rented for 10 days with a driver, total amount to be calculated as follows:

   Total rent = weeklyRent x 1 + dailyRent x 3 + dailyDriverCost x 10

2. Hire (with driver only) – These are based on packages such as airport drop, airport pickup, 100km per day package, 200km per day package etc. Standard rates are defined for a package type of a vehicle typeif that is applicable for that type of vehicle.For each package maximum km limit and maximum number of hours arealso defined. Extra km rate is also defined which is applicable if they run beyond the allocated km limit for the tour. For day tours if they exceed max hour limit,a waiting charge is applicable for extra hours. Driver overnight rate and vehicle night park rate also defined which is applicable for each night when the vehicle is hired for 2 or more days.

Activity 3

Function 1: **Rent calculation.**
Return the total rent_value when vehicle_no, rented_date, return_date, with_driver parameters are sent in. with_driver parameter is set to true or false depending whether the vehicle is rented with or without driver.

Function 2: **Day tour - hire calculation.**
Calculate total hire_value when vehicle_no, package_type, start_time, end_time, start_km_reading, end_km_reading parameters are sent in. Should return base_hire_charge, waiting_charge and extra_km_charge as output parameters.

Function 3: **Long tour - hire calculation.**
Calculate total hire_value when vehicle_no, package_type, start_date, end_date, start_km_reading, end_km_reading parameters are sent in. Should return base_hire_charge, overnight_stay_charge and extra_km_charge as output parameters.

Write suable algorithms for vehicle tariff calculation for rents and hires. Ideally 3 functions should be developed for this purpose as above. Use the visual studio IDE (using C#.net) to Implement the above algorithms and design the suitable database structure for keeping the tariffs for vehicle types and different packages which must be used for implementing the above functions.

Analyze the features of an Integrated Development Environment (IDE) and explain how those features help in application development. Evaluate the use of the Visual StudioIDE for your application development contrasted with not using an IDE.

Activity 4

4.1 Design and build a small system to calculate vehicle hire amounts and record them in a database for customer billing and management reporting forAyubo drive. This includes the completing the database design started in 3.2 and implementing one or more GUIs for vehicle, vehicle type, and package add/edit/delete functions. It essentially requires an interface for hire calculation and recording function described above. Generating customer reports and customer invoices are not required for this course work.

4.2 Explain debugging process and the features available in Visual studio IDE for debugging your code more easily. Evaluate how you used the debugging process to develop more secure, robust application with examples.

4.3 Outline the coding standards you have used in your application development. Critically evaluate why a coding standard is necessary for the team as well as for the individual.

Grading Rubric

Grading Criteria	Achieved	Feedback
LO1 Define basic algorithms to carry out an operation and outline the process of programming an application.		
P1Provide a definition of what an algorithm is and outline the process in building an application.		
M1Determine the steps taken from writing code to execution.		
D1 Evaluate the implementation of an algorithm in a suitable language. Evaluate the relationship between the written algorithm and the code variant		
LO2 Explain the characteristics of procedural, objectorientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE)		
P2Give explanations of what procedural, objectorientated, and eventdriven paradigms are; their characteristics and the relationship between them.		
M2 Compare and contrast the procedural, object orientated and event driven paradigms used in given source code of an application		

D2Critically evaluate the source code of an application which implements the programming paradigms, in terms of the code structure and characteristics.		
LO3Implement basic algorithms in code using an IDE.		
P3Write a program that implements an algorithm using an IDE.		
M3Use the IDE to manage the development process of the program.		
D3Evaluate the use of an IDE for development of applications contrasted with not using an IDE.		
LO4 Determine the debugging process and explain the importance of a coding standard		
P4Explain the debugging process and explain the debugging facilities available in the IDE.		
P5Outline the coding standard you have used in your code.		
M4Evaluate how the debugging process can be used to help develop more secure, robust applications.		

D4 Critically evaluate why a coding standard is necessary in a team as well as for the individual.		

Acknowledgment
I want to express my sincere gratitude to my professor Mr. Sachintha, who provided me with the fantastic opportunity to complete this excellent project on the topic of programming. They also enabled me to conduct a great deal of research, which allowed me to learn a lot of new information. Second, I'd want to thank my parents and friends who greatly contributed to the completion of this project within the allotted time. Finally, the learner should thank ESOFT Metro Campus for providing him with the information and resources he needed to complete the assignments.

Table of Contents

List of Figures

List of Tables

What is algorithm

A method for completing a computation or solving a problem is called an algorithm. Algorithms function as a precise sequence of instructions that guide hardware- or software-based routines through a series of prescribed actions step by step.

All aspects of information technology employ algorithms extensively. A simple technique that resolves a recurring issue is typically referred to as an algorithm in mathematics and computer science. Algorithms also serve as guidelines for data processing,

In addition to a starting input, algorithms also require a set of instructions. The input, which can be expressed in words or figures, contains the initial information required to make judgments. The supplied data is processed through a series of calculations, including arithmetic and decision-making procedures. An algorithm's final step, known as the output, typically results in more data.

A search algorithm might, for instance, take a search query as input and process it through a series of directives for searching a database for things that match the query. As automation adheres to a set of rules to execute tasks, automation software serves as another illustration of an algorithm. (*What is algorithm: Introduction to algorithms* 2022)

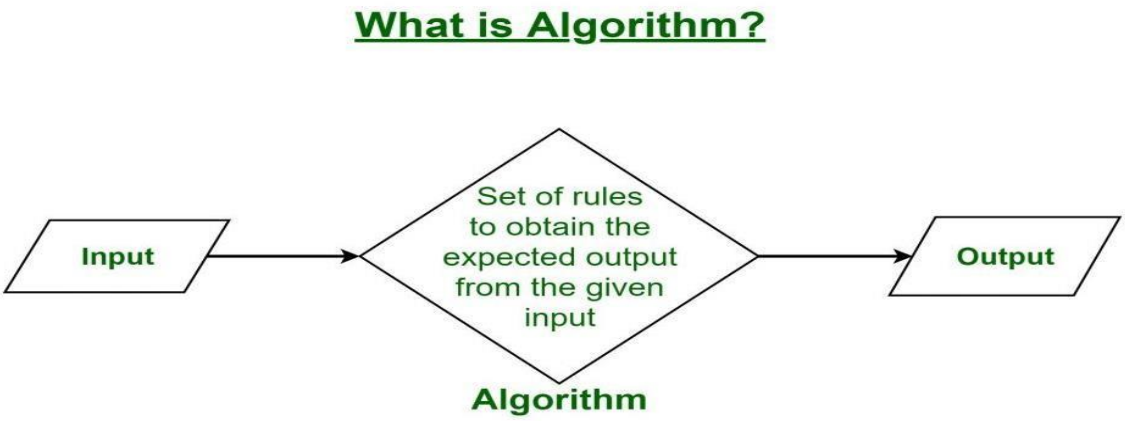


Figure 1- Algorithm

Characteristics of an Algorithm

Every algorithm should adhere to a certain set of traits. There are five distinct features that cover numerous algorithmic topics.

- Input specified
  -There may be a lot of inputs
- Output Specified
  -There should be at least one
- Definiteness
  There should be at least one output from it.
- Effectiveness
  It ought to produce Each step must be distinct, exact, and unambiguous.
  There must be no room for
  uncertain
  Fectivne
- Termination
  After a limited number of steps, an algorithm may come to an end.

Types of Algorithms

Algorithms come in a variety of forms, each intended to carry out a particular task. There are numerous sorts of algorithms available in both arithmetic and computer science, just like in real life. In other words, there are typically numerous approaches - involving various processes - to fixing a certain issue. Naturally, each of these algorithms need input in order to produce useful results.

- Greedy Algorithm
  An algorithmic type called a greedy algorithm is frequently employed to address optimization issues. Therefore, such an approach is used if one wants to extract the most in the shortest amount of time or with the fewest resources.

- Dynamic Programming Algorithm

By breaking down difficulties into smaller ones, this programme finds solutions. The outcomes are then saved for use in solving related challenges in the future. An algorithm for dynamic programming works by remembering the outcomes of past runs and applying them to generate new outcomes.

- Divide and Conquer Algorithm

  There are two parts to this typical method. One portion separates a problem into more manageable subproblems. The second section resolves these issues and then combines them to create a solution.

- Recursive Algorithm

  An algorithm that repeats steps until the issue is resolved is called a recursive algorithm. Every time a recursive function is used, recursive algorithms call themselves with a lower value.

- Backtracking Algorithm

  This algorithm addresses a given problem piecemeal, coming up with incremental solutions as it goes. If one is unable to move forward at any point, one backtracks, or reverses direction, in order to start over and discover a different approach to the problem. Therefore, the backtracking method solves a subproblem; if and when it fails to solve the original problem, the final step is undone, and the search for a solution is restarted from the beginning.

- Brute Force Algorithm

  This method iterates through every potential answer to a question while looking. It explores every option before coming up with a decision. A brute force algorithm's temporal complexity is frequently inversely correlated with the input size. Brute force methods are straightforward and reliable but extremely sluggish.

- Sorting Algorithm

Sorting algorithms are used to change the order of data in a data structure depending on a comparison operator. The new order of the elements in the relevant data structure is determined using the comparison operator. (*Most important type of algorithms* 2022)

Pseudo Code

The term "pseudo code" is frequently used in algorithm-based professions such as programming. It is a method that enables the programmer to depict how an algorithm is implemented. It's the fabricated representation of an algorithm, to put it simply. Pseudo codes are frequently used to depict algorithms because they may be understood by programmers with any level of programming experience.

Pseudo code for Fibonacci numbers

Start

Step 01: Input FibNumber

Step 02: x,y=0,1

Step 03: Total=0

Step 04: If tot<= FibNumber then

Step 05: For I in range (0, Fibnumer)

Step 06: Display (tot,end= “ ”)

Step 07: x=y

Step 08: y=total

Step 09: calculate tot=x+y

Step 10: End if

Step 11: Else

Step 12: Display

Step 13: End Else

End

Pseudo code for Factorial numbers

Start

Step 01: Input Number

Step 02: factorial=0

Step 03: If 0> num then

Step 04: Display (“please enter positive number”)

Step 05: End If

Step 06: If num==0 then

Step 07: Display (“the factorial of 0 is 1”)

Step 08: End If

Step 09: Else

Step 10: For I in range (0,num + 1)

Step 11: Calculate factorial = factorial*i

Step 12: Display (“the factorial of”,num, “is”,factorial)

Step 13: End Else

End

Python Coding

pro.py - C:/Users/C TECH/OneDrive/Documents/pro.py (3.10.5)

File Edit Format Run Options Window Help

```
fnumber=int(input("Enter your fibbonacci number:"))
X=0
Y=1
tot=0
if tot<=fibnumber:
   for i in range (0,fibnumber):
      print(tot,end="")
      x=y
      y=tot
      tot=x+y

else:
   print("please enter higher number!")
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/C TECH/OneDrive/Documents/pro.py ==============
Enter your fibbonacci number:
```

Ln: 5 Col: 29

Figure 2-Fibbonacci numbers python coding

pro.py - C:/Users/C TECH/OneDrive/Documents/pro.py (3.10.5)

File Edit Format Run Options Window Help

```
num=int(input("Enter a number:"))
factorial=1
if 0>num:
   print ("please enter possitive number")
if num** 0:
   print("the factorial of 0 is i")
else:
   for i in range(1,num+1):
      factorial+factorial*1
      print("the factorial of",num,"if",factorial)
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/C TECH/OneDrive/Documents/pro.py ==============
Enter a number:
```

Ln: 5 Col: 0

Figure 3- Factorial numbers python coding

Linear Search

Linear search and binary search are two common search techniques.

