

Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS			
Programme title	Unit 01-Programming Assignment 01		
Assessor	Mrs. Menisha silva	Internal Verifier	Mr.Vidura Botheju
Unit(s)			
Assignment title	Design & Implement a GUI based system using a suitable Integrated Development Environment		
Student's name	K A Thilina Shenal Kuruppu		
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	

Higher Nationals - Summative Assignment Feedback Form

Student Name/ID	K A Thilina Shenal Kuruppu / LH54647		
Unit Title	Unit 01-Programming Assignment		
Assignment Number	02	Assessor	Mrs. Mnisha Silva
Submission Date	2020/03/30	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 process of programming an application. Pass, Merit & Distinction Descriptors P1 <input type="checkbox"/> M1 <input type="checkbox"/> D1 <input type="checkbox"/>			
LO2. Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis and design of a program using an Integrated Development Environment (IDE). Pass, Merit & Distinction Descriptors P2 <input type="checkbox"/> M2 <input type="checkbox"/> D2 <input type="checkbox"/>			
LO3. Implement basic algorithms in code using an IDE. Pass, Merit & Distinction Descriptors P3 <input type="checkbox"/> M3 <input type="checkbox"/> D3 <input type="checkbox"/>			
LO4. Determine the debugging process and explain the importance of a coding standard. Pass, Merit & Distinction Descriptors P4 <input type="checkbox"/> P5 <input type="checkbox"/> M4 <input type="checkbox"/> D4 <input type="checkbox"/>			
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 be sure to fill the details correctly.
2. This entire brief should be attached in first before you start answering.
3. All the assignments should prepare using word processing software.
4. All the assignments should print in A4 sized paper, and make sure to only use one side printing.
5. Allow 1" margin on each side of the paper. But on the left side you will need to leave room for binding.

Word Processing Rules

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

Important Points:

1. Check carefully the hand in date and the instructions given with the assignment. Late submissions will not be accepted.
2. Ensure that you give yourself enough time to complete the assignment by the due date.
3. Don't leave things such as printing to the last minute – excuses of this nature will not be accepted for failure to hand in the work on time.
4. You must take responsibility for managing your own time effectively.
5. 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.
6. Failure to achieve at least a PASS grade will result in a REFERRAL grade being given.
7. Non-submission of work without valid reasons will lead to an automatic REFERRAL. You will then be asked to complete an alternative assignment.
8. Take great care that if you use other people's work or ideas in your assignment, you properly reference them, using the HARVARD referencing system, in you text and any bibliography, otherwise you may be guilty of plagiarism.
9. If you are caught plagiarising you could have your grade reduced to A REFERRAL or at worst you could be excluded 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.

Thilinashenal34@gmail.com

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

2020/04/06

Date:
(Provide Submission Date)

Higher National Diploma in Computing

Assignment Brief

Student Name /ID Number	K A Thilina Shenal Kuruppu/LH54647
Unit Number and Title	Unit 01: Programming
Academic Year	2019/20
Unit Tutor	
Assignment Title	Design & Implement a GUI based system using a suitable Integrated Development Environment
Issue Date	2020/03/01
Submission Date	2020/03/30
IV Name & Date	
Submission Format	
<p>This submission will have 3 components</p> <p>1. Written Report</p> <p>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)</p> <p>2. Implemented System (Software)</p> <p>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.</p> <p>3. Presentation</p> <p>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.</p>	
Unit Learning Outcomes:	
	<p>LO1. Define basic algorithms to carry out an operation and outline the process of programming an application.</p> <p>LO2. Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE).</p> <p>LO3. Implement basic algorithms in code using an IDE.</p>

	LO4. Determine the debugging process and explain the importance of a coding standard
--	---

Assignment Brief and Guidance:

Activity 1

Searching on an array/list is to find a given element on the array and return whether it is found or not and return its position if found. Linear search and binary search are two popular searching algorithms on arrays.

- 1.1 Define what an algorithm is and outline the characteristics of a good algorithm. Develop algorithms for linear search and binary search using Pseudo code.
- 1.2 Describe the steps involved in the process of writing and executing a program. Take an array of 10 or more elements and dry run the above two algorithms. Show the outputs at the end of each iteration and the final output.
- 1.3 Define 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 Define what is meant by a Programming Paradigm. Explain the main characteristics of Procedural, Object oriented and Event-driven paradigms and the relationships among them. . Introduction who found it. Why is the reason to have more programming paradigms?(Fincher &Robins,2018)
Similarities and dissimilarities. Pros and cons page 5.
- 2.2 Write small snippets of code as example for the above three programming paradigms using a suitable programming language(s).
- 2.3 Critically evaluate the code samples that you have 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), 7-seater 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:

$$\text{Total rent} = \text{weeklyRent} \times 1 + \text{dailyRent} \times 3 + \text{dailyDriverCost} \times 10$$

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 type if that is applicable for that type of vehicle. For each package maximum km limit and maximum number of hours are also 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

3.1 Design suitable algorithms for vehicle tariff calculation for rents and hires.

Ideally 3 functions should be developed for this purpose as follows:

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.

3.2 Implement the above algorithms using visual studio IDE (using C#.net) 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.

3.3 Analyze the features of an Integrated Development Environment (IDE) and explain how those features help in application development. Evaluate the use of the Visual Studio IDE 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 for Ayubo 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 c

4.2 calculation and recording function described above. Generating customer reports and customer invoices are not required for this course work.

4.3 What is debugging an application? Explain 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.

4.4 Explain 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.		
P1 Provide a definition of what an algorithm is and outline the process in building an application.	20-30	
M1 Determine the steps taken from writing code to execution.	30-35	
D1 Examine the implementation of an algorithm in a suitable language. Evaluate the relationship between the written algorithm and the code variant	30-35	
LO2 Explain the characteristics of procedural, object orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE)		

P2 Give explanations of what procedural, object orientated, and event driven paradigms are; their characteristics and the relationship between them.	38-43	
M2 Analyze the common features that a developer has access to in an IDE.	55-67	
D2 Critically evaluate the source code of an application which implements the programming paradigms, in terms of the code structure and characteristics.	70-88	
LO3 Implement basic algorithms in code using an IDE.		
P3 Write a program that implements an algorithm using an IDE.	52-55	
M3 Use the IDE to manage the development process of the program.	52-55	
D3 Evaluate the use of an IDE for development of applications contrasted with not using an IDE.	55-70	

LO4 Determine the debugging process and explain the importance of a coding standard		
P4 Explain the debugging process and explain the debugging facilities available in the IDE.	70-88	
P5 Outline the coding standard you have used in your code.	88-89	
M4 Evaluate how the debugging process can be used to help develop more secure, robust applications.	70-88	
D4 Critically evaluate why a coding standard is necessary in a team as well as for the individual.	88-89	

Content Page

Contents

Content Page	16
List of figures.....	18
List of tables.....	19
Activity 1	20
1.1 Define what an algorithm is and outline the characteristics of a good algorithm. Develop algorithms for linear search and binary search using Pseudo code.	20
1.2 Describe the steps involved in the process of writing and executing a program. Take an array of 10 or more elements and dry run the above two algorithms. Show the outputs at the end of each iteration and the final output.....	30
1.3 Define 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.	35
Activity 2	38
2.1 Define what is meant by a Programming Paradigm. Explain the main characteristics of Procedural, Object oriented and Event-driven paradigms and the relationships among them. Introduction who found it. Why is the reason to have more programing paradigms? (Fincher &Robins,2018).....	38
Similarities and dissimilarities. Pros and cons page 5.	38
2.2 Write small snippets of code as example for the above three programming paradigms using a suitable programming language(s).....	43
2.3 Critically evaluate the code samples that you have above in relation to their structure and the unique characteristics.....	48
Activity 3	52
3.1 Design suable algorithms for vehicle tariff calculation for rents and hires.	52
3.2 Implement the above algorithms using visual studio IDE (using C#.net) 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.	55
3.3 Analyze the features of an Integrated Development Environment (IDE) and explain how those features help in application development. Evaluate the use of the Visual Studio IDE for your application development contrasted with not using an IDE.	67
Activity 4	70
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 for Ayubo 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	

described above. Generating customer reports and customer invoices are not required for this course work.....70

4.2 What is debugging an application? Explain 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.88

4.3 Explain 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.89

List of Reference.....91

List of figures

Figure 1: Example for written code	31
Figure 2 – Example for test & debug	31
Figure 3- Programming paradigm chart.....	39
Figure 4-C++ program output.....	43
Figure 5- C# calculator interface before calculate	45
Figure 6- After calculate	46
Figure 7-Python output	47
Figure 8-Rent calculation flow chart	52
Figure 9-Day tour calculation flow chart.....	53
Figure 10- Long tour calculation flow chart.....	54
Figure 11-Rent calculation interface.....	55
Figure 12-After calculation.....	55
Figure 13-Day tour calculation interface	57
Figure 14-After calculate	57
Figure 15- Long tour calculation interface	59
Figure 16-After calculate	59
Figure 17-Select vehical type.....	61
Figure 18-Vehical registration interface	61
Figure 19-Data insert	62
Figure 20-Data update.....	63
Figure 21-Data delete.....	64
Figure 22-Data displayed.....	65
Figure 23-Form cleared.....	66
Figure 24-Toolbox	68
Figure 25-Solution explorer.....	69
Figure 26-Main menu interface	70
Figure 27-Data insert in rent calculation interface	71
Figure 28- Data update in rent calculation interface.....	72
Figure 29- Data delete in rent calculation interface.....	74
Figure 30- Data display in rent calculation interface.....	75
Figure 31- Data clear in rent calculation interface.....	76
Figure 32- Data insert in day tour calculation interface	77
Figure 33- Data update in day tour calculation interface.....	78
Figure 34- Data delete in day tour calculation interface.....	79
Figure 35- Data display in day tour calculation interface.....	80
Figure 36- Data clear in day tour calculation interface.....	81
Figure 37- Data insert in long tour calculation interface	82
Figure 38- Data update in long tour calculation interface	83
Figure 39- Data delete in long tour calculation interface	84
Figure 40- Data display in long tour calculation interface	85
Figure 41- Data clear in long tour calculation interface	86
Figure 42-Example for bug.....	88
Figure 43-Example comment.....	90

List of tables

Table 1- flowchart shapes	23
Table 2- Trace table using linear search	34
Table 3- Trace table using binary search	34
Table 4- Relationships among procedural, Object-oriented and Event-driven paradigms	42

Activity 1

1.1 Define what an algorithm is and outline the characteristics of a good algorithm.

Develop algorithms for linear search and binary search using Pseudo code.

An algorithm is a method that helps to solve problems according to a procedure. It is a small procedure regarding to mathematics and computer science. These are more using in IT field.

(www.techtarget.com/2020)

An algorithm is a clear and can understand because of there is a step by step problem solving procedure and its speed depend on the count of steps. Because of that we can increase speed by reducing steps and we can go to shortest path and make this simply.

(www.topcoder.com/2020)

There are 6 steps in process of writing an algorithm.

- Think about the result that you want after your code.
- Think about your starting point. In this step, we have to make answers to these questions to full of this process.
 - What are the available Data Inputs?
 - Where we can find that Data?
 - What are the applicable formulas according to problems?
 - What are the rules using when working with available Data?
 - How can join Data values to each other?
- Find the algorithm in the ending point. It is the next step in this process. When we are starting an algorithm, in that time we have to think about the end point by answering to these questions.
 - What we are learning in this process?
 - What is the thing that change when start and end?
 - What did we add to this process?
- Fourth step in this process is make a list of steps from start to end.
- Think about your list and think about your points of your list, that how you going to do one by one. In this step we have to concern about some questions.
 - What is the language that you are using?
 - What are the available resources that you can use?

- In that language, what is the best way to do in every point?

When you are explaining all the steps in this process you have to expand each point.

- Finally analyze the algorithm and make answers for these questions by yourself.
 - After finished this process, can you solve your problems by using this algorithm?
 - What about your algorithm's input and output, is it clearly defined?
 - Does your final result more specific?
 - Can you simplify any points of this process?
 - Is this algorithm, ending in the correct result?

(www.wikihow.com/2020)

An algorithm can represent into two main ways.

- Flowchart
- Pseudocode

Pseudocode

Pseudocode is writing in simple way of English. This is not a programming language. To write code for programs, pseudocode use short phrases in definite language. The results that you want to your program can achieve using pseudocode by creating statements.

(www.study.com/2020)

The main importance of pseudocode is, it can be writing directly in the program. You have to give all the details when you are coding in given programming language but when you write it in pseudocode, you no need to give all details. You only have to give the main idea of what the algorithm supposed to do. By the way Pseudocode is very easy to understand.

(www.quora.com/2020)

How to write Pseudocode?

- First of all, start with easy words to transcribed with your using algorithm.
- Notch when you are adding instructions within loop or a conditional clause. When set of instructions repeating, it is called "loop". Conditional clause is creating by using comparison.
- Do not use words with a fixed kind of a computer language.

Example:

structure.

- FOR/ENDFOR
- WHILE/ENDWHILE

And you can use these kinds of words for conditional clauses.

- IF/ENDIF
- WHILE/ENDWHILE
- CASE/ENDCASE


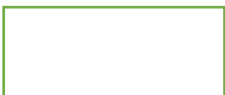
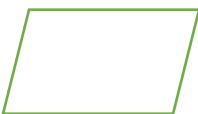
(www.study.com/2020)

Flowchart

Flowchart is an algorithm that process can clear and easy to understand. There are some shapes using to make these charts. Rectangle, Diamond, Oval and Parallelogram are shapes that using in these charts. Arrows are using to make connection step by step in this process. These charts are using in many fields. The fields are education, engineering, IT, business, etc. There are types of flowcharts. They are Document flowcharts, Data flowcharts, System flowcharts, Program flowcharts, etc.

(www.lucidchart.com)

Flowchart Shapes:

Shape	Function	Name
	Start/End	Oval
	Process	Rectangle
	Input/output	Parallelogram

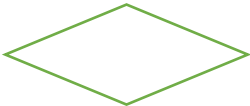

	Decision	Diamond
	Make Relationships	Arrow

Table 1- flowchart shapes

Example for Flowchart Algorithm;

Step 1: Start

Step 2: Create a variable

Step 3: Clear the variable

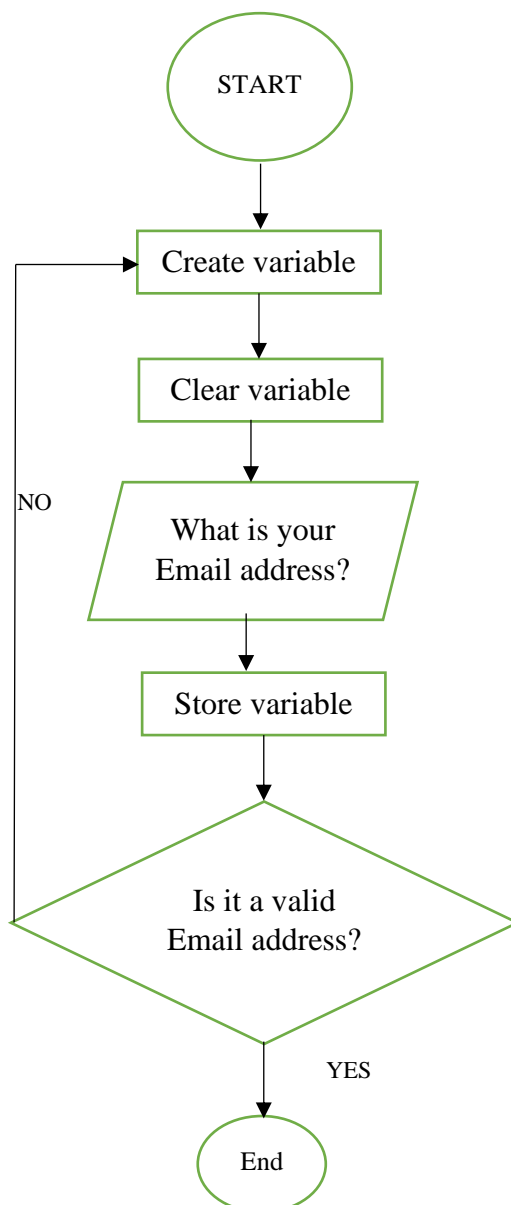
Step 4: Ask the users Email address

Step 5: Store the response in the variable

Step 6: Check the stored response

Step 7: Not valid? Go to step 3

Step 8: End



Characteristic of Good Algorithm

In every good algorithm, there should be these kinds of characteristics.

Precision – The steps are clearly noted.

Uniqueness – Each step, the results are clearly introducing and it only depend on the input and results of forwarding steps.

Finiteness – After finish the finite number of instructions execute, the algorithm will stop after that.

Input – Input receive to algorithm.

Output – Output produces by algorithm.

Generality – Set of input, own the algorithm.

Effectiveness – An algorithm should be effective.

(www.qsstudy.com/2020)

Linear Search Algorithm

Linear search is a search algorithm which is very basic and very simple. We search for an element or value in a given array in linear search by traversing the array from the start until the desired element or value is found.

Linear search algorithm finds a given element in a list of elements with time complexity of $O(n)$ where n is the total number of elements in the set. The search method starts to compare search element with the first item in the list. When both are matched, the result element will be identified otherwise the search element will be compared to the next item in the list.

Repeat the same until you compare search feature.

(www.studytonight.com/2020)

Implemented steps in Linear Search:

- Read the element.
- Compare the element with first element.
- Finish the function and display “Given element is found!!!” when they matched.
- If the elements not matched, compare again with another element.
- Until the search elements is compare with last element, repeat both step 3 and 4.
- Finish the function and display “Element is not found!!!” when the last element also not matched.

Example:

	0	1	2	3	4	5		6	7
List	12	25	51	45	30	65	20	95	

Search Element: 30

Step 1:

Search element (30) is compared with first element (12)

	0	1	2	3	4	5	6	7
List	12	25	51	45	30	65	20	95

Not matching, move on until the elements match.

Step 2:

	0	1	2	3	4	5	6	7
List	12	25	51	45	30	65	20	95

Both elements are matching at the index 4. Stop comparing and display element at index 4.

Pseudocode for Linear search

```
procedure linear_search (list, value)
```

```
  for each item in the list
```

```
    if match item == value
```

```
      return the item's location
```

```
    end if
```

```
  end for
```

```
end procedure
```

Binary Search Algorithm

Binary search algorithm can find a given element in a list of elements with time complexity of $O(\log n)$, where n is the total number of elements in the set. The binary search algorithm can only be used for a collection of various elements. That means that the binary search is only used for a list of elements already ordered in an order. A set of elements ordered in random order cannot be used for binary search. The search method starts a comparison between the main item and the middle item in the set. If both are matched then “item found” is the product. Otherwise we will test if the search element in the list is smaller or greater than the middle one. If the search element is smaller, we perform the same procedure for the left sublist of the middle element. If the search element is bigger, we perform the same procedure for right sublist of the middle element. This cycle is repeated until we find the matching element in the list or until we have reached a sublist containing just one element. And if the element does not suit the search element, the result is “Element not contained in the set.”

(www.btechsmartclass.com/2020)

Implemented steps in binary search:

- Read the element.
- Find the element in middle.
- Compare the search and middle elements.
- When they matched, display “Given element is found!!!” and finish the function.
- Not matched, check the search element is larger or smaller than the middle one.
- If it is smaller than middle one, repeat all the steps again for the left sublist of the middle one.
- If it is larger, repeat all the steps again for the right sublist of the middle one.
- Repeat the process until sublist contains only one element.
- If it is also does not match, display “Element is not found in the list!!!” and finish the process.

Example:

	0	1	2	3	4	5	6
List	12	25	51	45	30	65	20

Search element 25

Step 1:

25 compare with middle element (45)

	0	1	2	3	4	5	6
List	12	25	51	45	30	65	20

Not matching, 25 smaller than 45 then search only left sublist (i.e. 12, 25, 51).

	0	1	2	3	4	5	6
List	12	25	51	45	30	65	20

Step 2:

25 compare with new left sublist middle element (25).

	0	1	2	3	4	5	6
List	12	25	51	45	30	65	20

Both matching,

index 1" is the result.

"Element found at

Pseudocode for binary search

Procedure binary_search

Ar ← sorted array

As ← size of array

N ← value to be searched

Set lowerBound = 1

Set upperBound = u

while N not found

if upperBound < lowerBound

EXIT: N does not exists.

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

if Ar[midPoint] < N

set lowerBound = midPoint + 1

```
if Ar[midPoint] > N
    set upperBound = midPoint - 1

if Ar[midPoint] = N
    EXIT: N found at location midPoint
end while

end procedure
```

1.2 Describe the steps involved in the process of writing and executing a program. Take an array of 10 or more elements and dry run the above two algorithms. Show the outputs at the end of each iteration and the final output.

