Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS					
Programme title	Higher National Diploma in Computing				
Assessor		**Internal Verifier**			
Unit(s)					
Assignment title					
Student's name					
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			
Unit Title			
Assignment Number		**Assessor**	
Submission Date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback:

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

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

LO2. Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an ana Integrated Development Environment (IDE).

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

LO3. Implement basic algorithms in code using an IDE.

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

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

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

Grade:	**Assessor Signature:**	**Date:**
Resubmission Feedback:		
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
Signature & Date:		

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

Assignment Feedback

Formative Feedback: Assessor to Student			
Action Plan			
Summative feedback			
Feedback: Student to Assessor			
Assessor signature		**Date**	
Student signature		**Date**	

Pearson Higher Nationals in

Computing

Unit 01: Programming
Assignment 01

General Guidelines

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

Word Processing Rules

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

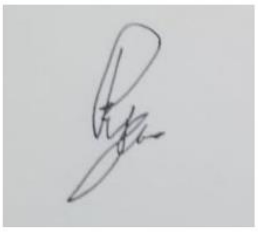
Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

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

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



ryandilthusha@gmail.com

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

04/02/2022

Date:
(*Provide Submission Date*)

Higher National Diploma in Computing

Assignment Brief

Student Name /ID Number	Ryan Wickramaratne (COL 00081762)
Unit Number and Title	**Unit 01: Programming**
Academic Year	2021/22
Unit Tutor	Mr. Praveen Croos
Assignment Title	**Design &Implement a GUI based system using a suitable Integrated Development Environment**
Issue Date	
Submission Date	
IV Name & Date	

Submission Format

This submission will have 3 components

1. Written Report

This submission is in the form of an individual written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system. (**The recommended word count is 1,500–2,000 words for the report excluding annexures)**

2. Implemented System (Software)

The student should submit a GUI based system developed using an IDE. The system should connect with a backend database and should have at least 5 different forms and suitable functionality including insert, edit and delete of main entities and transaction processing.

3. Presentation

With the submitted system student should do a presentation to demonstrate the system that was developed. Time allocated is 10 to 15 min. Student may use 5 to 10 PowerPoint slides while doing the presentation, but live demonstration of the system is required. Evaluator will also check the ability to modify and debug the system using the IDE.

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

Assignment Brief and Guidance:

Activity 1

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

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

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

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

n! = n * (n - 1) * …….. 1

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

Activity 2

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

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

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

The vehicles that it owns are hired or rented with or without a driver. The tariffs are based on the vehicle type. Some of the vehicle types that it operates are, small car, sedan car, SVUs, Jeep (WD), 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:

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

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

Activity 3

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

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

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

Write suable algorithms for vehicle tariff calculation for rents and hires. Ideally 3 functions should be developed for this purpose as above. Use the visual studio IDE (using C#.net) to Implement the above algorithms and design the suitable database structure for keeping the tariffs for vehicle types and different packages which must be used for implementing the above functions.
Analyze the features of an Integrated Development Environment (IDE) and explain how those features help in application development. Evaluate the use of the Visual StudioIDE for your application development contrasted with not using an IDE.

Activity 4

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

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

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

Grading Rubric

Grading Criteria	Achieved	Feedback
LO1 Define basic algorithms to carry out an operation and outline the process of programming an application.		
P1Provide a definition of what an algorithm is and outline the process in building an application.		
M1Determine the steps taken from writing code to execution.		
D1 Evaluate the implementation of an algorithm in a suitable language. Evaluate the relationship between the written algorithm and the code variant		
LO2 Explain the characteristics of procedural, object orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE)		
P2Give explanations of what procedural, object orientated, and event driven paradigms are; their characteristics and the relationship between them.		
M2 Compare and contrast the procedural, object orientated and event driven paradigms used in given source code of an application		
D2Critically evaluate the source code of an application which implements the programming paradigms, in terms of the code structure and characteristics.		

LO3Implement basic algorithms in code using an IDE.		
P3Write a program that implements an algorithm using an IDE.		
M3Use the IDE to manage the development process of the program.		
D3Evaluate the use of an IDE for development of applications contrasted with not using an IDE.		
LO4 Determine the debugging process and explain the importance of a coding standard		
P4Explain the debugging process and explain the debugging facilities available in the IDE.		
P5Outline the coding standard you have used in your code.		
M4Evaluate how the debugging process can be used to help develop more secure, robust applications.		
D4 Critically evaluate why a coding standard is necessary in a team as well as for the individual.		

Acknowledgement

I would like to express my special thanks of gratitude to my programing lecturer Mr. Praveen for providing invaluable guidance and giving immense amount of knowledge to work on this assignment perfectly. I specially thanks him because he helped us in doing a lot of research and I came to know about so many new things about the computer programming.

Secondly, I would like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Executive Summery

This entire assignment is based on an implementation of a Car rent and hire system design for selected company (Ayubo Drive). The purpose of this assignment is to improve programing skills.

This report includes basic programming principles, creating algorithms, understanding time complexities, creating basic programs by using Python language and C# language, understanding different programming paradigms, working with IDEs, Basic system creation using Microsoft SQL and Visual Studio C# windows forms application, debugging process and coding standards.

List of figures

List of Tables

TABLE OF CONTENTS

Activity 1

1.1 Algorithms

1.1.1 What is an Algorithm

A set of instructions for solving a problem is referred to as an algorithm. The best approach to comprehend an algorithm is to consider it as a recipe, which contains detailed guidelines for producing a dish. To accomplish its operations, every computer device utilizes algorithms in the form of hardware or software-based routines. In computer programming terms, an algorithm is a series of well-defined instructions for solving a certain problem. It accepts a collection of inputs and produces a desired output.

The Google Hangouts use audio and video compression algorithm to transmit live video across the Internet so quickly. The Google Maps use route finding algorithm to figure out to get one location to another. The Pixar use rendering algorithm to color a 3D model of a character based on the lighting in a virtual room. The Nasa use an optimization and scheduling algorithm to arrange the solar panels on the International Space Station. Those algorithms are more complex than our everyday algorithms like adding two numbers. But they follow the same thing to accomplish a task.

1.1.2 Characteristics of a good algorithm

Algorithms are language independent. In other words, they are simply directions that may be executed in any language and produce the same result. Not all written programming instructions are algorithms. Some instructions must have the following features in order to be considered an algorithm. These are the characteristics that a successful algorithm should possess.

1) Unambiguous and clear:

The algorithm should be straightforward and clear. Each of its phases should be obvious in every way and should only lead to one meaning. Algorithms must specify each step in the process, as well as the sequence in which the steps must be completed. Definiteness refers to the order in which operations are performed to convert input into output. The algorithm should be simple and straightforward. Each step's specifics must also be specified (including how to handle errors). Everything in it should be quantitative, not qualitative. For example, we can't expect a machine to comprehend something if we're unsure about it.

2) Well-Defined Inputs:

If an algorithm requires inputs, the inputs should be well-defined. The input is the data that will be modified to produce the output during the computation. There should be 0 or more well-defined inputs in an algorithm. Input precision demands knowing what type of data, how much of it there should be, and in what format it should be.

3) Well-Defined Outputs:

The algorithm must clearly specify and well-define what outcome will be produced. The data obtained as a result of the computation is the output. A well-defined algorithm should have one or more outputs that match the desired output. Precision in output also demands knowing what type of data, how much, and in what format the output should be.

4) Finiteness:

The algorithm must have a limited number of steps. In other words, the algorithm should not wind up in infinite repetitions or something like. The algorithm will finally come to a complete stop. Stopping could indicate that we have received the intended result. Algorithms must come to a stop after a certain number of steps. An algorithm should never be endless and should always end after a set number of steps. It's pointless to create an infinite algorithm because it will be useless to us.

5) Feasible:

The algorithm has to be simple, generic, and practical, and it must be able to be executed using the given resources. It must not include any futuristic technologies or anything else. All of the steps needed to get to the output must be achievable with the given resources for an algorithm to be effective. It should not include any extra or redundant steps that could make the algorithm useless. For example, if we are preparing a recipe and we cut vegetables that will not be utilized in the recipe, we are wasting our time.

6) Language Independence:

The proposed Algorithm must be language independent. In other words, it must be simple instructions that can be implemented in any language and produce the expected results. Step-by-step instructions should be included in an algorithm, and they should be independent of any programming code. It should be written in such a way that it may be used with any programming language.

1.1.3 Advantages and disadvantages of an Algorithm

Advantages of Algorithms :

- Algorithms are simple to comprehend.
- An algorithm is a logical representation of a solution to a problem.
- The problem is divided into smaller bits or phases in Algorithm. As a result, it is easy for the programmer to translate it into a working program.

If we know something about current algorithms, we can save time and make our programs run faster by using the correct one. Some of the uses of algorithms today world as below.

New algorithms are constantly being developed in the biological sciences for applications such as constructing the molecular structures that are at the core of disease-fighting medicines. Algorithms can be used in physics to simulate climate and weather patterns. Algorithms are used to search and analyze the large amounts of data collected by automated space telescopes on stars in the universe. In many fields, and even in everyday life, efficient algorithms are required to analyze massive data sets or to wisely choose from a wide number of possible decisions.

Disadvantages of Algorithms :

- It takes a long time to write an algorithm, hence it is time-consuming.
- Algorithms make it a lot harder to demonstrate Branching and Looping statements.

1.1.4 Difference between Algorithm and Flowchart

Algorithms and flowcharts both help programmers in the same way, yet they are slightly different. As previously stated, an algorithm is a step-by-step procedure for solving a certain problem. A flowchart is a visual depiction of an algorithm that uses geometrical diagrams and symbols to demonstrate how it works.

Flowcharts are widely used by programmers as a problem-solving technique. It employs symbols that are linked together to represent the flow of information and processing. The process of drawing a flowchart for an algorithm known as “flowcharting”.

Below table represent some differences between Algorithm and Flowchart

Algorithm	Flowchart
This is a step-by-step procedure to solve a problem.	This is a diagram made up of several shapes that displays data flow.
The program's pseudo code is called an algorithm.	A flowchart is just a graphical representation of the logic of an algorithm.
Complex to understand.	Easy to understand.
Plain texts are used in algorithms.	Symbols and Shapes are used in flowchart.
Easy to debug.	Hard to debug.
Difficult to construct.	Simple to construct.
Doesn’t follow any rules.	Follow rules to be constructed.

Table 1. 1 Differences between algorithm and flowchart.

1.1.5 Relationship between the written algorithm and the code variant

Mathematicians and programmers use algorithms to show and specify a set of steps and functions that must be followed in order to solve a problem. The sequence and these phases are both fixed; the only variables are the inputs. Algorithms can be used to solve anything from simple problems to complex functions, depending on the problem.

But in other hand code, often known as coding, is a set of instructions given to a machine or computer in such a way and language that it can understand and carry out the commands. These are usually in the form of high-level languages that machines can understand and translate.

Coding is a low-level programming language that is used to specify repeated functions like as controlling a device's display, validating transactions, and so on. Algorithms, on the other hand, are high-level and complicated systems capable of bringing projects, scientific curiosity, and research to life.

Algorithms are considered to be language independent and can be written in any language. If it fulfills the goal of ensuring steps for solving a problem, it will be considered an algorithm. But coding is dependent on language. If we are coding in HTML, we will be utilizing a defined set of terminology, commands, and instructions, but if we are working in C++, everything will be completely different. We are unable to write computer programming programs in any language.

1.2 The steps taken from writing code to execution

1.2.1 The Programming Process

Developing a software requires the same processes as any other problem-solving activity. The programming process is made up of five steps as followings.

1. Defining the problem
2. Planning the solution
3. Coding the program
4. Testing the program
5. Documenting the program

1.2.2 The Programming Process Step 01: Defining the problem

To study the problem, we meet with users from the client organization, or we meet with a systems analyst who describes the project. The task of describing the problem involves deciding what we already know (input-the data) and what we want to obtain (output-the result). Eventually, we establish a written agreement that describes the type of input, processing, and output required, among other things. This is not an easy task.

1.2.3 The Programming Process Step 02: Planning the solution

Drawing a flowchart and writing pseudocode, or maybe both, are two typical methods for preparing a problem's solution. A flowchart is basically a visual representation of a step-by-step solution to a problem. It is made of arrows that show the program's direction and boxes and other symbols that represent activities. It's a road map for what our program will accomplish and how it will accomplish it.

Pseudocode is a nonstandard English-like language that allows us to express our solution with greater detail than plain English. This requiring less precision than a formal programming language. Pseudocode allows us to focus on the program logic without having to worry about the specific syntax of a programming language just yet. Pseudocode, on the other hand, is not computer executable.

1.2.4 The Programming Process Step 03: Coding the program

The next stage for us as programmers is to code the program or express our solution in a programming language. We'll convert the logic from the flowchart to a programming language using pseudocode or another tool. A programming language is a set of rules for guiding the computer on what operations to perform. To make any program operate, we must strictly abide by the rule's syntax of the programming language.

1.2.5 The Programming Process Step 04: Testing the program

Some experts believe that if a program is well-designed, it can be developed right the first time. Most programmers, on the other hand, have become used to the fact that their freshly built programs are likely to include a few bugs. This can be discouraging at first, because programmers are typically accurate, careful, and detail-oriented people who take pride in their work. We must eventually be ready to test the software on the computer after we have finished developing it. These 2 phases are included in this process.

- Desk Checking :- The process of manually reviewing a program's source code is known as desk checking. It involves reading through the code and manually checking each function, frequently with different input values.

- Debugging :- Debugging is the process of finding and removing errors in software code that can cause it to act incorrectly or crash. The software will be ready to use after the bug has been repaired.

1.2.6 The Programming Process Step 05: Documenting the program

A written complete description of the work - flow and specific details about the program is known as documentation. Logic tools such as flowcharts and pseudocode, data-record descriptions, program listings, and testing results are typical program documentation resources. Comments within the program are also considered to be an important part of the documentation. Many programmers keep track of their work as they code. Program documentation, in a broader sense, might be included in the documentation for a whole system.

1.3 Asymptotic Analysis: Big-O Notation

1.3.1 What is Asymptotic Analysis

The amount of time, storage, and other resources required to execute an algorithm affects its efficiency. Asymptotic Notations are used to calculate efficiency. An algorithm's performance may differ depending on the type of input. The performance will vary as the input size increases. Asymptotic analysis is the study of how the algorithm's performance changes as the order of the input size changes.

As described, Asymptotic Notations are mathematical notations that are used to express the running duration of an algorithm when the input moves toward a specific value or a limiting value. For example, when an input array is already sorted, the algorithm takes linear time, which is the best scenario. However, when the input array is in reverse order, the algorithm takes the longest (quadratic) time to sort the items, which is the worst scenario. It takes average time when the input array is not sorted or in reverse order. Hence, Asymptotic notations are used to represent these durations.

There are three types of asymptotic notations:

1. Big-O notation → The upper bound of an algorithm's execution time is represented by the Big-O notation. As a result, it determines an algorithm's worst-case complexity.
2. Omega notation → The lower bound of an algorithm's running time is represented by the omega notation. As a result, it gives the algorithm's best-case complexity.
3. Theta notation → The function from above and below is enclosed in theta notation. It is used to analyze an algorithm's average-case complexity since it represents the upper and lower bounds of the algorithm's execution time.

1.3.2 Big O notation

In computer science, the Big O notation is used to express the performance or complexity of an algorithm. Big O is a term that describes the worst-case scenario and can be used to express the amount of time an algorithm takes to execute or the amount of space it takes up in memory or on disk.

As a programmer, the best approach to fully comprehend Big O is to use code examples. So, below are some popular growth orders, along with descriptions and examples with Python algorithms.

O(1) → Constant Time

O(1) also known as " constant time". Which means the run time of the algorithm takes the same amount of time to run regardless of the size of the input. Addition, subtraction and the majority of fundamental search operations are all assumed to run at constant time.

Below figure is an algorithm of reading index of an array by Python. When considering O(1) where “1” is the number of times, we want it to execute. No matter how long the array is, the algorithm runs 1 time when execute.

Python code:

```
my_List = ["Ryan","Dexter","Yagi"]

print (my_List[0])
```

Output :

```
C:\Us
Ryan
```

Figure 1. 1 Example Python code and output for demonstrate Big O notation “constant time”

O(n) → Linear Time

O(n) also known as "linear time". Which means the run time of the algorithm grows at the same pace as the input. For loops are an excellent example of a linear time operation.

Below figure is an algorithm of "for loop" by Python. When considering O(n) where "n" is the number of times, we want it to execute. According to below algorithm as we set range(5), it runs the algorithm 5 times which result printing 5 times of "Hello World". If we set range(100), it runs the algorithm 100 times which result printing 100 times of "Hello World".

Python code:

```
for i in range (5):
    print ("Hello World")
```

Output :

```
C:\Users\ryanc
Hello World
Hello World
Hello World
Hello World
Hello World
```

Figure 1. 2 Example Python code and output for demonstrate Big O notation "linear time"

O(n²) → Quadratic Time

O(n^2) also known as " Quadratic time". Which means that the calculation takes the squared size of the input data to perform. Nested For loops are an excellent example of a Quadratic time operation.

Below figure is an algorithm of " nested for loops" by Python. According to below algorithm as we set range(3) in outer loop, it runs 3 times. And as we set range(2) in inner loop, it runs 2 times for each iteration of the outer loop. As a result, this function takes O(n^2) times to perform.