Sequential search algorithm is another name for linear search. It is the most straightforward search algorithm. In a linear search, we merely go through the list in its whole and match each element with the object whose location needs to be determined. The algorithm returns the item's location if a match is made otherwise, it returns null. It is frequently used to search for a certain element in an unordered list, or a list where the entries are not sorted. The linear search's worstcase time complexity is o(n). (*Linear Search - javatpoint*)

Pseudo code for linear search

```
procedure LINEAR_SEARCH (array, key)

   for each item in the array
      if match element == key
         return element's index
      end if
   end for

end procedure
```

Figure 4- pseudo code for linear search

Binary Search

A sorted array can be searched using the binary search algorithm by continually halving the search interval. Utilizing the knowledge that the array is sorted, binary search attempts to minimise the time complexity to O (Log n).

Binary search algorithm

- Start your search using the middle member in the entire array.
- Return an index of the search key if the value of the search key equals the item.
- Alternately, if the search key's value is smaller than the item in the interval's midpoint, limit the interval to its lower half. Otherwise, cut it off at the top half.
- until the value is found or the interval is empty, check the second point repeatedly.

The binary search algorithms' pseudocode should resemble this.

```
Procedure binary_search
   A ← sorted array
   n ← size of array
   x ← value to be searched

   Set lowerBound = 1
   Set upperBound = n

   while x not found
      if upperBound < lowerBound
         EXIT: x does not exists.

      set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

      if A[midPoint] < x
         set lowerBound = midPoint + 1

      if A[midPoint] > x
         set upperBound = midPoint - 1

      if A[midPoint] = x
         EXIT: x found at location midPoint
   end while

end procedure
```

Figure 5- Binary search algorithms pseudo code

How binary search works

The target array must be sorted in order for a binary search to function. We will use a visual illustration to teach us how binary search works. Let's say that we need to use binary search to find the value 31 in the following sorted array.

First, we will use this algorithm to find

Mid = Low + (High-Low) /2

As you can see, 0 + (9 - 0) / 2 = 4. (Integer value of 4.5). 4 is therefore the middle of the array.

Now we contrast the value that is being searched, which is 31, with the value that is stored at location 4. The value at position 4 is found to be 27, which does not match. We also know that the target value must be in the top part of the array because the value is larger than 27, and the array is sorted.

We adjust our low to mid + 1, then recalculate the new mid value.

Low = Mid + 1

Mid = Low + (High-Low) /2

Our new mid is now seven. We contrast our desired value of 31 with the number kept at location 7

The value kept at location 7 does not match what we are looking for; rather, it is greater. Therefore, from this place, the value must be in the lower section.

Therefore, we recalculate the mid. This time, five.

We contrast our desired value with the value kept at location 5. We discover that it matches.

We come to the conclusion that the goal value, 31, is kept at position 5.

Programming Process

Every programming project includes developing a solution to a problem. The issues can be as significant in terms of science or national importance or as unimportant as quelling personal

ennui. Consider the information in this part as a general overview of what you should accomplish before venturing into the world of programming. It offers one method for resolving such issues.

Steps of programming process

- Identify the problem

  Finding the inputs, outputs, and actions that result in the inputs and outputs is what this stage entails. Requirements and specifications are the two stages of identifying a solution.

- Design a solution

  The next stage is to figure out exactly how you're going to translate that specification into a usable programme once you've determined what needs to be done to address your problem and stated what shape your solution will take. The hardest task is always this one.

- Write the program

  The task of programming then entails explaining your idea to the computer, or teaching it how to solve problems. Coding, Compiling and Debugging are the usually three stages to writing a program.

- Check the solution

  Testing your creation to ensure that it performs as intended is the last stage in the great programming process. Unfortunately, even though the compiler has verified that your programme is appropriately written, it is unable to verify whether the solution you have provided genuinely resolves your original problem. (*The programming process*)

Big O notation

One of the most fundamental methods used by computer scientists to evaluate the cost of an algorithm is the Big O notation. Software engineers would do well to comprehend it thoroughly as well. Big O notation is a type of mathematical notation that expresses how a function limits itself when the argument tends to zero or infinity.

Big O Notation provides an algorithm's worst-case complexity or upper-bound runtime. It evaluates and categorises algorithms based on how much space or time they require to run.

When an algorithm's input trends toward a particular or limiting value, this is referred to as having a high time complexity. It determines how long it takes an algorithm to run each code statement.

Big O Notation is a technique for expressing an algorithm's temporal complexity. As the input increases, the amount of time needed to perform an algorithm is calculated. It determines an algorithm's worst-case temporal complexity, to put it another way. (Huang, *What is big O notation explained: Space and time complexity* 2022)

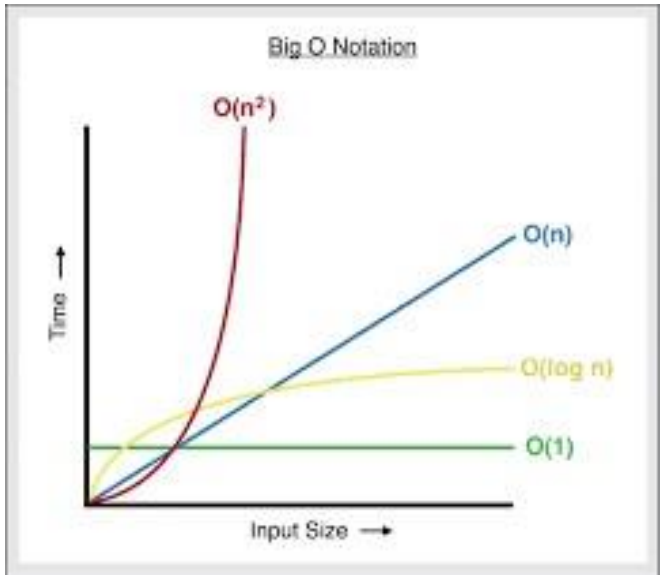


Figure 6- Big O Notation

Programming Paradigms

A paradigm is a strategy for tackling a challenge or completing a task. Programming paradigms are ways to solve problems using programming languages, or you might say they are ways to use tools and techniques that are already at our disposal to solve problems in a certain way. There are many well-known programming languages, but when they are used, they always need to adhere

to a philosophy or strategy called a paradigm. There are numerous paradigms to meet every need in addition to different programming languages.

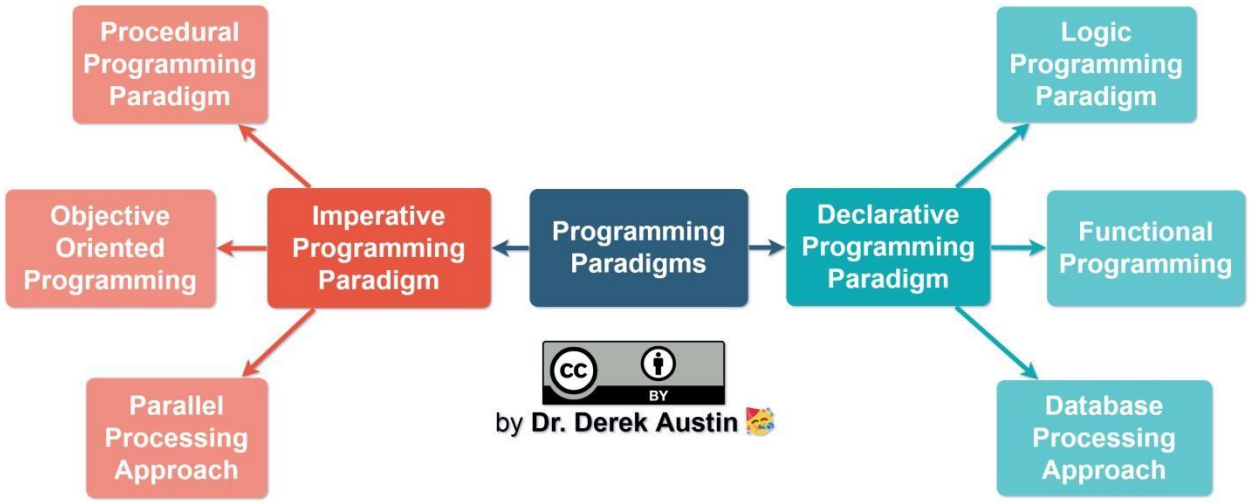


Figure 7- programming Paradigm

Imperative Programming Paradigm

It is one of the earliest paradigms in programming. It closely resembles machine architecture. On Von Neumann architecture is where it is based. By using assignment statements, it alters the program's state. By altering states, it completes tasks step by step. The method of achieving the goal is the major concern. The paradigm is made up of a number of statements, and once each one is executed, a result is saved.

```
// Imperative Programming

let array = [1, 2, 3, 4, 5, 6]

var evenNumbers: [Int] = []

for i in 0..<array.count {
    if array[i] % 2 == 0 {
        evenNumbers.append(array[i])
    }
}
```

Source: https://medium.com/@vincentbacalso/imperative-vs-declarative-programming-f886d3b65595

Figure 8- imperative Programming

Advantages ○ simple to read ○ Comparatively simple to learn. ○ The conceptual model is relatively simple to comprehend for novices.

- Applications-specific characteristics may be taken into consideration.

Disadvantages ○ Code soon grows confusingly long and complex. ○ More potential for errors when editing.

- Because of system-oriented programming, it is impossible to design new applications.
- Extension and optimization are more challenging.

Imperative programming can be categorized into three main groups.

- Procedural Programming Paradigm

  This paradigm places a strong emphasis on the underpinning machine model's procedure. The imperative approach and the procedural approach are identical. As a result of its capacity to reuse code, it was a huge help when it was first put to use.

- Object Oriented Paradigm

  The program is written as a set of communication-oriented classes and objects. The lowest and most fundamental entity is an object, and only objects are used for all computations. Data is prioritized over procedure. It can deal with practically any situation that arises in real life nowadays.

- Parallel Processing Approach

  The division of program instructions among several processors is known as parallel processing. A parallel processing system has a large number of processors with the goal of dividing a program's execution time. This strategy appears to be a divide and conquer strategy. Examples include NESL, which is among the oldest, and C/C++, which also allows due to several library functions.