Program is a software that runs on computer and it is like a script but it is larger than the script size and it does not need scripting engine to run.

Examples to Programs:

- Video games
- Web browsers
- Applications

Programs create programmers by coding instructs the computer what should do.

There are some general steps that following by programmers.

1. Understand the problem
2. Design solutions
3. Draw flow charts
4. Write pseudo-code
5. Write code
6. Test & Debug
7. Test with real users
8. Release programme
9. Iterate steps to another version

Understand the problem

Before making a programme, first programmers have to do is they have to listen to the client's problem and they must be understanding the situation. This is the first step.

Design solutions

Regarding to the first step programmers must be design a solution to that problems.

Draw flow charts

After designed the solutions programmers must draw flow charts to that solutions.

Write pseudo-code

The third step is writing a pseudo-code. In this step programmers follows the flow chart and they start to write the correct suitable pseudo-code.

Write code

programmers do the main part of coding. It is the important part on these steps.

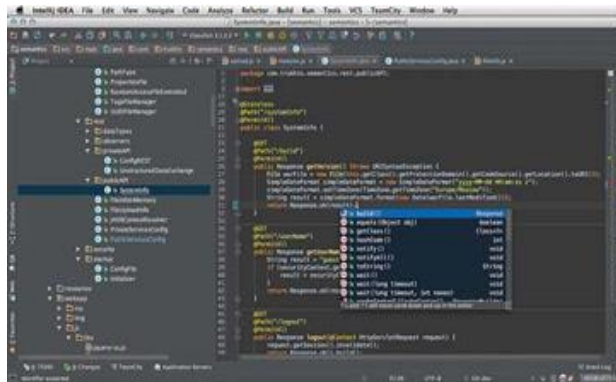


Figure 1: Example for written code

Test & Debug

After develop the programme the next step is to test the programme to find the outputs and functions as desired. Debugging is the step by step process of find bugs or errors in programme. The tools that use to debug are called debuggers. There are some debugging systems. They are Step by step, breaking point, etc.

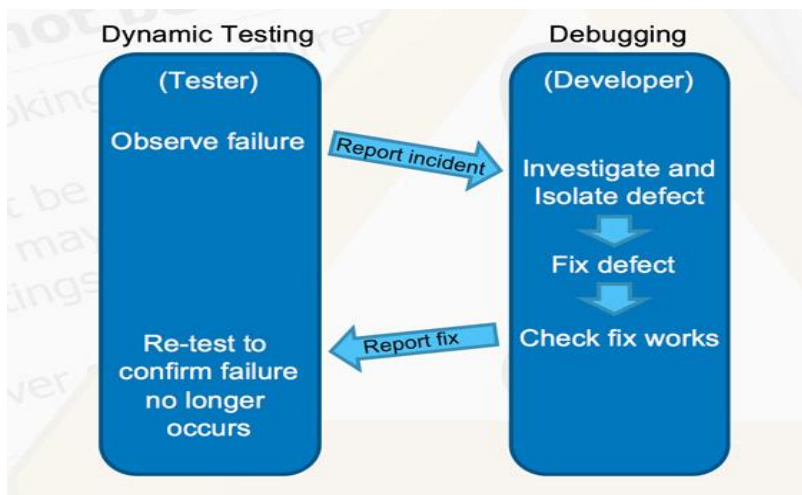


Figure 2 – Example for test & debug

Test with Real Users

After testing and debugging programmers do give their system experience to users and they are getting the users ideas and test their system through them.

Release Programme

After testing finish programmers finally check again system and correct mistakes and release their quality system to the users.

Iterate steps to another version

After released the programme the programmers are not stop their job. They will always try to iterate steps to another version. Then only programmer can keep the demand to their programme.

What is dry run?

A dry run may be a practice some time recently the genuine method. A dry run in testing carries the same idea. We as a rule perform dry runs with the taking after purposes:

- Approve a test case and progress it sometime recently recording it steps and declarations down.
- Confirm that the computer program beneath test does not contain any dazzling bugs.
- Decide whether the test case is automatable in terms of specialized challenges and commerce esteem.
- Distinguish commerce dangers as before long as conceivable accepting that the genuine test run takes as well long or troublesome to execute with in the appreciate way.
- Recognize the requirements for administrations virtualization whereas a few computer program components are still missing and code integration is not prepared however amid improvement.

What is an array?

An array may be an information structure that contains a bunch of components. Ordinarily these components are all of the same information sort, such as a numbers or string. Arrays are commonly utilized in computer programs to organize information so that a related set of values can be effortlessly sorted or looked.

(www.techterms.com/2020)

For case, a look motor may utilize an array to store web pages found in a look performed by the client. When showing the comes about, the program will yield one component of the array

at a time. This may be done for an

indicated number of values or until all the values put away within the array have been yield. Whereas the program may make a modern variable for each result found, putting away the comes about in an array is much more productive way to oversee memory.

(www.techterms.com/2020)

The sentence structure for putting away and showing the values in an array ordinarily looks something like this:

array name [0] = "This";

array name [1] = "is";

array name [2] = "very easy.";

print array name [0];

print array name [1];

print array name [2];

These commands print by using "while" or "for" loop, like "This is very easy."

Dry run for Linear search using trace table.

Array = { 12, 45, 65, 28, 96, 84, 56 }

Array	Input	Index	Input=Index	Result
12, 45, 65, 28, 96, 84, 56	12	0	True	No. Found on List
12, 45, 65, 28, 96, 84, 56	28	0	False	No. Found in List
		1	False	
		2	False	
		3	True	
12, 45, 65, 28, 96, 84, 56	84	0	False	No. Found in List
		1	False	
		2	False	
		3	False	
		4	False	
		5	True	
12, 45, 65, 28, 96, 84, 56	13	0	False	No.
		1	False	

		2	False	Not Found in List.
		3	False	
		4	False	
		5	False	
		6	False	

Table 2- Trace table using linear search

Dry run for Binary search using trace table.

Array = {5, 8, 15, 22, 26, 28, 30}

Input	Mid=Input	Input>Mid	Input<Mid	Mid-1	Mid+1	Result Displayed
22	yes	no	no	no	no	Number Identified in Array
30	no	yes	no	no	no	Number Identified in Array
8	no	no	yes	no	no	Number identified in Array
40	no	no	no	no	no	Number not identified in Array

Table 3- Trace table using binary search

1.3 Define 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.

Big O notation is employed in computer science to explain the performance or complexity of an algorithm. Big O specifically describes the worst-case scenario, and might be wont to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

(www.rob-bell.net/2020)

Anyone who is read Programming Pearls or the other computer science books and does not have a grounding in mathematics will have hit a wall after they reached chapters that mention $O(N \log N)$ or other seemingly crazy syntax. Hopefully this text will facilitate your gain an understanding of the fundamentals of huge O and Logarithm.

(www.rob-bell.net/2020)

$O(1)$

$O(1)$ describe an algorithm which will always execute within the same time (or space) no matter the scale of the input file set.

$O(N)$

$O(N)$ describes an algorithm whose performance will grow linearly and in direct proportion to the scale of the computer file set. The instance below also demonstrates how big O favours the worst-case performance scenario; an identical string can be found during any iteration of the for loop and therefore the function would return early, but big O notation will always assume the upper limit where the algorithm will perform the greatest number of iterations.

(www.rob-bell.net/2020)

$O(N^2)$

$O(N^2)$ represents an algorithm whose performance is directly proportional to the square of the scale of the input file set. This is often common with algorithms that involve nested iterations over the information set. Deeper nested iterations will lead to $O(N^3)$, $O(N^4)$ etc.

$O(\log N)$

This notation is additionally called logarithm. This suggest that the time consumed increases when the scale of the information set inserted but are going to be reduced once the information set is halved inorder to look the specified element. The foremost suitable example for this can be the binary search, where the element is searched by dividing the information set into 2. Using this sort of notation, it makes binary search efficient upto the extent best when understanding with large volumes of knowledge sets.

(www.rob-bell.net/2020)

$O(2^N)$

$O(2^N)$ denotes an algorithm whose grows doubles with each addition to the computer file set. The expansion curve of an $O(2^N)$ function is exponential – coming out very shallow, then rising meteorically. An example of an $O(2^N)$ function is that the recursive calculation of Fibonacci numbers.

(www.rob-bell.net/2020)

Python code for Linear Search

```
nlist = [12, 5, 9, 16, 7, 68, 33, 75, 16, 24]

print('Items has been in the list: ', nlist)

searchItem = int(input('Input a number to search for: '))

found = False

for i in range(len(nlist)):

    if nlist[i] == searchItem:

        found = True

print(searchItem, ' was found within the list at index ', i)

break

if found == False:
```

Python code for Binary search

```
def binary_search(Arr, lw, hgh, c):  
  
    if hgh >= lw:  
  
        mid = lw + (hgh - lw)/2  
  
        if Arr[mid] == c:  
  
            return 'c is found'  
  
        elif Arr[mid] > c:  
  
            return binary_search(Arr, lw, mid-1w, c)  
  
        else:  
  
            return binary_search(Arr, mid+1w, hgh, c)  
  
    else:  
  
        return 'c is not found'
```

Activity 2

2.1 Define what is meant by a Programming Paradigm. Explain the main characteristics of Procedural, Object oriented and Event-driven paradigms and the relationships among them. Introduction who found it. Why is the reason to have more programming paradigms? (Fincher &Robins,2018)

Similarities and dissimilarities. Pros and cons page 5.

Paradigm also can be termed as technique to clear up some trouble or perform a little task. Programming paradigm is an method to clear up problems using few programming language or additionally we are able to say it is miles a method to clear up a trouble the usage of gear and techniques that are to be had to us following some technique. There are masses for programing language which can be recognized however they all want observe a few approaches whilst they are applied and this system/strategy is paradigm. Other than styles of programing language there are masses of paradigms to meet every and each call for.

(www.geeksforgeeks.org/2020)

Examples for Programming Paradigms:

- Procedural Paradigm
- Object Oriented Paradigm
- Event Driven Paradigm
- Logic Paradigm
- Functional Paradigm
- Database Paradigm

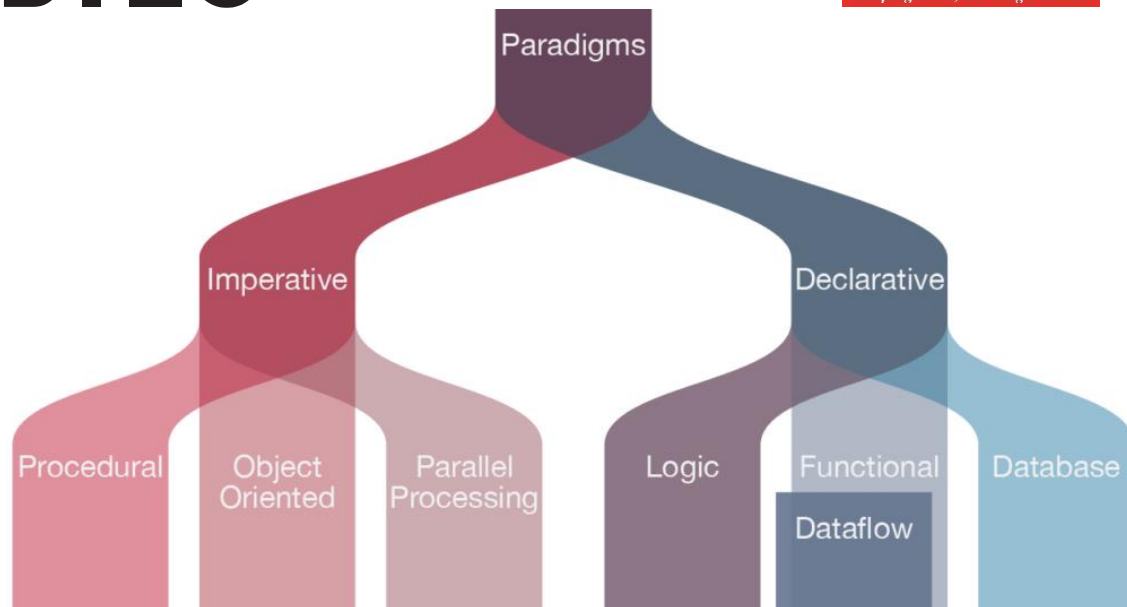


Figure 3- Programming paradigm chart

1. Procedural paradigm:

Procedural programming is a technique of the programming which has assist of splitting the functionalities into a number of strategies. In procedural programming, a huge program is damaged down into smaller potential components called approaches or features. Here, priority is given on functions instead of statistics. In a procedural programming language, an application basically includes a sequence of commands each of which tells the PC to do something together with studying inputs from the consumer, doing the important calculation, displaying output.

(www.tuannguyen.tech/2020)

Characteristics of Procedural paradigm:

- Global data changing by features. Worldwide records can be altered in feature chains.
- Global variables can be share by different features.
- This puts a good deal significance on matters to be executed.
- Large troubles are divided into smaller packages referred to as features.
- Statistics flow brazenly around the gadget from function to function.
- Functions transfer information from one shape to another.
- Employs pinnacle-down method in programming designing.
- Within the cases of massive program, bringing taste is difficult and time eating.

- Suitable and powerful techniques are

unavailable to comfortable statistics of a function from others.

2. Object Oriented paradigm:

Inside the elegance-based object-oriented programming paradigm, ‘Object’ refers to a specific example of a category wherein the item may be a combination of variables, features and statistics structures. A terrific information of object-oriented paradigms standards can help in decision making when designing a software. The way you must layout an application and what language have to be used. This is a programming style that is related to the ideas like elegance, object, inheritance, encapsulation, abstraction and polymorphism.

(www.blog.usejournal.com/2020)

Characteristics of Object-Oriented paradigm:

- Emphasis on facts rather than process
- Programs are divided into entities referred to as item
- Facts structures are designed such that they signify gadgets
- Features that operate on records of an object are tied collectively in records systems
- Data is hidden and can not be accessed by way of outside features
- Objects communicate with every other through capabilities
- New facts and features may be easily delivered each time vital

3. Event Driven paradigm:

This is a computer programming paradigm wherein manipulate go with the flow of the program is determined by way of the prevalence of activities. These activities are monitored with the aid of code known as an occasion listener. If it detects that an assigned occasion has passed off. It is running an event handler.

(www.computerhope.com/2020)

Characteristics of Event-Driven paradigm:

- The components of an application do not interact directly with each other, but through implicit invocation
- The implicit invocation is facilitated by an infrastructure called Event service
- Implicit invocation may happen according to the variants

Relationships among procedural, Object-oriented and Event-driven paradigms.

	Advantages	Disadvantages	Differences
Procedural	<p>Easy and simple to code.</p> <p>Codes have ability to be reused in several parts of the program.</p> <p>This takes less memory on computer.</p> <p>Easy to tracking the flow of codes.</p> <p>Regarded best for general programming implement.</p>	<p>Difficult to identify global data within the large programs.</p> <p>Maintaining and enhancing is difficult because of global data.</p> <p>Have to focus on functions rather than data.</p>	<p>Provide character user interface.</p> <p>Commands are writing and execute in linear fashion.</p> <p>Focus on sequential execution of steps.</p> <p>Most common languages which follow this.</p>
Object-Oriented	<p>Easy to manage codes.</p> <p>Easy to maintain codes.</p> <p>Useful data only visible in abstraction.</p> <p>Code is reusability.</p> <p>Codes are easy to manage.</p>	<p>Programmer need many skills.</p> <p>Designing and implementation is complex.</p> <p>This got large size comparison.</p> <p>Slow because of large size.</p>	<p>Provides command writing in module.</p> <p>Functions are preparing for interaction to perform specific task.</p> <p>Focus on object or data.</p> <p>Most common languages which follow this.</p>

Event-Driven	<p>Allows more interactive programs.</p> <p>Simple to add new functionality.</p> <p>Complete control over scheduling decisions.</p> <p>Code can reuse.</p>	<p>Tricky to code.</p> <p>Slower to execute.</p> <p>Failure can halt whole server.</p> <p>Harder to debug system level errors.</p>	<p>Provides graphical user interface.</p> <p>Actions are defined on events.</p> <p>Focus on selecting user interface.</p> <p>Most common languages which follow this.</p>
--------------	--	--	---

Table 4- Relationships among procedural, Object-oriented and Event-driven paradigms

2.2 Write small snippets of code as example for the above three programming paradigms using a suitable programming language(s).

Procedural Programming using C++:

The language used to represent the paradigm is C++. The program is where the user is able to enter two numbers to the system and the sum is shown in console application.

Code:

```
#include <iostream>

using namespace std;

int main ()
{
    int a = 50;
    int b = 28;
    int total=0;

    total = a + b;

    cout<<"total is "<<total<<endl;
    system("pause");

    return 0;
}
```

Output:

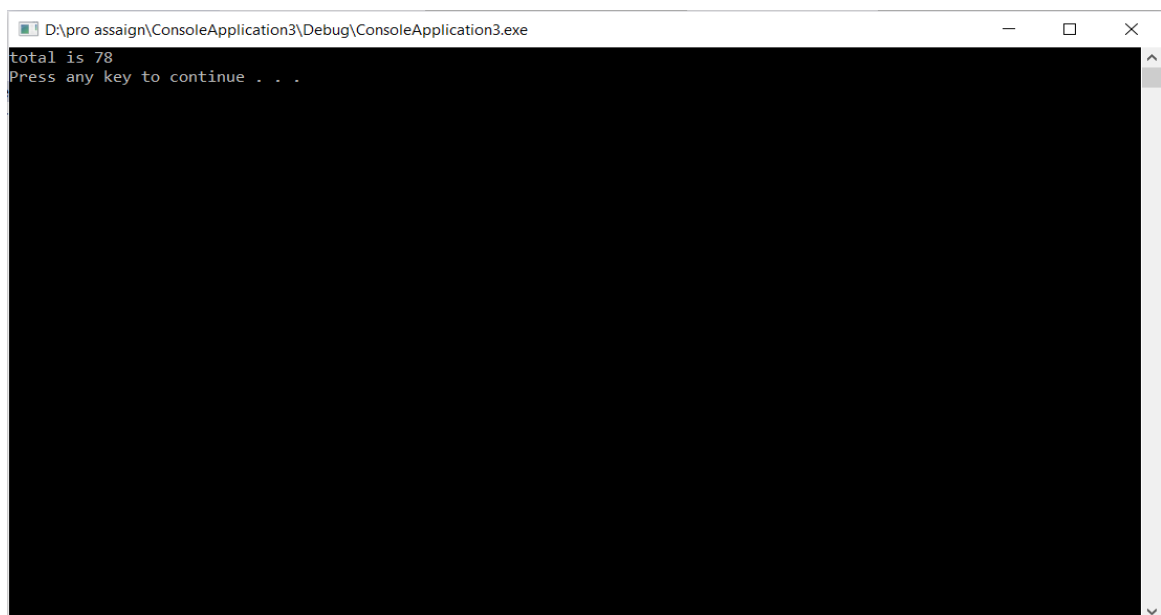


Figure 4-C++ program output

Object-Oriented programming using c#:

The language used to represent this paradigm is C#. A program is being designed to calculate the result when numbers are entered. The result is to be displayed by a message box. Three labels along with 3 textboxes have been used together with four buttons in order to obtain the expected result.

Code:

```
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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button5_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            int INPUT_1, INPUT_2, INPUT_3;
            INPUT_1 = int.Parse(textBox1.Text);
            INPUT_2 = int.Parse(textBox2.Text);

            INPUT_3 = INPUT_1 + INPUT_2;
            textBox3.Text = INPUT_3.ToString();

            MessageBox.Show("Result is " + INPUT_3.ToString());
        }

        private void button4_Click(object sender, EventArgs e)
        {
            int INPUT_1, INPUT_2, INPUT_3;
            INPUT_1 = int.Parse(textBox1.Text);
            INPUT_2 = int.Parse(textBox2.Text);
```

```

INPUT_3 = INPUT_1 / INPUT_2;
textBox3.Text = INPUT_3.ToString();

    MessageBox.Show("Result is " + INPUT_3.ToString());
}

private void button1_Click(object sender, EventArgs e)
{
    int INPUT_1, INPUT_2, INPUT_3;
    INPUT_1 = int.Parse(textBox1.Text);
    INPUT_2 = int.Parse(textBox2.Text);

    INPUT_3 = INPUT_1 * INPUT_2;
    textBox3.Text = INPUT_3.ToString();

    MessageBox.Show("Result is " + INPUT_3.ToString());
}

private void button2_Click(object sender, EventArgs e)
{
    int INPUT_1, INPUT_2, INPUT_3;
    INPUT_1 = int.Parse(textBox1.Text);
    INPUT_2 = int.Parse(textBox2.Text);
    INPUT_3 = INPUT_1 - INPUT_2;
    textBox3.Text = INPUT_3.ToString();

    MessageBox.Show("Result is " + INPUT_3.ToString());
}
}
}

```

Before Output:

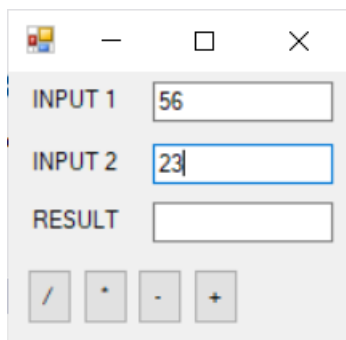


Figure 5- C# calculator interface before calculate

Output:

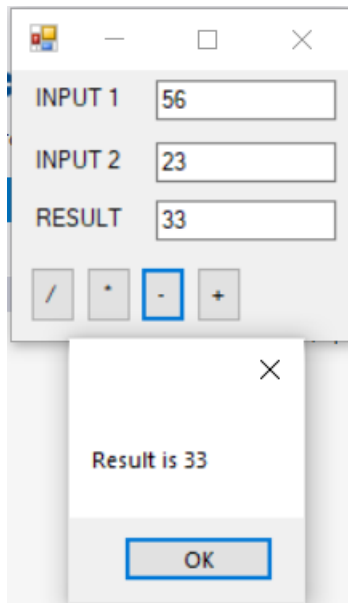


Figure 6- After calculate

Event-driven programming using Python:

The language used to represent this paradigm is Python. The system is created to display the square root of the particular number which is entered by the user.