Python code:

```
for i in range (3):
    for j in range (2):
        print ("Hello World")
```

Output :

```
C:\Users\rya
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

Figure 1. 3 Example Python code and output for demonstrate Big O notation "quadratic time"

1.3.3 Calculating time complexity

Knowing the Big O Notation, how it's computed, and what constitutes an acceptable time complexity for an algorithm will give us an edge over competitors when it comes to job seeking. One thing to be noted is that time complexity is analyzed for very large input sizes and worst-case scenario.

Big O can be calculated using the following five steps:

- Divide our algorithm or function into separate operations.
- Determine each operation's Big O value.
- Add all of the operations' Big Os together.
- Get rid of the variables.
- Find the highest order word – this is what our algorithm/function will be called the Big O.

Let's start easy and compute the Big O of a simple function with this in mind.

I'll take a polynomial example and try to calculate its time complexity.

$T(n) = 2n^2 + 3n + 1$

First step is to drop the lower order terms. That are 3n + 1 in this case.

$T(n) = 2n^2 + \sout{3n + 1}$

Next step is to drop all the constant multipliers. Hence, I can drop the constant that is 2.

$T(n) = \sout{2}n^2 + \sout{3n + 1}$

So, I can get the time complexity as big O of n^2 as following.

Time complexity → $O(n^2)$

Now let's look at some rules for finding out the time complexity in algorithms.

1.3.4 Finding out the time complexity for simple operations

First let us look at how do we calculate the time complexity of a simple calculation.

```
total = num1 + num2

print (total)
```

Figure 1. 4 Finding time complexity for simple operations sample question

Whatever the inputs are, this procedure will take the same amount of time to complete.

Hence, time complexity will be constant for this operation which is a constant taken as "C".

Therefore, time complexity will come out as C times. → C

But with Big O terms, we can neglect the constant "C". → ~~C~~

And simply we can write the time complexity as big O of n as following.

Time complexity → O (1)

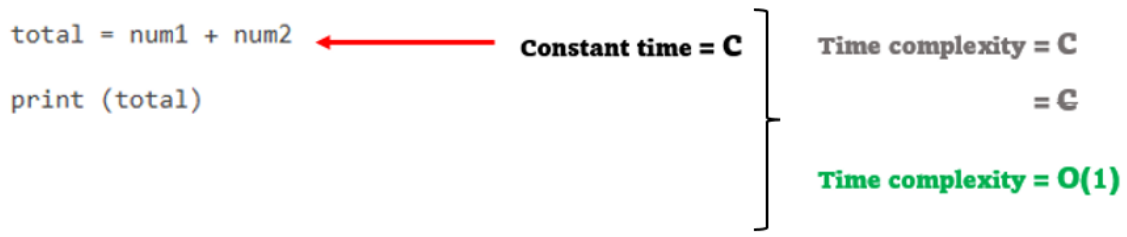


Figure 1. 5 Finding time complexity for simple operations answer for the question

1.3.5 Finding out the time complexity for Single Loop

First let us look at how do we calculate the time complexity of a single loop.

```
for (i = 1; i <= n; i++)
{
    a = 1+ 2
}
```

Figure 1. 6 Finding out the time complexity for Single Loop sample question

So above represent is a “for loop” which runs from 1 to n. “n” could be any number. Then there is simple mathematical operation inside the for loop.

The operation inside the for loop takes constant time to execute. Hence, time complexity will be n times time is taken for this operation which is a constant taken as “C”.

Therefore, time complexity will come out as C times n. → C n

But with Big O terms, we can neglect the constant “C”. → ∈ n

And simply we can write the time complexity as big O of n as following.

Time complexity → O (n)

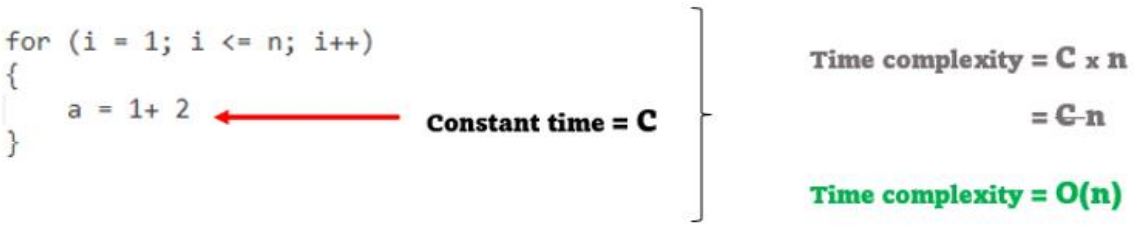


Figure 1. 7 Finding out the time complexity for Single Loop answer for the question

1.3.6 Finding out the time complexity for Nested Loops

Now let's calculate the time complexity of a nested loops.

```
for (i = 1; i <= n; i++)
{
    for (k = 1; k <= n; k++)
    {
        a = 1+ 2
    }
}
```

Figure 1. 8 Finding out the time complexity for Nested Loops sample question

As we can see, I have represented 2 loops which are nested. As I did before the operation inside of the inner for loop takes constant time to execute. Hence, time complexity will be n times time is taken for this operation which is a constant taken as "C". Now I can simply multiply the time complexity of each loop. Hence, outer loop runs n times. For each run of the outer loop, the inner loop will also run n times.

So, the total running time will be Constant "C" time multiplied by into n into n. → C n^2

But with Big O terms, we can neglect the constant "C". → ~~C~~ n^2

And simply we can write the time complexity as big O of n as following.

Time complexity → O (n^2)

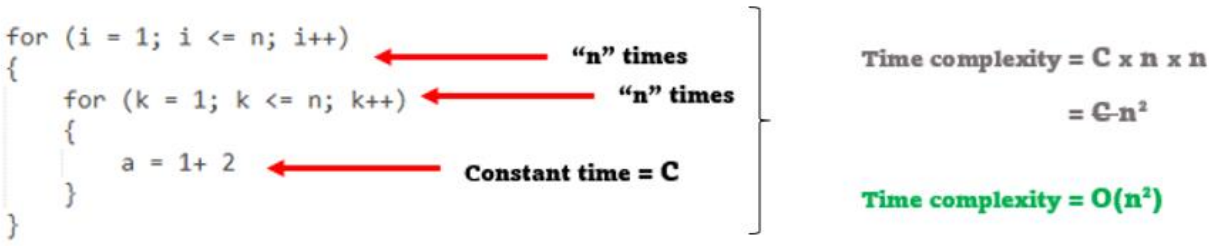


Figure 1. 9 Finding out the time complexity for Nested Loops answer for the question

1.3.7 Finding out the time complexity for Sequential Statements

Now let's calculate the time complexity of a nested loops.

```
a = 1 + 2

for (i = 1; i <= n; i++)
{
    print ("Hello")

for (k = 1; k <= n; k++)
{
    print ("Hello World")
}
```

Figure 1. 10 Finding out the time complexity for Sequential Statements sample question

The first operation of above expression takes constant time to run. → C_1.

After that we have this "for loop". As I explained before, for loop will take C into "n" times to run. → C_2 n

Then we have another for loop which takes C_3 into n times to run. → C_3 n

So, when we have sequential statements like these, we simply add the time taken for each statement. → $C_1 + C_2 n + C_3 n$

As I told before in Big-O terms, we can ignore constants which are C_1, C_2 and C_3.

→ ~~C_1~~ + ~~C_2~~ n + ~~C_3~~ n

And simply we can write the time complexity as big O of n as following.

Time complexity → O (n)

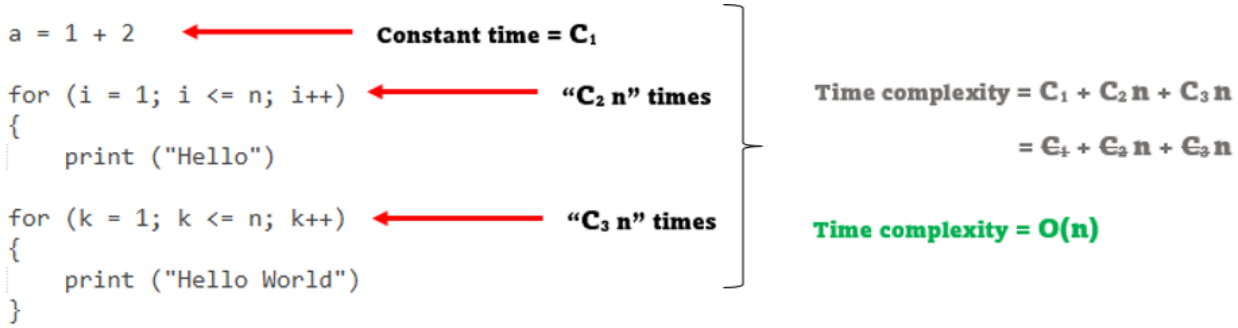


Figure 1. 11 Finding out the time complexity for Sequential Statements answer for the question

1.3.8 Finding out the time complexity for Conditional Statements

In conditional statements such as "if else" conditions, we evaluate each situation.

```
if
{
    //for loop
}

else
{
    //netsted for loop
}
```

Figure 1. 12 Finding out the time complexity for Conditional Statements sample question

Let's say inside if condition we have a for loop whose time complexity is → O (n)

And let's say inside else condition we have a nested for loop whose time complexity is

→ O (n^2)

We can consider the else condition as it has a greater time complexity.

Hence the overall time complexity will come out to be → O (n^2).

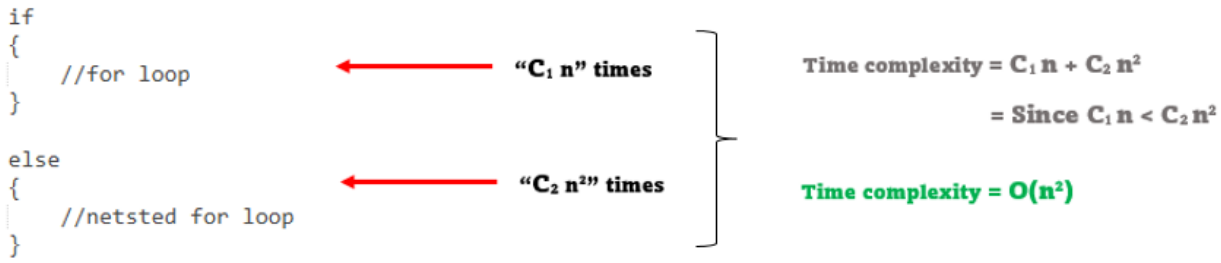


Figure 1. 13 Finding out the time complexity for Conditional Statements answer for the question

1.3.9 Comparison of time complexities

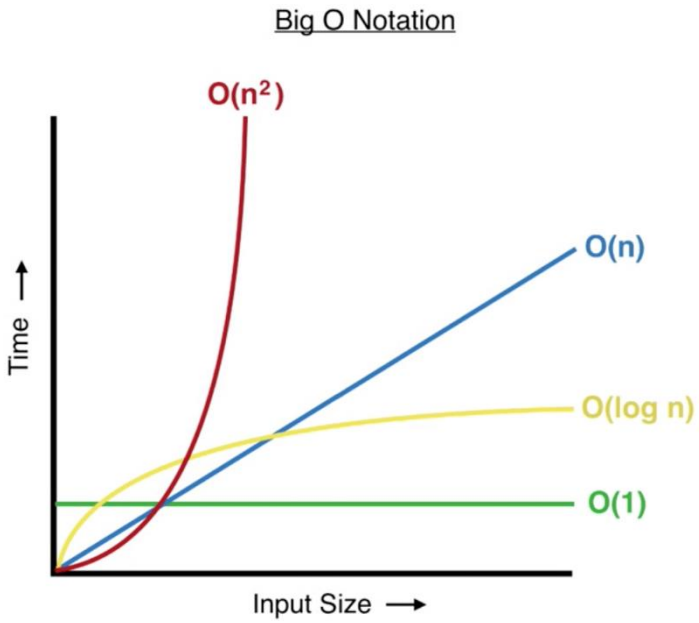


Figure 1. 14 Time Complexities Comparison Graph

According to above figure we can visually observe how the time increases as the amount of the input grows larger. And as we know time is proportional for number of operations. This will give us an idea of what an efficient runtime for more basic algorithms looks like.

O (1) → Big O of 1 is the smallest and the best.

Then Respectively O (log n) < O (n) < O (n^2)

O (n!) → Big O of n factorial is the biggest and the worst.

1.4 Algorithm for Fibonacci series

1.4.1 What is Fibonacci series

One of the most well-known mathematical formulas is the Fibonacci sequence. The sum of the two numbers before it determines the next number in the sequence.

Hence the order of the sequence as follows : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so forth.

The mathematical equation describing it is → $F_n = F_{n-1} + F_{n-2}$

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
x_n =	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	...

Figure 1. 15 Fibonacci series

By adding the two numbers before it, the following number is found:

- The 2 is obtained by multiplying the two numbers preceding it. (1+1)
- The 3 is obtained by multiplying the two numbers preceding it. (1+2)
- The 5 is obtained by multiplying the two numbers preceding it. (2+3)

1.4.2 Algorithm to display Fibonacci series for a given number using Pseudo code

STEP 1 : START

STEP 2 : Read nth value from user and store it as integer in “n” variable

STEP 3 : if n < 0 ask positive number and go to STEP 10 else go to STEP 4

STEP 4 : Declare following variables.

*Firstnum = 0 (string variable)

*Secondnum = 1 (string variable)

STEP 5 : Declare an array to store Fibonacci numbers

My_List = [ ]

STEP 6 : Use “for loop” for following steps until i = n +1 else go to STEP 7

- STEP 6.1 : Append Firstnum to My_List
- STEP 6.2 : Convert Firstnum and Secondnum string variables into integer
- STEP 6.3 : Next = Firstnum + Secondnum
- STEP 6.4 : Firstnum = Secondnum
- STEP 6.5 : Secondnum = Next
- STEP 6.6 : Convert Firstnum integer variable into string
- STEP 6.7 : i = i+1
- STEP 6.8 : Repeat STEP 6 until i = n +1

STEP 7 : print My_List to show Fibonacci series until to nth value

STEP 8 : Make reverse the My_List elements

STEP 9 : print 1st element of the My_List to show nth value of Fibonacci series

STEP 10 : STOP

1.4.3 Dry run the Fibonacci series algorithm by using pseudocode

The sample number used for Dry Run is number 3

STEP 1 : START

STEP 2 : 3 (Sample number which reading by the user)

STEP 3 : since 3 > 0 go to step 4

STEP 4 : Declare following variables.

*Firstnum = 0 (string variable)

*Secondnum = 1 (string variable)

STEP 5 : Declare an array to store Fibonacci numbers

My_List = [ ]

STEP 6 : Use “for loop” for following steps until i = 4 else go to STEP 7

STEP 6.1 : Append Firstnum = 0 to My_List → [0]

STEP 6.2 : Convert Firstnum = 0 and Secondnum = 1 variables into integer

STEP 6.3 : Next = Firstnum + Secondnum = 0 + 1

STEP 6.4 : Firstnum = Secondnum =1

STEP 6.5 : Secondnum = Next = 1

STEP 6.7 : Convert Firstnum = 1 into string

STEP 6.8 : i = 0 + 1 = 1

STEP 6.9 : Repeat STEP 6 until i = 4

STEP 6.10 : Append Firstnum = 1 to My_List → [0,1]

STEP 6.12 : Convert Firstnum = 1 and Secondnum = 1 variables into integer

STEP 6.13 : Next = 1 + 1

STEP 6.14 : Firstnum = Secondnum = 1

STEP 6.15 : Secondnum = Next = 2

STEP 6.17 : Convert Firstnum = 1 into string

STEP 6.18 : i = 1 + 1 = 2

STEP 6.19 : Repeat STEP 6 until i = 4

STEP 6.20 : Append Firstnum = 1 to My_List → [0,1,1]

STEP 6.21 : Convert Firstnum = 1 and Secondnum = 2 variables into integer

STEP 6.22 : Next = Firstnum + Secondnum = 1 + 2

STEP 6.23 : Firstnum = Secondnum = 2

STEP 6.24 : Secondnum = Next = 3

STEP 6.25 : Convert Firstnum = 2 into string

STEP 6.26 : i = 2 + 1 = 3

STEP 6.27 : Repeat STEP 6 until i = 4

STEP 6.28 : Append Firstnum = 2 to My_List → [0,1,1,2]

STEP 6.29 : Convert Firstnum = 2 and Secondnum = 3 variables into integer

STEP 6.30 : Next = Firstnum + Secondnum = 2 + 3

STEP 6.31 : Firstnum = Secondnum = 3

STEP 6.32 : Secondnum = Next = 5

STEP 6.33 : Convert Firstnum = 3 into string

STEP 6.34 : i = 3 + 1 = 4

STEP 6.35 : Since i (4) = 4 Go to STEP 7

STEP 7 : print My_List → [0,1,1,2] to show Fibonacci series until to nth value

STEP 8 : Make reverse the My_List → [2,1,1,0] elements

STEP 9 : print 1st element of My_List → "2" to show nth value of Fibonacci series

STEP 10 : STOP

Output is → [0,1,1,2] with 2

1.4.4 Dry run the above Fibonacci series algorithm by using Python

Fibonacci 1.py | Fibonacci 2.py | main.py | Test.py | Fibonacci 3.py | Fibonacci 4.py

```
# Fibonacci Sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……

n = int(input("Enter a value (nth value) to display nth Fibonacci number: "))

if n < 0:
    print("Please enter positive integer")

else:
    firstnum = "0"
    secondnum = "1"
    my_list = []

    for i in range(n+1):
        firstnum = str(firstnum)
        my_list.append(firstnum)

        firstnum = int(firstnum)
        secondnum = int(secondnum)
        next = firstnum + secondnum
        firstnum = secondnum
        secondnum = next

    #Display the fibonacci sequence
    print("Fibonacci Sequence is: ", " , ".join(my_list))