Declarative Programming Paradigm

The categories are Logic, Functional, and Database. Declarative programming is a method of creating programs in computer science that expresses computing theory without discussing its control flow. Programs are frequently seen as theories of some logic. It might make it easier to write parallel programs. The emphasis is on what must be done, not how it should be done, and on what the code itself is accomplishing. It simply states the desired outcome without specifying how it was achieved. The only distinction between declarative (what to do) and imperative (how to do) programming paradigms is this.

```
class Button extends React.Component{
  this.state = { color: 'red' }
  handleChange = () => {
    const color = this.state.color === 'red' ? 'blue' : 'red';
    this.setState({ color });
  }
  render() {
    return (<div>
      <button
         className=`btn ${this.state.color}`
         onClick={this.handleChange}>
      </button>
    </div>);
  }
}
```

Figure 9- Declarative programme Paradigm

Advantages ○ Short and

effective code.

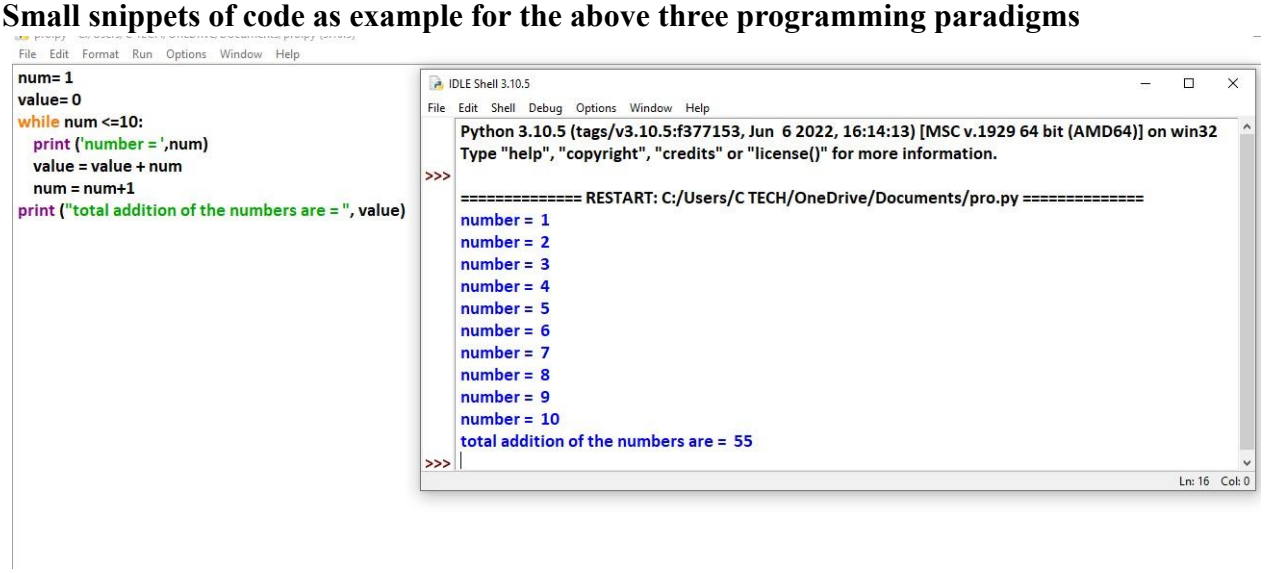
- Can be put into practice using techniques that weren't yet recognized when programming.
- Simple optimization because algorithmic control over implementation.
- Independent of application development, maintenance is possible.

Disadvantages ○ For outsiders, understanding can occasionally be challenging.

- Based on a strange conception of humanity.

(*Introduction of programming paradigms* 2022)

Small snippets of code as example for the above three programming paradigms



```
File  Edit  Format  Run  Options  Window  Help
class dog:
  species = "animal"
  def __init__(self,name,age):
    self.name = name
    self.age = age

Blacky = dog("Blacky",4)
Snowy = dog ("Snowy",6)
print ("Blcky is a {}".format(Blacky.__class__.species))
print ("Snowy is also a {}".format(Snowy.__class__.species))
print ("{} is {} months old".format (Blacky.name,Blacky.age))
print ("{} is {} months old".format (Snowy.name,Snowy.age))
```

```
IDLE Shell 3.10.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/C TECH/OneDrive/Documents/pro.py ==============
Blcky is a animal
Snowy is also a animal
Blacky is 4 months old
Snowy is 6 months old
>>>
```

Figure 10-Python code

Table 1-Declarative programming paradigm

Event driven method	Procedural programming method	Object oriented programming
Enables easy addition of new features to the program.	Due to its lengthy code, adding new functions is possible but slightly more difficult.	Anywhere in the code, data as objects can be added, removed, or called again.
one of the simplest programming languages	The ideal general-purpose programming approach	programming approach that is a little difficult

The code merely provides a complete explanation of the application's logic.	Understanding the code makes it simpler to comprehend how the application functions.	Most sections will use classes, and knowing which ones to use will assist programmers grasp dataflow and reasoning.

Algorithm For rent calculator

```
Start Long Tour calculator
   Enter Start date
   Enter End date
   Enter Day charge
   Enter Driver cost per day
   Enter Start K/M
   Enter End K/M
   Enter Extra K/M Cost

Total days = (End date Start Date)
Charge= Day charge * Total days
Driver cost = Total days * Driver cost per day
Total K/M= (End K/M- Start K/M)

IF(Driver is selected)
    Total charge = Charge + Driver cost
    Display Total charge
Else
    Display Total charge
End IF

End Long tour caleulator
```

Figure 11- Algorithm for rent calculation

The aforementioned algorithm demonstrates how the code's implementation will calculate the customer's rent. The application will then display the rent, total hired days, total hired months, total hired weeks, and the rent when the process of calculating the rent has been completed.

Algorithm for day rent calculation

```
Start Long Tour calculator
   Enter Start date
   Enter End date
   Enter Day charge
   Enter Driver cost per day
   Enter Start K/M
   Enter End K/M
   Enter Extra K/M Cost

Total days = (End date Start Date)
Charge= Day charge * Total days
Driver cost = Total days * Driver cost per day
Total K/M= (End K/M- Start K/M)

IF(Driver is selected)
    Total charge = Charge + Driver cost
    Display Total charge
Else
    Display Total charge
End IF

End Long tour caleulator
```

Figure 12- Algorithm for day rent calculation

The long tour calculator will operate according to the aforementioned algorithm. This algorithm describes how the total number of days will be determined, how the day charge will be multiplied by the total number of days, and how the charge will be determined. And if a driver is chosen, their fee will be doubled by the number of days they are employed.

Algorithms implementation in C#

1.Hire Long tour

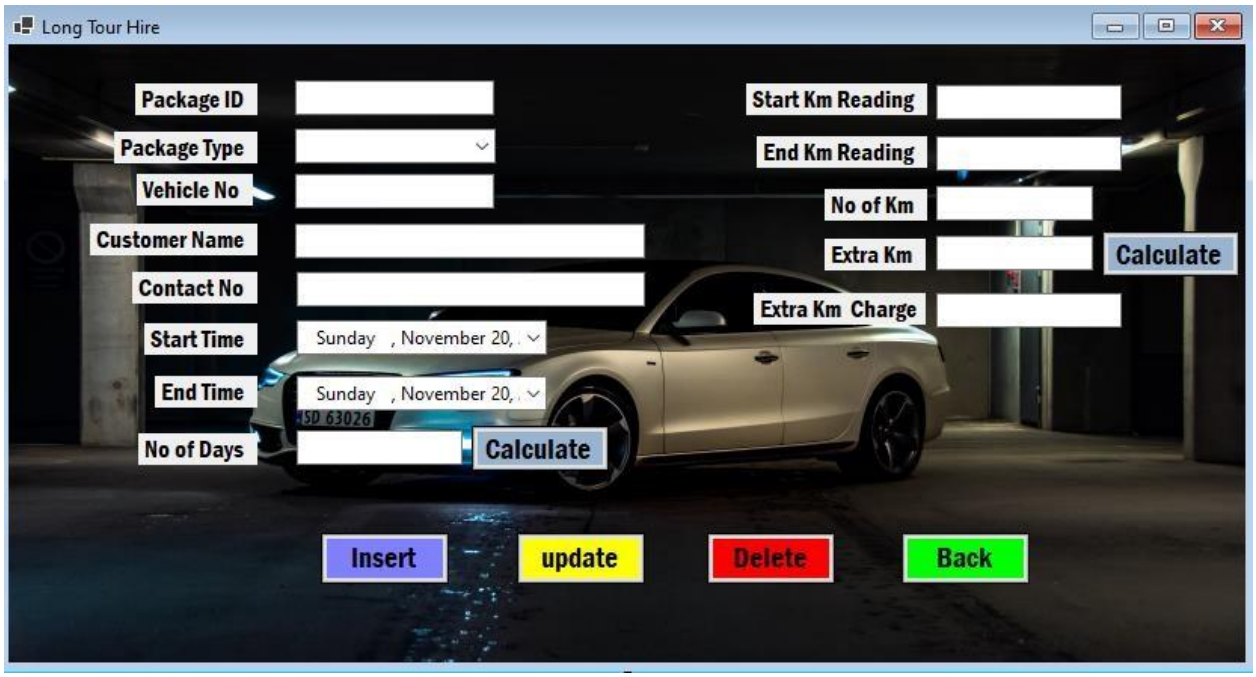


Figure 13- hire Long tour

```
            MessageBox.Show("recoard insert done");

            con.Open();
            SqlDataAdapter sqldata = new SqlDataAdapter("select * from hirelongtour", con);
            DataTable dt = new DataTable();
            sqldata.Fill(dt);
            dataGridView1.DataSource = dt;
            con.Close();
        }

        1 reference
        private void button6_Click(object sender, EventArgs e)
        {
            int input1 = Convert.ToInt32(txt_start_km_reader.Text);
            int input2 = Convert.ToInt32(txt_end_km_reader.Text);


            int total = (input1 - input2) + 0;
            txt_no_of_km.Text = total.ToString();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            DateTime sdt = dateTimePicker1.Value.Date;
            DateTime edt = dateTimePicker2.Value.Date;

            TimeSpan ts = sdt - edt;
            int days = ts.Days;
            int total = days +0;

            txt_no_of_days.Text = total.ToString();



        }

        1 reference
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
```