Code:

```
# Python Program to calculate the square root
```

```
# Note: change this value for a different result
```

```
num = 12
```

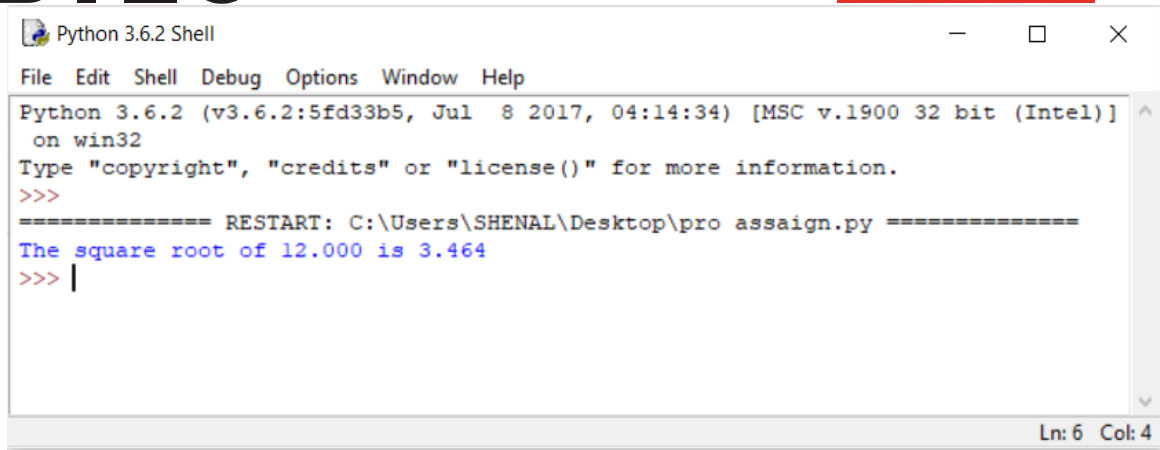
```
# To take the input from the user
```

```
#num = float(input('Enter a number: '))
```

```
num_sqrt = num ** 0.5
```

```
print("The square root of %0.3f is %0.3f"%(num ,num_sqrt))
```

Output:



```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\SHENAL\Desktop\pro assaign.py =====
The square root of 12.000 is 3.464
>>> |
```

Ln: 6 Col: 4

Figure 7-Python output

2.3 Critically evaluate the code samples that you have above in relation to their structure and the unique characteristics.

Procedural Programming using C++:

Code:

```
#include <iostream>

using namespace std;

int main ()
{
    int a = 50;
    int b = 28;
    int total=0;

    total = a + b;

    cout<<"total is "<<total<<endl;
    system("pause");

    return 0;
}
```

```
#include <iostream>
```

Iostream stands for standard input-output stream. This header record contains definitions to objects like cin, cout, cerr etc.

(www.geeksforgeeks.org/2020)

```
using namespace std;
```

This implies we utilize the namespace named std. Std is a shortened form for standard. So, which means we utilize all the things with in std namespace.

(www.medium.com/2020)

```
int main ()
```

This implies that our work has to return a few numbers at the conclusion of the execution and we do so by returning at the conclusion of the program. 0 is the standard for the “successful execution of the program”.

(www.codesdope.com/2020)

c#:

```
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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button5_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            int INPUT_1, INPUT_2, INPUT_3;
            INPUT_1 = int.Parse(textBox1.Text);
            INPUT_2 = int.Parse(textBox2.Text);

            INPUT_3 = INPUT_1 + INPUT_2;
            textBox3.Text = INPUT_3.ToString();

            MessageBox.Show("Result is " + INPUT_3.ToString());
        }

        private void button4_Click(object sender, EventArgs e)
        {
            int INPUT_1, INPUT_2, INPUT_3;
            INPUT_1 = int.Parse(textBox1.Text);
            INPUT_2 = int.Parse(textBox2.Text);

            INPUT_3 = INPUT_1 / INPUT_2;
            textBox3.Text = INPUT_3.ToString();

            MessageBox.Show("Result is " + INPUT_3.ToString());
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int INPUT_1, INPUT_2, INPUT_3;
            INPUT_1 = int.Parse(textBox1.Text);
            INPUT_2 = int.Parse(textBox2.Text);
```

```

INPUT_3 = INPUT_1 * INPUT_2;
textBox3.Text = INPUT_3.ToString();

    MessageBox.Show("Result is " + INPUT_3.ToString());
}

private void button2_Click(object sender, EventArgs e)
{
    int INPUT_1, INPUT_2, INPUT_3;
    INPUT_1 = int.Parse(textBox1.Text);
    INPUT_2 = int.Parse(textBox2.Text);
    INPUT_3 = INPUT_1 - INPUT_2;
    textBox3.Text = INPUT_3.ToString();

    MessageBox.Show("Result is " + INPUT_3.ToString());
}
}
}

```

partial class

It gives an uncommon capacity to execute the usefulness of a single course into different records and all these records are combined into a single course record when the application is compiled.

(www.geeksforgeeks.org/2020)

Public

This is a keyword use to declare accessibility of type and type member such that access is not limited. This is one of access modifier provide complete visibility to all types and members.

(www.techopedia.com/2020)

Private

Type or member can as it were be gotten to by code within the same class or struct.

(www.stackoverflow.com/2020)

Event-driven programming using Python:

Code:

```
# Python Program to calculate the square root

# Note: change this value for a different result
num = 12

# To take the input from the user
#num = float(input('Enter a number: '))

num_sqrt = num ** 0.5
print("The square root of %0.3f is %0.3f"%(num ,num_sqrt))

print ()
```

This work prints the required message to the screen, or other standard output device. The message can be string, or any other protest, the question will be changed over into a string sometime recently composed to the screen.

(www.w3schools.com/2020)

Activity 3

3.1 Design suitable algorithms for vehicle tariff calculation for rents and hires.

Flow chart to rent calculation:

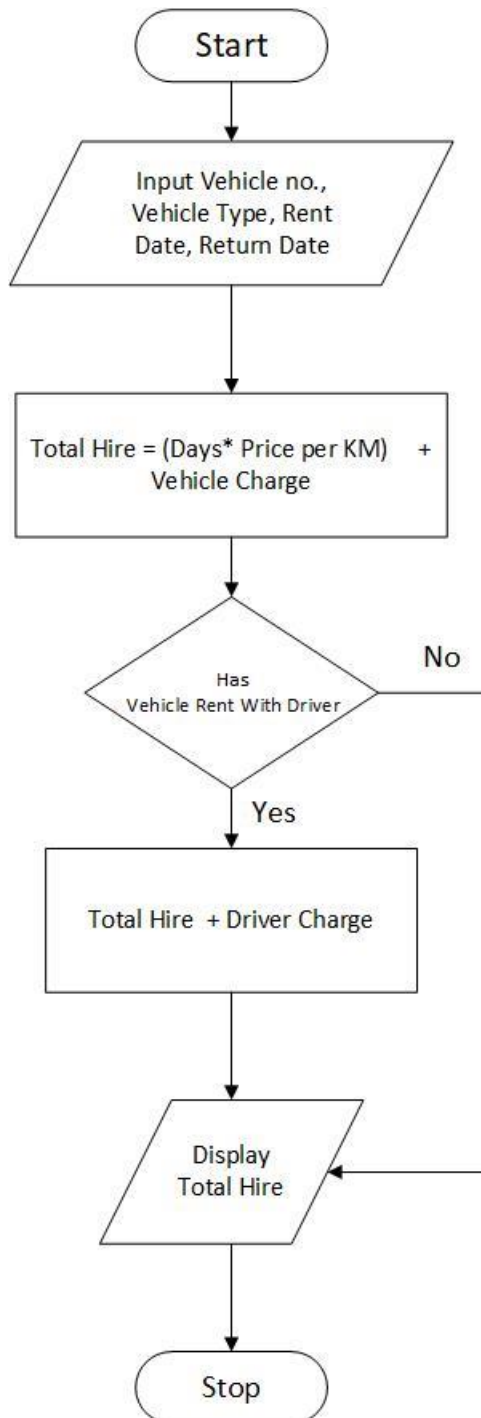


Figure 8-Rent calculation flow chart

Flowchart to day tour calculation:

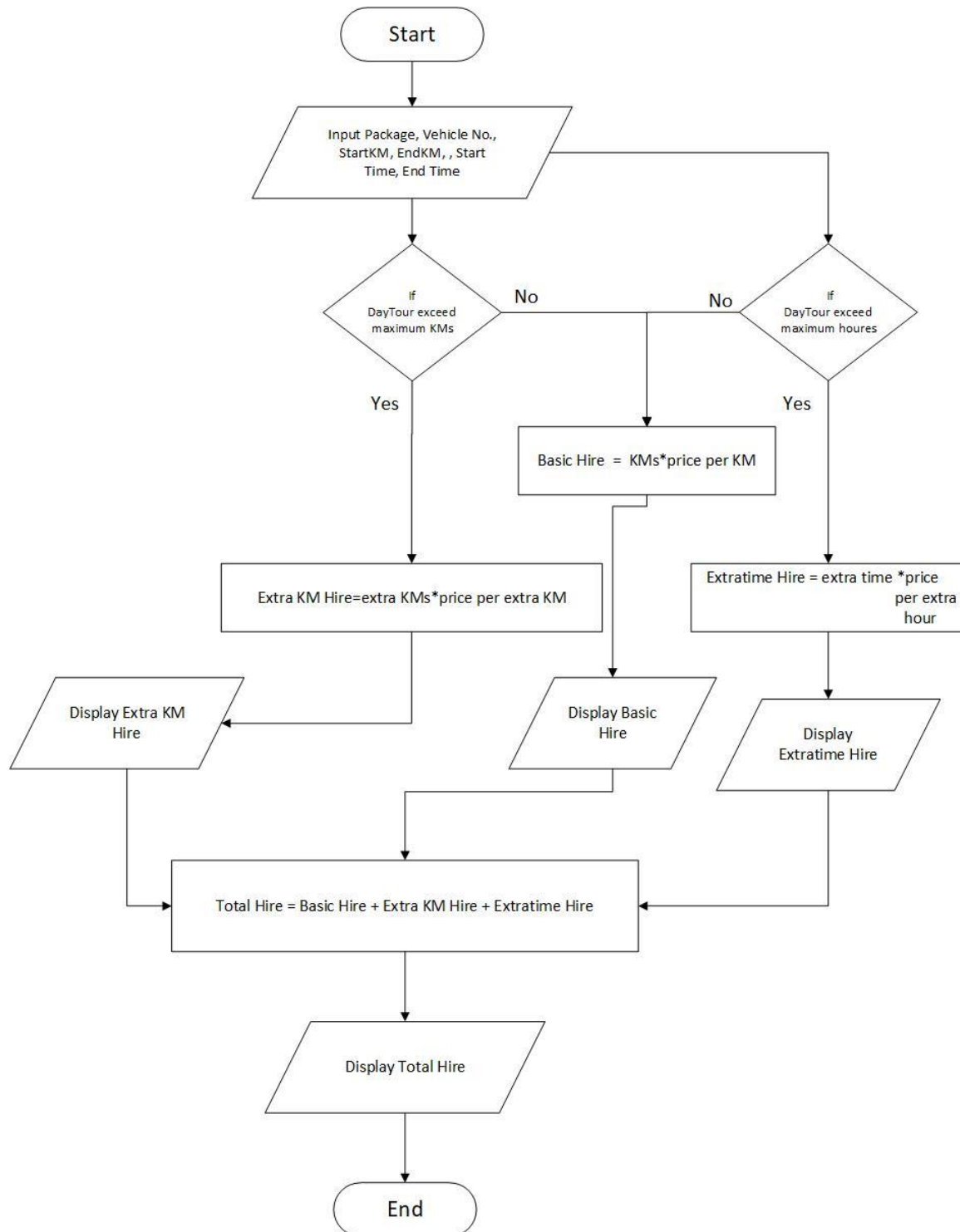


Figure 9-Day tour calculation flow chart

Flowchart to long tour calculation:

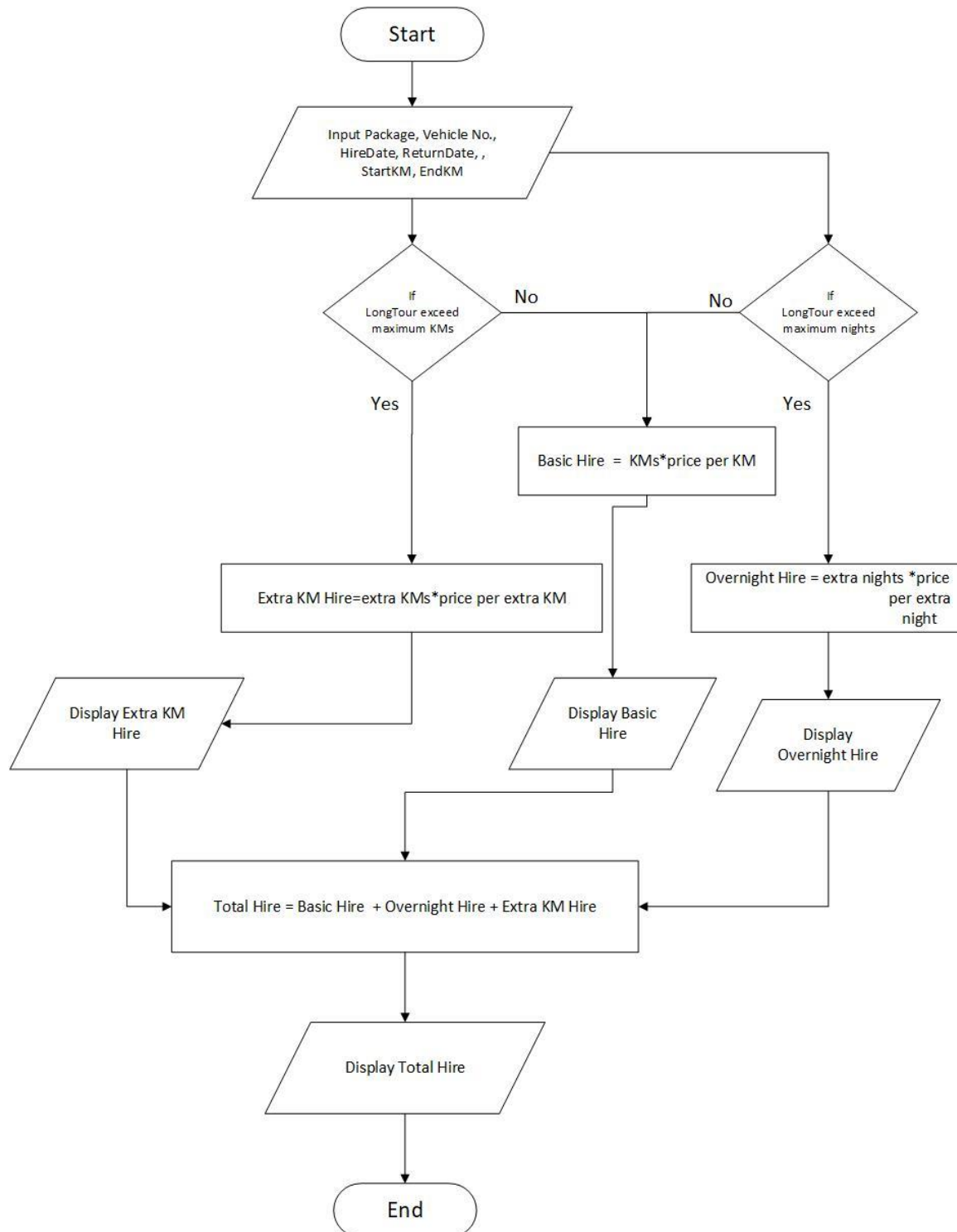
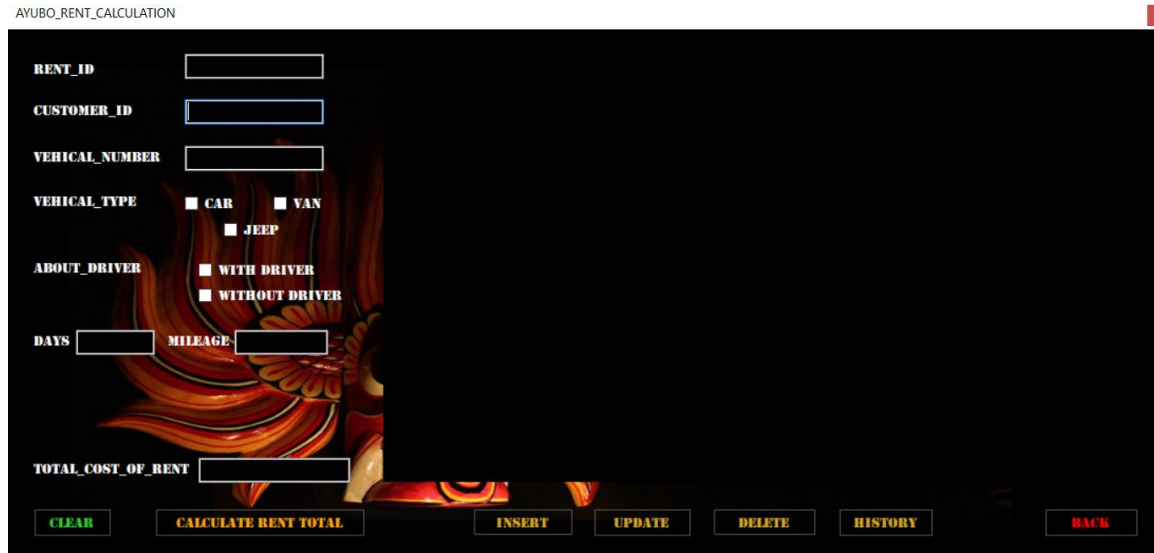


Figure 10- Long tour calculation flow chart

3.2 Implement the above algorithms using visual studio IDE (using C#.net) 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.