    #Display nth value
    my_list.reverse()
    fib = int(my_list[0])
    print("nth value is : " ,fib)
```

Figure 1. 16 Fibonacci series algorithm by using Python

Output →

The sample number used for Dry Run is number 3

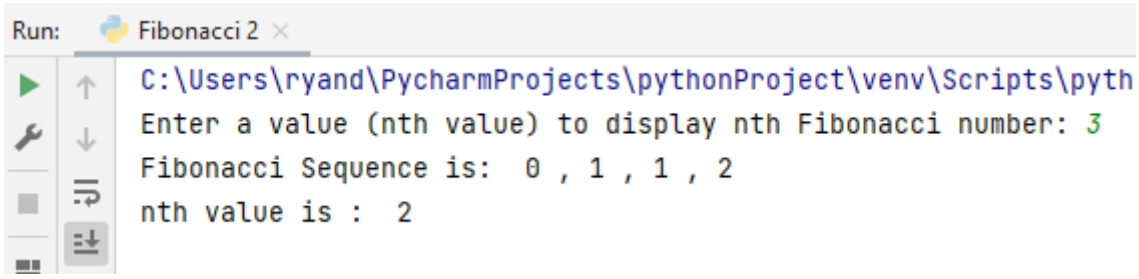


Figure 1. 17 Fibonacci series Dry Run taking sample number 3

The sample number used for Dry Run is number 2

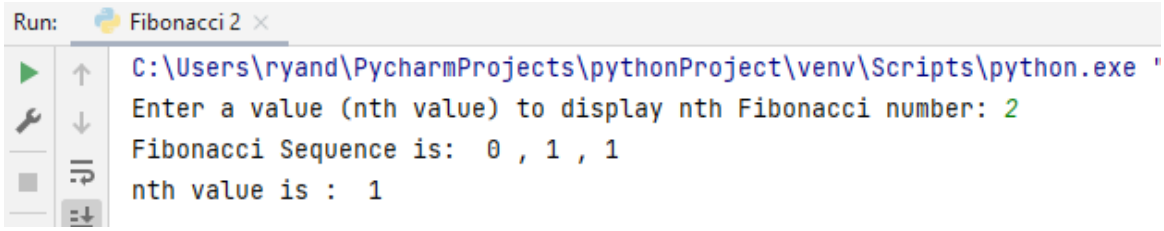


Figure 1. 18 Fibonacci series Dry Run taking sample number 2

The sample number used for Dry Run is number 1

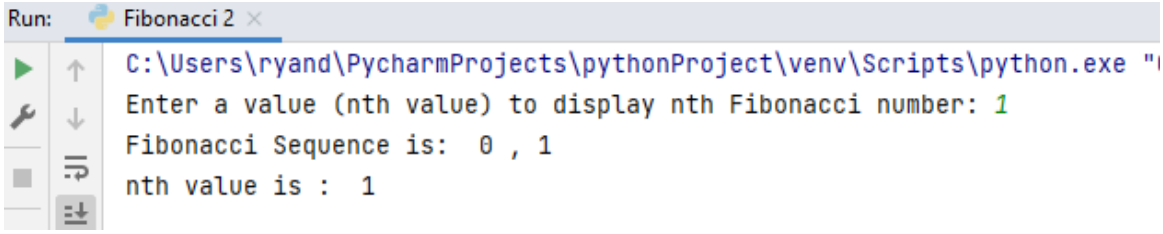


Figure 1. 19 Fibonacci series Dry Run taking sample number 1

The sample number used for Dry Run is number 0

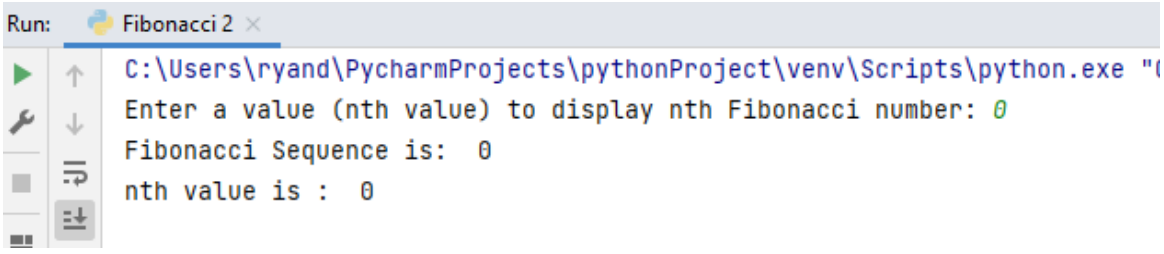


Figure 1. 20 Fibonacci series Dry Run taking sample number 0

The sample number used for Dry Run is number 10

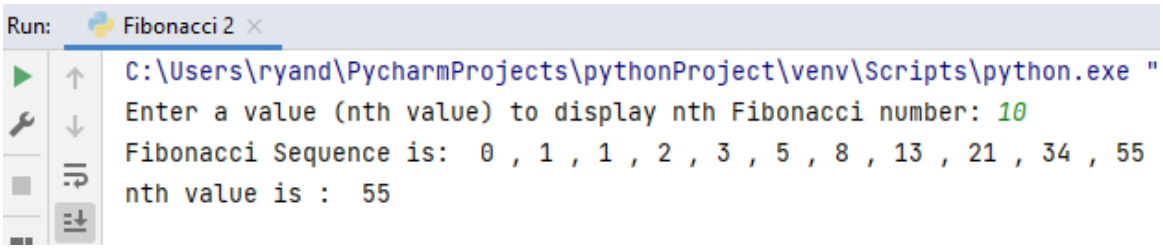


Figure 1. 21 Fibonacci series Dry Run taking sample number 10

1.4.5 Evaluation efficiency of Fibonacci series algorithm using Big-O notation

```
n = int(input("Enter a value (nth value) to display nth Fibonacci number: "))

if n < 0:
    print("Please enter positive integer")

else:
    firstnum = "0"
    secondnum = "1"
    my_list = []

    for i in range(n+1):
        firstnum = str(firstnum)
        my_list.append(firstnum)

        firstnum = int(firstnum)
        secondnum = int(secondnum)
        next = firstnum + secondnum
        firstnum = secondnum
        secondnum = next

    #Display the fibonacci sequence
    print("Fibonacci Sequence is: ", " , ".join(my_list))

    #Display nth value
    my_list.reverse()
    fib = int(my_list[0])
    print("nth value is : " ,fib)
```

Constant time = C_1

"C_2 n" times

Constant time = C_3

Constant time = C_4

$$\text{Time complexity} = C_1 + C_2 n + C_3 + C_4$$

$$= C_1 + C_2 n + C_3 + C_4$$

Time complexity = O(n)

Figure 1. 22 Evaluation efficiency of Fibonacci series algorithm using Big-O notation

Let's Evaluation efficiency of Fibonacci series algorithm using Big-O notation. As we learned to find out the time complexity for Conditional Statements, lets evaluate each situation.

The first operation in the if condition takes constant time to run. → C_1.

After in else condition we have this "for loop". As I explained in big O notation chapter, for loop will take C into "n" times to run. → C_2 n

Then after Conditional Statements there is a execute operation with an in-built function which takes constant time to run. → C_3.

The time complexity of accessing the value of an element at a certain index is constant. Whether the list has 5 or 500 elements, the complexity of retrieving the value at index 0 takes constant time to run. This is because, regardless of how many elements are in the array, the operations needed to access an element in memory remain constant.

Hence, despite of having in built function, the final part of the algorithm is fundamental accessing element of an array operation which takes also constant time to run. → C_4.

So, when we have sequential statements like these, we simply add the time taken for each statement.

→ $C_1 + C_2 n + C_3 + C_4$

As I told before in Big-O terms, we can ignore constants which are C_1, C_2 , C_3 and C_4.

→ ~~C_1~~ + ~~C_2~~ n + ~~C_3~~ + ~~C_4~~

And simply we can write the time complexity as big O of n as following.

Time complexity → O (n)

1.5 Algorithm for Factorial Value

1.5.1 What is Factorial Value

The factorial function (symbol is !) tells us to multiply all whole integers from 1 to our selected value.

n	n!		
1	**1**	1	1
2	2 × **1**	= 2 × **1!**	= 2
3	3 × **2 × 1**	= 3 × **2!**	= 6
4	4 × **3 × 2 × 1**	= 4 × **3!**	= 24
5	5 × **4 × 3 × 2 × 1**	= 5 × **4!**	= 120

Figure 1. 23 Factorial Value

The mathematical equation describing it is → $n! = n \times (n-1)!$

1.5.2 Algorithm to display Factorial Value for a given number using Pseudo code.

STEP 1 : START

STEP 2 : Read nth value from user and store it as integer in “n” variable

STEP 3 : Declare following variables.

*Count = n (integer variable)

*Fac = 1 ((integer variable)

STEP 4 : Declare an array to store Factorial numbers

My_List = [ ]

STEP 5 : Use “while loop” for following steps until count > 0 else go to STEP 6

- STEP 5.1 : Convert Count integer variables into string
- STEP 5.2 : Append Count to My_List
- STEP 5.3 : Convert Count string variables into integer
- STEP 5.4 : Count = Count -1
- STEP 5.5 : Repeat STEP 5 until count = 1

STEP 6 : Use “while loop” for following steps until n > 0 else go to STEP7

- STEP 6.1 : fac = fac * n
- STEP 6.2 : n = n -1
- STEP 6.3 : Repeat STEP 6 until n = 1

STEP 7 : print My_List to show Factorial Numbers until to nth value

STEP 8 : print fac to show the value of Factorial Numbers series

STEP 9 : STOP

1.5.3 Dry run the above Factorial Value algorithm by using pseudocode

STEP 1 : START

STEP 2 : 3 (Sample number which reading by the user)

STEP 3 : Declare following variables.

*Count = n = 3 (integer variable)

*Fac = 1 ((integer variable)

STEP 4 : Declare an array to store Factorial numbers

My_List = [ ]

STEP 5 : Use “while loop” for following steps until Count (3) > 0 else go to STEP 6

STEP 5.1 : Convert Count = 3 integer variables into string

STEP 5.2 : Append Count (3) to My_List → [3]

STEP 5.3 : Convert Count = 3 string variables into integer

STEP 5.4 : Count = Count -1 → Count = 3 -1 = 2

STEP 5.5 : Repeat STEP 5 until Count (2) = 0

STEP 5.6 : Convert Count = 2 integer variables into string

STEP 5.7 : Append Count (2) to My_List → [3,2]

STEP 5.8 : Convert Count = 2 string variables into integer

STEP 5.9 : Count = Count -1 → Count = 2 -1 = 1

STEP 5.10 : Repeat STEP 5 until Count (1) = 0

STEP 5.11 : Convert Count = 1 integer variables into string

STEP 5.12 : Append Count (1) to My_List → [3,2,1]

STEP 5.13 : Convert Count = 1 string variables into integer

STEP 5.14 : Count = Count -1 → Count = 1 -1 = 0

STEP 5.15 : Since Count (0) = 0 go to STEP 6

STEP 6 : Use “while loop” for following steps until n (3) > 0 else go to STEP7

STEP 6.1 : fac = fac * n → fac = 1 * 3 = 3

STEP 6.2 : n = n -1 → n = 3 – 1 = 2

STEP 6.3 : Repeat STEP 6 until n (2) = 0

STEP 6.4 : fac = fac * n → fac = 3 * 2

STEP 6.5 : n = n -1 → n = 2 – 1 = 1

STEP 6.6 : Repeat STEP 6 until n (1) = 0

STEP 6.7 : fac = fac * n → fac = 6 * 1

STEP 6.8 : n = n -1 → n = 1 – 1 = 0

STEP 6.9 : Since n (0) = 0 go to STEP 7

STEP 7 : print My_List → [3,2,1] to show Factorial Numbers until to nth value

STEP 8 : print fac → “6” to show the value of Factorial Numbers series

STEP 9 : STOP

Output is → [3,2,1] with 6

1.5.4 Dry run the above Factorial Value algorithm by using Python

ibonacci 2.py | Factorial Func 4.py | main.py | Test.py | Fibonacci 3.py | Fibonacci 4.py

```
1
2   n = int(input("Enter factorial number (!n): "))
3
4
5   count = n   #define variable for 1st loop
6   fac = 1
7   my_list = []
8
9   #Using loop to show Factorial number sequence
10  while count > 0 :
11      count = str(count)  #To convert integer to string when adding to list
12      my_list.append(count)
13
14      count = int(count)  #To convert string to integer when calculating
15      count = count -1
16
17  #Using loop to get Factorial number sequence final answer
18  while n > 0 :
19      fac = fac * n
20      n = n -1
21
22
23  print("!n = ", " * ".join(my_list), " = ",fac)
24
```

Figure 1. 24 Factorial Value algorithm by using Python

Output →

The sample number used for Dry Run is number 3

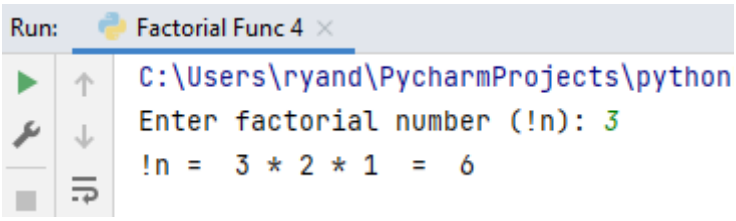


Figure 1. 25 Factorial Value Dry Run taking sample number 3

The sample number used for Dry Run is number 2

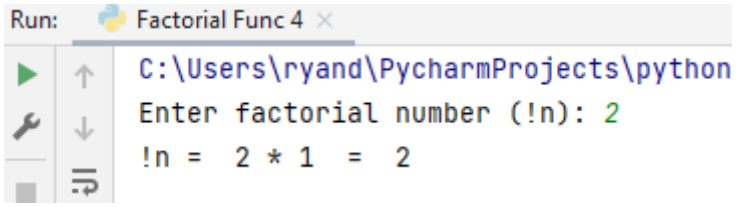


Figure 1. 26 Factorial Value Dry Run taking sample number 2

The sample number used for Dry Run is number 1

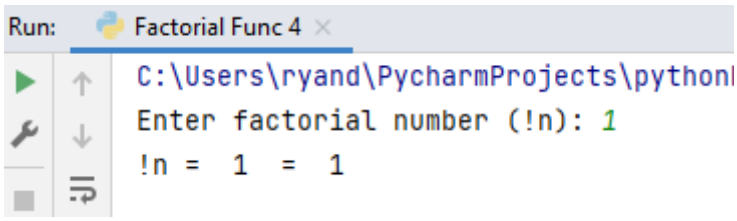


Figure 1. 27 Factorial Value Dry Run taking sample number 1

The sample number used for Dry Run is number 0

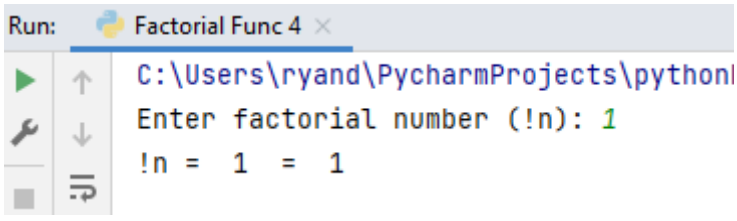


Figure 1. 28 Factorial Value Dry Run taking sample number 0

The Zero Factorial is exciting since it is widely accepted that 0! = 1.

It may seem strange that multiplying no numbers results in 1

The sample number used for Dry Run is number 10

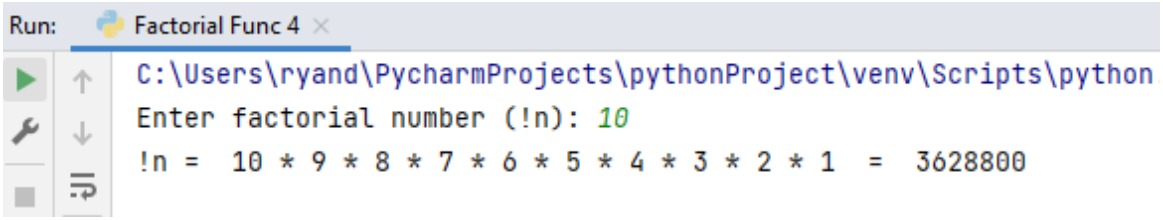


Figure 1. 29 Factorial Value Dry Run taking sample number 10

1.5.5 Evaluation efficiency of Factorial Value algorithm using Big-O notation

```
n = int(input("Enter factorial number (!n): "))


count = n   #define variable for 1st loop
fac = 1
my_list = []

#Using loop to show Factorial number sequence
while count > 0 :
    count = str(count)  #To convert integer to string when adding to list
    my_list.append(count)

    count = int(count)  #To convert string to integer when calculating
    count = count -1