Figure 14- Hire long tour coding

2.Day Tour Hire

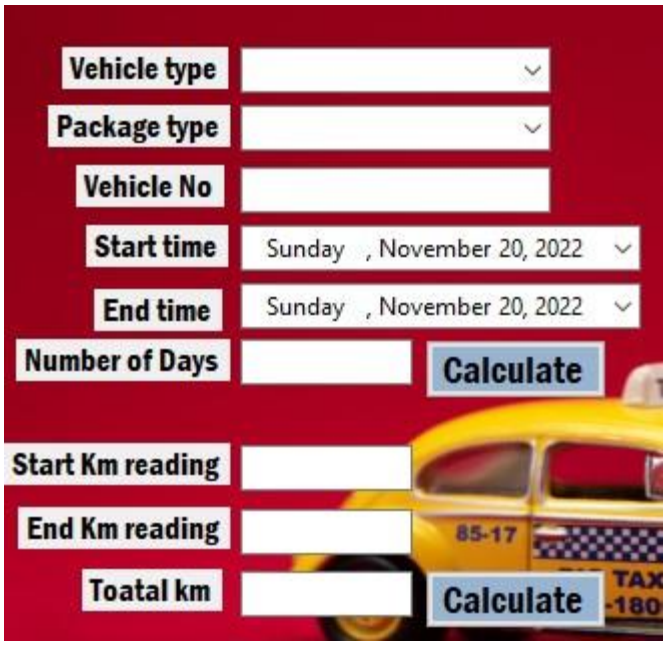


Figure 15- Day tour Hire

Database development

Vehicle Detail:

44 K.A Bhashitha Maduwantha E159257
Programming

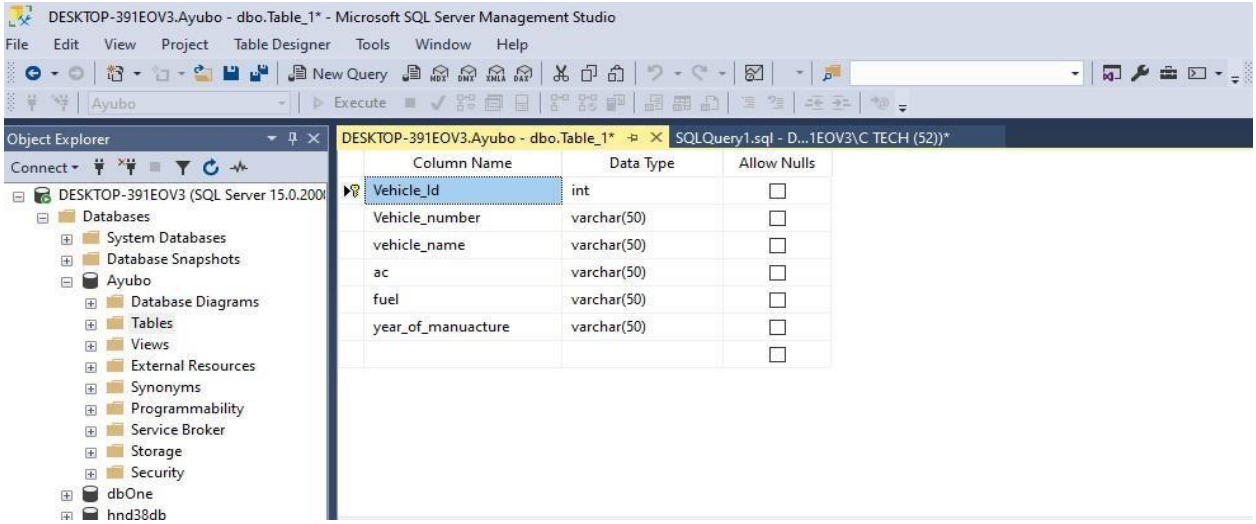


Figure 16- vehicle detail DB

Rent Details:

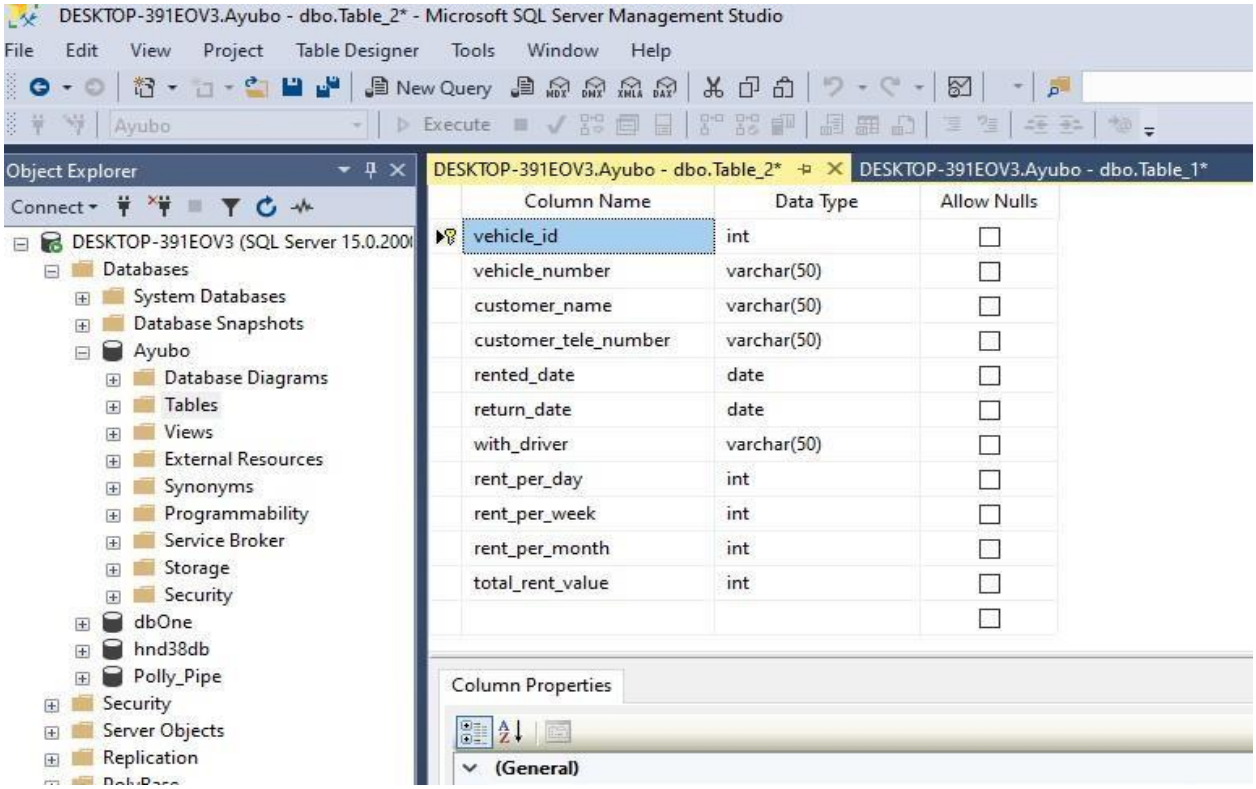


Figure 17- rent Details DB

Hire Long Tour:

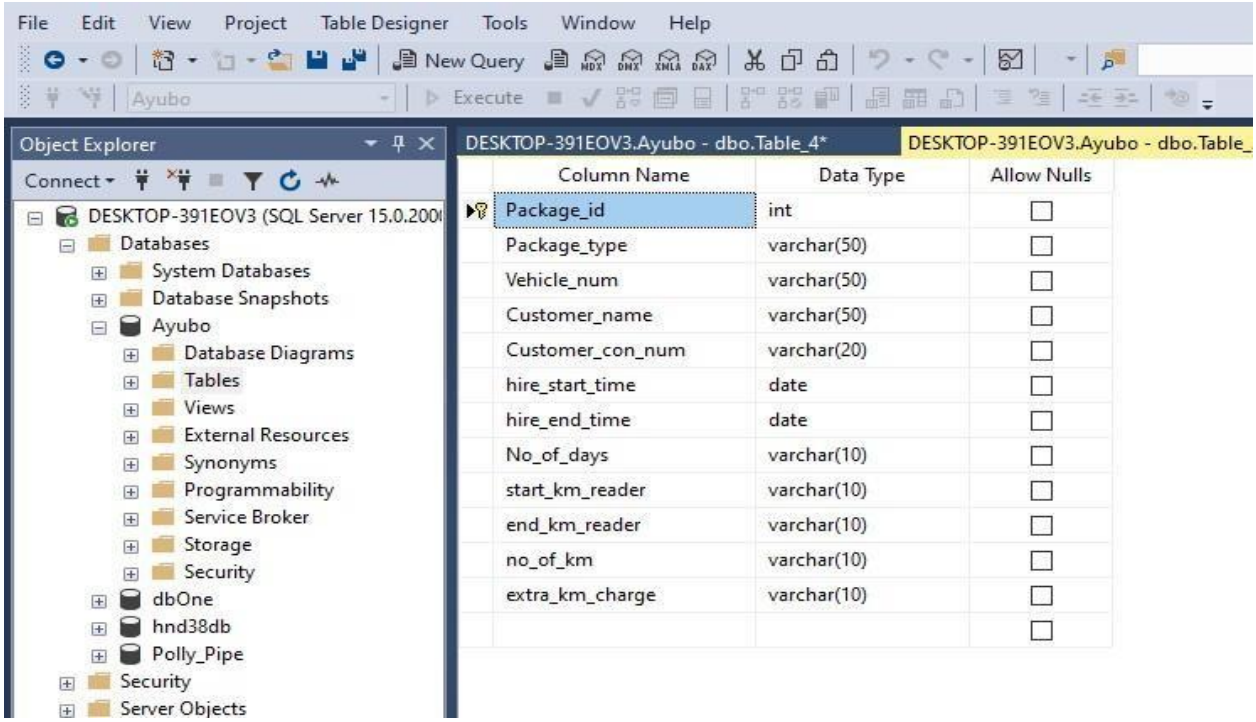


Figure 18-Hire long tour DB

Day tour hire:

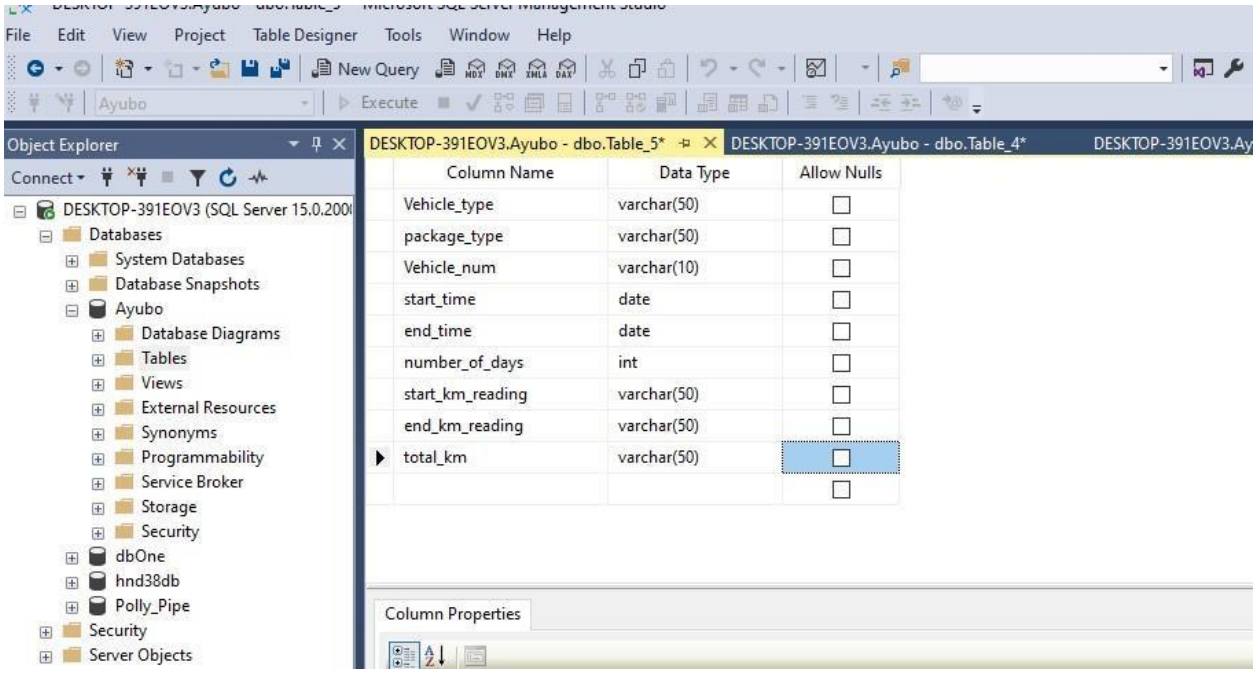


Figure 19-Day tour hire DB

Driver Registration:

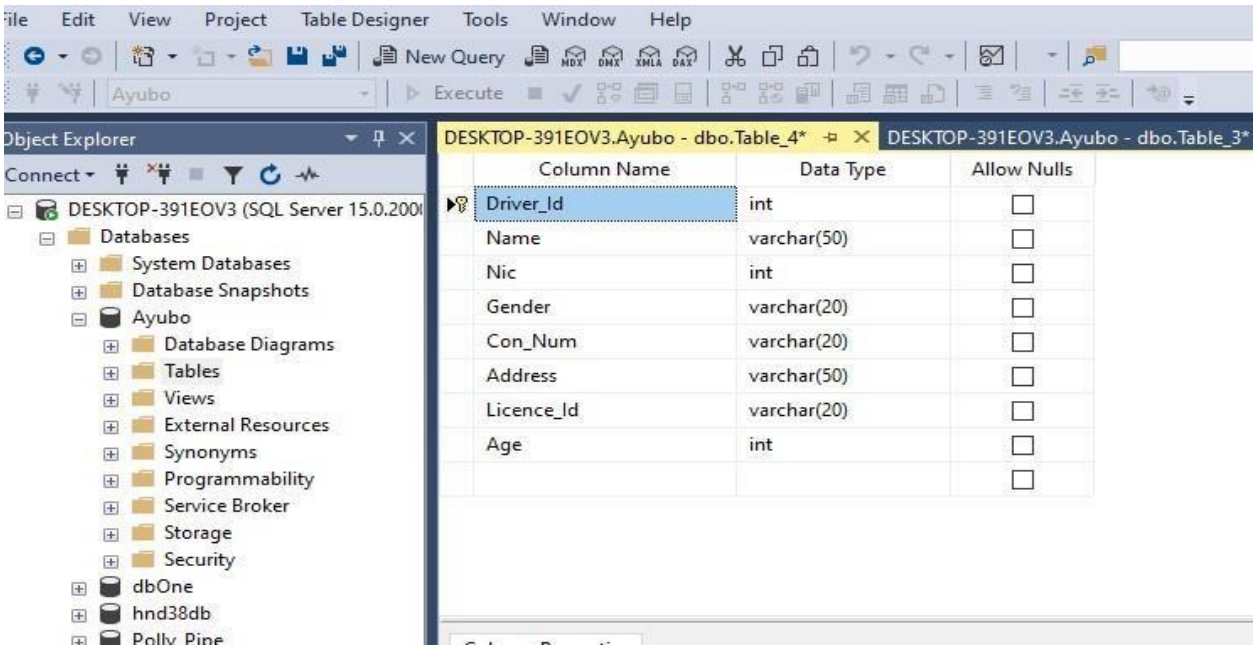


Figure 20- Driver Registration

What is an IDE

Software for creating applications known as an integrated development environment (IDE) combines standard developer tools into a single graphical user interface.

An IDE usually includes:

- Source code editor
- Local build automation
- Debugger

Because several tools don't need to be manually configured and integrated as part of the setup process, an IDE enables developers to start developing new apps rapidly. Additionally, because every utility is available on the same workbench, developers don't have to spend hours learning how to use each one individually.

Benefits of IDEs

- Impose project or corporate standards
- Less time and effort
- Managing the project
- Accelerating the process of problem-solving

Advantages of using visual studio as the IDE of the application which we are developing

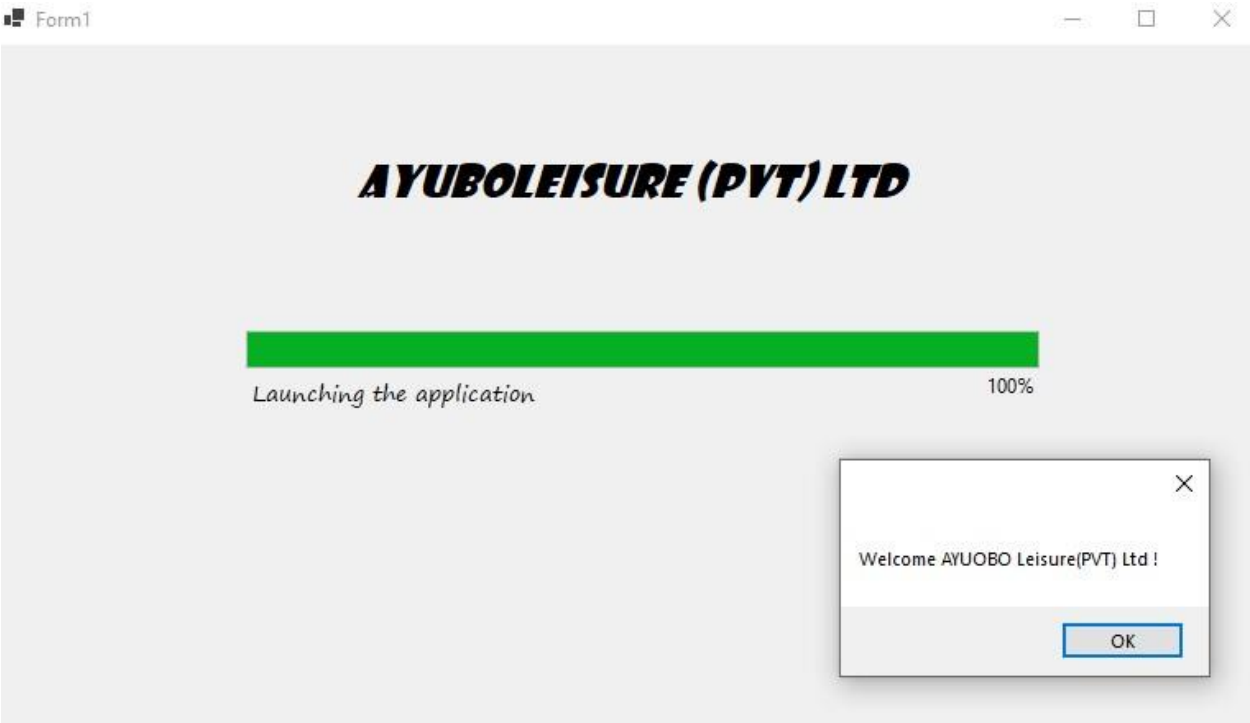
- The IDE underlines syntax and other mistakes in red for any programming language and offers suggestions as fixes, allowing developers to create apps more quickly than with other applications on this system.
- Coding is made simple with Visible Studios' IDE thanks to auto-ideas and syntax checks.
- Even when just one line of code has been correctly written, the developer can easily run the application and verify that it functions as intended.
- The specifics of any modifications you make in the IDE are kept in the Visual Studio project files.

It is an easy-to-use tool that primarily helps pupils by streamlining the learning process.The student has utilised Visual studios since they make coding and debugging straightforward.

Designing and developing the application

Launching the application

Figure 21-Launching the application



```
4 references
public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        InitializeComponent();
    }

    1 reference
    private void timer1_Tick(object sender, EventArgs e)
    {
        if(PBloading1.Value < 100)
        {
            PBloading1.Value += 1;
            LblReportprogress.Text = PBloading1.Value.ToString() + "%";
        }
        else
        {
            timer1.Stop();
            MessageBox.Show("Welcome AYUOBO Leisure(PVT) Ltd !");
            //go to the Main form
            this.Hide();
            Main_Form main_Form = new Main_Form();
            main_Form.ShowDialog();
        }
    }

    1 reference
    private void Form1_Load(object sender, EventArgs e)
    {
        timer1.Start();
    }
```

Figure 22- Launching the application code

Login Page

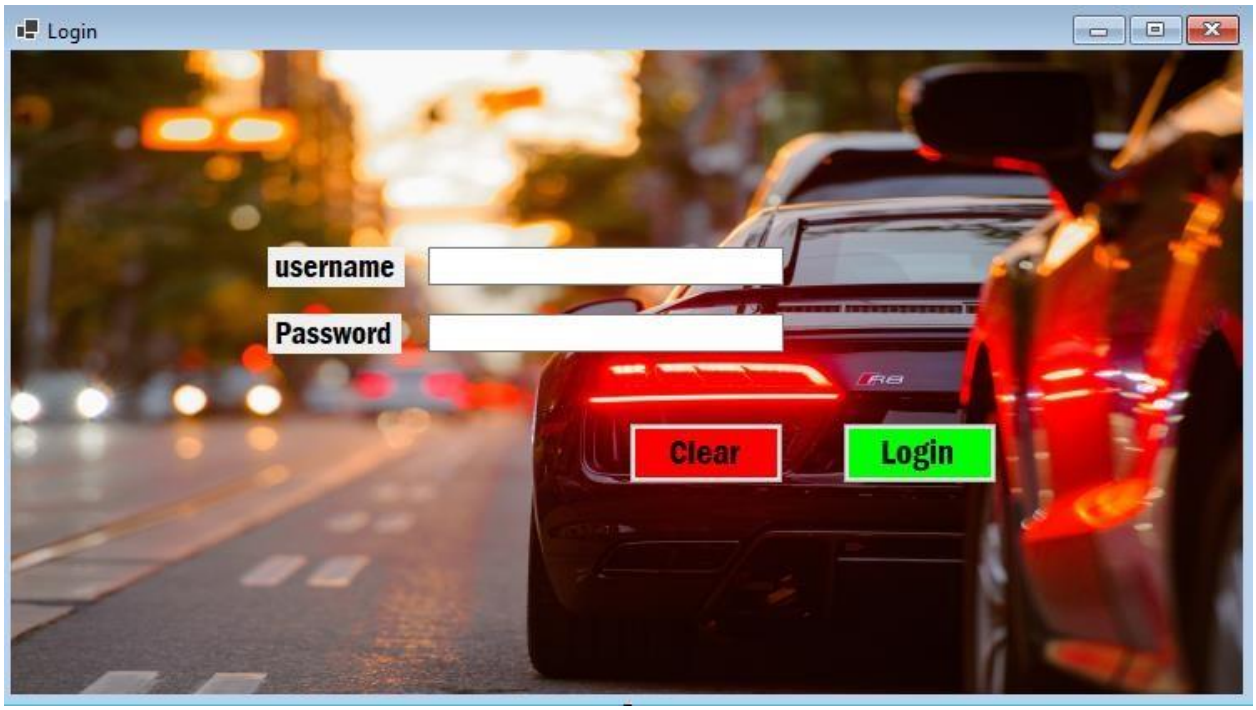


Figure 23-Login page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace programme_assignment
{
    4 references
    public partial class login : Form
    {
        1 reference
        public login()
        {
            InitializeComponent();
        }
        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-B7C25O4;Initial Catalog=ayuboDB;Integrated Security=True");