Interface of Rent Calculation:



AYUBO_RENT_CALCULATION

RENT_ID:

CUSTOMER_ID:

VEHICAL_NUMBER:

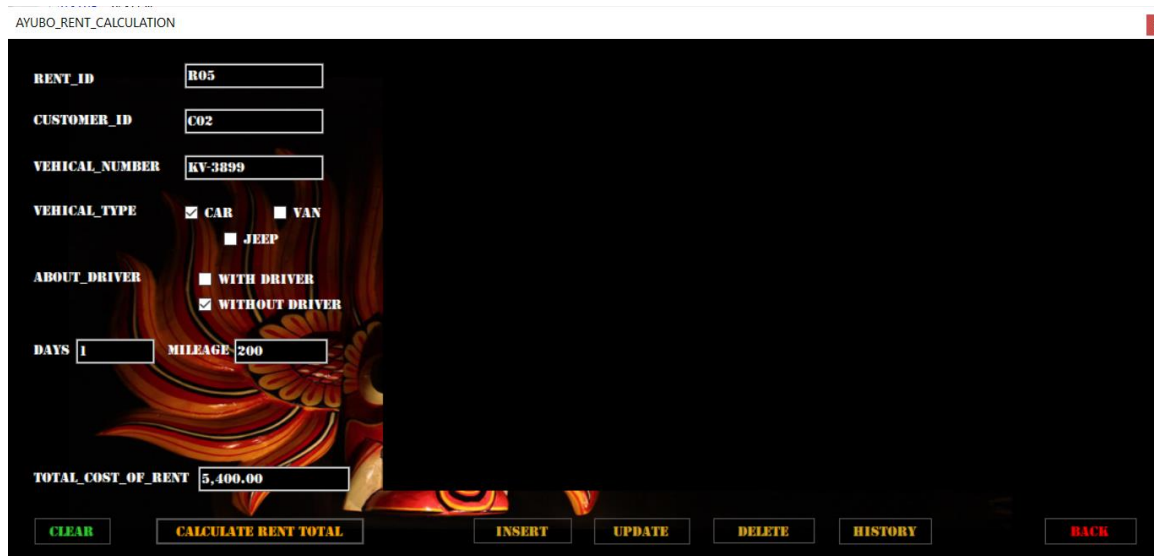
VEHICAL_TYPE: ☐ CAR ☐ VAN ☐ JEEP

ABOUT_DRIVER: ☐ WITH DRIVER ☐ WITHOUT DRIVER

DAYS: MILEAGE:

TOTAL_COST_OF_RENT:

Figure 11-Rent calculation interface



AYUBO_RENT_CALCULATION

RENT_ID:

CUSTOMER_ID:

VEHICAL_NUMBER:

VEHICAL_TYPE: ☒ CAR ☐ VAN ☐ JEEP

ABOUT_DRIVER: ☐ WITH DRIVER ☒ WITHOUT DRIVER

DAYS: MILEAGE:

TOTAL_COST_OF_RENT:

Figure 12-After calculation

```
private void button1_Click(object sender, EventArgs e) //total calculation button
code
{
    string CUSTOMER_ID, ABOUT_DRIVER, VEHICAL_TYPE, VEHICAL_NUMBER, RENT_ID;
    decimal TOTAL_COST_OF_RENT, DAYS, MILEAGE;
    RENT_ID = textBox8.Text;
    CUSTOMER_ID = textBox1.Text;
    ABOUT_DRIVER = string.Empty;
    if (checkBox4.Checked)
        if (checkBox5.Checked)
            VEHICAL_TYPE = String.Empty;
    if (checkBox1.Checked)
        if (checkBox2.Checked)
            if (checkBox3.Checked)

                VEHICAL_NUMBER = textBox3.Text;
    DAYS = decimal.Parse(textBox5.Text);
    MILEAGE = decimal.Parse(textBox6.Text);
    TOTAL_COST_OF_RENT = 0;

    if (checkBox1.Checked)
//when select vehical type car
    {
        if (checkBox4.Checked)
//when select vehical type car and with driver

            TOTAL_COST_OF_RENT = (1500 + (20 * MILEAGE)) - (100 * DAYS) +
2500;

        else
//when select vehical type car and without driver
            TOTAL_COST_OF_RENT = (1500 + (20 * MILEAGE)) - (100 * DAYS);
    }
    else if (checkBox2.Checked)
//when select vehical type jeep
    {
        if (checkBox4.Checked)
//when select vehical type jeep and with driver
            TOTAL_COST_OF_RENT = (3000 + (30 * MILEAGE)) - (80 * DAYS) + 2500;
        else
//when select vehical type jeep and without driver
            TOTAL_COST_OF_RENT = (3000 + (30 * MILEAGE)) - (80 * DAYS);
    }
    else if (checkBox3.Checked)
//when select vehical type van
    {
        if (checkBox4.Checked)
//when select vehical type van and with driver
            TOTAL_COST_OF_RENT = (4500 + (50 * MILEAGE)) - (60 * DAYS) + 2500;
        else
//when select vehical type van and without driver
            TOTAL_COST_OF_RENT = (4500 + (50 * MILEAGE)) - (60 * DAYS);
    }
    else
    {
        MessageBox.Show("INVALID !");
    }
}

    textBox7.Text = TOTAL_COST_OF_RENT.ToString("N");
//after calculation finish, result will be show on total cost of rent's textbox
```



```

    }
    MessageBox.Show("YOUR TOTAL COST OF RENT IS Rs:" + textBox7.Text);
    //after calculation finish, result will be show in a messagebox

}

```

Interface for Day Tour calculation:

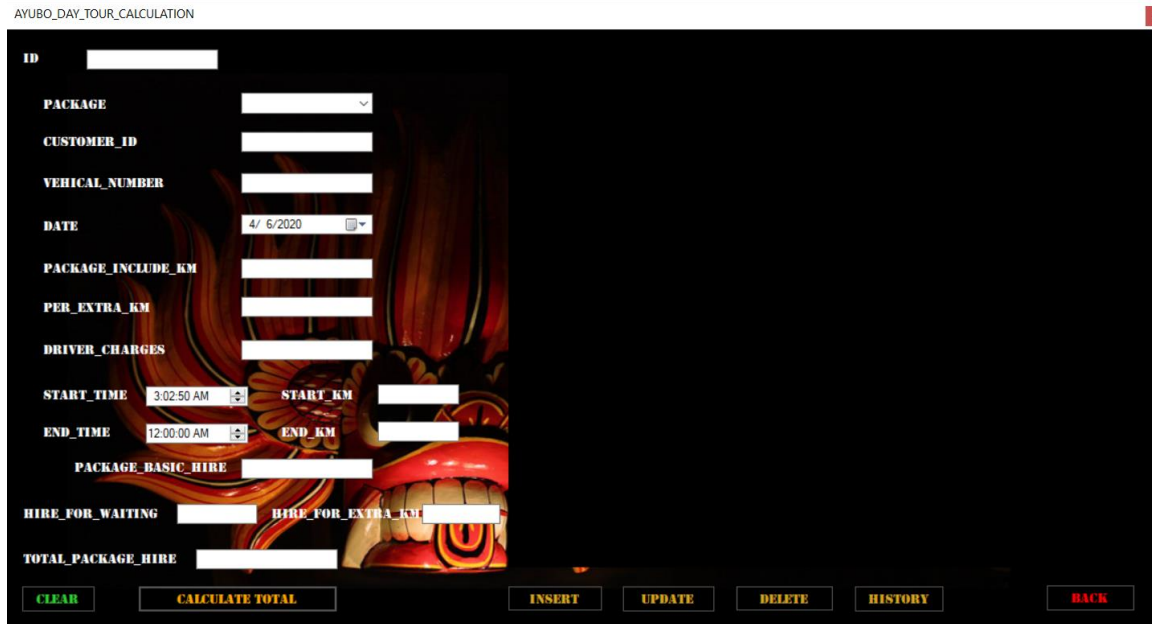


Figure 13-Day tour calculation interface

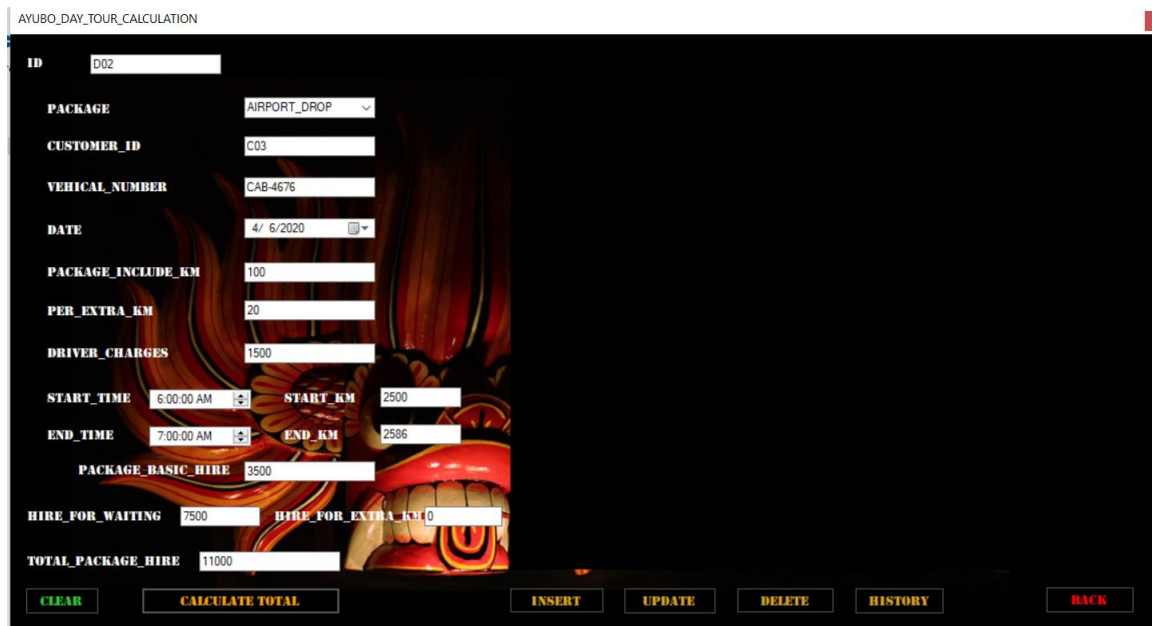


Figure 14-After calculate

Day Tour calculation C# code:

```

private void button1_Click(object sender, EventArgs e)    //calculate button code

```

```

    {
        int HIRE_FOR_WAITING = 0;
        int HIRE_FOR_EXTRA_KM = 0;

        DateTime ST = dateTimePicker2.Value;
        DateTime ET = dateTimePicker3.Value;
        int TT = (int)ET.Subtract(ST).TotalHours + 1;
        int SK = int.Parse(textBox3.Text);
        int EK = int.Parse(textBox4.Text);
        int TK = EK - SK;
        int PACKAGE_BASIC_HIRE, PACKAGE_INCLUDED_KM, PER_EXTRA_KM, DRIVER_CHARGES,
TOTAL_PACKAGE_HIRE;
        PACKAGE_INCLUDED_KM = int.Parse(textBox2.Text);
        PER_EXTRA_KM = int.Parse(textBox6.Text);
        DRIVER_CHARGES = int.Parse(textBox7.Text);

        PACKAGE_BASIC_HIRE = PACKAGE_INCLUDED_KM * PER_EXTRA_KM + DRIVER_CHARGES;
        textBox5.Text = PACKAGE_BASIC_HIRE.ToString();

        if (TK > 250)
        {
            HIRE_FOR_EXTRA_KM = (TK - 250) * 100;           //Rs.100
per Extra KM Charges
        }

        if (TT > 4 || TT <= 12)                             //4 hours
is time allowed.
        {
            HIRE_FOR_WAITING = (4 - TT) * 150;

        }

        TOTAL_PACKAGE_HIRE = PACKAGE_BASIC_HIRE + HIRE_FOR_WAITING +
HIRE_FOR_EXTRA_KM;
        textBox12.Text = TOTAL_PACKAGE_HIRE.ToString(); ;
        textBox8.Text = HIRE_FOR_WAITING.ToString();

        textBox9.Text = HIRE_FOR_EXTRA_KM.ToString();
        textBox12.Text = TOTAL_PACKAGE_HIRE.ToString();

    }

```

Interface for Long Tour calculation:

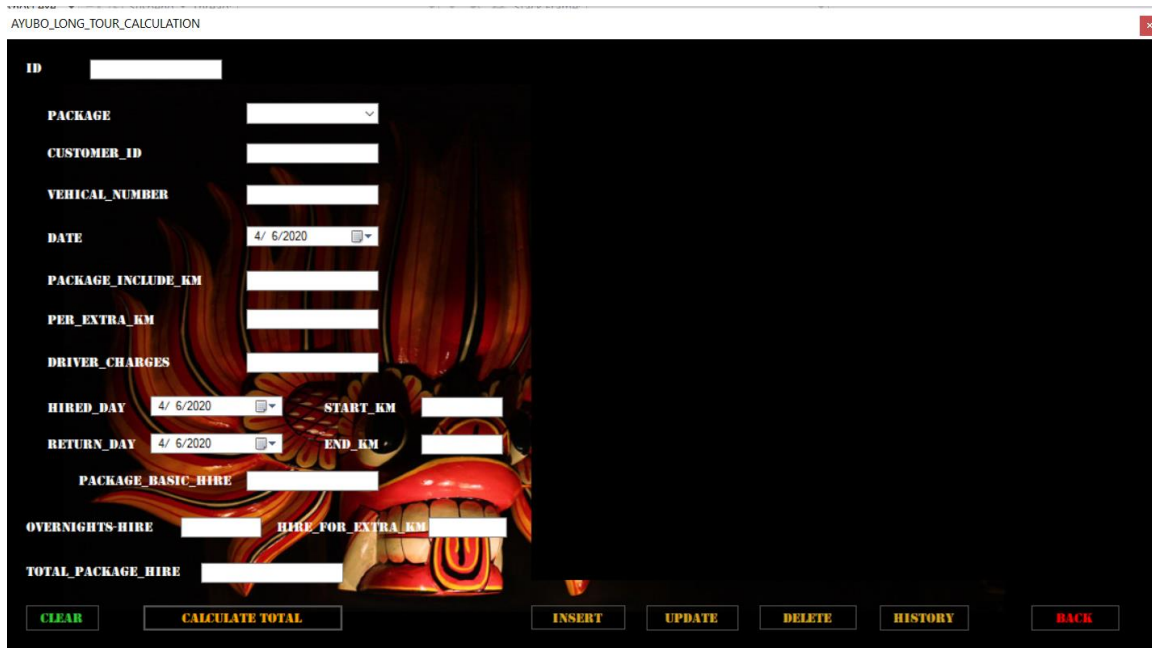


Figure 15- Long tour calculation interface

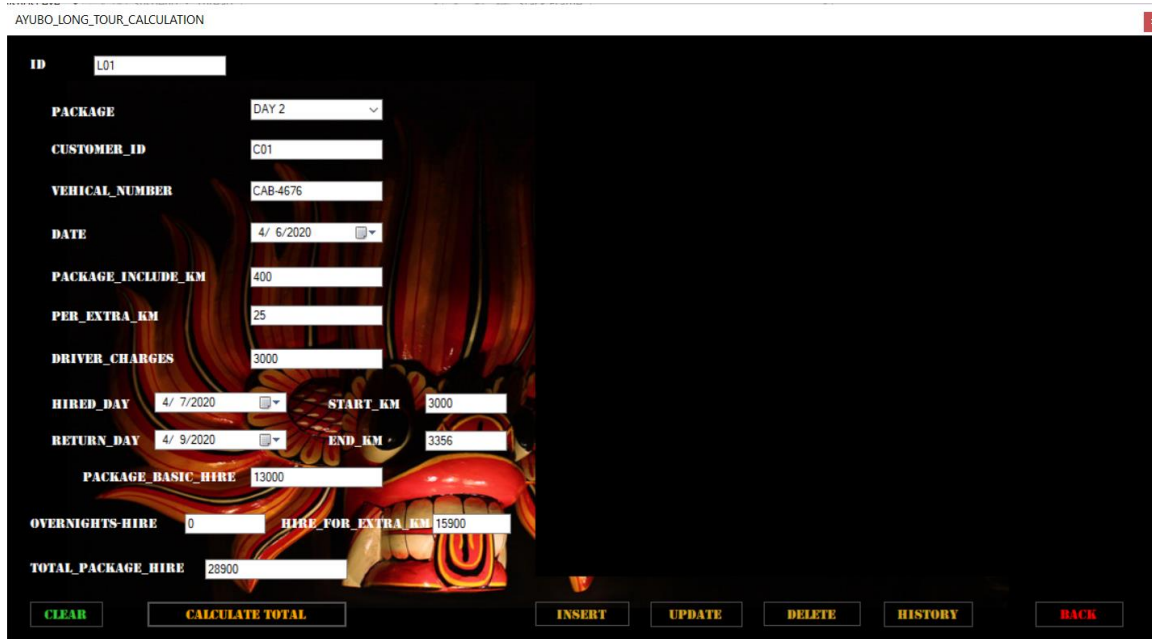


Figure 16-After calculate

Long Tour calculation C# code:

```
private void button1_Click(object sender, EventArgs e) //calculate button
code
{
    int PACKAGE_INCLUDE_KM, PER_EXTRA_KM,
    DRIVER_CHARGES, PACKAGE_BASIC_HIRE, TOTAL_PACKAGE_HIRE;

    int OVERNIGHTS_HIRE = 0;
    int HIRE_FOR_EXTRA_KM = 0;
```

```

DateTime D1 = dateTimePicker2.Value;
DateTime D2 = dateTimePicker3.Value;
int ND = (int)D2.Subtract(D1).TotalDays + 1;
int SK = int.Parse(textBox3.Text);
int EK = int.Parse(textBox4.Text);
int TEK = EK - SK;

if (TEK > 250)
{
    HIRE_FOR_EXTRA_KM = (TEK - 250) * 150;           //150
Extra KM CHARGE
}
if (ND > 3)
{
    OVERNIGHTS_HIRE = (ND - 3) * 500 ;           //500
Extra night
}

PACKAGE_INCLUDE_KM = int.Parse(textBox2.Text);
PER_EXTRA_KM = int.Parse(textBox6.Text);
DRIVER_CHARGES = int.Parse(textBox7.Text);
PACKAGE_BASIC_HIRE = PACKAGE_INCLUDE_KM * PER_EXTRA_KM + DRIVER_CHARGES;
textBox5.Text = PACKAGE_BASIC_HIRE.ToString();

TOTAL_PACKAGE_HIRE = OVERNIGHTS_HIRE + HIRE_FOR_EXTRA_KM +
PACKAGE_BASIC_HIRE;
textBox9.Text = HIRE_FOR_EXTRA_KM.ToString();
textBox8.Text = OVERNIGHTS_HIRE.ToString();
textBox12.Text = TOTAL_PACKAGE_HIRE.ToString();

}

```

Interface for Vehicle Registration:

AYUBO_VEHICAL_REGISTRATION

VEHICAL_NUMBER	<input type="text"/>
REGISTER_DATE	Monday April <input type="text"/>
MAKE	<input type="text"/>
MODEL	<input type="text"/>
MAKE_YEAR	<input type="text"/>
VEHICAL_TYPE	<input type="text"/>
FUEL_TYPE	<input type="text"/>
NUMBER_OF_SEATS	<input type="text"/>
COLOUR	<input type="text"/>
TRANSMISSION	<input type="text"/>
OWNER	<input type="text"/>
OWNER'S_CONTACT_NUMBER	<input type="text"/>

Figure 18-Vehical registration interface

Select vehicle type:

AYUBO_VEHICAL_REGISTRATION

VEHICAL_NUMBER	CAV-9663
REGISTER_DATE	Monday April <input type="text"/>
MAKE	HONDA
MODEL	INSIGHT
MAKE_YEAR	2015
VEHICAL_TYPE	<input type="text"/>
FUEL_TYPE	<input type="text"/>
NUMBER_OF_SEATS	<input type="text"/>
COLOUR	<input type="text"/>
TRANSMISSION	<input type="text"/>
OWNER	<input type="text"/>
OWNER'S_CONTACT_NUMBER	<input type="text"/>

Figure 17-Select vehical type

Vehicle register data insert:

AYUBO_VEHICAL_REGISTRATION

VEHICAL_NUMBER	CAV-9663
REGISTER_DATE	Monday April 6
MAKE	HONDA
MODEL	INSIGHT
MAKE_YEAR	2015
VEHICAL_TYPE	CAR
FUEL_TYPE	HYBRID
NUMBER_OF_SEATS	5
COLOUR	GREY
TRANSMISSION	AUTO
OWNER	MUDITHA
OWNER'S_CONTACT_NUMBER	0720548914

DATA INSERTED !

OK

CLEAR INSERT UPDATE DELETE SHOW BACK

Figure 19-Data insert

Data insert code:

```
private void button2_Click(object sender, EventArgs e) //insert button code
{
    //insert data to database
    con.Open();
    SqlCommand cmd = new SqlCommand("insert into
[vreg](VEHICAL_NUMBER,REGISTER_DATE,MAKE,MODEL,MAKE_YEAR,VEHICAL_TYPE,FUEL_TYPE,NUMBER
_OF_SEATS,COLOUR,TRANSMISSION,OWNER,OWNER_CONTACT_NUMBER)
values(@VEHICAL_NUMBER,@REGISTER_DATE,@MAKE,@MODEL,@MAKE_YEAR,@VEHICAL_TYPE,@FUEL_TYPE
,@NUMBER_OF_SEATS,@COLOUR,@TRANSMISSION,@OWNER,@OWNER_CONTACT_NUMBER)", con);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
    cmd.Parameters.AddWithValue("@REGISTER_DATE", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("@MAKE", comboBox1.Text);
    cmd.Parameters.AddWithValue("@MODEL", textBox2.Text);
    cmd.Parameters.AddWithValue("@MAKE_YEAR", textBox3.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_TYPE", comboBox2.Text);
    cmd.Parameters.AddWithValue("@FUEL_TYPE", comboBox3.Text);
    cmd.Parameters.AddWithValue("@NUMBER_OF_SEATS", textBox4.Text);
    cmd.Parameters.AddWithValue("@COLOUR", textBox5.Text);
    cmd.Parameters.AddWithValue("@TRANSMISSION", comboBox4.Text);
    cmd.Parameters.AddWithValue("@OWNER", textBox6.Text);
    cmd.Parameters.AddWithValue("@OWNER_CONTACT_NUMBER", textBox7.Text);
    cmd.ExecuteNonQuery();
    MessageBox.Show("DATA INSERTED !"); //after insert message
    con.Close();
    disp_data();
}
```


Vehicle register data update:

AYUBO_VEHICAL_REGISTRATION

VEHICAL_NUMBER
REGISTER_DATE
MAKE
MODEL
MAKE_YEAR
VEHICAL_TYPE
FUEL_TYPE
NUMBER_OF_SEATS
COLOUR
TRANSMISSION
OWNER
OWNER'S_CONTACT_NUMBER

VEHICAL_NUMBE	REGISTER_DATE	MAKE	MODEL	MAKE_YEAR	VEHICAL_TYPE	FUEL
CAB-4576	Saturday, April 4, ...	TOYOTA	AXIO	2015	CAR	HYBR
CAV-9663	Monday, April 6, ...	HONDA	INSIGHT	2015	CAR	HYBR

DATA UPDATED !