#Using loop to get Factorial number sequence final answer
while n > 0 :
    fac = fac * n
    n = n -1

print("!n = " , " * ".join(my_list) , " = " ,fac)
```

"C_1 n" times

"C_2 n" times

Constant time = C_3

$$\text{Time complexity} = C_1 n + C_2 n + C_3$$

$$= C_1 n + C_2 n + C_3$$

$$\text{Time complexity} = O(n)$$

Figure 1. 30 Evaluation efficiency of Factorial Value algorithm using Big-O notation

Let's Evaluation efficiency of Factorial Value algorithm using Big-O notation. As we learned to find out the time complexity for Sequential Statements, lets evaluate each situation.

The first operation is the "while loop". As I explained in big O notation chapter, while loop is very similar to for loop will take C into "n" times to run. → C_1 n

After the while loop there is another while loop which will take C into "n" times to run. → C_2 n

Finally, there is a execute operation with an in-built function which takes constant time to run. → C_3.

So, when we have sequential statements like these, we simply add the time taken for each statement.

→ C_1 n + C_2 n + C_3

As I told before in Big-O terms, we can ignore constants which are C_1, C_2 , C_3 and C_4.

→ ~~C_1~~ n + ~~C_2~~ n + ~~C_3~~

And simply we can write the time complexity as big O of n as following.

Time complexity → O (n)

1.6 The process in building an application

App development may be a hard and time-consuming process that, at times, involves far too many operations for a single person to efficiently execute. There are 5 steps involving in this process as following.

1. Defining and analyzing the requirements
2. Designing the application
3. Developing the application
4. Testing the application
5. The application's deployment

Step 1: Defining and analyzing the requirements

The software developers gather detailed project-related information from the customer's supporting documentation. Then, using the requirements document, project plan, use cases, and requirements traceability, software developers create and evaluate the requirements for the application.

Step 2: Designing the application

Software developers will create the application in this stage, depending on the requirements, use cases, and project scope agreed upon in the previous stage. The design will be tested against the customer's specifications after it's finished. The project plan will be verified, and a cost estimate will be made. At this point, any necessary revisions are made.

Step 3: Developing the application

Developers will create the actual code based on the finalized design papers during the development stage. After that, the application will be put to the test against the customer's specifications and test scenarios. This step involves performing code testing, quality testing, and customer acceptance tests, as well as receiving feedback from the customer. This is also where debugging takes place. The customer's acceptance of the developed application brings this stage to a close.

Step 4: Testing the application

Frontend or user interface testing and backend testing, which checks database behavior, are the two phases of application testing. In this step software developer can eliminate the bug, reduce the development cost and future maintenance cos and improve the performance of the application.

Step 5: The application's deployment

The software program will be developed according to the deployment plan in the final stage of deployment. After the customer accepts the delivered software, the developer can maintain track of the customer and continue to provide assistance.

Activity 2

2.1 What is Programming Paradigm

2.1.1 What is Programming Paradigm

Computer programs are basically built and developed to solve a certain problem. Various programming techniques have evolved during the previous few decades. Each programming paradigm offers a set of principles and rules that must be followed by the programming language. During the software compilation step, the programming language compiler follows the programming paradigm. It is essential for computer science students, as well as me, to understand the concept of programming paradigms, various types, and corresponding languages.

Each paradigm emphasizes a certain way of managing program code. A programming language is actually a strategy for solving problems. There are numerous solutions to each problem. Furthermore, each solution may take a different approach to solving the problem. Different programming languages are used to develop the computer programs.

Each programming language has a specific programming style that follows a certain programming paradigm. The programming paradigm is everything about the writing style and how the program code is arranged.

Examples :-

- C programming language belongs to the procedural programming paradigm.
- C++, Python, and Java are examples of object-oriented programming (OOP) paradigms.

Programming paradigms are divided into two categories. The paradigm type is determined by the features of the programming language as well as the way in which the program code is organized.

1. Imperative Paradigm.
2. Declarative Paradigm.

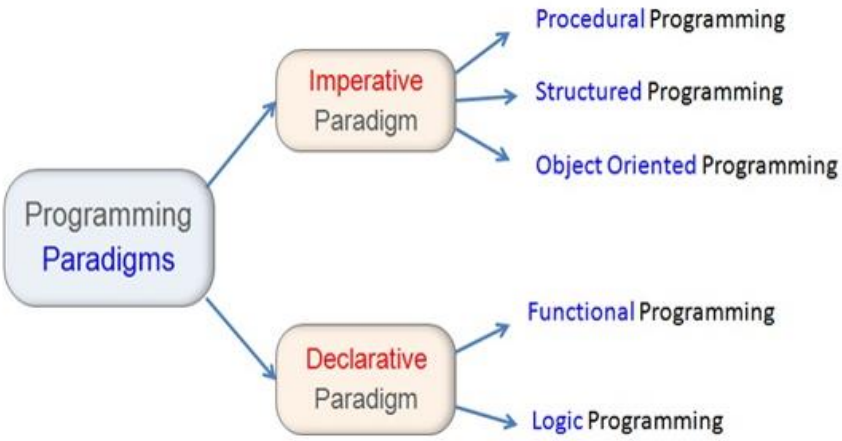


Figure 2. 1 Types of programming Paradigms

2.1.2 Imperative Paradigm.

Imperative paradigm is one of the oldest programming paradigms. It looks very similar to machine architecture. Its architecture is based on Von Neumann's. The imperative paradigm is defined as command driven. It works by changing the program's state with assignment statements. It completes tasks in a step-by-step manner by changing states. Which means in an imperative paradigm programming language, the program code leads program execution as a series of statements executed one by one. The imperative style program is made up of a series of program statements.

Each sentence instructs the computer to carry out a specific task. The programmer must elaborate on each statement in an imperative style program. Each statement describes what should be done and how it should be done. The control flow statements determine how the program statements are executed. Furthermore, the program flow can be adjusted based on the program logic. The main objective of imperative paradigm is on how to achieve the goal.

Imperative Programming Types:

1. Structured Programming.
2. Procedural Programming.
3. Object Oriented Programming.
4. Database Query Language.

2.1.3 Declarative Paradigm.

Declarative programming is a way of constructing programs in computer science that expresses computing logic without detailing its control flow. It is a programming paradigm that focuses on program logic and the final outcome. The control flow is not the most important aspect of the program under this paradigm. The basic goal of declarative programming is to achieve the desired result. While writing the program code, this paradigm is straightforward and to the point. It could make writing parallel programs easier.

The emphasis is on what must be done rather than how it should be done, emphasizing what code really does. It just declares the desired outcome rather than how it was achieved. The sole difference between imperative (how to do) and declarative (what to do) programming paradigms is that this is the only difference.

Declarative Programming Types:

1. Functional Programming.
2. Logic Programming.

2.2 Procedural Programming Paradigms

2.2.1 Main characteristics of Procedural Programming Paradigms

Procedural Programming is likely to be a new developer's first programming paradigm. A procedural paradigm program is made up of a set of procedures. Simply put, procedural programming is writing down a list of instructions that instruct the computer how to accomplish a task step by step.

Functions, methods, and subroutines are all terms used to describe procedures. In procedural programming, the program structure is made up of a series of functions. Each one carries out a unique operation. The function is made up of a series of computational steps that tell the computer how to complete a given task. Once defined, the function can be called multiple times throughout the program to repeat the same operation. The programmer has the choice of using standard library functions or creating a user-defined function library.

The C programming language is the most widely used and broadly applied programming language in the software industry. C++, Java, ColdFusion, Pascal languages are some examples for Procedural programming paradigm.

2.2.2 Advantages and Disadvantages of Procedural Programming

Advantages :-

- For general-purpose programming, procedural programming is suitable.
- A wide range of books and online course materials on tested algorithms are available, making it easy to study.
- Because the source code is portable, it can also be used to target a different CPU.
- It is not necessary to duplicate the code because it can be reused in different parts of the program.
- The memory need is also reduced using the Procedural Programming technique.
- The program's flow can be simply followed.

Disadvantages :-

- When Procedural Programming is used, the computer code is more difficult to write.
- Since procedural code is frequently not reusable, it may be necessary to rebuild it if it is required for usage in another application.
- Relationships with real-world objects are complicated.
- The operation is prioritized over the data, which may cause problems in data-sensitive situations.
- Since the data is visible to the entire program, it is not very secure.

2.2.3 Procedural Programming Paradigm example snippets of code

Example Python Code :-

```
print ("Enter your makrks for each subjecrt please")
print ("========================================")
Mark1 = int(input("First subject mark :- "))
Mark2 = int(input("Secont subject mark :- "))
Mark3 = int(input("Third subject mark :- "))

SUM = Mark1 + Mark2 + Mark3
Average = SUM/3
print("Sum of the marks are:- " , SUM)
print("Average of the marks are:- " , Average)

if (Average>=75):
    print("Grade is A")
elif (Average>=60):
    print("Grade is B")
elif (Average>=40):
    print("Grade is C")
else:
    print("Grade is F")
```

Figure 2. 2 Procedural Programming Paradigm example Python code

Output :-

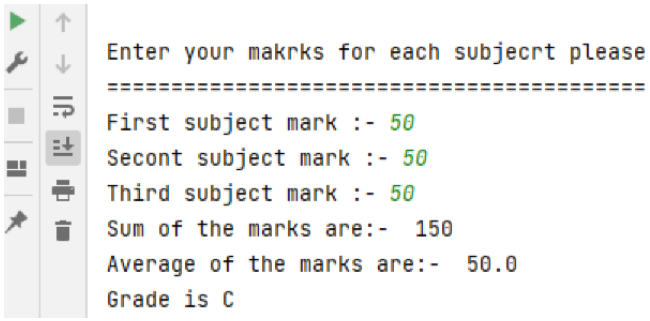


Figure 2. 3 Procedural Programming Paradigm example Python code output

The basic grade calculation shown above. To determine the grades, this python code follows a step-by-step approach. The first two lines of code print a string. The first step is to collect the three subject marks. The second step is to add up the marks to get the total. The average is then calculated by dividing the total by the number of inputs. Finally, the program compares the average marks to the predefined evaluation grades to calculate the grade using the if condition.

The program clearly follows a step-by-step approach, as can be seen. We can see how the program flows by looking at the results. It runs line by line, step by step to produce the desired results.

Below here stated another few examples of codes for Procedural Programming Paradigm.

Example Python Code :-

```
def even_check (num_list) :

    for x in num_list:
        if x % 2 == 0:
            return x
        else:
            pass

x = even_check([1,2,3,4])
print(x)
```

1 User defined function

2 Call the function and print it

Figure 2. 4 Procedural Programming Paradigm example Python code

Output :-

```
Run:  main  Test
C:\Users\ryand\PycharmProjects\p
2
```

Figure 2. 5 Procedural Programming Paradigm example Python code output

Example C# Code :-

```
0 references
static void Main(string[] args)
{
    GetMax(1, 10);
    Console.ReadLine();
}

1 reference
static void GetMax (int num1 , int num2)
{
    int result;
    if (num1>num2)
    {
        result = num1;
        Console.WriteLine(result);

    }
    else
    {
        result = num2;
        Console.WriteLine(result);
    }

}
```

2 — Call the user defined method in Main Method and Read it.

1 — User defined new method below Main Method

Figure 2. 6 Procedural Programming Paradigm example C# code

Output :-

```
C:\Users\ryand\source\repos\Testing\Testing\bin\Debug\Testing.exe
10
```

Figure 2. 7 Procedural Programming Paradigm example C# code output

2.3 Object-Oriented Programming Paradigms

2.3.1 Main characteristics of Object-Oriented Programming Paradigms

Object-oriented programming (OOP) is a style of structured programming in which program components are represented as objects. The program is composed of a set of classes and objects that are used to communicate. All program components are represented as objects in OOP programming. Objects are the smallest and most basic entities, and all computations are done on them.

Data and associated methods are bound together as a single unit by an object. As a result, the programmer can define the access specifier to manage data access permissions. The focus is on data rather than procedure. The data in the program is protected by OOP programming from unintentional operations performed by other methods. As a result, object-oriented programming provides strong security features.

For enterprise-level software projects, OOP-compliant languages such as C++, Java, and Python are widely used.

2.3.2 Advantages and Disadvantages of Object-Oriented Programming

Advantages :-

- OOP is easy to maintain because of its flexibility and encapsulation.
- OOP simulates the real world, making it simpler to understand.
- Since objects are complete in and of themselves, they can be reused in other programs.

Disadvantages :-

- Object-Oriented programs are typically slower and consume a lot of memory.
- Over-generalization
- It is possible that programs designed using this paradigm will take longer to develop.

2.3.3 Object-Oriented Programming Paradigm example snippets of code

Example Python Code :-

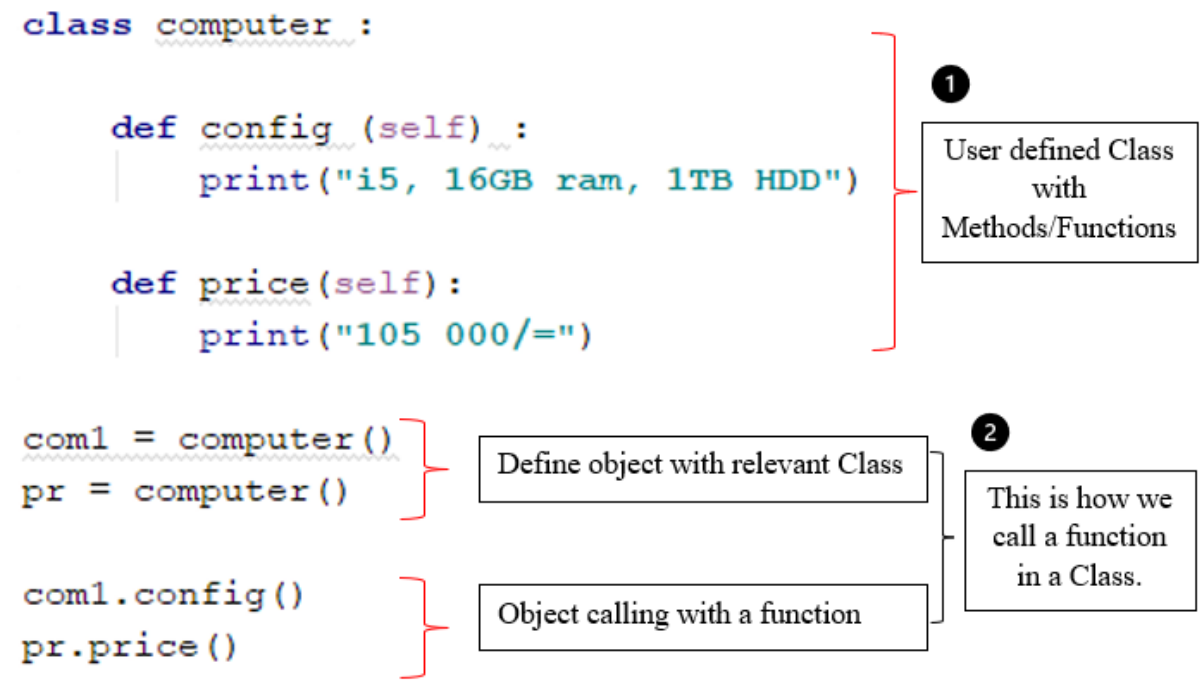


Figure 2. 8 Object-Oriented Programming Paradigm example Python code

Output :-

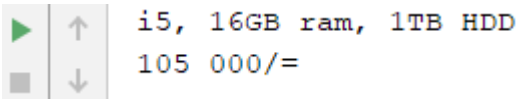


Figure 2. 9 Object-Oriented Programming Paradigm example Python code output

When looking at the above python code, I have defined 1 Class called Computer. In this class I have defined two methods named "config" and "price" then I've introduced a specific task for each method also. Later at the end I've defined 2 objects under this class then called out the methods by the object's name.

The programmer can capture data and keep it safe and secure from outside interfaces using the OOP programming paradigm. This is known as Encapsulation. The sharable class is a big benefit because it allows us to reuse the same code without duplicating it. If the programmer has to add additional data or functionalities, he or she can do so easily.

Below here stated another few examples of codes for Object-Oriented Programming Paradigm.

Example C# Code :-

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        //create objects for the class
        Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();

        //perform methods
        s1.attend_class("Ryan","Science");
        s3.sport();

        Console.ReadLine();
    }
}
```

2 Create objects

3 Call methods

Main Class

```
6 references
class Student
{
    //Define common methods
    1 reference
    public void attend_class(string name , string subject)
    {
        Console.WriteLine("Student " +name);
        Console.WriteLine("Is going to " + subject);
    }

    1 reference
    public void sport()
    {
        Console.WriteLine("Student play Sports");
    }
}
```

1 User defined methods in user defined Class "Student"

User defined Class

Figure 2. 10 Object-Oriented Programming Paradigm example C# code

Output :-

```
C:\Users\ryand\source\repos\Testing\Testing\bin\Debug\Testing.exe
Student Ryan
Is going to Science
Student play Sports
```

Figure 2. 11 Object-Oriented Programming Paradigm example C# code output

2.4 Event-Driven Programming Paradigms

2.4.1 Main characteristics of Event-Driven Programming Paradigms

The concept behind event-driven programming is that the program is built to react. Event-driven programming is a programming paradigm in which events control the flow of program execution. It responds to specified types of user input, such as a command button click, a drop-down list selection, a text box entry, a mouse click, a key press, a message from the operating system or another program, or other user events. An event-driven program is one that identifies events as they happen and reacts with an appropriate event-handling procedure.

Event-driven programs can be created in any programming language, while some languages such as Visual Basic are built specifically for event-driven programming. They also have an integrated development environment (IDE) that partially automates code development and includes a large number of built-in objects and controls, each of which can react to a number of different events. Event-driven programming is supported by almost all object-oriented and graphic programming languages. Examples of such languages are Visual Basic, Visual C++, and Java.

Many visual programming environments will even include code templates for event-handlers, requiring the programmer to simply input the code that defines the program's response to the event. Each event handler on a form is generally tied to a specific object or control. Any additional subroutines, methods, or function procedures are usually kept in their own code module and can be called from other parts of the program when needed.

2.4.2 Advantages and Disadvantages of Object-Oriented Programming

Advantages :-

- If a programmer needs to modify something, they can quickly adjust event-driven programming because of its flexibility.
- Event-driven allows the user to choose from a variety of tools on the toolbar to create exactly what they need, such as buttons, radio buttons, and other controls.
- By allowing users to directly modify the object for which they want the code, hence event-driven programming can make programming easier for some.
- More interactive programs are accessible with event-driven programming. Event-driven programming is used in almost all recent GUI apps.