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            // login button

            SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-B7C25O4;Initial Catalog=ayuboDB;Integrated Security=True");
            string query = "select * from login where username = '" + textBox1.Text.Trim() + "' and password = '" + textBox2.Text.Trim() + "'";
            SqlDataAdapter sda = new SqlDataAdapter(query, con);
            DataTable dtbl = new DataTable();
            sda.Fill(dtbl);
            if (dtbl.Rows.Count == 1)
            {
                Dashboard main = new Dashboard();
                this.Hide();
                main.Show();
            }
            else
            {
                MessageBox.Show("Check the username and password");
            }

        }

        1 reference
```

Figure 24- Login page Code

Dash Board

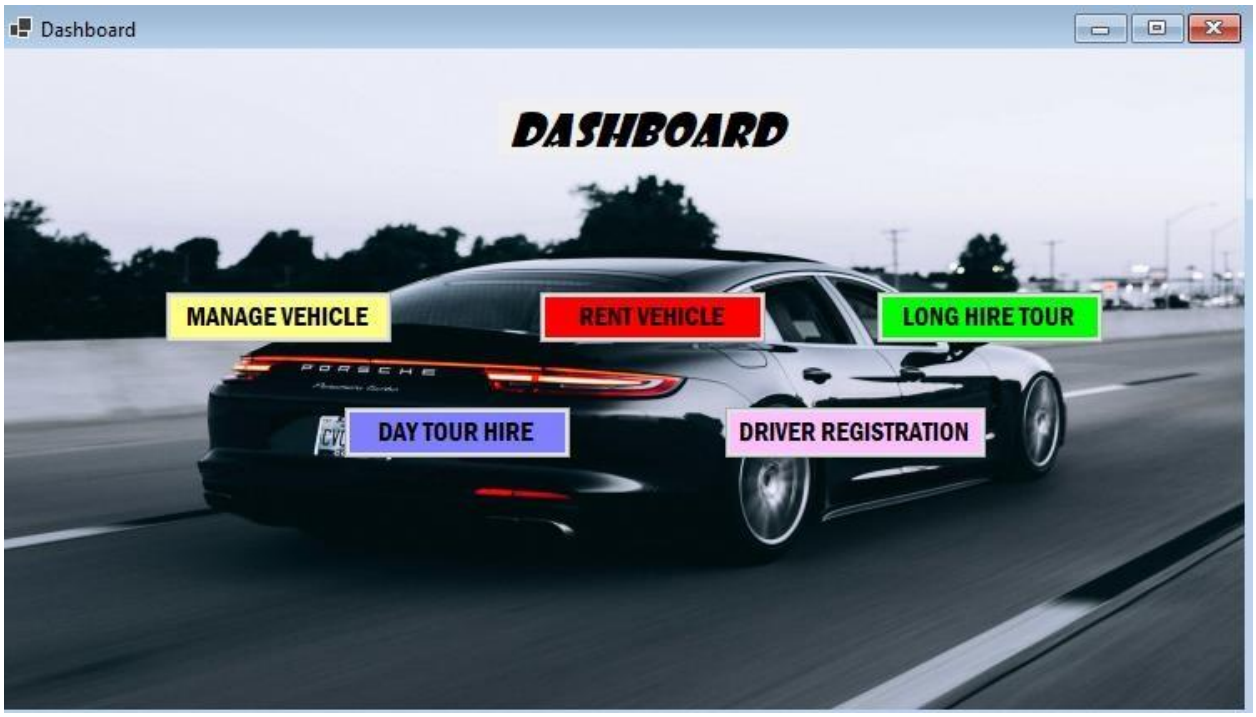


Figure 25- Dashboard

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace programme_assignment
{
    15 references
    public partial class Dashboard : Form
    {
        6 references
        public Dashboard()
        {
            InitializeComponent();
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 f3 = new Form3();
            f3.Show();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form5 f5 = new Form5();
            f5.Show();
        }

        1 reference
        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4();
            f4.Show();
        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
```

Manage Vehicle page

Figure 26- Dashboard coding

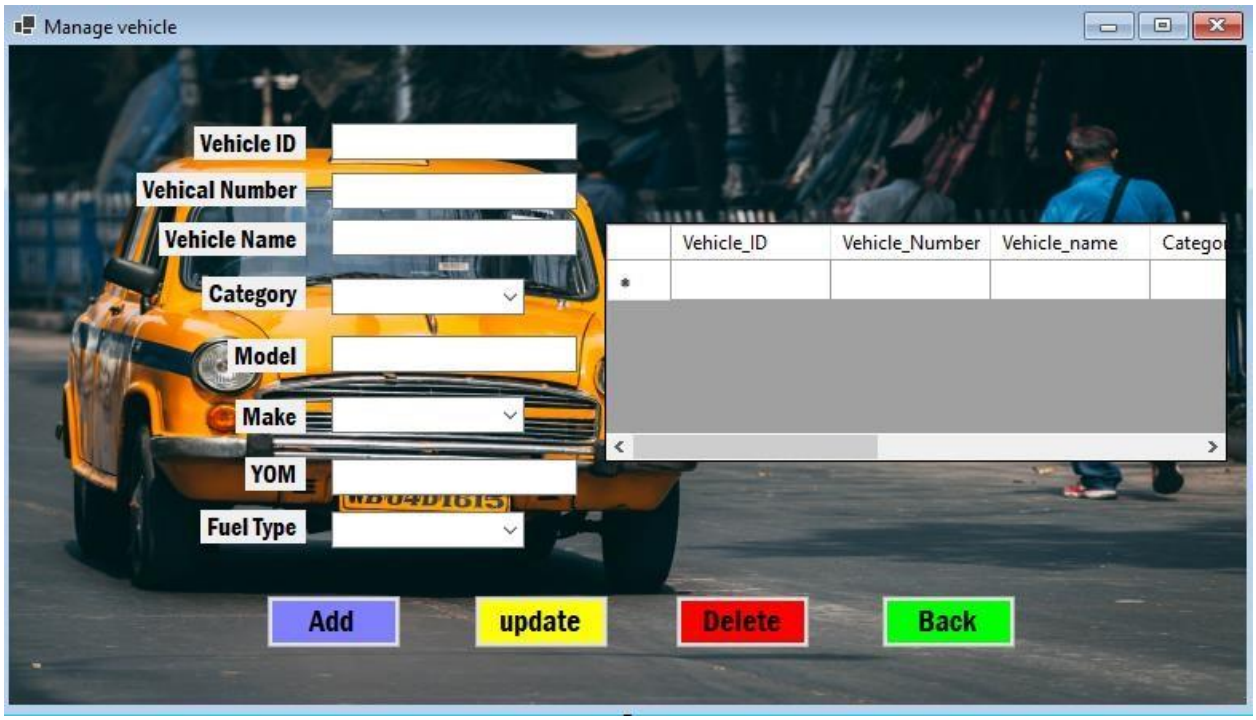


Figure 27- Manage Vehicle Page

```
18          {
19              InitializeComponent();
20
21              con.Open();
22              SqlDataAdapter sqldata = new SqlDataAdapter("select * from vehicle_details", con);
23              DataTable dt = new DataTable();
24              sqldata.Fill(dt);
25              dataGridView1.DataSource = dt;
26              con.Close();
27          }
28          SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-B7C25O4;Initial Catalog=ayuboDB;Integrated Security=True");
29
            1 reference
30          private void button5_Click(object sender, EventArgs e)
31          {
32              this.Hide();
33              Dashboard f2 = new Dashboard();
34              f2.ShowDialog();
35          }
36
            1 reference
37          private void button2_Click(object sender, EventArgs e)
38          {
39              con.Open();
40              SqlCommand cmd = con.CreateCommand();
41              cmd.CommandType = CommandType.Text;
42              cmd.CommandText = "insert into vehicle_details values('" + txt_vehicle_id.Text + "','" + txt_vehicle_number.Text + "','" + txt_vehicle
43              cmd.ExecuteNonQuery();
44              con.Close();
45
46              txt_vehicle_id.Clear();
47              txt_vehicle_number.Clear();
48              txt_vehicle_name.Clear();
49              txt_ac.Clear();
50              txt_year_of_manufacture.Clear();
51
52              MessageBox.Show("recoard insert done");
53              con.Open();
54              SqlDataAdapter sqldata = new SqlDataAdapter("select * from vehicle_details", con);
55              DataTable dt = new DataTable();
56              sqldata.Fill(dt);
57              dataGridView1.DataSource = dt;
58              con.Close();
59          }
60
61
```