OK

CLEAR

INSERT

UPDATE

DELETE

SHOW

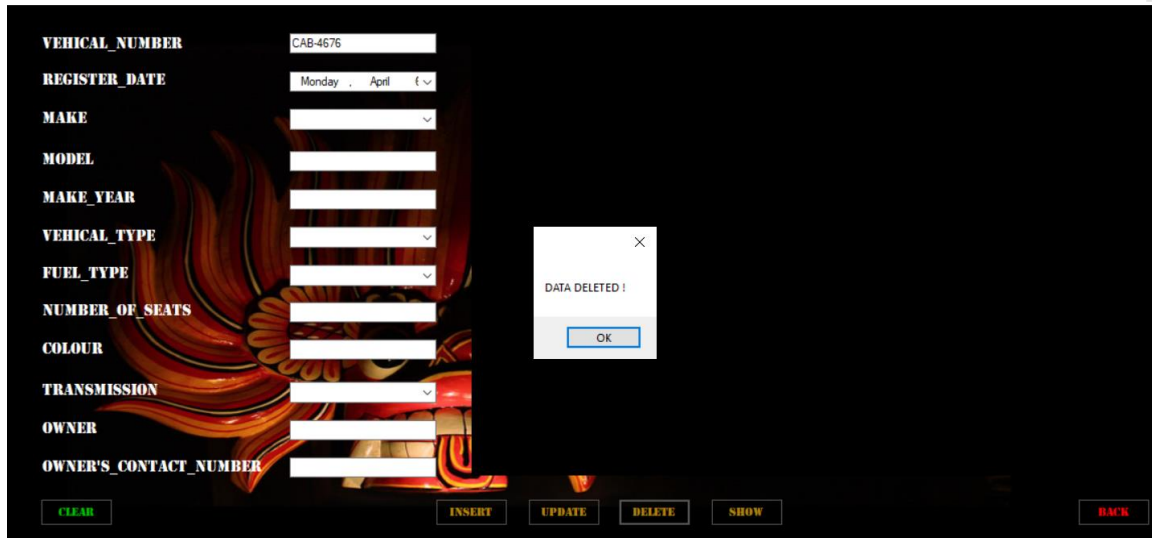
BACK

Figure 20-Data update

Data Update code:

```
private void button3_Click(object sender, EventArgs e) //update button code
{
    //update selected data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("update vreg set
REGISTER_DATE=@REGISTER_DATE,MAKE=@MAKE,MODEL=@MODEL,MAKE_YEAR=@MAKE_YEAR,VEHICAL_TYPE
=@VEHICAL_TYPE,FUEL_TYPE=@FUEL_TYPE,NUMBER_OF_SEATS=@NUMBER_OF_SEATS,COLOUR=@COLOUR,TR
ANSMISSION=@TRANSMISSION,OWNER=@OWNER,OWNER_CONTACT_NUMBER=@OWNER_CONTACT_NUMBER where
VEHICAL_NUMBER='" + textBox1.Text + "'", con);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
    cmd.Parameters.AddWithValue("@REGISTER_DATE", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("@MAKE", comboBox1.Text);
    cmd.Parameters.AddWithValue("@MODEL", textBox2.Text);
    cmd.Parameters.AddWithValue("@MAKE_YEAR", textBox3.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_TYPE", comboBox2.Text);
    cmd.Parameters.AddWithValue("@FUEL_TYPE", comboBox3.Text);
    cmd.Parameters.AddWithValue("@NUMBER_OF_SEATS", textBox4.Text);
    cmd.Parameters.AddWithValue("@COLOUR", textBox5.Text);
    cmd.Parameters.AddWithValue("@TRANSMISSION", comboBox4.Text);
    cmd.Parameters.AddWithValue("@OWNER", textBox6.Text);
    cmd.Parameters.AddWithValue("@OWNER_CONTACT_NUMBER", textBox7.Text);
    cmd.ExecuteNonQuery();
    MessageBox.Show("DATA UPDATED !"); //after update message
    con.Close();
    disp_data();
}
```

AYUBO_VEHICAL_REGISTRATION



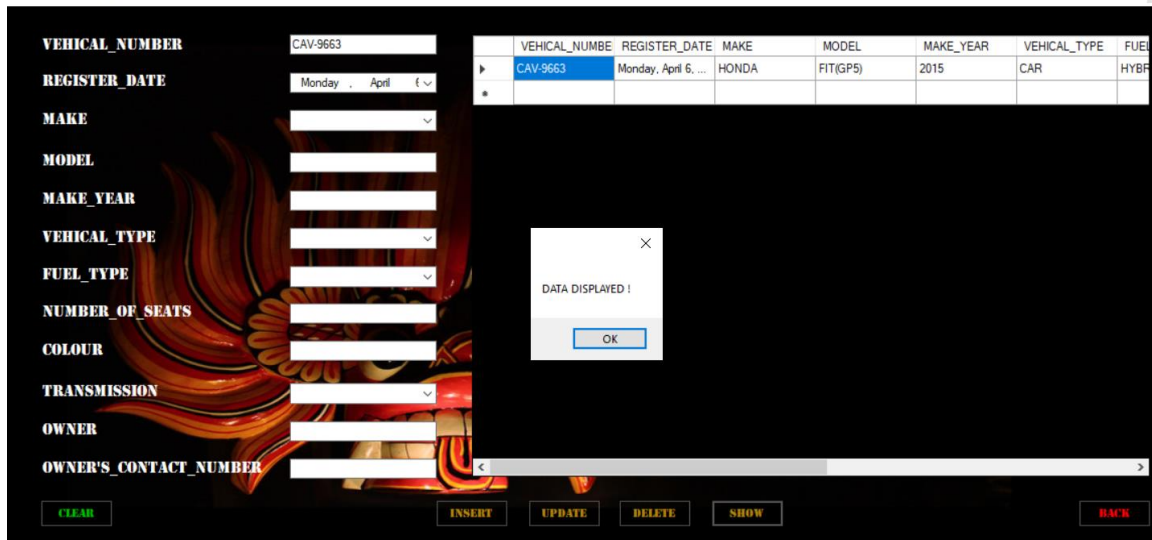
The screenshot shows a web application for vehicle registration. On the left, there is a form with the following fields: VEHICAL_NUMBER (text input with value 'CAB-4676'), REGISTER_DATE (date picker showing 'Monday . April'), MAKE (dropdown), MODEL (text input), MAKE_YEAR (text input), VEHICAL_TYPE (dropdown), FUEL_TYPE (dropdown), NUMBER_OF_SEATS (text input), COLOUR (text input), TRANSMISSION (dropdown), OWNER (text input), and OWNER'S_CONTACT_NUMBER (text input). Below the form are buttons for CLEAR, INSERT, UPDATE, DELETE, and SHOW. A modal dialog box is open in the center, displaying 'DATA DELETED !' with an OK button.

Figure 21-Data delete

Data delete code:

```
private void button4_Click(object sender, EventArgs e) //delete button code
{
    //delete selected data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("delete from vreg where VEHICAL_NUMBER='"
+ textBox1.Text + "'", con);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("DATA DELETED !");
    disp_data();
}
```


AYUBO_VEHICAL_REGISTRATION



VEHICAL_NUMBE	REGISTER_DATE	MAKE	MODEL	MAKE_YEAR	VEHICAL_TYPE	FUEL
CAV-9663	Monday, April 6, ...	HONDA	FIT(GP5)	2015	CAR	HYBR

Figure 22-Data displayed

Selected data display code:

```
private void button5_Click(object sender, EventArgs e) //show button code
{
    //display selected data from database on datagridview
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from vreg WHERE VEHICAL_NUMBER='" +
textBox1.Text + "'";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
    MessageBox.Show("DATA DISPLAYED !"); //after display message
}
```

Vehicle register windows form clear:

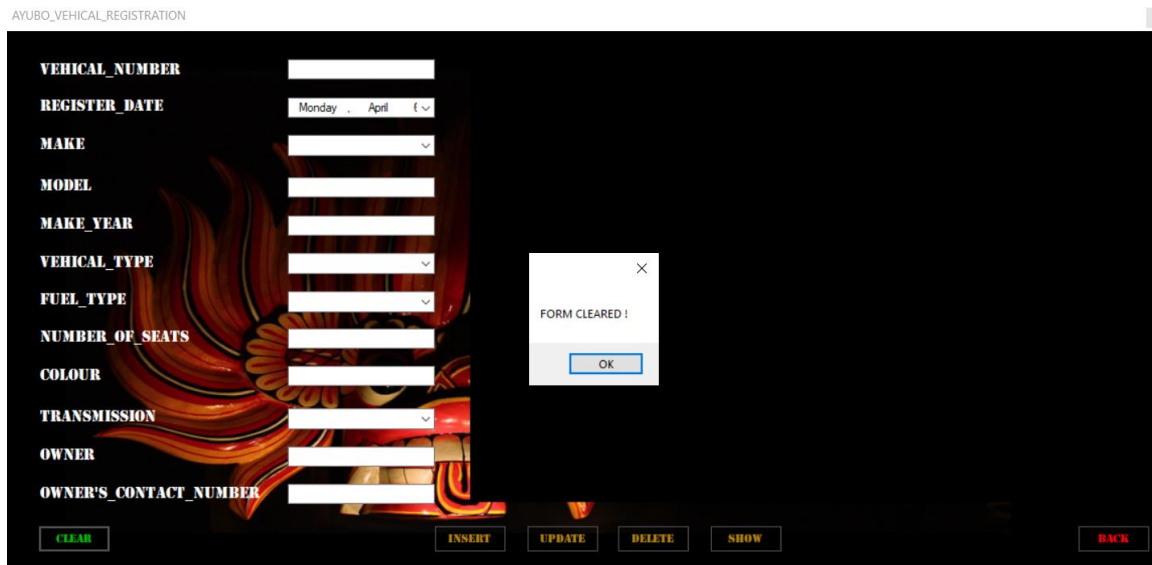


Figure 23-Form cleared

Form clear code:

```
public void clearinputfields()
{
    //clearing Textbox
    textBox1.Text = string.Empty;
    textBox2.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox4.Text = string.Empty;
    textBox5.Text = string.Empty;
    textBox6.Text = string.Empty;
    textBox7.Text = string.Empty;

    //clearing combobox
    comboBox1.Text = string.Empty;
    comboBox2.Text = string.Empty;
    comboBox3.Text = string.Empty;
    comboBox4.Text = string.Empty;

    //clearing datetimepicker
    dateTimePicker1.Text = string.Empty;
    //clear datagridview
    dataGridView1.DataSource = null;
    dataGridView1.Rows.Clear();

    MessageBox.Show("FORM CLEARED !"); //after clear message
}
private void button1_Click(object sender, EventArgs e) //clear button
{
    clearinputfields();
}
```

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

What is IDE?

This is a software application with all tools and feature that needed by a software developer. This has a graphical user interface.

(www.study.com/2020)

Role of IDE:

IDE fundamentally makes a difference you type in your code and see it with a supportive editor with highlights like code highlight, code overlay, imply and blunder / caution maker, arrange it effectively with built in formatter, construct it effortlessly with built in maven / subterranean insect / gradle back, compile, run and investigate effortlessly.

(www.quora.com/2020)

Features of IDE:

Text editor:

Essentially each IDE will have a text editor outlined to compose and control source code.

Few tools may have visual components to drag and drop front-end components, but most have a basic interface with language-specific language structure highlighting.

(www.learn.g2.com/2020)

Debugger:

These tools help clients in recognizing and curing blunders inside source code. They frequently simulate real-world scenarios to test usefulness and execution.

(www.learn.g2.com/2020)

Compiler:

Compilers are components that translate programming language into a frame machines can prepare, such as binary code.

(www.learn.g2.com/2020)

Code completion:

Code total highlights help programmers by intelligently distinguishing and inserting common code components.

(www.learn.g2.com/2020)

Visual Studio IDE

This used to develop computer programs, websites, web apps, web services and mobile apps.

This uses Microsoft software development platforms. This supports thirty-six programming languages. C, C++, C#, Visual Basic.NET, JavaScript, CSS, HTML etc.

Features of Visual Studio:

Window Layout:

This is straightforwardly related to the way various windows are orchestrated and laid out in a Visual Studio IDE.

(www.c-sharpcorner.com/2020)

Tool Box:

This shows all tools that you can add to your projects. Click on View menu on the toolbar and when double click on the toolbox, can easily open toolbox window.

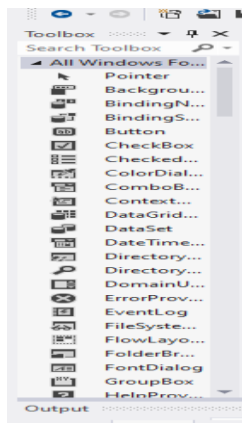


Figure 24-Toolbox

Projects and Solutions:

In spite of its title, an arrangement is not an “answer”. An arrangement is essentially a holder utilized by Visual Studio to organize one or more related projects

(www.docs.microsoft.com/2020)

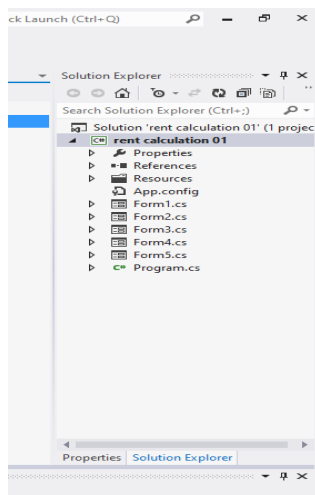


Figure 25-Solution explorer

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 for Ayubo 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.

Ayubo drive company is one of rent a car company. Regarding to the system there are 5 pages and can go for 4 pages through one main page. These all pages are link with database.

Main page called “AYUBO_MAIN_MENU”.

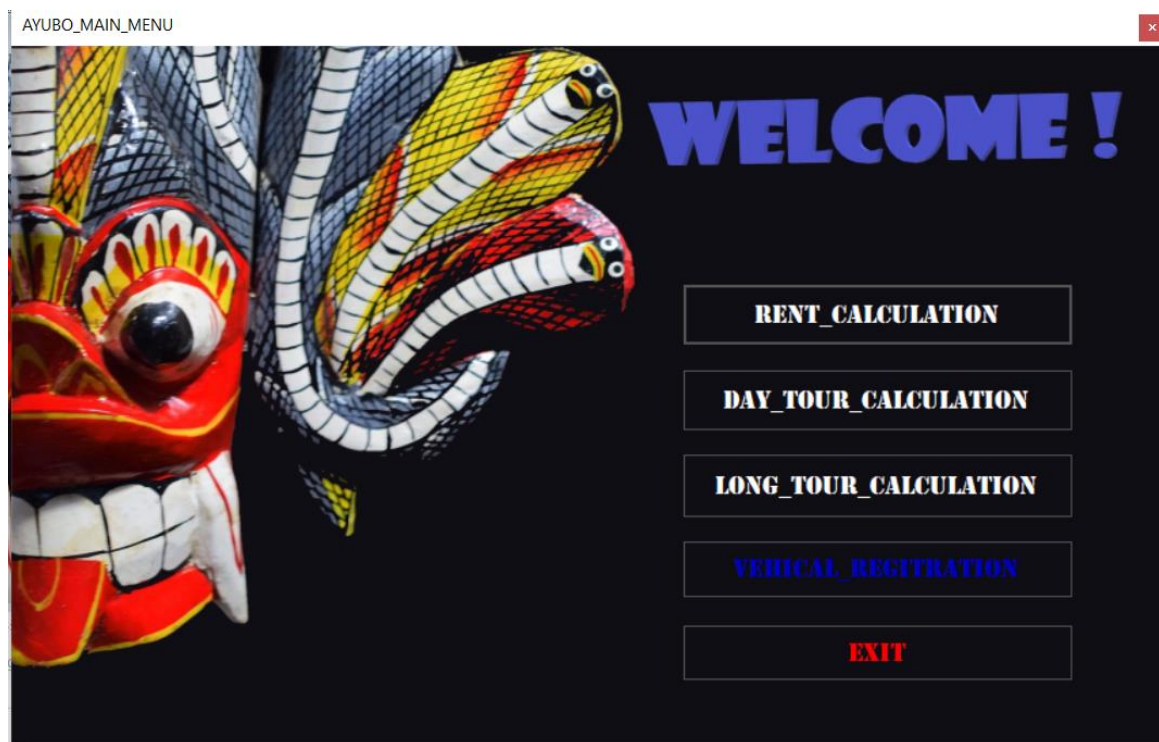


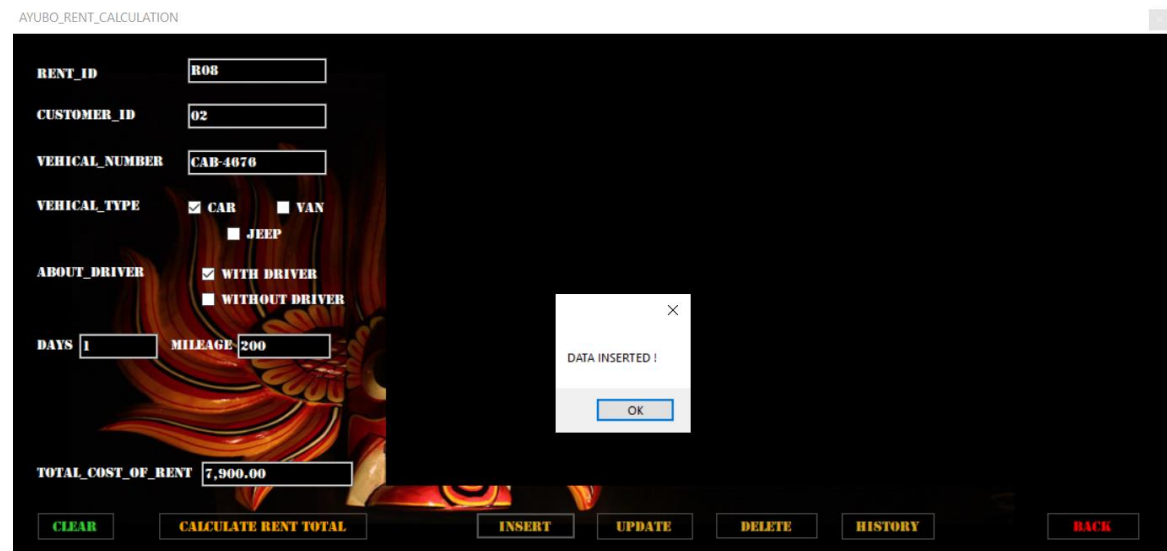
Figure 26-Main menu interface

Rent calculation Page:

Can easily calculate rent through this page.

There are functions buttons in this page to insert, update, delete, view data and clear the form.

Insert button



The screenshot shows a web application titled "AYUBO_RENT_CALCULATION". The form contains the following fields and options:

- RENT_ID**: Text box with value "R08"
- CUSTOMER_ID**: Text box with value "02"
- VEHICAL_NUMBER**: Text box with value "CAB-4070"
- VEHICAL_TYPE**: Radio buttons for ☒ CAR, ☐ VAN, and ☐ JEEP
- ABOUT_DRIVER**: Radio buttons for ☒ WITH DRIVER and ☐ WITHOUT DRIVER
- DAYS**: Text box with value "1"
- MILEAGE**: Text box with value "200"
- TOTAL_COST_OF_RENT**: Text box with value "7,900.00"

At the bottom, there are buttons: CLEAR, CALCULATE RENT TOTAL, INSERT, UPDATE, DELETE, HISTORY, and BACK. A small dialog box titled "DATA INSERTED !" with an "OK" button is visible in the center.