Disadvantages :-

- Event-driven programming is usually more difficult and time-consuming for basic programs.
- Errors can be harder to notice in more complex systems than in simpler, procedural programs.
- Complex graphical user interfaces may take longer to load and operate than simpler programs, especially if RAM is insufficient.

2.4.3 Event-Driven Programming Paradigm example snippets of code

Example C# Windows Forms Code :-

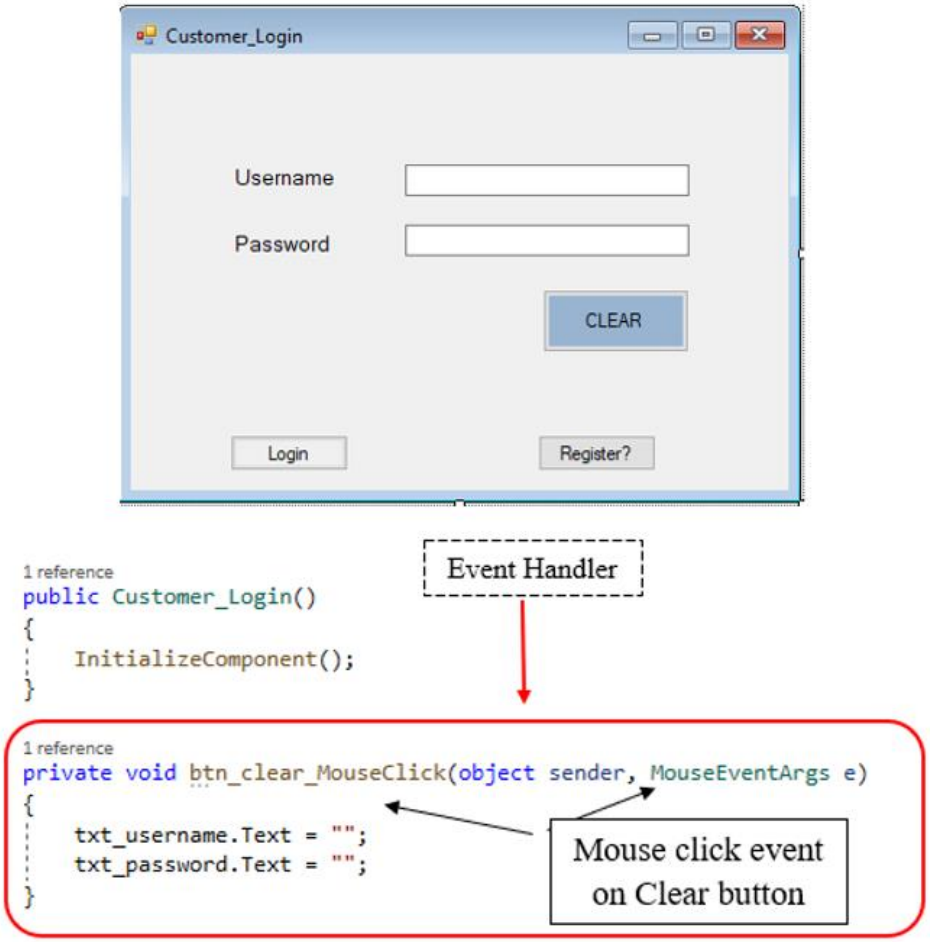


Figure 2. 12 Event-Driven Programming Paradigm example C# Windows Forms code

According to above example Mouse event must be triggered in the program for them to occur. Which means the user must interact with the targeted item in the program. For above code example “Clear Button” clicking by a mouse triggers the program.

When a specific event is triggered, event handlers are a sort of function or method that performs a specific action. For example, an event handler may be a button that displays a message when the user clicks it and closes the message when the user clicks the button again. For above example the event handler is clearing Username and Password textboxes fields when triggered.

If I further explained, when an event occurs, we usually need to pass along some information about it. This is done by using EventArgs. In above example I have created a custom EventArgs class for “Clear Button” which including clearing textboxes fields as event details. The above given event sensitive for mouse click as it refers. When I click mouse button on Clear Button the event triggers then perform specific tasks which I’ve given in the event handler.

2.5 Relationships among Procedural, Object oriented and Event-driven paradigms

Procedural Programming Paradigm	Object Oriented Programming Paradigm	Event-Driven Programming Paradigm
Helps to write commands using a character user interface.	It helps to write commands in modules.	Creates applications using a graphical user interface.
Commands are written in a linear format and are also executed in a linear format.	Objects and functions are established to interact with one another in order to complete specified tasks.	On events, actions are defined. These events can be triggered by mouse clicks and movements, as well as keyboard strokes.
It focuses on the execution of steps in a specific order.	Concentrates on objects or data and makes it easier to secure them from illegal access.	Concentrate on choosing on a user interface.
Basic, Fortran, and COBOL are the most popular languages that follow this paradigm.	Smalltalk, C++, and JAVA are the most common languages that follow this paradigm.	Visual Basic and C# are the most common languages that follow this approach.

Table 2. 1 Relationships among Procedural, Object oriented and Event-driven paradigms

2.6 Best programming paradigm to use

2.6.1 Advantages of Object-Oriented programming (OOP) and Event-Driven Programming (EDP) over pure Procedural Programming (PP)

Over pure Procedural Programming, Object-Oriented Programming has a number of advantages. The first benefit is that OOP and EDP make software development faster. Objects and classes used in OOP allow code to be reused as needed, as contrast to Procedural Programming, which needs rewriting for each usage.

The second benefit is that OOP makes debugging easier. Unlike Procedural Programming, which requires the execution of fully completed code, OOP allows the execution of individual classes. This makes error detection and troubleshooting much easy. OOP also leads to increased software development productivity.

To execute a program, EDP allows the capture of single instances of events such as mouse clicks or key presses. The user is separated from the internal procedures required to complete a task and is only responsible for triggering the program's events. Procedural Programming, on the other hand, uses only built-in instructions and cannot receive external instructions after they have been executed. Each instruction necessary to run the software must be specified by the user.

2.6.2 Conclusion

Different programming languages from various paradigms can be used to create computer programs. As previously said, each approach to the solution will have certain advantages and disadvantages. To put it another way, the programming paradigm refers to the various approaches that are utilized to discover a software solution to a problem. Each solution to the problem has its own set of benefits and drawbacks.

Programming paradigms have been constantly evolving due to the diversification of computer use and the necessity for larger and more complex applications. Object-Oriented programming and Event-Driven Programming are the most popular programming paradigms in today's world. These paradigms isolate the user from the program's internal workings, allowing them to accept various and complicated instructions as needed. These applications also allow for code reuse, which means that other programs can utilize the same codes to do identical functions. Because of these benefits, procedural programming has been pushed out in favor of OOP and EDP. Hence, Object-Oriented Programming and Event-Driven Programming are the most user-friendly paradigms to use.

Activity 3

3.1 Integrated Development Environment (IDE)

3.1.1 What is an Integrated Development Environment (IDE)

Creating software applications is a challenging task. Developers must create a variety of interconnected components, including code, user interfaces, project structures, environment configurations and more.

An integrated development environment, or IDE, is a software application that brings together all of the tools required for a software development project in one place. Users can utilize IDEs to write code, organize text groupings, and manage programming redundancies on a more basic level. IDEs, on the other hand, combine the capabilities of different programming processes into a single application.

An IDE, at a minimum level, includes an editor, compiler, and debugger, as well as code completion and general code management. Advanced capabilities like data visualization, tracing, and cross-referencing are available in some IDEs. Some IDEs specialize in a single programming language, such as Python or Java, although many others support many languages. In terms of text editing, IDEs usually include or allow the use of frameworks and element libraries to extend the capability of base-level programming.

Most IDEs have built-in debuggers that activate when the project is built. Many IDEs include visual debuggers as a feature. Users are shown which areas of the code have difficulties if any bugs or errors are found. Complex programming benefited from IDEs because they provide greater coding assistance, code completion, debugging, visual representation of code, and deep program analysis to improve the overall development experience.

3.1.2 Common characteristics of an IDE

Having a Text Editor :

Almost every IDE will have a text editor for writing and manipulating source code.

Although some programs feature visual components that allow users to drag and drop front-end components. The majority have a simple interface that highlights language-specific syntax.

Having a Debugger :

Debugging tools helps users in finding and fixing bugs in source code. To evaluate functionality and performance, they frequently recreate real-world scenarios. Before an application is deployed, programmers and software engineers may usually test the various code parts and discover issues.

Having a compiler :

Compilers are pieces of software that convert programming languages into machine-readable formats, such as binary code. To guarantee that the machine code is accurate, it is examined. After that, the compiler parses and optimizes the code to improve performance.

Having code completers :

Code completion tools help programmers by finding and adding common code components intelligently. These features lower the possibility of errors and bugs while saving developers time when developing code.

Having Plugins and Integrations :

Since an IDE serves as development portal, integrating all of your other development tools will boost development workflows and productivity. Poor integrations can generate plenty of problems and difficulties.

3.1.3 Benefits of IDE

- Most developers utilize three key tools which are source code editors, debuggers, and compilers. These all 3 key tools centralized on the IDE platform. This allows users to develop, test, and process code all in the same place.
- With IDE it's also easier to navigate the source code when these tools are all in one place. Many come with extra features like code testing, organization, and restructuring.
- In addition, Developers' capabilities and development speed are greatly enhanced by IDE features like as autocomplete, build, and deployment features.
- The capable of completing code improves the programming process.
- Checks for mistakes automatically to ensure high-quality code.
- Developers can use refactoring to create comprehensive and error-free name changes.
- Ensure that the development process runs smoothly.
- Enhance developer productivity and happiness.

3.1.4 Types of IDE

1) IDE for a specific language

There are IDEs intended exclusively for developers who work in a single language. Jikes and Jcreator for Java, CodeLite and C-Free for C/C++, and Idle for Python are just a few examples.

2) Multi-language IDE

As a new developer, we should look into learning how to use a multi-language IDE. Visual Studio, for example, is a multi-language IDE that is well-known for its extensive capabilities and ongoing support for extensions and upgrades. Introducing support for a new programming language is as simple as adding an extension.

3) IDE for mobile development

Many new tools are becoming available as the mobile app development market grows. To create successful and resourceful apps, mobile app developers need a platform specialized to this type of development. Android Studio and Xcode, for example, are mobile development IDEs for the Android and iOS platforms.

4) Web/Cloud-Based IDE

When compared to local development environments, cloud-based IDEs provide numerous advantages. A SaaS IDE can do long-running jobs without using a local workstation's computational resources. Cloud IDEs are typically platform-independent, allowing connectivity to a variety of cloud providers.

3.1.5 Best IDEs to use

The ability to customize an environment with plugins and integrations is one of the most important reasons that IDEs might be more advantageous than other tools. Plugins allow us to change workflows and add new features. Color themes and schedules are examples of simple plugins, while continuous deployment and database development extensions are examples of more complex plugins. Some of the best multifunctional IDE solutions for C, C++, PHP, Java, JavaScript, Python, and other languages are listed below.

- Visual Studio
- Pycharm
- Xcode
- Eclipse
- PhpStorm
- WebStorm
- Netbeans
- AWS Cloud 9

Figure 3. 1 Best IDEs to use

3.1.6 The use of the Visual Studio IDE

Visual studio is the best IDE for any kind of coding or development. It is one of the greatest IDEs for programming because it has an extension and updates feature that allows us to utilize java, JavaScript, Python, and other languages in a single IDE since its interface is very user-friendly, allowing users to use it quickly and get the most out of our code.

C++ and C# are the most commonly used languages. We can however do program Android, Windows Phone, Mac, and Windows desktop computers. And also, different templates are supported by Visual Studio. Overall, this Visual Studio IDE is fantastic for developers, whether they're working on embedded systems, websites, games, or graphics. If we can't locate what you're looking for, there's a comprehensive search engine that looks through all of Visual Studio's features and plugins.

3.2 What happens when developing a system without an IDE?

3.2.1 Problems usually occurs when developing a system without an IDE

Following problems usually occurs when developer doesn't use an IDE for coding

- Developer won't have a compiler that can detect syntax errors.
- Developer can't execute, change, and revise the code till it works.
- Developer is unable to perform test cases on the code.
- Developer won't having a debugger to step through the code.

3.2.2 Ways to developing a system without an IDE

Before IDE come to play a major role in programmer's life, developers used text editors and code editors back then.

A text editor is simply a text editing computer software. It is one of the most essential tools for programmers because it allows them to type and edit text, most commonly programming language files. Text editors have nothing to do with programming. In reality, they're made to function with whichever framework or language we choose. The most popular text editors that come packaged with Microsoft Windows are WordPad and Notepad. When we examine these editors, we notice that they all perform the same basic text editing functions. Simply put, they take some input, change it, and then produce some output.

One of the most important tools for programmers is a code editor, which is created specifically to modify the source code of computer programs. A code editor is essentially a text editor that also serves as a tool for writing code. It allows us to color the code and gives access to more complex tools to make coding easier. It's similar to a text editor, but with extra capabilities and features built in. A basic code editor might be a small program or a component of an IDE or a web browser.

3.2.3 Difference between IDE and Text Editor

A text editor is a computer program and tool that allows you to edit plain text. An IDE, on the other hand, is a full-fledged software environment that brings together all of the essential developer tools needed to create and test software.

In a text editor, the text comes into focus. But IDEs are bringing together several parts of a computer program into a single graphical user interface (GUI). IDEs enable a group of developers to work on different modules of the same project in a logical order.

Text editors are typically simpler than integrated development environments (IDEs), but IDEs are unquestionably more feature-rich, with features like as code intelligence, project management, debugging, compilation, and more.

3.2.4 Difference between IDE and Code Editor

Code editors are text editors that have sophisticated built-in features and specific functionalities that make code editing easier and faster. An integrated development environment (IDE) is a collection of software development tools that make coding easier. It simplifies the software development process by combining the various features of a computer program into a single GUI.

An integrated development environment (IDE) combines a text editor, a code editor, a debugger, a compiler, and other tools into a single utility belt. A code editor can be a standalone application or a component of an IDE or web browser. It only allows us to color our code and gives us more powerful tools to assist us code more efficiently.