Figure 28- Manage Vehicle page coding

Day Tour Hire Page

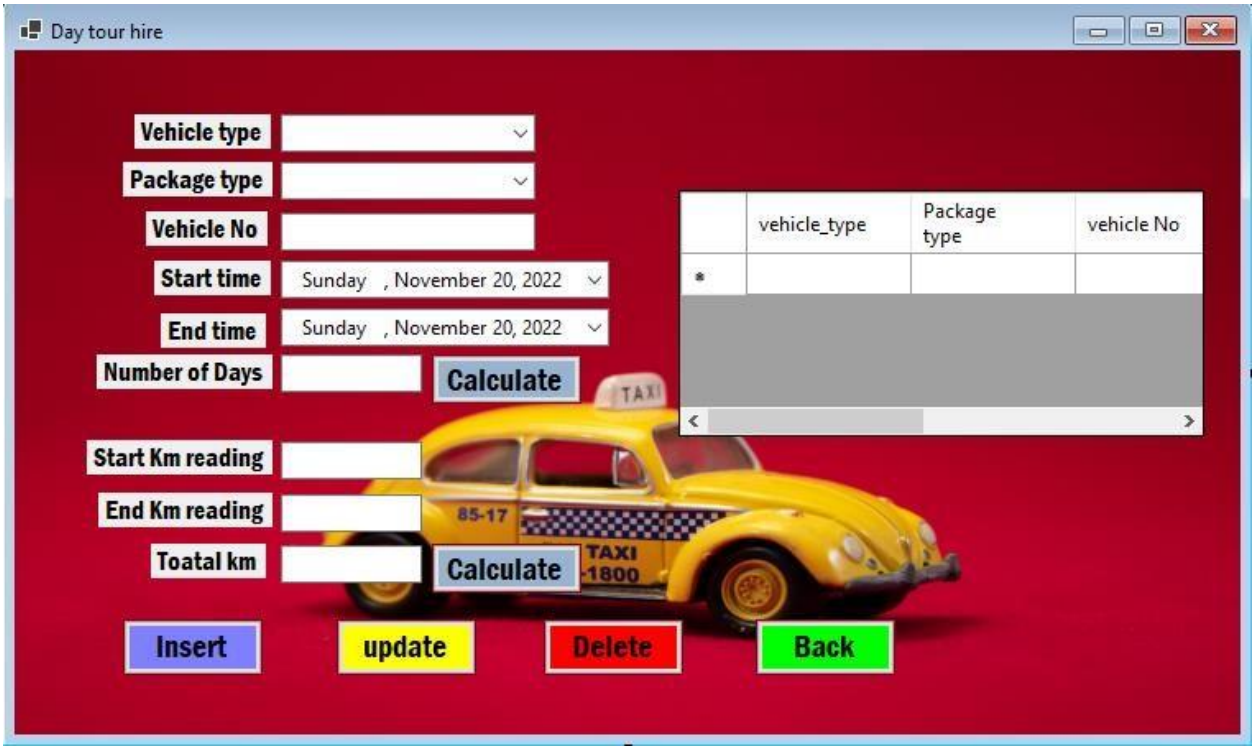


Figure 29- Day tour Hire

```
1 reference
private void btn_calculate_km_Click(object sender, EventArgs e)
{
    int input1 = Convert.ToInt32(txt_start_km_reading.Text);
    int input2 = Convert.ToInt32(txt_end_km_reading.Text);


    int total = (input1 - input2) + 0;
    txt_total_km.Text = total.ToString();
}

1 reference
private void btn_inset_Click(object sender, EventArgs e)
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "insert into daytourhire values('" + comboBox1.Text + "','" + comboBox2.Text + "','" + txt_vehicle_no.Text + "','" +
    cmd.ExecuteNonQuery();
    con.Close();

    txt_vehicle_no.Clear();
    txt_number_of_days.Clear();
    txt_start_km_reading.Clear();
    txt_end_km_reading.Clear();
    txt_total_km.Clear();

    MessageBox.Show("recoard insert done");

    con.Open();
    SqlDataAdapter sqldata = new SqlDataAdapter("select * from daytourhirea", con);
    DataTable dt = new DataTable();
    sqldata.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
}

1 reference
private void btn_update_Click(object sender, EventArgs e)
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "update Daytourhire  set  vehicle_type='" + comboBox1.Text + "',package_type ='" + comboBox2.Text + "', vehicle_no =
    cmd.ExecuteNonQuery();
    con.Close();
```

Figure 30- Day tour hire coding

Rent vehicle Page

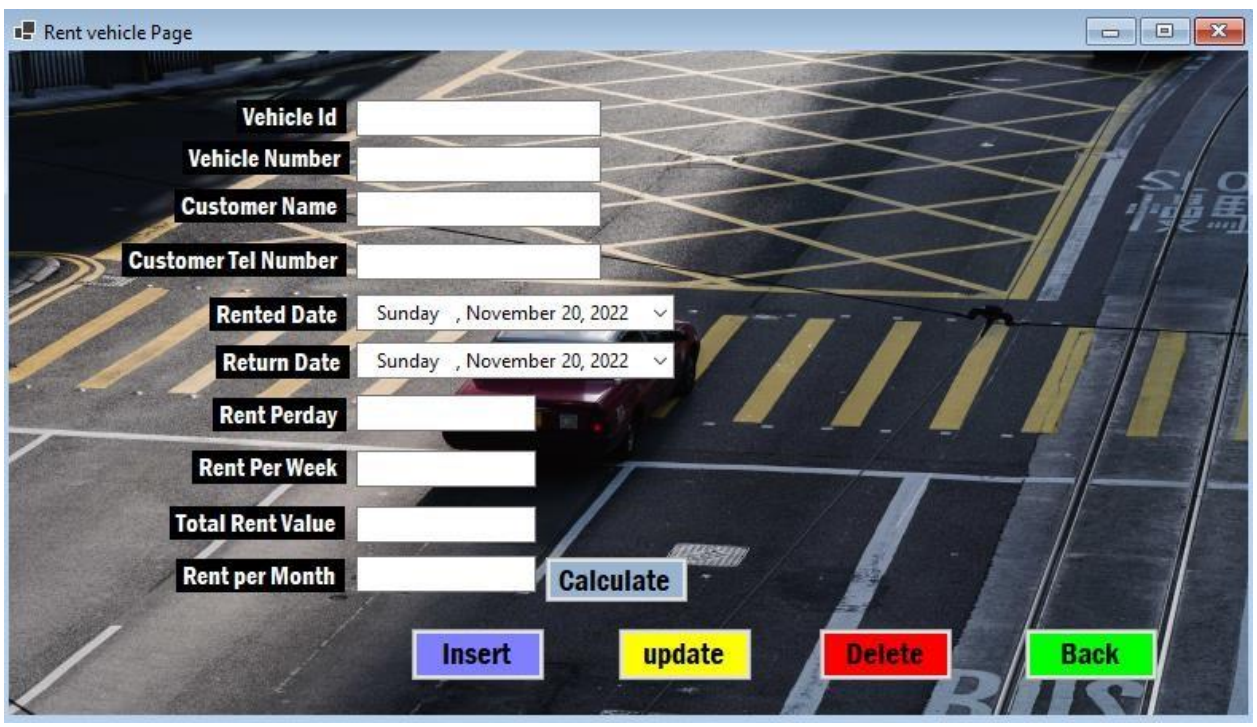


Figure 31- Rent Vehicle Page

```
            this.Hide();
            Dashboard f2 = new Dashboard();
            f2.Show();
        }

        1 reference
        private void Form3_Load(object sender, EventArgs e)
        {

        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "insert into rent values('" + txt_id.Text+ "','" + txt_vehicle_number.Text + "','" + txt_customer_name.Text + "','"
            cmd.ExecuteNonQuery();
            con.Close();

            txt_id.Clear();
            txt_vehicle_number.Clear();
            txt_customer_name.Clear();
            txt_customer_tele_number.Clear();
            txt_rent_per_day.Clear();
            txt_rent_per_week.Clear();
            txt_rent_per_month.Clear();
            txt_total_rent_value.Clear();


            MessageBox.Show("recoard insert done");
            con.Open();
            SqlDataAdapter sqldata = new SqlDataAdapter("select * from rent", con);
            DataTable dt = new DataTable();
            sqldata.Fill(dt);
            dataGridView1.DataSource = dt;
            con.Close();
        }

        1 reference
        private void btn_calculate_Click(object sender, EventArgs e)
        {
            int input1 = Convert.ToInt32(txt_rent_per_day.Text);
            int input2 = Convert.ToInt32(txt_rent_per_week.Text);
            int input3 = Convert.ToInt32(txt_rent_per_month.Text);
```

Figure 32- Rent Vehicle coding

Long Tour Hire

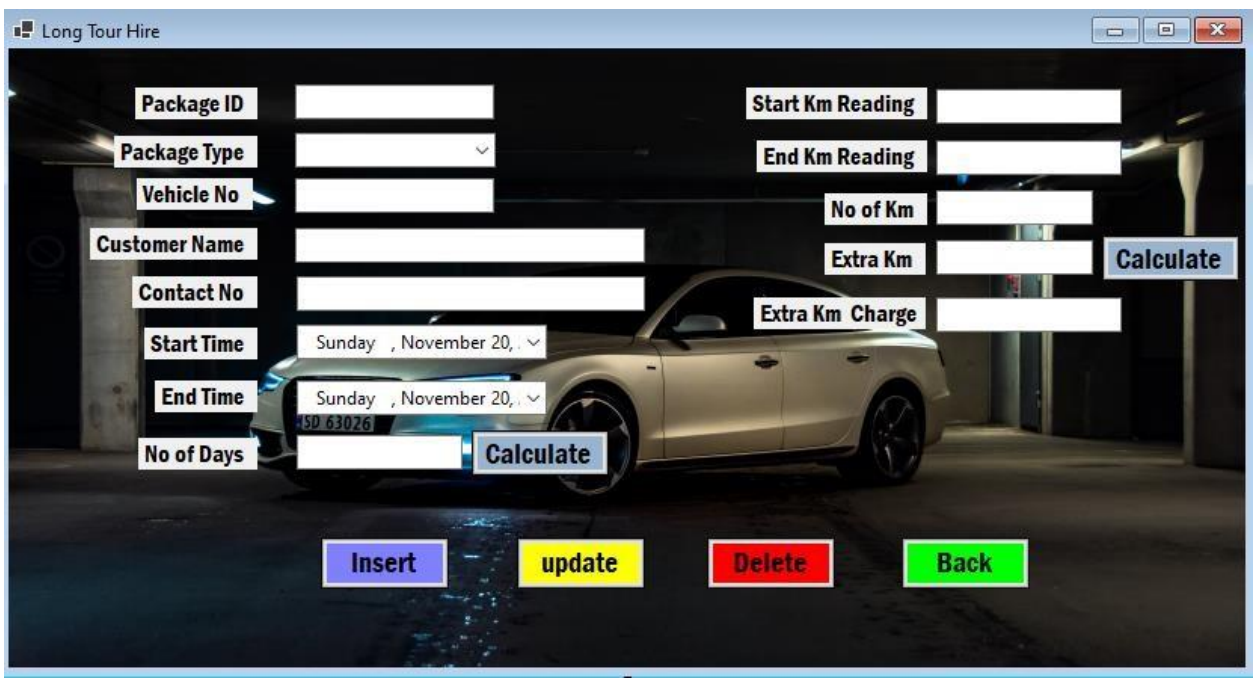


Figure 33-Long tour Hire

```
        }

        1 reference
        private void label8_Click(object sender, EventArgs e)
        {

        }

        0 references
        private void button5_Click_1(object sender, EventArgs e)
        {

        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "UPDATE hirelongtour  SET  package_type='" +comboBox1.Text + "',vehicle_number ='" + txt_vehicle_number.Text + "', c
            cmd.ExecuteNonQuery();
            con.Close();

            MessageBox.Show("recoard update done");

            con.Open();
            SqlDataAdapter sqldata = new SqlDataAdapter("select * from hirelongtour ", con);
            DataTable dt = new DataTable();
            sqldata.Fill(dt);
            dataGridView1.DataSource = dt;
            con.Close();
        }

        1 reference
        private void button5_Click_2(object sender, EventArgs e)
        {
            this.Hide();
            Dashboard f2 = new Dashboard();
            f2.ShowDialog();
        }

        1 reference
        private void Form4_Load(object sender, EventArgs e)
        {
            con.Open();
```