Figure 27-Data insert in rent calculation interface

Code:

```
private void button4_Click(object sender, EventArgs e)    //insert button code
{
    //insert data to database
    con.Open();
    if (checkBox1.Checked == true)
    {
        type = "CAR";
    }
    else if (checkBox2.Checked == true)
    {
        type = "JEEP";
    }
    else if (checkBox3.Checked == true)
    {
        type = "VAN";
    }
    if (checkBox4.Checked == true)
    {
        gen = "WITH_DRIVER";
    }
    else if (checkBox5.Checked == true)
    {
        gen = "WITHOUT_DRIVER";
    }

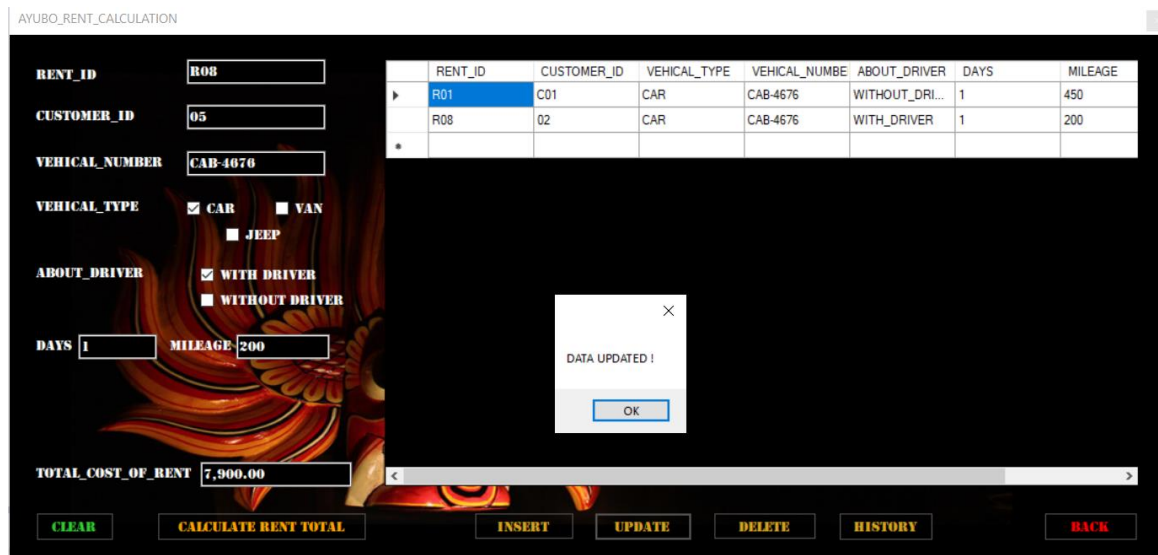
    SqlCommand cmd = new SqlCommand("insert into
[rentdetails](RENT_ID,CUSTOMER_ID,VEHICAL_TYPE,VEHICAL_NUMBER,ABOUT_DRIVER,DAYS,MILEAG
```

```

E, TOTAL_COST_OF_RENT)
values(@RENT_ID,@CUSTOMER_ID,@VEHICAL_TYPE,@VEHICAL_NUMBER,@ABOUT_DRIVER,@DAYS,@MILEAG
E,@TOTAL_COST_OF_RENT)", con);
cmd.Parameters.AddWithValue("@RENT_ID", textBox8.Text);
cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox1.Text);
cmd.Parameters.AddWithValue("@VEHICAL_TYPE", type);
cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox3.Text);
cmd.Parameters.AddWithValue("@ABOUT_DRIVER", gen);
cmd.Parameters.AddWithValue("@DAYS", textBox5.Text);
cmd.Parameters.AddWithValue("@MILEAGE", textBox6.Text);
cmd.Parameters.AddWithValue("@TOTAL_COST_OF_RENT", textBox7.Text);
cmd.ExecuteNonQuery();
MessageBox.Show("DATA INSERTED !");
//after insert message
con.Close();
disp_data();
}

```

Update button



RENT_ID	CUSTOMER_ID	VEHICAL_TYPE	VEHICAL_NUMBE	ABOUT_DRIVER	DAYS	MILEAGE
R01	C01	CAR	CAB-4676	WITHOUT_DRI...	1	450
R08	02	CAR	CAB-4676	WITH_DRIVER	1	200

Figure 28- Data update in rent calculation interface

Code:

```

private void button5_Click(object sender, EventArgs e) //update button code
{
    //update selected data in database
    con.Open();
    if (checkBox1.Checked == true)
    {
        type = "CAR";
    }
    else if (checkBox2.Checked == true)
    {
        type = "JEEP";
    }
    else if (checkBox3.Checked == true)
    {
        type = "VAN";
    }
}

```



```

    }
    if (checkBox4.Checked == true)
    {
        gen = "WITH_DRIVER";
    }
    else if (checkBox5.Checked == true)
    {
        gen = "WITHOUT_DRIVER";
    }
    SqlCommand cmd = new SqlCommand("update rentdetails set
CUSTOMER_ID=@CUSTOMER_ID,VEHICAL_TYPE=@VEHICAL_TYPE,VEHICAL_NUMBER=@VEHICAL_NUMBER,ABOUT_DRIVER=@ABOUT_DRIVER,DAYS=@DAYS,MILEAGE=@MILEAGE,TOTAL_COST_OF_RENT=@TOTAL_COST_OF_RENT where RENT_ID='" + textBox8.Text + "'", con);

    cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox1.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_TYPE", type);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox3.Text);
    cmd.Parameters.AddWithValue("@ABOUT_DRIVER", gen);
    cmd.Parameters.AddWithValue("@DAYS", textBox5.Text);
    cmd.Parameters.AddWithValue("@MILEAGE", textBox6.Text);
    cmd.Parameters.AddWithValue("@TOTAL_COST_OF_RENT", textBox7.Text);
    cmd.ExecuteNonQuery();

    MessageBox.Show("DATA UPDATED !");           //after data update
message

    con.Close();
    disp_data();
}

```

Delete button

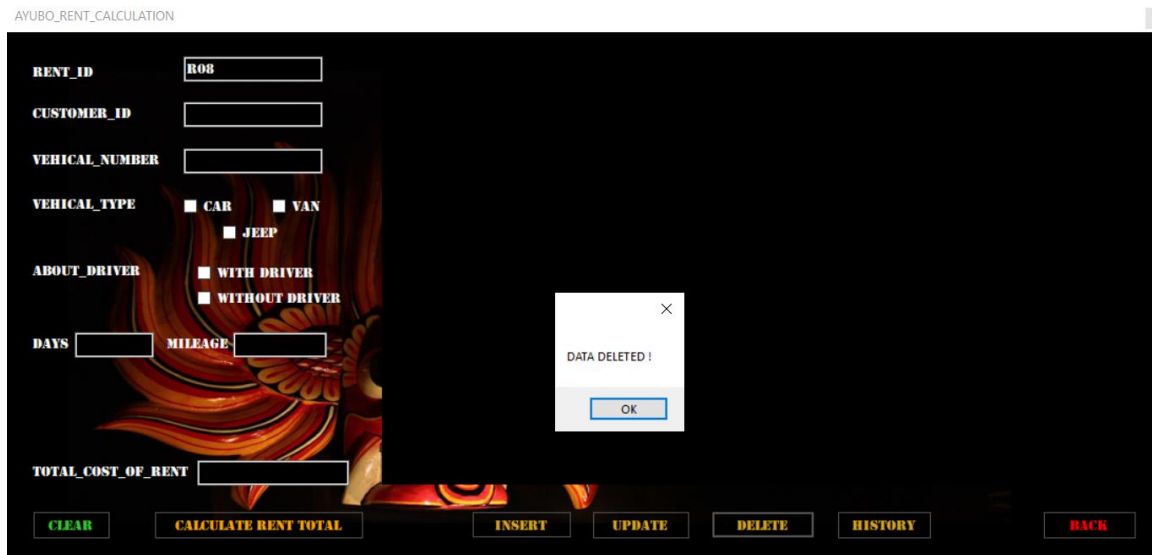


Figure 29- Data delete in rent calculation interface

Code:

```
private void button6_Click_1(object sender, EventArgs e) // delete button code
{
    //delete selected data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("delete from rentdetails where RENT_ID='"
+ textBox8.Text + "'", con);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("DATA DELETED !");
    disp_data();
}
```

Datagridview Code:

```
public void disp_data() //datagridview code
{
    //display data from database
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from rentdetails";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
}
```

History button

AYUBO_RENT_CALCULATION

RENT_ID:

CUSTOMER_ID:

VEHICAL_NUMBER:

VEHICAL_TYPE: ☐ CAR ☐ VAN ☐ JEEP

ABOUT_DRIVER: ☐ WITH DRIVER ☐ WITHOUT DRIVER

DAYS: MILEAGE:

TOTAL_COST_OF_RENT:

RENT_ID	CUSTOMER_ID	VEHICAL_TYPE	VEHICAL_NUMBE	ABOUT_DRIVER	DAYS	MILEAGE
R08	05	CAR	CAB-4676	WITH_DRIVER	1	200
*						

DATA DISPLAYED !

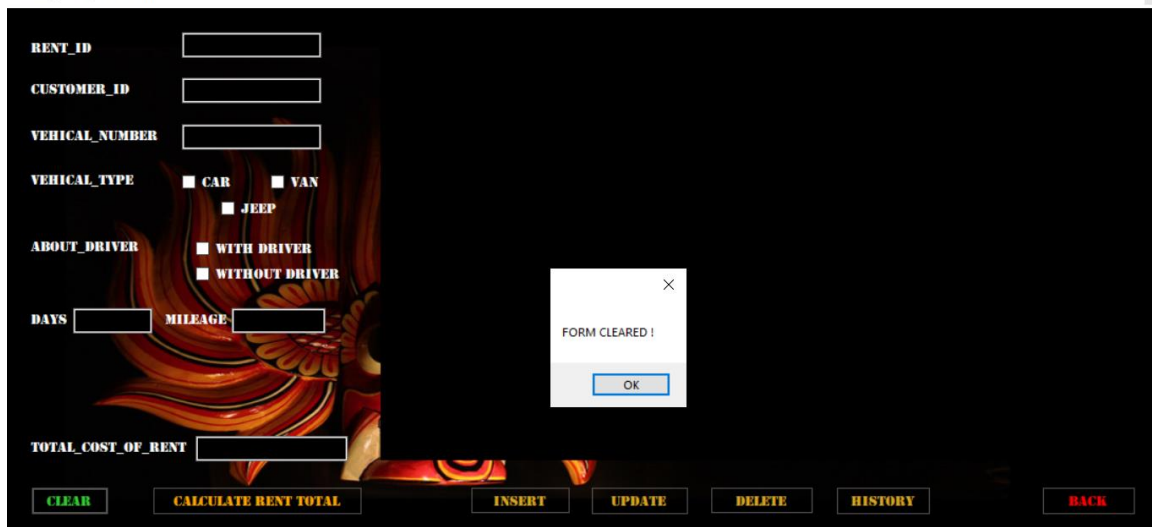
Figure 30- Data display in rent calculation interface

Code:

```
private void button7_Click(object sender, EventArgs e) //history button
code
{
    //get selected data from database and show on datagridview
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from rentdetails WHERE RENT_ID='" +
textBox8.Text + "'";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
    MessageBox.Show("DATA DISPLAYED !"); //after display message
}
```

Clear button

AYUBO_RENT_CALCULATION



The screenshot shows a Windows application window titled 'AYUBO_RENT_CALCULATION'. The interface has a dark background with a red and orange flame-like graphic on the left. It contains several input fields: 'RENT_ID', 'CUSTOMER_ID', 'VEHICAL_NUMBER', 'VEHICAL_TYPE' (with checkboxes for CAR, VAN, JEEP), 'ABOUT_DRIVER' (with checkboxes for WITH DRIVER, WITHOUT DRIVER), 'DAYS', 'MILEAGE', and 'TOTAL_COST_OF_RENT'. At the bottom, there are buttons for 'CLEAR', 'CALCULATE RENT TOTAL', 'INSERT', 'UPDATE', 'DELETE', 'HISTORY', and 'BACK'. A small 'FORM CLEARED!' message box with an 'OK' button is displayed in the center of the window.

Figure 31- Data clear in rent calculation interface

Code:

```
public void clearinputfields()
{
    //clearing Textbox
    textBox1.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox5.Text = string.Empty;
    textBox6.Text = string.Empty;
    textBox7.Text = string.Empty;
    textBox8.Text = string.Empty;

    //clearing CheckBox
    checkBox1.Checked = false;
    checkBox2.Checked = false;
    checkBox3.Checked = false;
    checkBox4.Checked = false;
    checkBox5.Checked = false;
    //clearing datagridview
    dataGridView1.DataSource = null;
    dataGridView1.Rows.Clear();

    MessageBox.Show("FORM CLEARED !"); //clear message
}
private void button2_Click(object sender, EventArgs e) //clear button code
{
    clearinputfields();
}
```

Back button Code:

```
private void button3_Click(object sender, EventArgs e) //back button code
{
    AYUBO_MAIN_MENU acc = new AYUBO_MAIN_MENU();
    acc.Show();
    Form1 f = new Form1();
}
```

Day tour calculation page:

Can easily calculate charges of day tour packages through this page.

Insert button

AYUBO_DAY_TOUR_CALCULATION

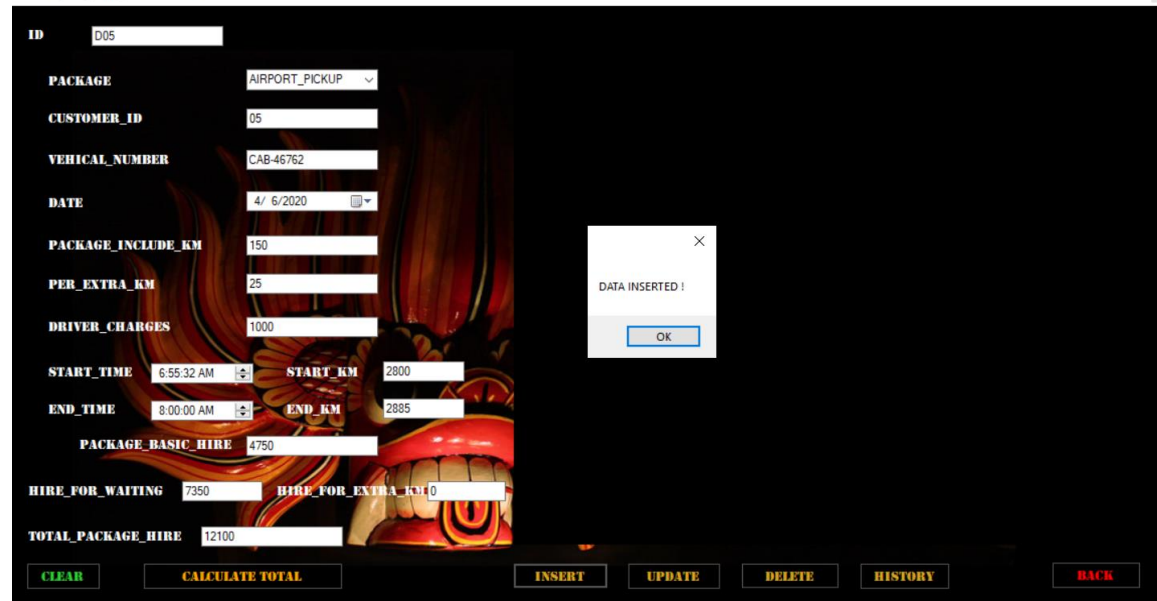


Figure 32- Data insert in day tour calculation interface

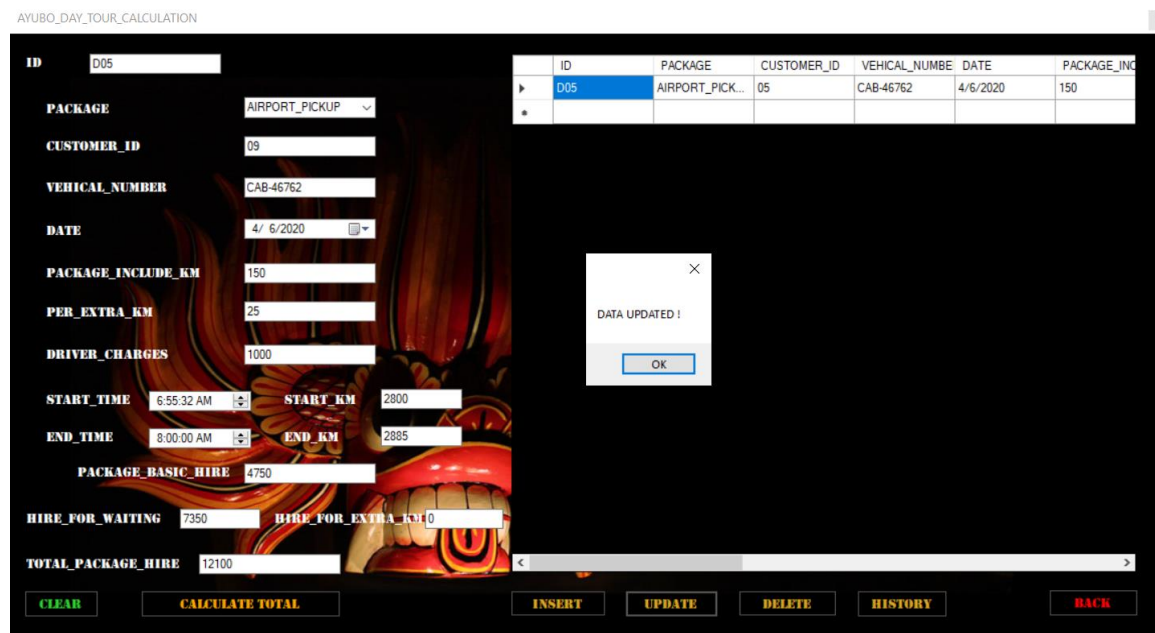
Code:

```
private void button4_Click(object sender, EventArgs e) //insert button code
{
    //insert data to the database
    con.Open();
    SqlCommand cmd = new SqlCommand("insert into
[daytour](ID,PACKAGE,CUSTOMER_ID,VEHICAL_NUMBER,DATE,PACKAGE_INCLUDE_KM,PER_EXTRA_KM,D
RIVER_CHARGES,START_KM,END_KM,START_TIME,END_TIME,PACKAGE_BASIC_HIRE,HIRE_FOR_WAITING,
HIRE_FOR_EXTRA_KM,TOTAL_PACKAGE_HIRE)
values(@ID,@PACKAGE,@CUSTOMER_ID,@VEHICAL_NUMBER,@DATE,@PACKAGE_INCLUDE_KM,@PER_EXTRA_
KM,@DRIVER_CHARGES,@START_KM,@END_KM,@START_TIME,@END_TIME,@PACKAGE_BASIC_HIRE,@HIRE_F
OR_WAITING,@HIRE_FOR_EXTRA_KM,@TOTAL_PACKAGE_HIRE)", con);
    cmd.Parameters.AddWithValue("@ID", textBox10.Text);
    cmd.Parameters.AddWithValue("@PACKAGE", comboBox1.Text);
    cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox11.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
    cmd.Parameters.AddWithValue("@DATE", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("@PACKAGE_INCLUDE_KM", textBox2.Text);
    cmd.Parameters.AddWithValue("@PER_EXTRA_KM", textBox6.Text);
    cmd.Parameters.AddWithValue("@DRIVER_CHARGES", textBox7.Text);
    cmd.Parameters.AddWithValue("@START_KM", textBox3.Text);
    cmd.Parameters.AddWithValue("@END_KM", textBox4.Text);
    cmd.Parameters.AddWithValue("@START_TIME", dateTimePicker2.Text);
    cmd.Parameters.AddWithValue("@END_TIME", dateTimePicker3.Text);
    cmd.Parameters.AddWithValue("@PACKAGE_BASIC_HIRE", textBox5.Text);
    cmd.Parameters.AddWithValue("@HIRE_FOR_WAITING", textBox8.Text);
    cmd.Parameters.AddWithValue("@HIRE_FOR_EXTRA_KM", textBox9.Text);
    cmd.Parameters.AddWithValue("@TOTAL_PACKAGE_HIRE", textBox12.Text);
    cmd.ExecuteNonQuery();
}
```

```
INSERTED !"); //after insert message
con.Close();
disp_data();
}
```

Update button

AYUBO_DAY_TOUR_CALCULATION



The screenshot shows a web application interface for calculating day tour costs. On the left, there is a form with various input fields: ID (D05), PACKAGE (AIRPORT_PICKUP), CUSTOMER_ID (09), VEHICAL_NUMBER (CAB-46762), DATE (4/6/2020), PACKAGE_INCLUDE_KM (150), PER_EXTRA_KM (25), DRIVER_CHARGES (1000), START_TIME (6:55:32 AM), START_KM (2800), END_TIME (8:00:00 AM), END_KM (2885), PACKAGE_BASIC_HIRE (4750), HIRE_FOR_WAITING (7350), HIRE_FOR_EXTRA_KM (0), and TOTAL_PACKAGE_HIRE (12100). At the bottom of the form are buttons for CLEAR, CALCULATE TOTAL, INSERT, UPDATE, DELETE, HISTORY, and BACK. On the right, there is a table with columns: ID, PACKAGE, CUSTOMER_ID, VEHICAL_NUMBE, DATE, and PACKAGE_INC. The table contains one row with data: D05, AIRPORT_PICK..., 09, CAB-46762, 4/6/2020, and 150. A modal dialog box titled 'DATA UPDATED !' with an 'OK' button is displayed in the center of the screen.