3.3 Algorithms and design system for vehicle tariff calculation for Rents

3.3.1 Flow Chart for Vehicle tariff calculation for rents

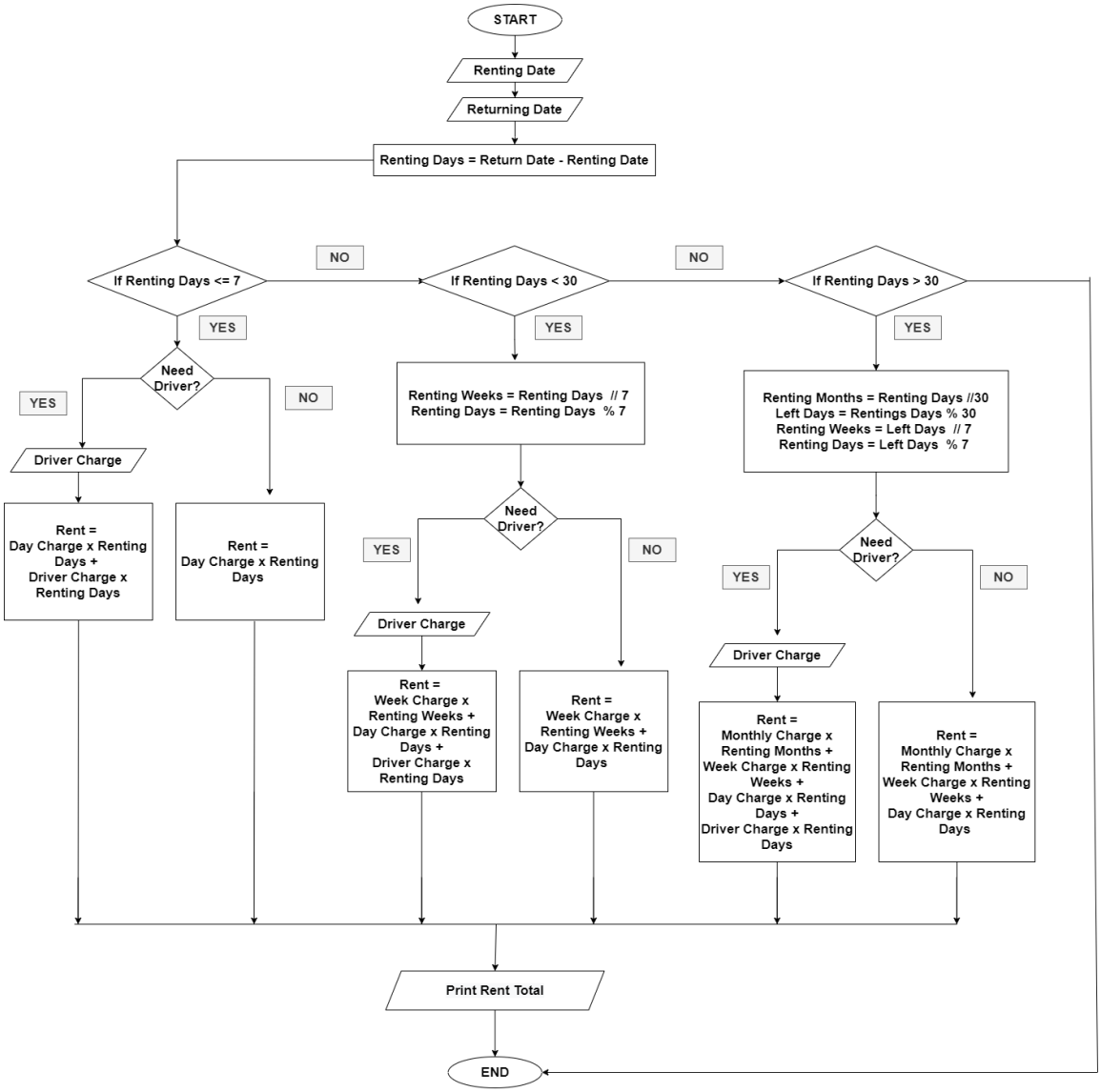


Figure 3. 2 Flow Chart for Vehicle tariff calculation for rents

3.3.2 Visual Studio IDE interface to calculate vehicle rents

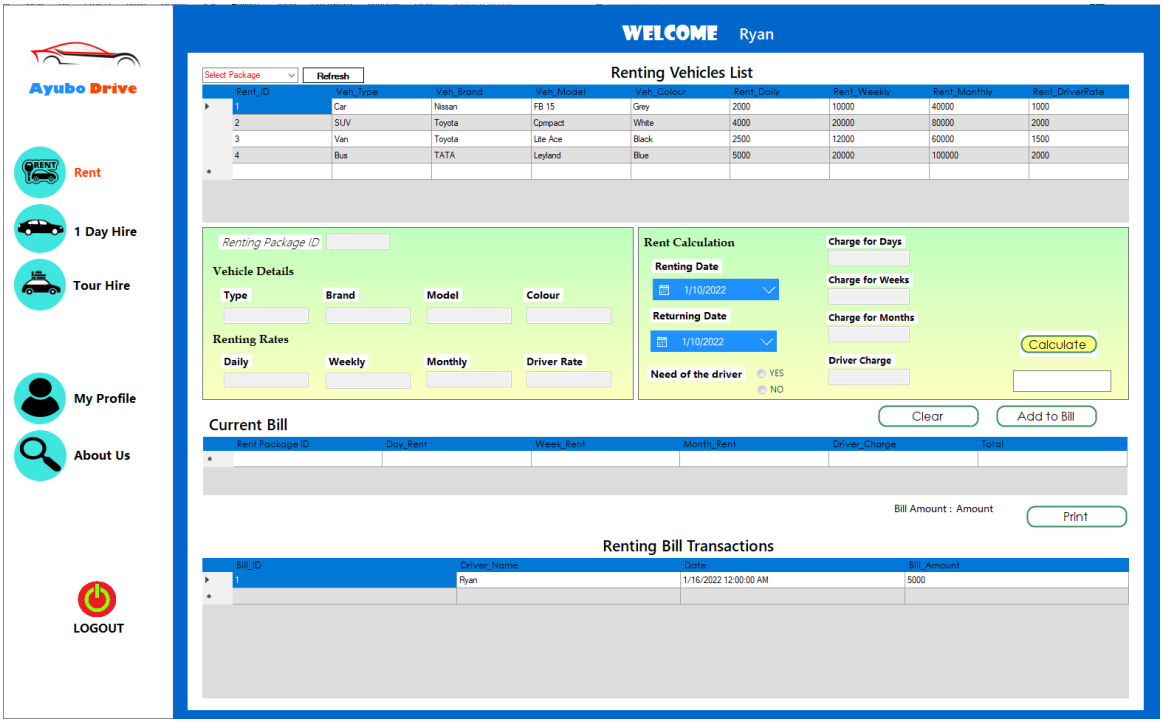
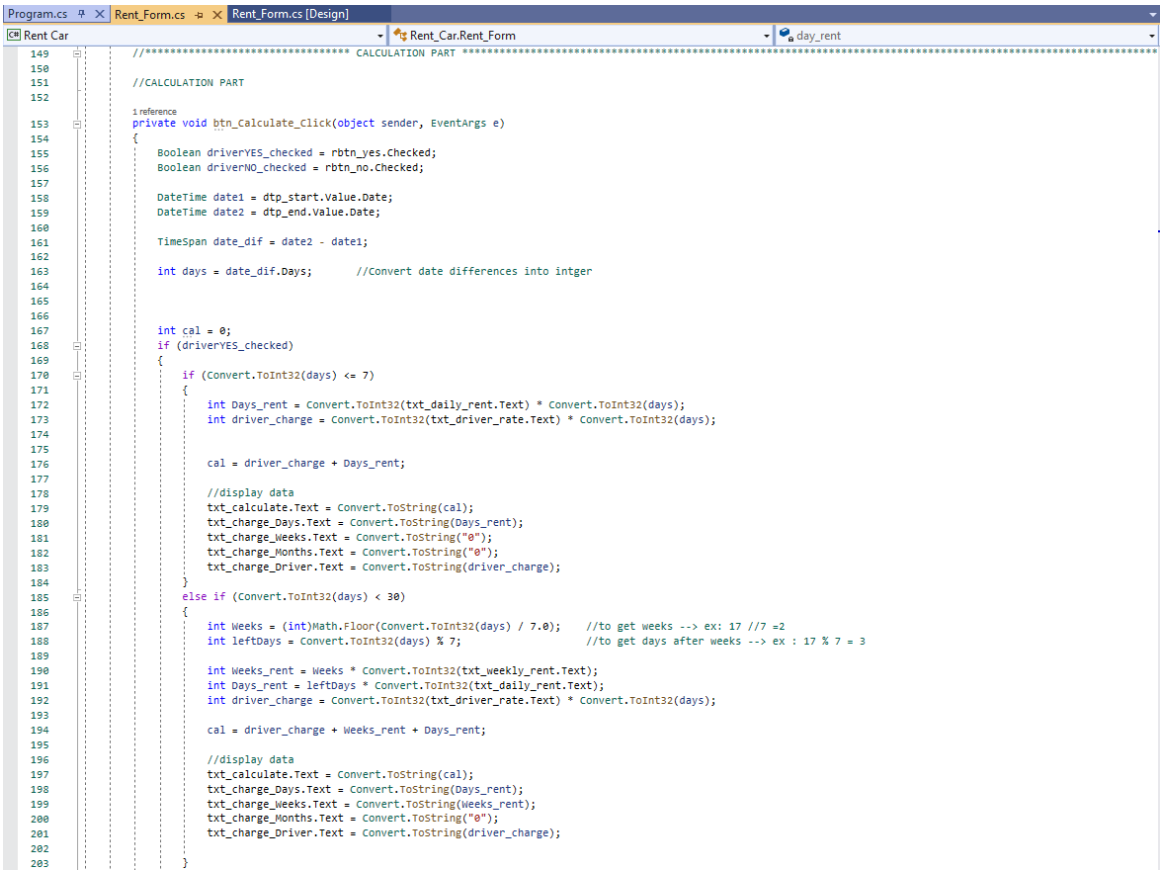


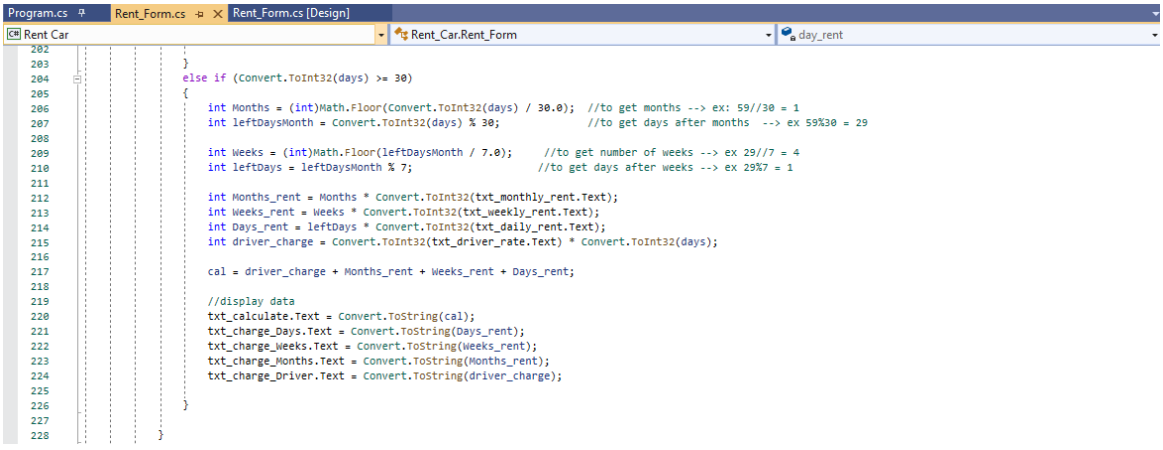
Figure 3. 3 Visual Studio IDE interface to calculate vehicle rents

3.3.3 Visual Studio IDE coding to calculate vehicle rents



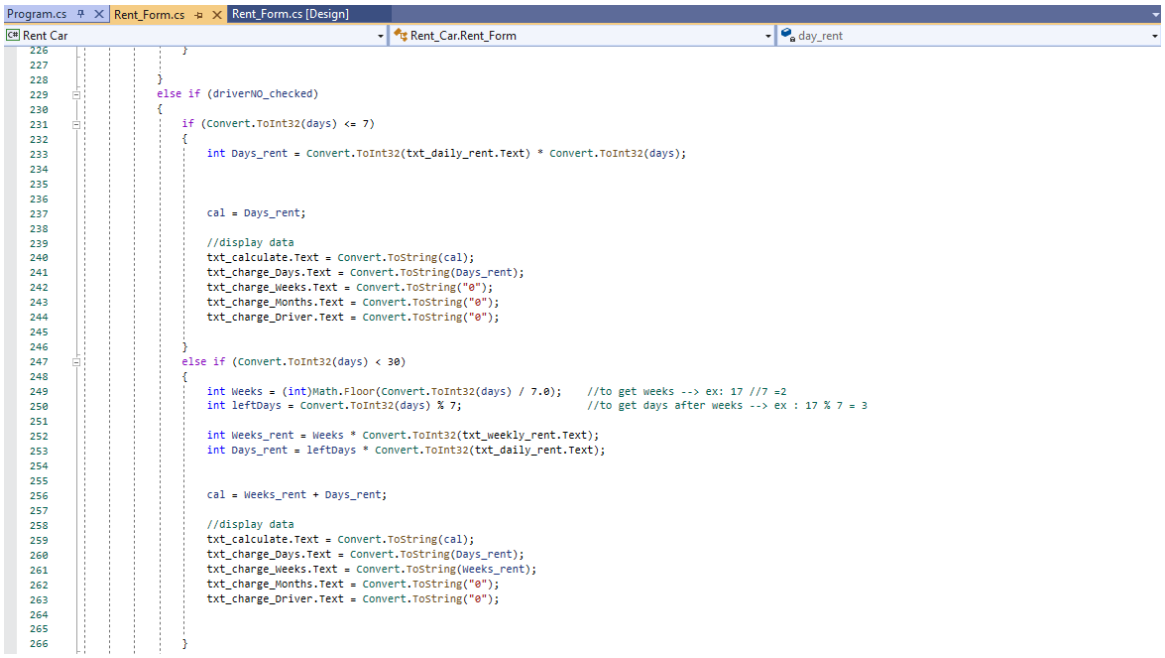
```
Program.cs | Rent_Form.cs | Rent_Form.cs [Design]
Rent Car | Rent_Car.Rent_Form | day_rent
149     //********************************* CALCULATION PART ***********************************************************************************************************
150
151     //CALCULATION PART
152
            1 reference
153     private void btn_Calculate_Click(object sender, EventArgs e)
154     {
155         Boolean driverYES_checked = rbtn_yes.Checked;
156         Boolean driverNO_checked = rbtn_no.Checked;
157
158         DateTime date1 = dtp_start.Value.Date;
159         DateTime date2 = dtp_end.Value.Date;
160
161         TimeSpan date_dif = date2 - date1;
162
163         int days = date_dif.Days;       //Convert date differences into intger
164
165
166
167         int cal = 0;
168         if (driverYES_checked)
169         {
170             if (Convert.ToInt32(days) <= 7)
171             {
172                 int Days_rent = Convert.ToInt32(txt_daily_rent.Text) * Convert.ToInt32(days);
173                 int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);
174
175
176                 cal = driver_charge + Days_rent;
177
178                 //display data
179                 txt_calculate.Text = Convert.ToString(cal);
180                 txt_charge_Days.Text = Convert.ToString(Days_rent);
181                 txt_charge_Weeks.Text = Convert.ToString("0");
182                 txt_charge_Months.Text = Convert.ToString("0");
183                 txt_charge_Driver.Text = Convert.ToString(driver_charge);
184             }
185             else if (Convert.ToInt32(days) < 30)
186             {
187                 int Weeks = (int)Math.Floor(Convert.ToInt32(days) / 7.0);    //to get weeks --> ex: 17 //7 =2
188                 int leftDays = Convert.ToInt32(days) % 7;                    //to get days after weeks --> ex : 17 % 7 = 3
189
190                 int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
191                 int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
192                 int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);
193
194                 cal = driver_charge + Weeks_rent + Days_rent;
195
196                 //display data
197                 txt_calculate.Text = Convert.ToString(cal);
198                 txt_charge_Days.Text = Convert.ToString(Days_rent);
199                 txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
200                 txt_charge_Months.Text = Convert.ToString("0");
201                 txt_charge_Driver.Text = Convert.ToString(driver_charge);
202
203             }
```

Figure 3. 4 Visual Studio IDE coding to calculate vehicle rents - part 1



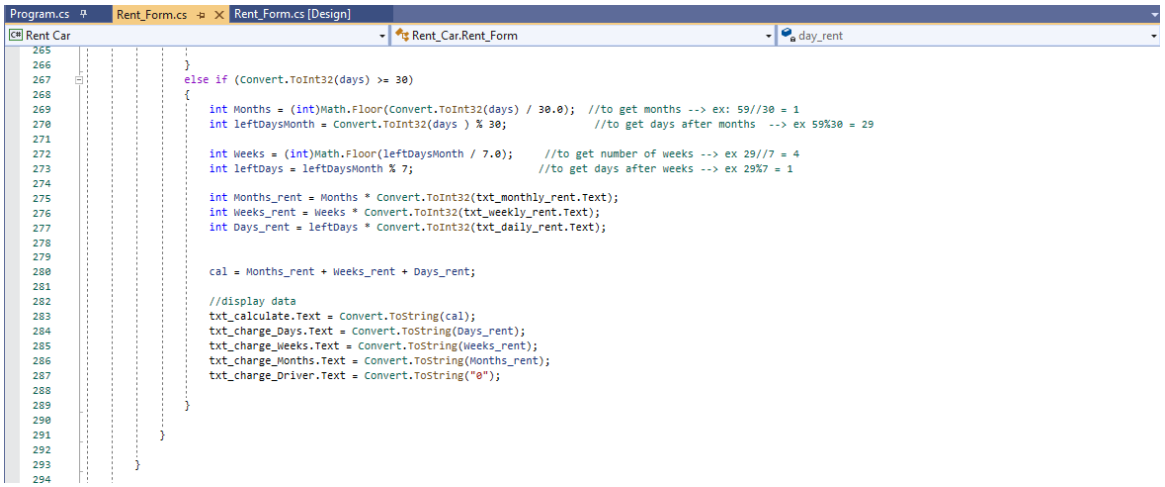
```
Program.cs | Rent_Form.cs | Rent_Form.cs [Design]
Rent Car | Rent_Car.Rent_Form | day_rent
202
203             }
204             else if (Convert.ToInt32(days) >= 30)
205             {
206                 int Months = (int)Math.Floor(Convert.ToInt32(days) / 30.0);  //to get months --> ex: 59//30 = 1
207                 int leftDaysMonth = Convert.ToInt32(days) % 30;              //to get days after months  --> ex 59%30 = 29
208
209                 int Weeks = (int)Math.Floor(leftDaysMonth / 7.0);     //to get number of weeks --> ex 29//7 = 4
210                 int leftDays = leftDaysMonth % 7;                    //to get days after weeks --> ex 29%7 = 1
211
212                 int Months_rent = Months * Convert.ToInt32(txt_monthly_rent.Text);
213                 int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
214                 int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
215                 int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);
216
217                 cal = driver_charge + Months_rent + Weeks_rent + Days_rent;
218
219                 //display data
220                 txt_calculate.Text = Convert.ToString(cal);
221                 txt_charge_Days.Text = Convert.ToString(Days_rent);
222                 txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
223                 txt_charge_Months.Text = Convert.ToString(Months_rent);
224                 txt_charge_Driver.Text = Convert.ToString(driver_charge);
225
226             }
227
228         }
```

Figure 3. 5 Visual Studio IDE coding to calculate vehicle rents - part 2




```
226                 }
227
228             }
229             else if (driverNO_checked)
230             {
231                 if (Convert.ToInt32(days) <= 7)
232                 {
233                     int Days_rent = Convert.ToInt32(txt_daily_rent.Text) * Convert.ToInt32(days);
234
235
236
237                     cal = Days_rent;
238
239                     //display data
240                     txt_calculate.Text = Convert.ToString(cal);
241                     txt_charge_Days.Text = Convert.ToString(Days_rent);
242                     txt_charge_Weeks.Text = Convert.ToString("0");
243                     txt_charge_Months.Text = Convert.ToString("0");
244                     txt_charge_Driver.Text = Convert.ToString("0");
245
246                 }
247                 else if (Convert.ToInt32(days) < 30)
248                 {
249                     int Weeks = (int)Math.Floor(Convert.ToInt32(days) / 7.0);    //to get weeks --> ex: 17 //7 =2
250                     int leftDays = Convert.ToInt32(days) % 7;                    //to get days after weeks --> ex : 17 % 7 = 3
251
252                     int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
253                     int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
254
255
256                     cal = Weeks_rent + Days_rent;
257
258                     //display data
259                     txt_calculate.Text = Convert.ToString(cal);
260                     txt_charge_Days.Text = Convert.ToString(Days_rent);
261                     txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
262                     txt_charge_Months.Text = Convert.ToString("0");
263                     txt_charge_Driver.Text = Convert.ToString("0");
264
265
266                 }
```