Figure 34- Long tour hire coding

Driver Registration

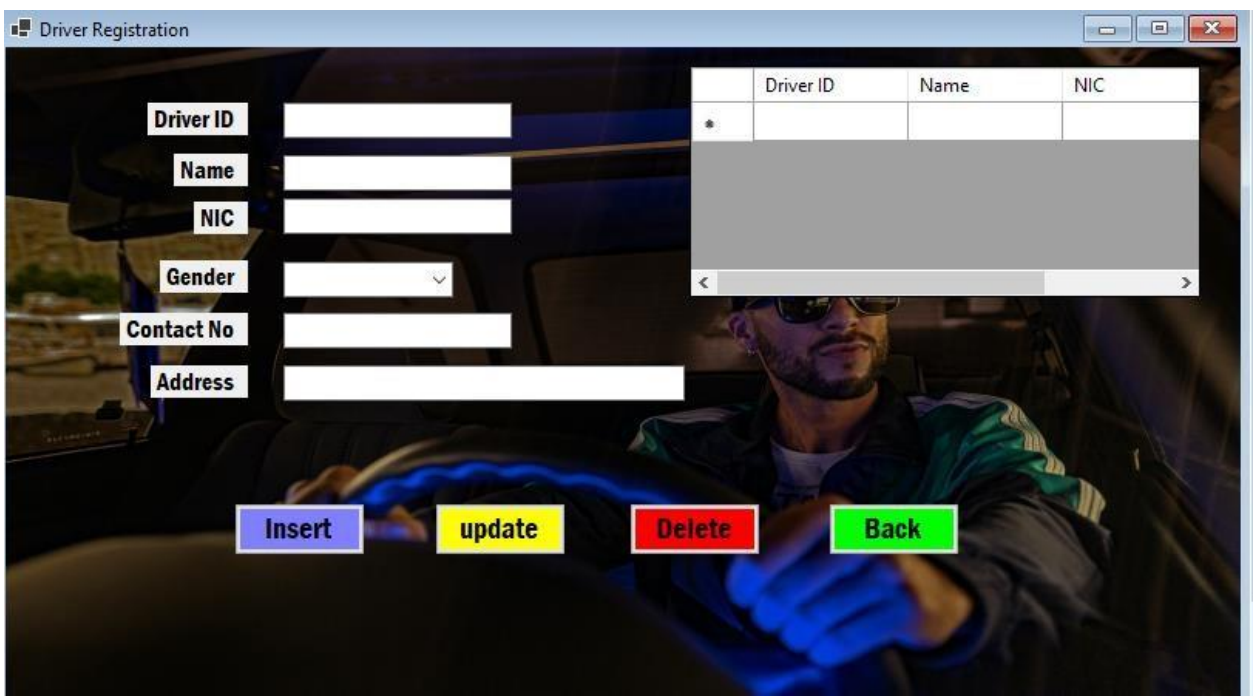


Figure 35- Driver registration

```
            InitializeComponent();
        }
        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-B7C25O4;Initial Catalog=ayuboDB;Integrated Security=True");

        1 reference
        private void driver_registartion_Load(object sender, EventArgs e)
        {
            SqlDataAdapter sqldata = new SqlDataAdapter("select * from Driver_Registartion", con);
            DataTable dt = new DataTable();
            sqldata.Fill(dt);
            dataGridView1.DataSource = dt;
            con.Close();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "insert into Driver_Registartion values('" + txt_driver_id.Text + "','" + txt_name.Text + "','" + txt_nic.Text + "',
            cmd.ExecuteNonQuery();
            con.Close();

            txt_driver_id.Clear();
            txt_name.Clear();
            txt_nic.Clear();
            txt_contact_number.Clear();
            txt_address.Clear();

            MessageBox.Show("recoard insert done");

            con.Open();

            SqlDataAdapter sqldata = new SqlDataAdapter("select * from Driver_Registartion", con);
            DataTable dt = new DataTable();
            sqldata.Fill(dt);
            dataGridView1.DataSource = dt;
            con.Close();
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
```

Figure 36- Driver registration Coding

Debugging

The act of debugging involves identifying and fixing code errors in computer programs. In the fields of engineering and information technology, the words "bug" and "error" are interchangeable. Debugging's objective is to locate and address an error's primary cause.

Debugging is a critical component of both software development and code handling. Debugging is frequently described by experts as involving "people, procedures, and systems" that will aid in resolving any problems with an existing code base.

How Debugging works

Debugging typically begins as soon as code is created and proceeds in phases when code is coupled with other programming units to create a software product. Using techniques like unit tests, code reviews, and pair programming can make troubleshooting a huge application with tens of thousands of lines of code easier.

Look at the code's logs and use a standalone debugger tool or the debug mode of an integrated development environment to find issues. The developer's knowledge of standard error messages

may be useful at this point. But even the cleanest code might be difficult to debug if authors don't sufficiently remark their code. (Heusser, *What is debugging?* 2019)

Sometimes the line of code itself is evident, but the module that exposes the issue is not. Unit tests, which allow the programmer to execute a particular function with a specified set of inputs, can be useful in this situation. Examples include JUnit and xUnit.

Figure 37- Debugging

Importance of Debugging

- Debugging helps you identify and fix pointless problems that are preventing your programme from functioning properly, which increases your programming dexterity. By doing this, you can improve your problem-solving abilities and attention to detail.
- When developers are able to precisely search for possible and existing problems that could disrupt their code, they become more resilient. Coders learn the patience necessary to construct successful, bug-free programmes through trial and error.
- Your development experience will increase as a result of debugging. Debugging procedures and mistakes-learning are important parts of building up your professional experience. The tedious and difficult process of removing unnecessary distractions from your code tends to increase your understanding of the topic, making you an expert.
- The many hours spent debugging, let's face it, can be a programmer's biggest source of annoyance and agony, but it is very essential for his or her development into a seasoned developer. Always consider it an opportunity to get better, get outside of your comfort zone, and pick up new skills.

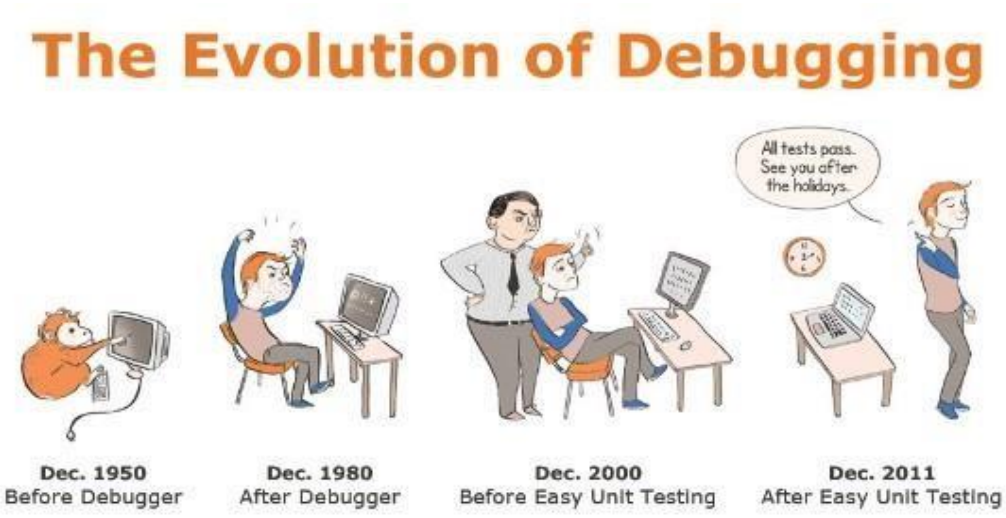


Figure 38- Evolution of Debugging

Coding Standard

The Coding phase involves the coding of several modules that are specified in the design document in accordance with the module specification. The primary objective of the coding phase is to write high-level code using the design document created following the design phase, and to unit test this code later.

Coding standards are a set of guidelines that reputable software development companies expect their programmers to adhere to. According to their organization's needs and the kinds of software they create, they typically create their own coding standards and rules. Maintaining coding standards is crucial for programmers; otherwise, the code would be rejected during code review.

Purpose of having coding standards

- It also minimises complexity while enhancing readability, maintainability, and intricacy of the code.
- It facilitates code reuse and makes error detection simple.
- It encourages sound programming techniques and boosts programmers' productivity.

- A coding standard offers the codes created by various engineers a uniform appearance.

The following list of coding standards includes some:

- Limited use of globals
- Standard headers for different modules
- Naming standards for constants, functions, global variables, and local variables
- Indentation
- Conventions for managing exceptions and error return values • A code style that is too challenging to grasp should be avoided
- Avoid using the same identification more than once.
- Code must be properly documented.
- Function length shouldn't be excessively long.

The following list of factors explains why a developer must adhere to a coding standard in order to successfully complete the program.

- It decreases complexity and increases readability, maintainability, and intricacy of the code.
- It encourages sound programming techniques and boosts programmers' productivity.
- The codes created by many engineers have a consistent appearance thanks to coding standards.
- It promotes code reuse and makes error detection simple.

The developer can create a better application by applying coding standards efficiently, and this will also ensure that the application contains the fewest possible faults and problems.

References

What is algorithm: Introduction to algorithms (2022) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/introduction-to-algorithms/ (Accessed: November 20, 2022).

Most important type of algorithms (2022) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/most-important-type-of-algorithms/ (Accessed: November 20, 2022).

Introduction of programming paradigms (2022) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/introduction-of-programming-paradigms/ (Accessed: November 20, 2022).

Huang, S. (2022) *What is big O notation explained: Space and time complexity*, *freeCodeCamp.org*. freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt1674cfa8a23c/ (Accessed: November 20, 2022).

Heusser, M. (2019) *What is debugging?*, *SearchSoftwareQuality*. TechTarget. Available at: https://www.techtarget.com/searchsoftwarequality/definition/debugging#:~:text=Debugging%2C%20in%20computer%20programming%20and,and%20make%20sure%20it%20works. (Accessed: November 20, 2022).

(no date) *The programming process*. Available at: https://www.cs.bham.ac.uk/~rxb/java/intro/2programming.html#:~:text=There%20are%20usually%20three%20stages,Debugging (Accessed: November 20, 2022).

Linear Search - javatpoint (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/linear-search (Accessed: November 20, 2022).