Figure 33- Data update in day tour calculation interface

Code;

```
private void button5_Click(object sender, EventArgs e) //update button code
{
    //update data in database
    con.Open();
    SqlCommand cmd = new SqlCommand("update daytour set
PACKAGE=@PACKAGE,CUSTOMER_ID=@CUSTOMER_ID,VEHICAL_NUMBER=@VEHICAL_NUMBER,DATE=@DATE,PA
CKAGE_INCLUDE_KM=@PACKAGE_INCLUDE_KM,PER_EXTRA_KM=@PER_EXTRA_KM,DRIVER_CHARGES=@DRIVER
_CHARGES,START_KM=@START_KM,END_KM=@END_KM,START_TIME=@START_TIME,END_TIME=@END_TIME,P
ACKAGE_BASIC_HIRE=@PACKAGE_BASIC_HIRE,HIRE_FOR_WAITING=@HIRE_FOR_WAITING,HIRE_FOR_EXTR
A_KM=@HIRE_FOR_EXTRA_KM,TOTAL_PACKAGE_HIRE=@TOTAL_PACKAGE_HIRE where ID='" +
textBox10.Text + "'", con);
    cmd.Parameters.AddWithValue("@ID", textBox10.Text);
    cmd.Parameters.AddWithValue("@PACKAGE", comboBox1.Text);
    cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox11.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
    cmd.Parameters.AddWithValue("@DATE", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("@PACKAGE_INCLUDE_KM", textBox2.Text);
    cmd.Parameters.AddWithValue("@PER_EXTRA_KM", textBox6.Text);
    cmd.Parameters.AddWithValue("@DRIVER_CHARGES", textBox7.Text);
    cmd.Parameters.AddWithValue("@START_KM", textBox3.Text);
    cmd.Parameters.AddWithValue("@END_KM", textBox4.Text);
    cmd.Parameters.AddWithValue("@START_TIME", dateTimePicker2.Text);
    cmd.Parameters.AddWithValue("@END_TIME", dateTimePicker3.Text);
    cmd.Parameters.AddWithValue("@PACKAGE_BASIC_HIRE", textBox5.Text);
    cmd.Parameters.AddWithValue("@HIRE_FOR_WAITING", textBox8.Text);
```

```
cmd.Parameters.AddWithValue("@HIRE_FOR_EXTRA_KM", textBox9.Text);
cmd.Parameters.AddWithValue("@TOTAL_PACKAGE_HIRE", textBox12.Text);
cmd.ExecuteNonQuery();
MessageBox.Show("DATA UPDATED !"); //after update
message
con.Close();
disp_data();
}
```

Delete button

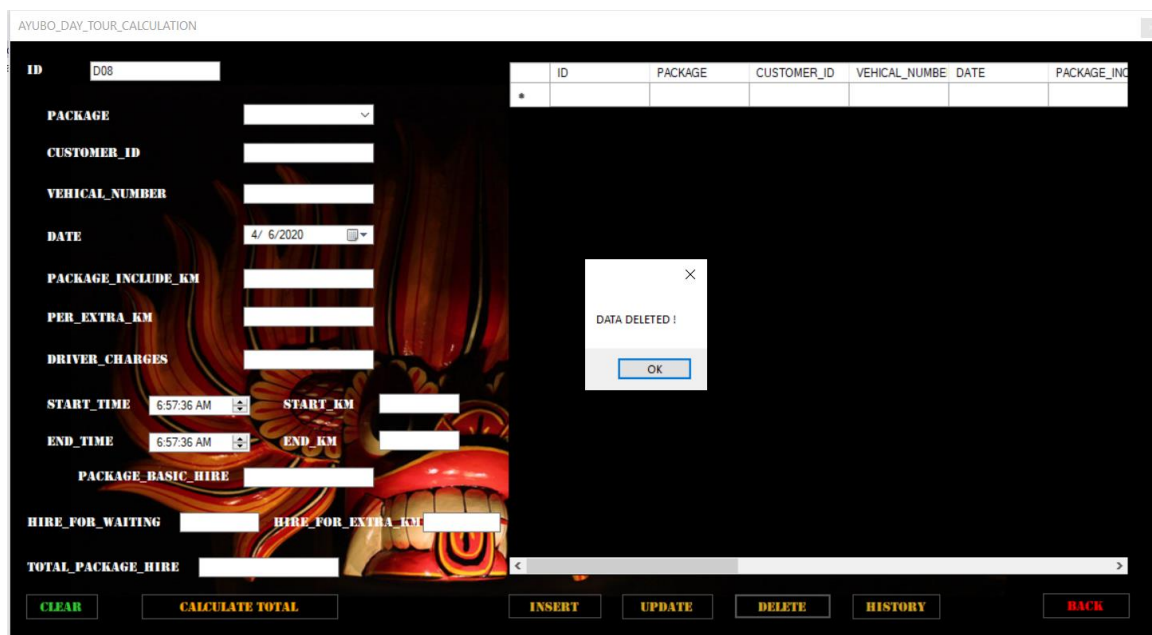


Figure 34- Data delete in day tour calculation interface

Code;

```
private void button6_Click(object sender, EventArgs e) //delete button code
{
    //delete data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("delete from daytour where ID='" +
textBox10.Text + "'", con);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("DATA DELETED !"); //after delete message
}
```

Datagridview code:

```
public void disp_data() //datagridview code
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from daytour"; //get data from database
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
}
```



```

        da.Fill(dt);
        dataGridView1.DataSource = dt;
        con.Close();
    }

```

History button

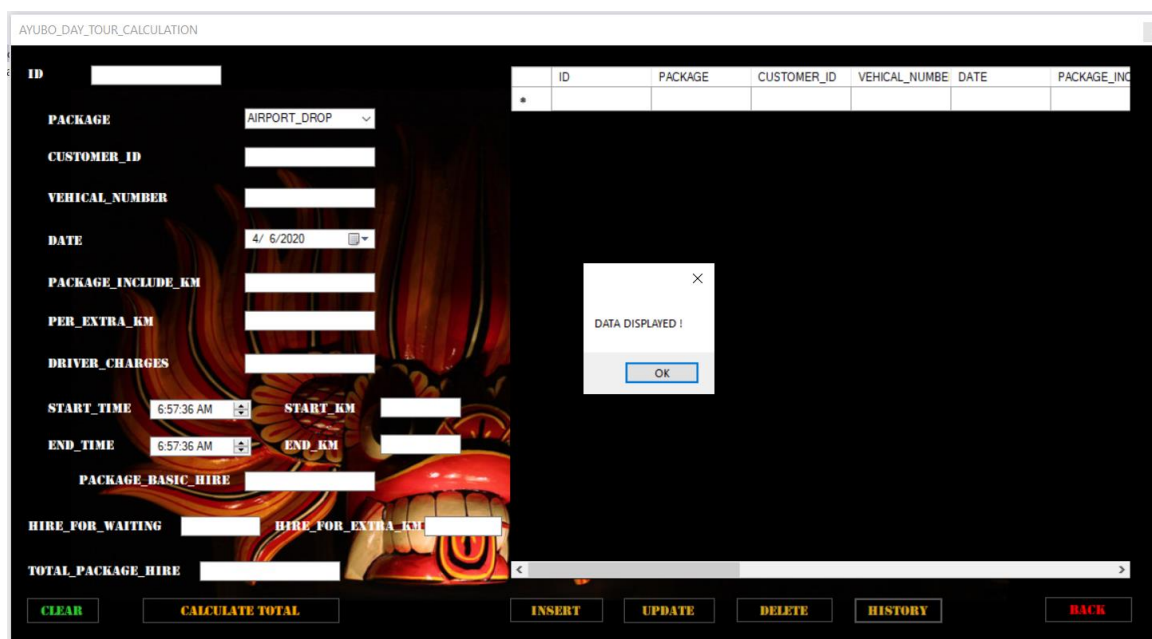


Figure 35- Data display in day tour calculation interface

Code;

```

private void button7_Click(object sender, EventArgs e) //history button code
{
    // get selected data from database and display on the datagridview
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from daytour WHERE ID='" + textBox10.Text +
    "'";

    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
    MessageBox.Show("DATA DISPLAYED !"); //after display message
}

```


Clear button

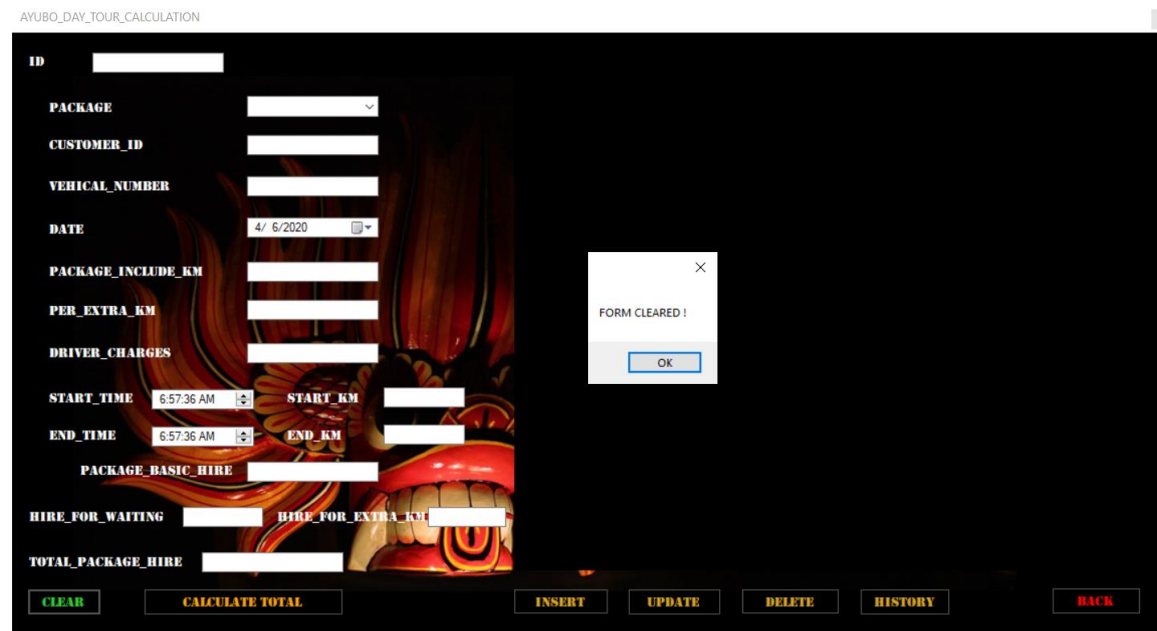


Figure 36- Data clear in day tour calculation interface

Code;

```
public void clearinputfields()
{
    //clear texboxes
    textBox1.Text = string.Empty;
    textBox2.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox4.Text = string.Empty;
    textBox5.Text = string.Empty;
    textBox6.Text = string.Empty;
    textBox7.Text = string.Empty;
    textBox8.Text = string.Empty;
    textBox9.Text = string.Empty;
    textBox10.Text = string.Empty;
    textBox11.Text = string.Empty;
    textBox12.Text = string.Empty;
    //clear comboboxes
    comboBox1.Text = string.Empty;
    //clear datetimepickers
    dateTimePicker1.Text = string.Empty;
    dateTimePicker2.Text = string.Empty;
    dateTimePicker3.Text = string.Empty;
    //clear datagridview
    dataGridView1.DataSource = null;
    dataGridView1.Rows.Clear();

    //after clear message
    MessageBox.Show("FORM CLEARED !");
}
private void button2_Click(object sender, EventArgs e) //clear button code
{
    clearinputfields();
}
```

```
private void button3_Click(object sender, EventArgs e) //back button code
{
    AYUBO_MAIN_MENU acc = new AYUBO_MAIN_MENU();
    acc.Show();
    AYUBO_DAY_TOUR_CALCULATION adtc= new AYUBO_DAY_TOUR_CALCULATION();
    this.Visible = false;
    this.Hide();
}
```

Long tour calculation page:

Can easily calculate charges of long tour packages through this page.

Insert button

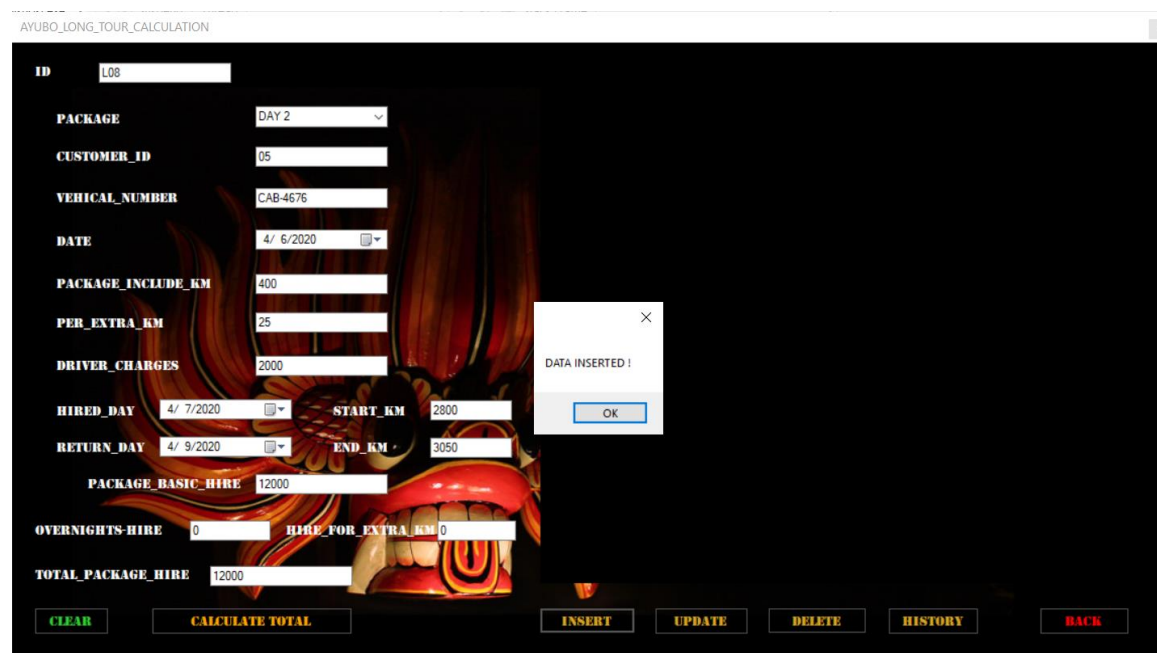


Figure 37- Data insert in long tour calculation interface

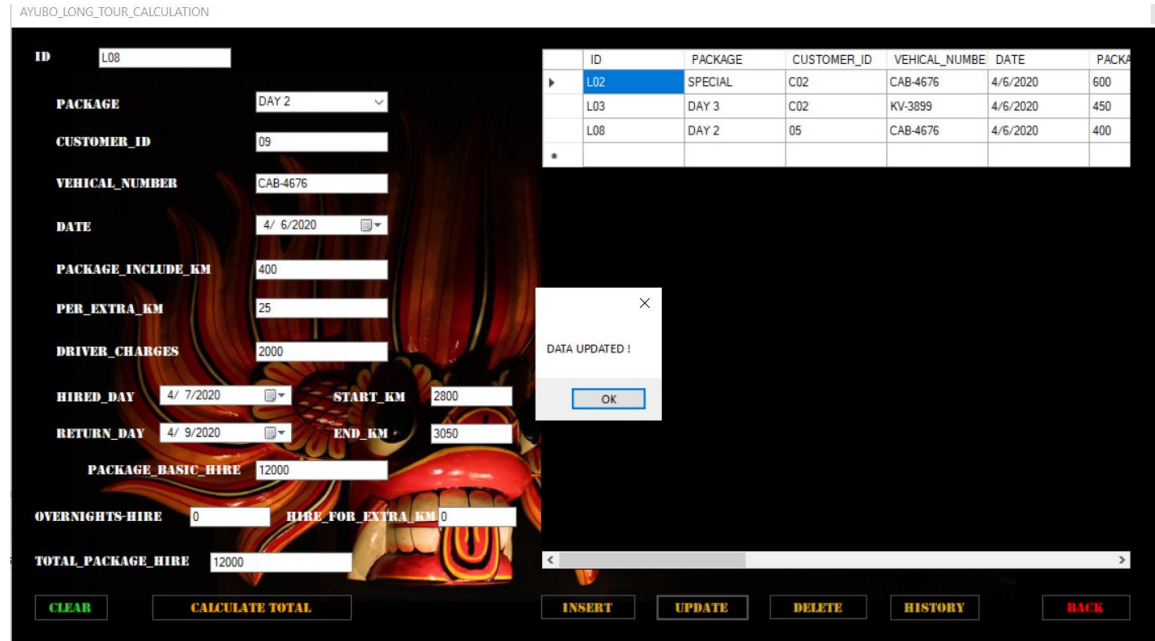
Code:

```
private void button4_Click(object sender, EventArgs e) //insert button code
{
    //insert data to database
    con.Open();
    SqlCommand cmd = new SqlCommand("insert into
[longtour](ID,PACKAGE,CUSTOMER_ID,VEHICAL_NUMBER,DATE,PACKAGE_INCLUDE_KM,PER_EXTRA_KM,
DRIVER_CHARGES,START_KM,END_KM,HIRED_DAY,RETURN_DAY,PACKAGE_BASIC_HIRE,OVERNIGHTS_HIRE
,HIRE_FOR_EXTRA_KM,TOTAL_PACKAGE_HIRE)
values(@ID,@PACKAGE,@CUSTOMER_ID,@VEHICAL_NUMBER,@DATE,@PACKAGE_INCLUDE_KM,@PER_EXTRA_
KM,@DRIVER_CHARGES,@START_KM,@END_KM,@HIRED_DAY,@RETURN_DAY,@PACKAGE_BASIC_HIRE,@OVERN
IGHTS_HIRE,@HIRE_FOR_EXTRA_KM,@TOTAL_PACKAGE_HIRE)", con);
    cmd.Parameters.AddWithValue("@ID", textBox10.Text);
    cmd.Parameters.AddWithValue("@PACKAGE", comboBox1.Text);
    cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox11.Text);
    cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
    cmd.Parameters.AddWithValue("@DATE", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("@PACKAGE_INCLUDE_KM", textBox2.Text);
```

```
cmd.Parameters.AddWithValue("@PER_EXTRA_KM", textBox6.Text);
cmd.Parameters.AddWithValue("@DRIVER_CHARGES", textBox7.Text);
cmd.Parameters.AddWithValue("@START_KM", textBox3.Text);
cmd.Parameters.AddWithValue("@END_KM", textBox4.Text);
cmd.Parameters.AddWithValue("@HIRED_DAY", dateTimePicker2.Text);
cmd.Parameters.AddWithValue("@RETURN_DAY", dateTimePicker3.Text);
cmd.Parameters.AddWithValue("@PACKAGE_BASIC_HIRE", textBox5.Text);
cmd.Parameters.AddWithValue("@OVERNIGHTS_HIRE", textBox8.Text);
cmd.Parameters.AddWithValue("@HIRE_FOR_EXTRA_KM", textBox9.Text);
cmd.Parameters.AddWithValue("@TOTAL_PACKAGE_HIRE", textBox12.Text);
cmd.ExecuteNonQuery();
MessageBox.Show("DATA INSERTED !"); //insert message
con.Close();
disp_data();
}
```

Update button

AYUBO_LONG_TOUR_CALCULATION



The screenshot shows a web application interface for calculating long tour costs. On the left, there is a form with various input fields: ID (L08), PACKAGE (DAY 2), CUSTOMER_ID (09), VEHICAL_NUMBER (CAB-4676), DATE (4/ 6/2020), PACKAGE_INCLUDE_KM (400), PER_EXTRA_KM (25), DRIVER_CHARGES (2000), HIRED_DAY (4/ 7/2020), START_KM (2800), RETURN_DAY (4/ 9/2020), END_KM (3050), PACKAGE_BASIC_HIRE (12000), OVERNIGHTS_HIRE (0), HIRE_FOR_EXTRA_KM (0), and TOTAL_PACKAGE_HIRE (12000). At the bottom of the form are buttons for CLEAR, CALCULATE TOTAL, INSERT, UPDATE, DELETE, HISTORY, and BACK. On the right, there is a table with columns: ID, PACKAGE, CUSTOMER_ID, VEHICAL_NUMBE, DATE, and PACKA. The table contains three rows: L02 (SPECIAL, C02, CAB-4676, 4/6/2020, 600), L03 (DAY 3, C02, KV-3899, 4/6/2020, 450), and L08 (DAY 2, 05, CAB-4676, 4/6/2020, 400). A confirmation dialog box is open in the center, displaying "DATA UPDATED !" and an OK button.