Figure 3. 6 Visual Studio IDE coding to calculate vehicle rents - part 3




```
265
266                 }
267                 else if (Convert.ToInt32(days) >= 30)
268                 {
269                     int Months = (int)Math.Floor(Convert.ToInt32(days) / 30.0);  //to get months --> ex: 59//30 = 1
270                     int leftDaysMonth = Convert.ToInt32(days ) % 30;              //to get days after months  --> ex 59%30 = 29
271
272                     int Weeks = (int)Math.Floor(leftDaysMonth / 7.0);     //to get number of weeks --> ex 29//7 = 4
273                     int leftDays = leftDaysMonth % 7;                    //to get days after weeks --> ex 29%7 = 1
274
275                     int Months_rent = Months * Convert.ToInt32(txt_monthly_rent.Text);
276                     int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
277                     int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
278
279
280                     cal = Months_rent + Weeks_rent + Days_rent;
281
282                     //display data
283                     txt_calculate.Text = Convert.ToString(cal);
284                     txt_charge_Days.Text = Convert.ToString(Days_rent);
285                     txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
286                     txt_charge_Months.Text = Convert.ToString(Months_rent);
287                     txt_charge_Driver.Text = Convert.ToString("0");
288
289                 }
290
291             }
292
293         }
294
```

Figure 3. 7 Visual Studio IDE coding to calculate vehicle rents - part 4

3.3.4 Create SQL table to store Vehicle Rent details

	Column Name	Data Type	Allow Null
🔑	Rent_ID	int	☐
	Veh_Type	varchar(50)	☐
	Veh_Brand	varchar(50)	☐
	Veh_Model	varchar(50)	☐
	Veh_Colour	varchar(50)	☐
	Rent_Daily	int	☐
	Rent_Weekly	int	☐
	Rent_Monthly	int	☐
	Rent_DriverRate	int	☐
			☐

Figure 3. 8 "Details_Renting_Table" Table to store Vehicle Rent details

	Rent_ID	Veh_Type	Veh_Brand	Veh_Model	Veh_Colour	Rent_Daily	Rent_Weekly	Rent_Monthly	Rent_DriverRate
1	1	Car	Nissan	FB 15	Grey	2000	10000	40000	1000
2	2	SUV	Toyota	Cpmpact	White	4000	20000	80000	2000
3	3	Van	Toyota	Lite Ace	Black	2500	12000	60000	1500
4	4	Bus	TATA	Leyland	Blue	5000	20000	100000	2000

Figure 3. 9 Stored table values in "Details_Renting_Table"

3.3.5 Create SQL table to bill Vehicle Rent details

	Column Name	Data Type	Allow Nulls
🔑	Bill_ID	int	☐
	Driver_Name	varchar(50)	☐
	Date	varchar(50)	☐
	Bill_Amount	int	☐
			☐

Figure 3. 10 "Bill_Rent_Table" Table to bill Vehicle Rent details

	Bill_ID	Driver_Name	Date	Bill_Amount
1	1	Ryan	1/16/2022 12:00:00 AM	5000
2	2	Ryan	2/3/2022 12:00:00 AM	146000
3	3	Ryan	2/3/2022 12:00:00 AM	354000

Figure 3. 11 Stored table values in "Bill_Rent_Table"