Figure 38- Data update in long tour calculation interface

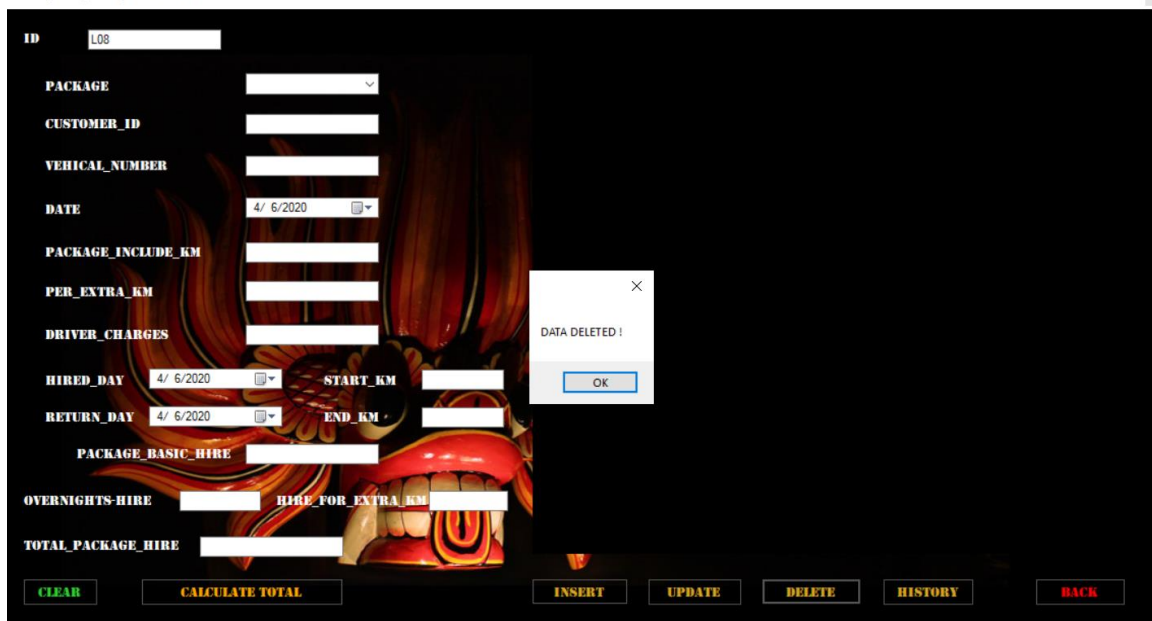
Code;

```
private void button5_Click(object sender, EventArgs e) //update button code
{
    //update selected data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("update longtour set
PACKAGE=@PACKAGE,CUSTOMER_ID=@CUSTOMER_ID,VEHICAL_NUMBER=@VEHICAL_NUMBER,DATE=@DATE,PA
CKAGE_INCLUDE_KM=@PACKAGE_INCLUDE_KM,PER_EXTRA_KM=@PER_EXTRA_KM,DRIVER_CHARGES=@DRIVER
_CHARGES,START_KM=@START_KM,END_KM=@END_KM,HIRED_DAY=@HIRED_DAY,RETURN_DAY=@RETURN_DAY
,PAGE_BASIC_HIRE=@PACKAGE_BASIC_HIRE,OVERNIGHTS_HIRE=@OVERNIGHTS_HIRE,HIRE_FOR_EXTR
A_KM=@HIRE_FOR_EXTRA_KM,TOTAL_PACKAGE_HIRE=@TOTAL_PACKAGE_HIRE where
ID='"+textBox10.Text+"'", con);
cmd.Parameters.AddWithValue("@ID", textBox10.Text);
cmd.Parameters.AddWithValue("@PACKAGE", comboBox1.Text);
cmd.Parameters.AddWithValue("@CUSTOMER_ID", textBox11.Text);
cmd.Parameters.AddWithValue("@VEHICAL_NUMBER", textBox1.Text);
cmd.Parameters.AddWithValue("@DATE", dateTimePicker1.Text);
cmd.Parameters.AddWithValue("@PACKAGE_INCLUDE_KM", textBox2.Text);
cmd.Parameters.AddWithValue("@PER_EXTRA_KM", textBox6.Text);
cmd.Parameters.AddWithValue("@DRIVER_CHARGES", textBox7.Text);
```

```
cmd.Parameters.AddWithValue("@START_KM", textBox3.Text);
cmd.Parameters.AddWithValue("@END_KM", textBox4.Text);
cmd.Parameters.AddWithValue("@HIRED_DAY", dateTimePicker2.Text);
cmd.Parameters.AddWithValue("@RETURN_DAY", dateTimePicker3.Text);
cmd.Parameters.AddWithValue("@PACKAGE_BASIC_HIRE", textBox5.Text);
cmd.Parameters.AddWithValue("@OVERNIGHTS_HIRE", textBox8.Text);
cmd.Parameters.AddWithValue("@HIRE_FOR_EXTRA_KM", textBox9.Text);
cmd.Parameters.AddWithValue("@TOTAL_PACKAGE_HIRE", textBox12.Text);
cmd.ExecuteNonQuery();
MessageBox.Show("DATA UPDATED !"); //update message
con.Close();
disp_data();
}
```

Delete button

AYUBO_LONG_TOUR_CALCULATION



The screenshot shows a web application interface for 'AYUBO_LONG_TOUR_CALCULATION'. It features a form with various input fields for tour details, including ID, Package, Customer ID, Vehicle Number, Date, Package Include KM, Per Extra KM, Driver Charges, Hired Day, Return Day, Package Basic Hire, Overnights Hire, Hire for Extra KM, and Total Package Hire. A 'DELETE' button is visible at the bottom. A modal dialog box is open, displaying the message 'DATA DELETED !' with an 'OK' button.

Figure 39- Data delete in long tour calculation interface

Code;

```
private void button6_Click(object sender, EventArgs e) // delete button
code
{
    //delete selected data from database
    con.Open();
    SqlCommand cmd = new SqlCommand("delete from longtour where ID='" +
textBox10.Text + "'", con);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("DATA DELETED !"); //delete message
    disp_data();
}
```

Datagridview code:

```
public void disp_data() // datagridview code
{
    // get data from database and display on datagridview
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from longtour";
}
```

```

        cmd.ExecuteNonQuery();
        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        dataGridView1.DataSource = dt;
        con.Close();
    }
}

```

History button

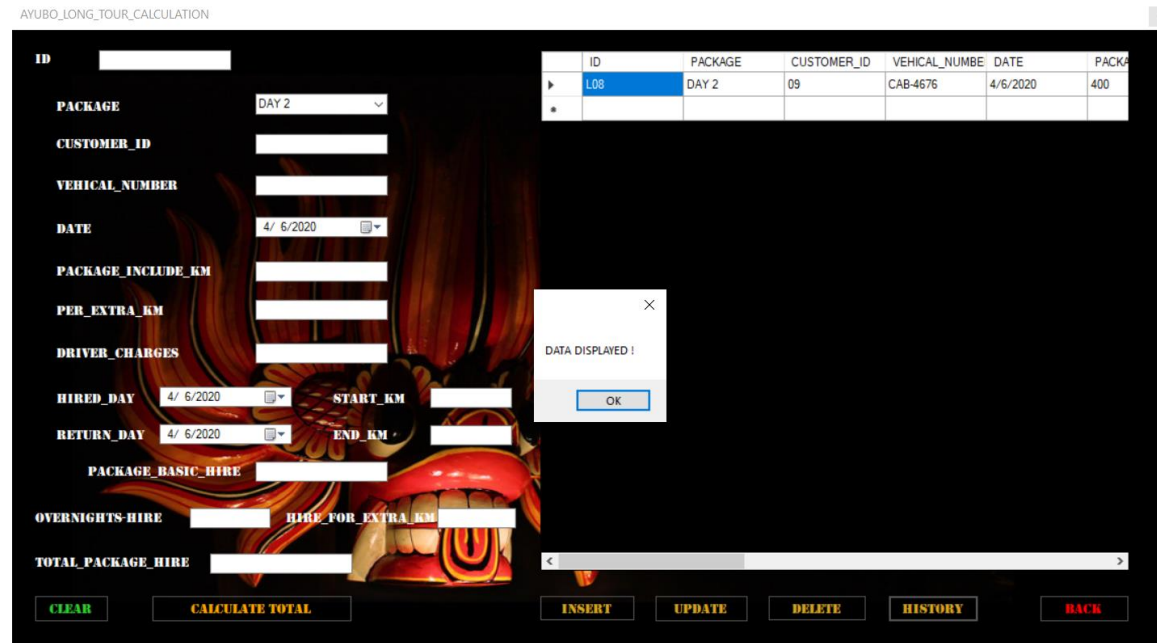


Figure 40- Data display in long tour calculation interface

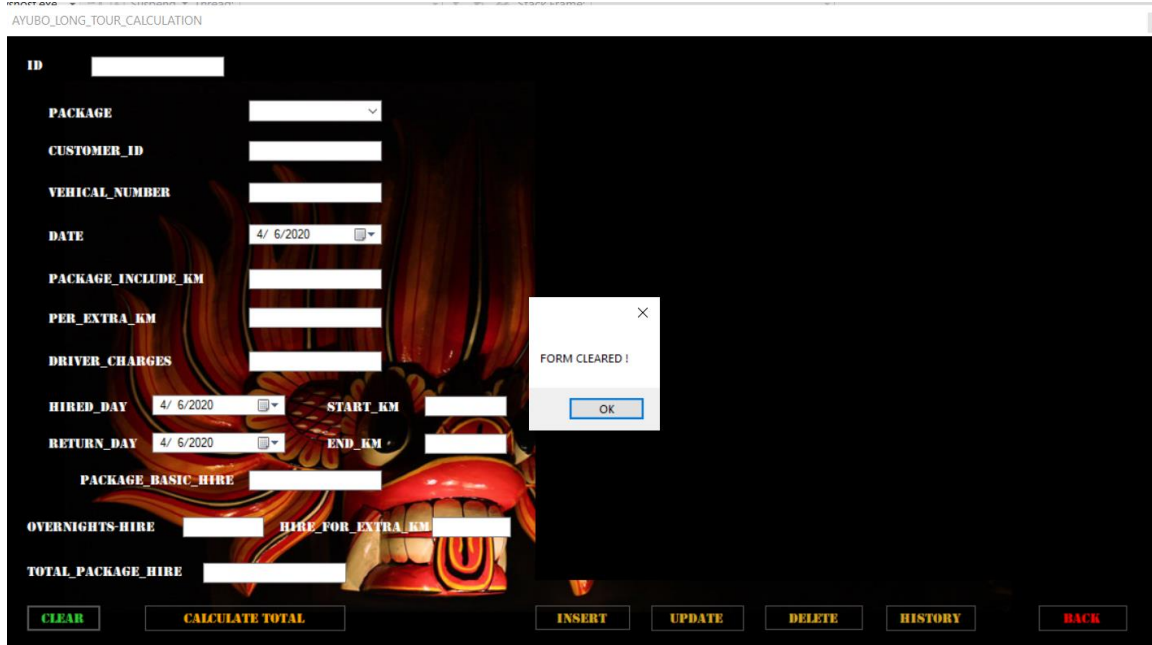
Code;

```

private void button7_Click(object sender, EventArgs e) //history button code
{
    //display selected data from database and show on datagridview
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from longtour WHERE PACKAGE='" +
comboBox1.Text + "'";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    con.Close();
    MessageBox.Show("DATA DISPLAYED !"); //display message
}

```


Clear button



The screenshot shows a web application interface for 'AYUBO_LONG_TOUR_CALCULATION'. The interface has a dark background with a large, stylized illustration of a person's face in the background. On the left, there is a form with various input fields and labels: ID, PACKAGE (dropdown), CUSTOMER_ID, VEHICAL_NUMBER, DATE (4/6/2020), PACKAGE_INCLUDE_KM, PER_EXTRA_KM, DRIVER_CHARGES, HIRED_DAY (4/6/2020), START_KM, RETURN_DAY (4/6/2020), END_KM, PACKAGE_BASIC_HIRE, OVERNIGHTS_HIRE, HIRE_FOR_EXTRA_KM, and TOTAL_PACKAGE_HIRE. At the bottom, there are buttons: CLEAR, CALCULATE TOTAL, INSERT, UPDATE, DELETE, HISTORY, and BACK. A small dialog box with the title 'FORM CLEARED!' and an 'OK' button is overlaid on the right side of the form.

Figure 41- Data clear in long tour calculation interface

Code;

```
public void clearinputfields()
{
    //clear textboxes
    textBox1.Text = string.Empty;
    textBox2.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox4.Text = string.Empty;
    textBox5.Text = string.Empty;
    textBox6.Text = string.Empty;
    textBox7.Text = string.Empty;
    textBox8.Text = string.Empty;
    textBox9.Text = string.Empty;
    textBox10.Text = string.Empty;
    textBox11.Text = string.Empty;
    textBox12.Text = string.Empty;
    //clear combobox
    comboBox1.Text = string.Empty;
    //clear datetimepickers
    dateTimePicker1.Text = string.Empty;
    dateTimePicker2.Text = string.Empty;
    dateTimePicker3.Text = string.Empty;
    //clear datagridview
    dataGridView1.DataSource = null;
    dataGridView1.Rows.Clear();
}
```

```
CLEARED !");           //clear message
    }
    private void button2_Click(object sender, EventArgs e) //clear button code
    {
        clearinputfields();
    }
```

Back button Code;

```
private void button3_Click(object sender, EventArgs e)           //back button code
{
    AYUBO_MAIN_MENU acc = new AYUBO_MAIN_MENU();
    acc.Show();
    AYUBO_LONG_TOUR_CALCULATION adtc = new AYUBO_LONG_TOUR_CALCULATION();
    this.Visible = false;
    this.Hide();
}
```

4.2 What is debugging an application? Explain 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.

What is bug?

Bug is an error in software or program that causes program to malfunction.



Figure 42-Example for bug

Debug

In common, debug alludes to the method of analyzing and removing errors from a program's source code. For case, a designer may send a debug command through a program to see where within the code an error happens so it can be settled or bypassed.

Steps of debugging:

- Identify the errors.
- Find the error location.
- Analyze the error using a code.
- Prove the analysis.
- If Cover lateral damaged all the unit test needed to be create.
- Finally fix the error and validate the error.

4.3 Explain 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.

Coding Standards:

Great computer program advancement organizations need their programmers to preserve to a few well-defined and standard fashion of coding called coding standards.

Advantages of Coding Standards:

Enhanced Efficiency

Coding standards are help team to find the problems early completely. In the software process efficiency will increase by this.

Risk of project failure is reduced

When software developing, there can be more failures because of coding problems. Coding standard is reduced this problems and risk of projects failures.

Minimal complexity

There are more chances to fail codes when they are complex. Coding standards are help to make them less complex and less errors.

Easy to Maintain

When follow coding standards, can easy to maintain and the code is consistent. Then anyone can understand.

Bug Rectification

If the code is written in consistent manner, can easy to locate and correct bugs in the program.

A Comprehensive look

If source code written in consistently, the source code can view in clearly.

Cost-Efficient

Can reuse the code whenever required. This will reduce the cost along with the efforts put in the development.

Comments

When add a comment to a code, can easily find the meaning of the code and the location.

```
if (TK > 250)
{
    HIRE_FOR_EXTRA_KM = (TK - 250) * 100;           //Rs.100 per Extra KM Charges
}

if (TT > 4 || TT <= 12)                           //4 hours is time allowed.
{
    HIRE_FOR_WAITING = (4 - TT) * 150;
}
```

Figure 43-Example comment

List of Reference

techtarget.com 2020. [ONLINE] Available at:

<https://whatis.techtarget.com/definition/algorithm>

[Accessed 05/03/2020]

topcoder.com 2020. [ONLINE] Available at:

<https://www.topcoder.com/community/competitive-programming/tutorials/the-importance-of-algorithms/>

[Accessed 05/03/2020]

wikihow.com 2020. [ONLINE] Available at:

<https://www.wikihow.com/Write-an-Algorithm-in-Programming-Language>

[Accessed 05/03/2020]

study.com 2020. [ONLINE] Available at:

<https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html>

[Accessed 05/03/2020]

quora.com 2020. [ONLINE] Available at:

<https://www.quora.com/What-is-important-in-Pseudocode>

[Accessed 05/03/2020]

study.com 2020. [ONLINE] Available at:

<https://study.com/academy/lesson/writing-pseudocode-algorithms-examples.html>

[Accessed 05/03/2020]

lucidchart.com 2020. [ONLINE] Available at:

<https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial>

[Accessed 05/03/2020]

qsstudy.com 2020. [ONLINE] Available at:

<https://www.qsstudy.com/technology/characteristics-of-an-algorithm>

[Accessed 05/03/2020]

studytoneight.com 2020. [ONLINE] Available at:

<https://www.studytoneight.com/data-structures/linear-search-algorithm>

[Accessed 05/03/2020]

btechsmartclass.com 2020. [ONLINE] Available at:

http://www.btechsmartclass.com/data_structures/binary-search.html

[Accessed 05/03/2020]

techterms.com 2020. [ONLINE] Available at:

<https://techterms.com/definition/array>

[Accessed 05/03/2020]

Rob-bell.net 2020. [ONLINE] Available at:

<https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/>

[Accessed 05/03/2020]

geeksforgeeks.org 2020. [ONLINE] Available at:

<https://www.geeksforgeeks.org/introduction-of-programming-paradigms/>

[Accessed 05/03/2020]

tuannnguyem.tech 2020. [ONLINE] Available at:

<https://www.tuannnguyen.tech/2019/05/programming-paradigms-what-is-procedural-programming/>

[Accessed 05/03/2020]

Blog.usejournal.com 2020. [ONLINE] Available at:

<https://blog.usejournal.com/object-oriented-programming-concepts-in-simple-english-3db22065d7d0?gi=a500dd587f6>

[Accessed 05/03/2020]

computerhope.com 2020. [ONLINE] Available at:

<https://www.computerhope.com/jargon/e/event-driven-prog.htm>

[Accessed 05/03/2020]

geeksforgeeks.org 2020. [ONLINE] Available at:

<https://www.geeksforgeeks.org/basic-input-output-c/>

[Accessed 05/03/2020]

medium.com 2020. [ONLINE] Available at:

<https://medium.com/breaktheloop/why-using-namespace-std-is-used-after-including-iostream-dc5ae45db652>

[Accessed 05/03/2020]

codesdope.com 2020. [ONLINE] Available at:

<https://www.codesdope.com/blog/article/int-main-vs-void-main-vs-int-mainvoid-in-c-c/>

[Accessed 05/03/2020]

geeksforgeeks.org 2020. [ONLINE] Available at:

<https://www.geeksforgeeks.org/partial-classes-in-c-sharp/>

[Accessed 05/03/2020]

techopedia.com 2020. [ONLINE] Available at:

<https://www.techopedia.com/definition/27985/public-c>

[Accessed 05/03/2020]

stackoverflow.com 2020. [ONLINE] Available at:

<https://stackoverflow.com/questions/614818/in-c-what-is-the-difference-between-public-private-protected-and-having-no>

[Accessed 05/03/2020]

W3schools.com 2020. [ONLINE] Available at:

https://www.w3schools.com/python/ref_func_print.asp

[Accessed 05/03/2020]

study.com 2020. [ONLINE] Available at:

<https://study.com/academy/lesson/ide-in-software-definition-examples.html>

[Accessed 05/03/2020]

quora.com 2020. [ONLINE] Available at:

<https://www.quora.com/What-is-the-function-of-IDE>

[Accessed 05/03/2020]

Learn.g2.com 2020. [ONLINE] Available at:

<https://learn.g2.com/ide>

[Accessed 05/03/2020]

c-sharpcorner.com 2020. [ONLINE] Available at:

<https://www.c-sharpcorner.com/UploadFile/84c85b/visual-studio-2015-feature-series-sharp2-window-layout/>

[Accessed 05/03/2020]

Docs.microsoft.com 2020. [ONLINE] Available at:

<https://docs.microsoft.com/en-us/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019>

[Accessed 05/03/2020]