3.3.6 Test Cases for vehicle Rent

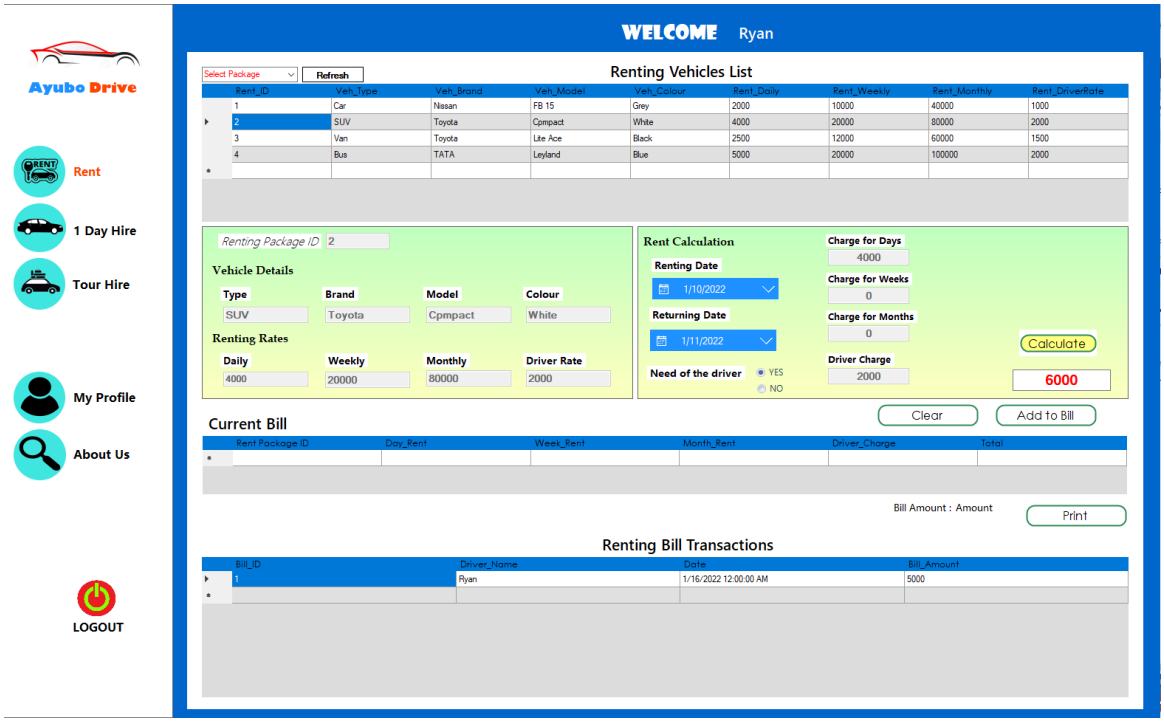


Figure 3. 12 Rent vehicle for 1 day with driver

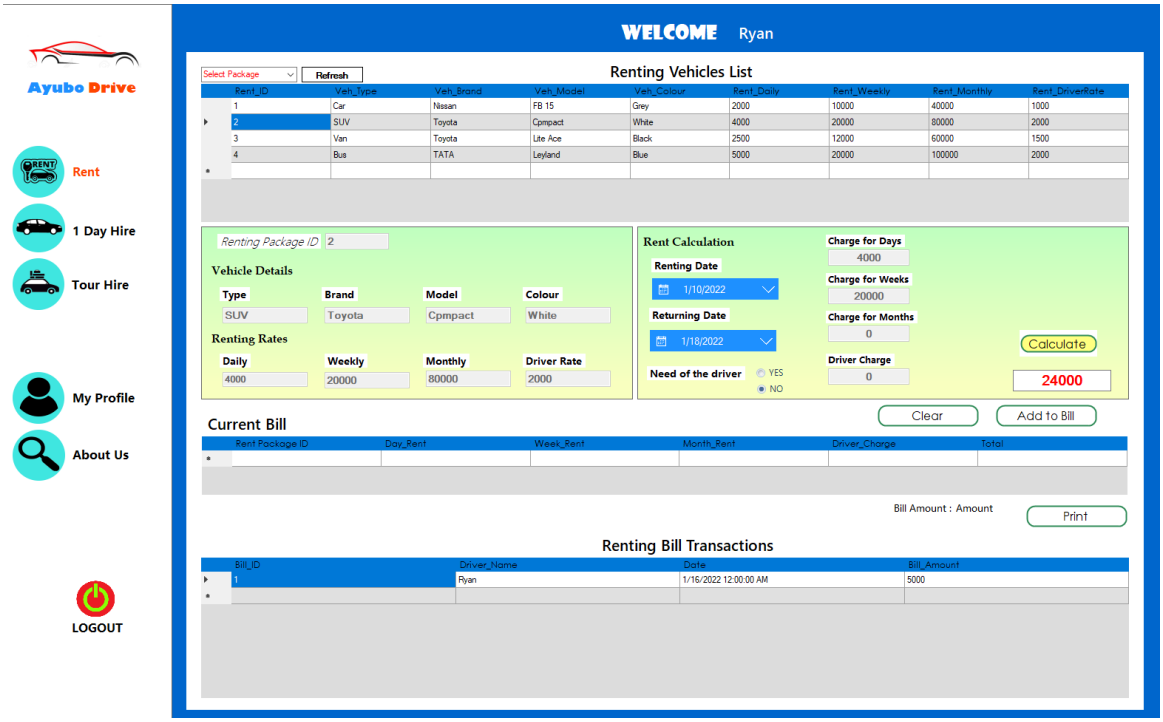


Figure 3. 13 Rent vehicle for 8 days without driver

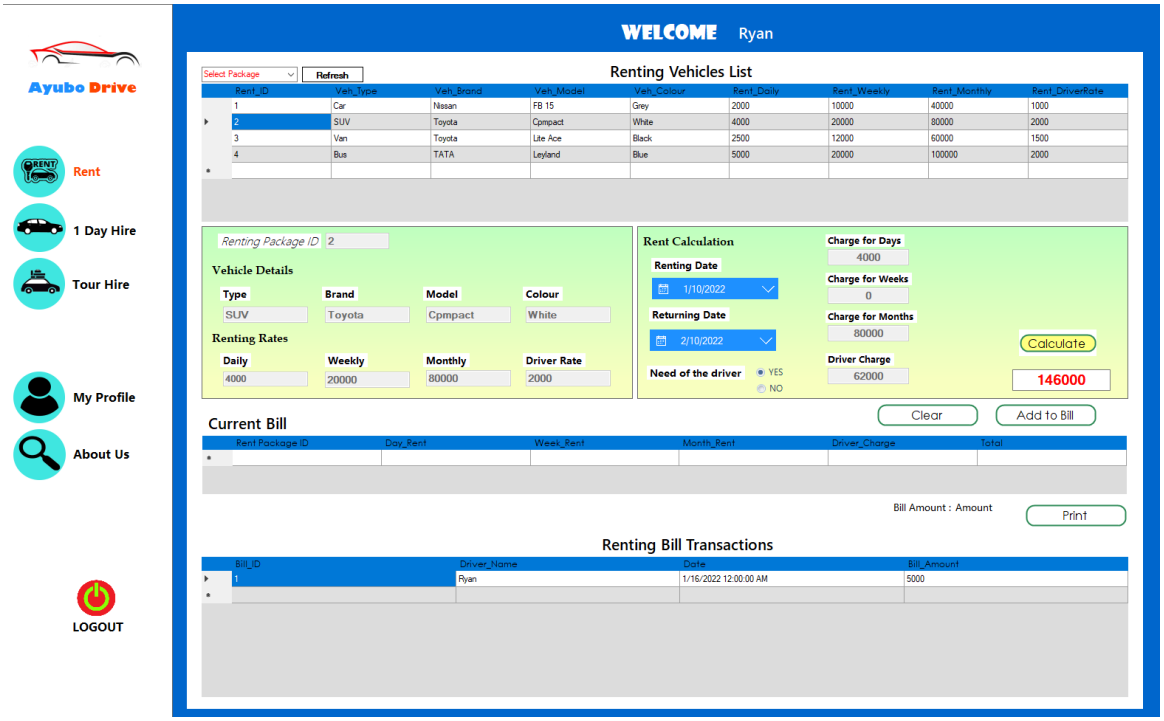


Figure 3. 14 Rent vehicle for 31 days with driver

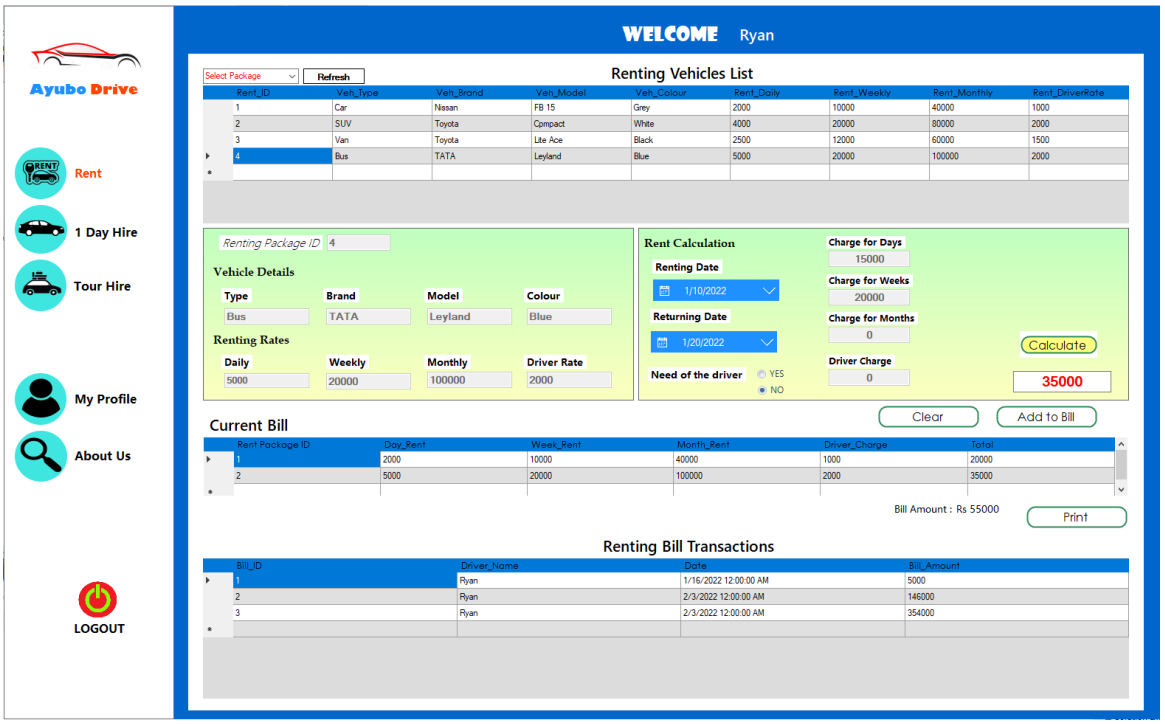


Figure 3. 15 Vehicle rent values adding to temporary bill table

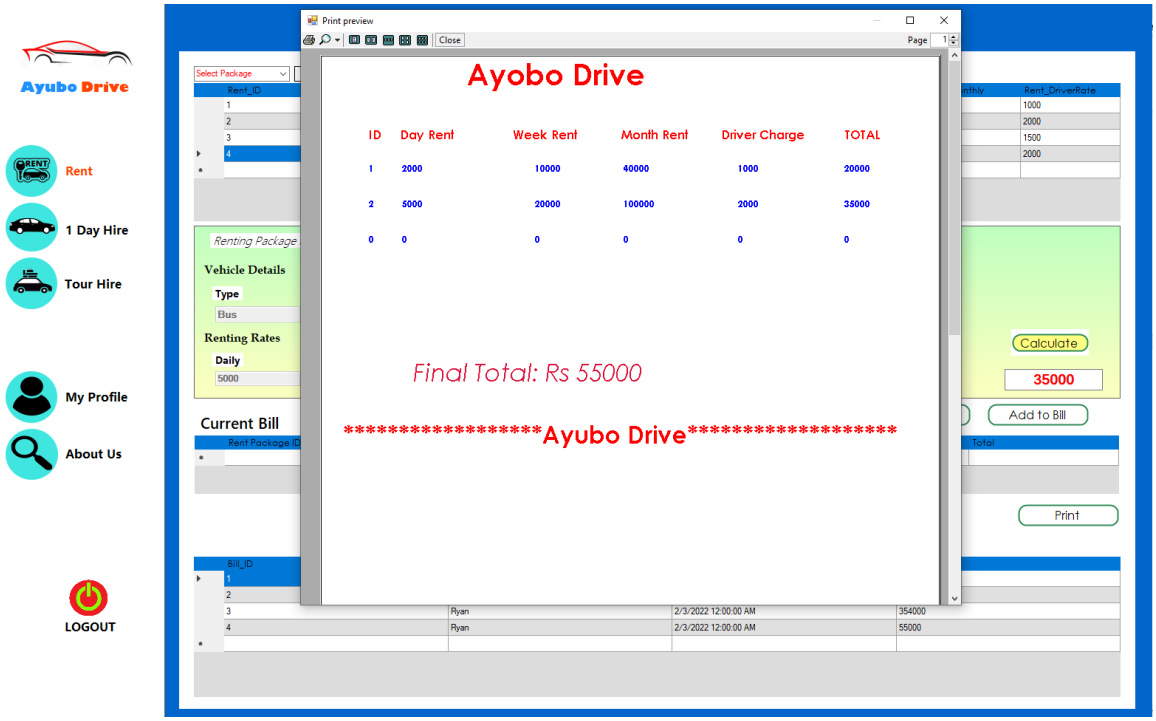


Figure 3. 16 Vehicle rent values adding to rent bill table and printing the bill

3.4 Algorithms and design system for vehicle tariff calculation for One Day Hires

3.4.1 Flow Chart for Vehicle tariff calculation for One Day Hires

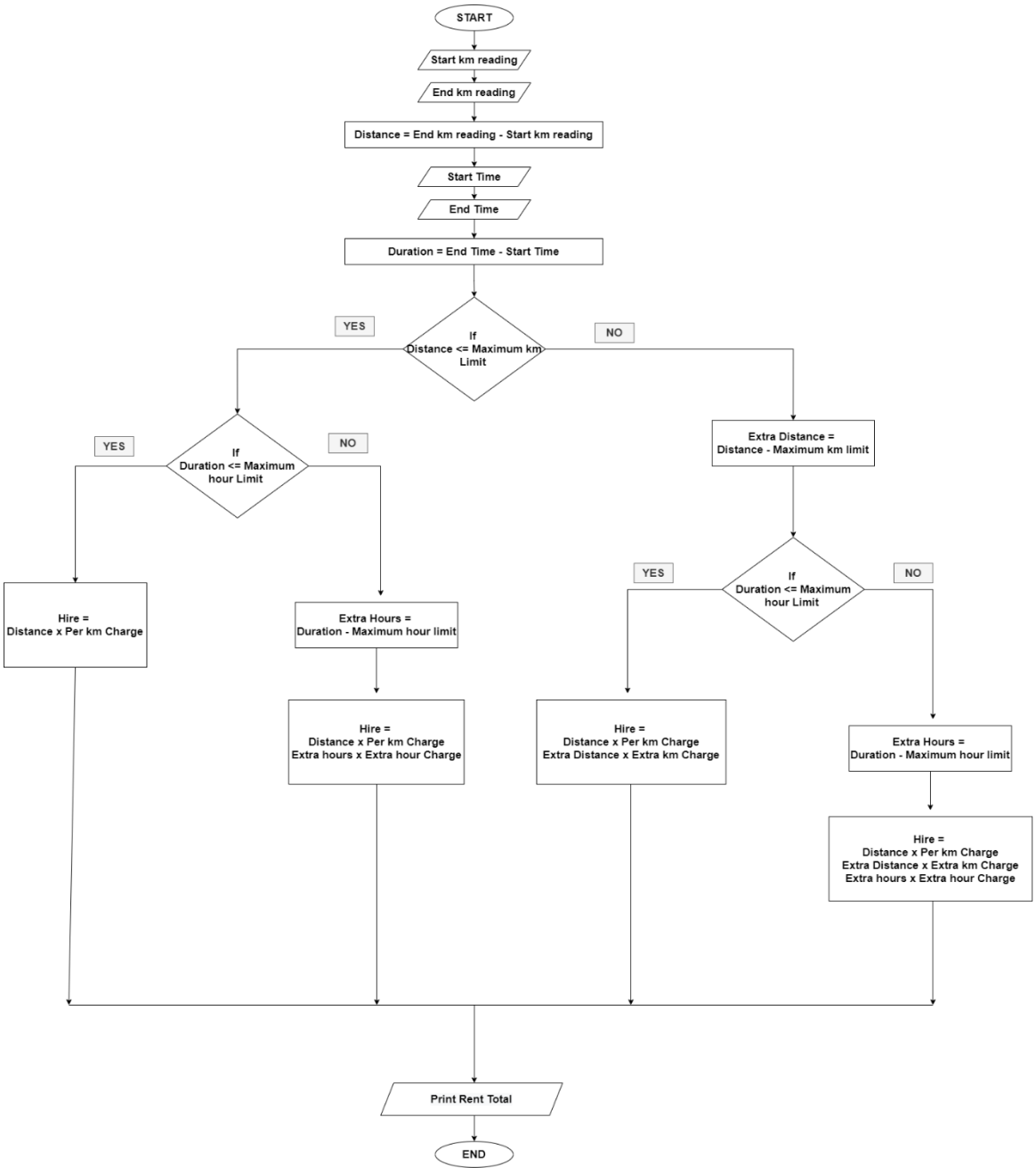


Figure 3. 17 Flow Chart for Vehicle tariff calculation for One Day Hires

3.4.2 Visual Studio IDE interface to calculate vehicle One Day Hires

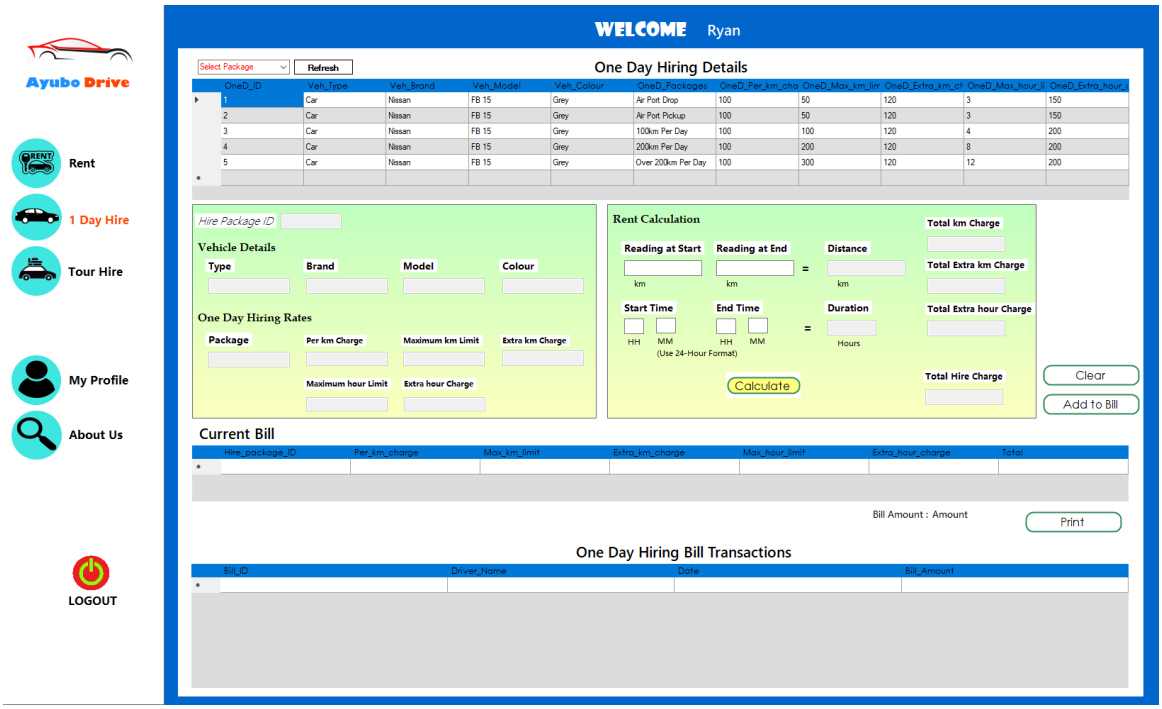
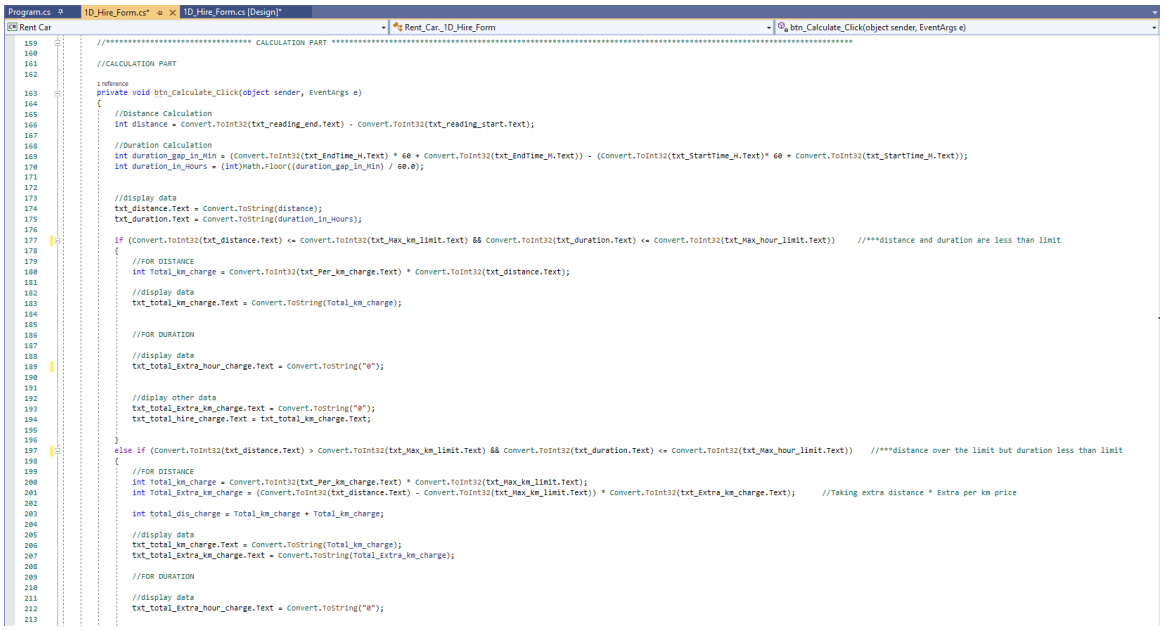


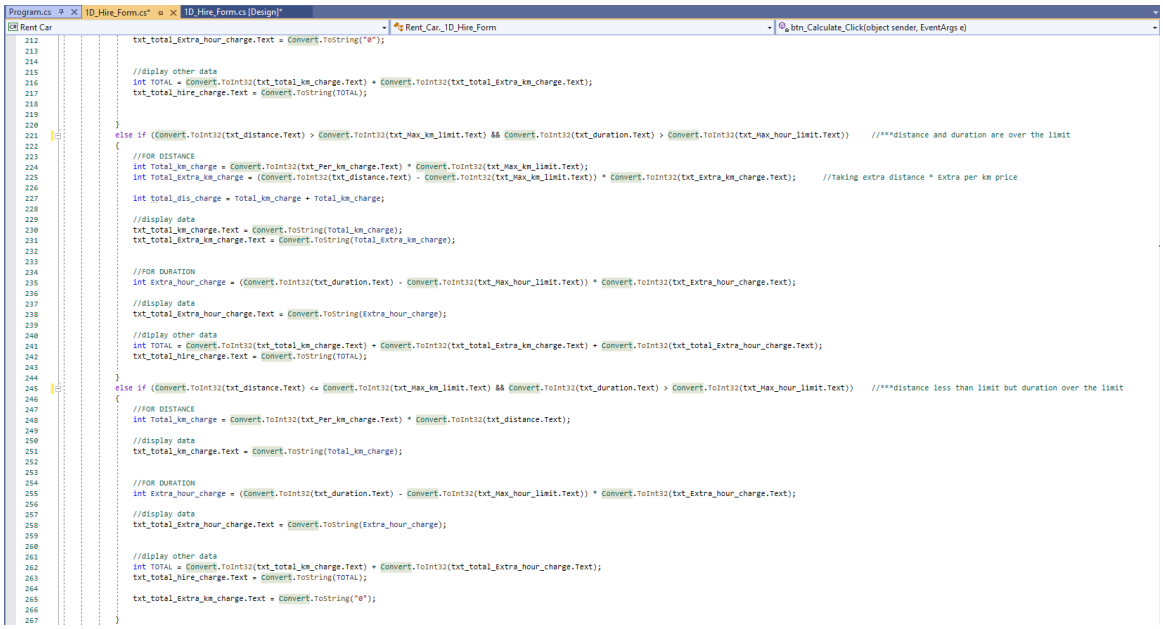
Figure 3. 18 Visual Studio IDE interface to calculate vehicle One Day Hires

3.4.3 Visual Studio IDE coding to calculate vehicle One Day Hires



```
Program.cs | 1D_Hire_Form.cs* | 1D_Hire_Form.cs [Design]*
Rent Car | Rent_Car_1D_Hire_Form | btn_Calculate_Click(object sender, EventArgs e)

159     //********************************** CALCULATION PART ***********************************************************************************************************
160
161     //CALCULATION PART
162
163     1 reference
        private void btn_Calculate_Click(object sender, EventArgs e)
164     {
165         //Distance Calculation
166         int distance = Convert.ToInt32(txt_reading_end.Text) - Convert.ToInt32(txt_reading_start.Text);
167
168         //Duration Calculation
169         int duration_gap_in_Min = (Convert.ToInt32(txt_EndTime_H.Text) * 60 + Convert.ToInt32(txt_EndTime_M.Text)) - (Convert.ToInt32(txt_StartTime_H.Text)* 60 + Convert.ToInt32(txt_StartTime_M.Text));
170         int duration_in_hours = (int)Math.Floor((duration_gap_in_Min) / 60.0);
171
172
173         //display data
174         txt_distance.Text = Convert.ToString(distance);
175         txt_duration.Text = Convert.ToString(duration_in_hours);
176
177         if (Convert.ToInt32(txt_distance.Text) <= Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) <= Convert.ToInt32(txt_Max_hour_limit.Text))     //***distance and duration are less than limit
178         {
179             //FOR DISTANCE
180             int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_distance.Text);
181
182             //display data
183             txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
184
185
186             //FOR DURATION
187
188             //display data
189             txt_total_extra_hour_charge.Text = Convert.ToString("0");
190
191
192             //diplay other data
193             txt_total_extra_km_charge.Text = Convert.ToString("0");
194             txt_total_hire_charge.Text = txt_total_km_charge.Text;
195
196         }
197         else if (Convert.ToInt32(txt_distance.Text) > Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) <= Convert.ToInt32(txt_Max_hour_limit.Text))    //***distance over the limit but duration less than limit
198         {
199             //FOR DISTANCE
200             int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_Max_km_limit.Text);
201             int Total_Extra_km_charge = (Convert.ToInt32(txt_distance.Text) - Convert.ToInt32(txt_Max_km_limit.Text)) * Convert.ToInt32(txt_Extra_km_charge.Text);      //Taking extra distance * Extra per km price
202
203             int total_dis_charge = Total_km_charge + Total_km_charge;
204
205             //display data
206             txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
207             txt_total_extra_km_charge.Text = Convert.ToString(Total_Extra_km_charge);
208
209             //FOR DURATION
210
211             //display data
212             txt_total_extra_hour_charge.Text = Convert.ToString("0");
213
```

Figure 3. 19 Visual Studio IDE coding to calculate vehicle One Day Hires - part 1



```
Program.cs | 1D_Hire_Form.cs* | 1D_Hire_Form.cs [Design]*
Rent Car | Rent_Car_1D_Hire_Form | btn_Calculate_Click(object sender, EventArgs e)

212             txt_total_extra_hour_charge.Text = Convert.ToString("0");
213
214
215             //diplay other data
216             int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text);
217             txt_total_hire_charge.Text = Convert.ToString(TOTAL);
218
219
220         }
221         else if (Convert.ToInt32(txt_distance.Text) > Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) > Convert.ToInt32(txt_Max_hour_limit.Text))     //***distance and duration are over the limit
222         {
223             //FOR DISTANCE
224             int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_Max_km_limit.Text);
225             int Total_Extra_km_charge = (Convert.ToInt32(txt_distance.Text) - Convert.ToInt32(txt_Max_km_limit.Text)) * Convert.ToInt32(txt_Extra_km_charge.Text);      //Taking extra distance * Extra per km price
226
227             int total_dis_charge = Total_km_charge + Total_km_charge;
228
229             //display data
230             txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
231             txt_total_extra_km_charge.Text = Convert.ToString(Total_Extra_km_charge);
232
233
234             //FOR DURATION
235             int Extra_hour_charge = (Convert.ToInt32(txt_duration.Text) - Convert.ToInt32(txt_Max_hour_limit.Text)) * Convert.ToInt32(txt_Extra_hour_charge.Text);
236
237             //display data
238             txt_total_extra_hour_charge.Text = Convert.ToString(Extra_hour_charge);
239
240             //diplay other data
241             int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text) + Convert.ToInt32(txt_total_Extra_hour_charge.Text);
242             txt_total_hire_charge.Text = Convert.ToString(TOTAL);
243
244         }
245         else if (Convert.ToInt32(txt_distance.Text) <= Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) > Convert.ToInt32(txt_Max_hour_limit.Text))    //***distance less than limit but duration over the limit
246         {
247             //FOR DISTANCE
248             int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_distance.Text);
249
250             //display data
251             txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
252
253
254             //FOR DURATION
255             int Extra_hour_charge = (Convert.ToInt32(txt_duration.Text) - Convert.ToInt32(txt_Max_hour_limit.Text)) * Convert.ToInt32(txt_Extra_hour_charge.Text);
256
257             //display data
258             txt_total_extra_hour_charge.Text = Convert.ToString(Extra_hour_charge);
259
260
261             //diplay other data
262             int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_hour_charge.Text);
263             txt_total_hire_charge.Text = Convert.ToString(TOTAL);
264
265             txt_total_extra_km_charge.Text = Convert.ToString("0");
266
267         }
```

Figure 3. 20 Visual Studio IDE coding to calculate vehicle One Day Hires - part 2

3.4.4 Create SQL table to store Vehicle One Day Hires details

	Column Name	Data Type	Allow Nulls
▶	OneD_ID	int	☐
	Veh_Type	varchar(50)	☐
	Veh_Brand	varchar(50)	☐
	Veh_Model	varchar(50)	☐
	Veh_Colour	varchar(50)	☐
	OneD_Packages	varchar(50)	☐
	OneD_Per_km_charge	int	☐
	OneD_Max_km_limit	int	☐
	OneD_Extra_km_charge	int	☐
	OneD_Max_hour_limit	int	☐
	OneD_Extra_hour_charge	int	☐
			☐

Figure 3. 21 "Details_OneDay_Hire_Table" Table to store Vehicle One Day Hires details

	OneD_ID	Veh_Type	Veh_Brand	Veh_Model	Veh_Colour	OneD_Packages	OneD_Per_km_charge	OneD_Max_km_limit	OneD_Extra_km_charge	OneD_Max_hour_limit	OneD_Extra_hour_charge
1	1	Car	Nissan	FB 15	Grey	Air Port Drop	100	50	120	3	150
2	2	Car	Nissan	FB 15	Grey	Air Port Pickup	100	50	120	3	150
3	3	Car	Nissan	FB 15	Grey	100km Per Day	100	100	120	4	200
4	4	Car	Nissan	FB 15	Grey	200km Per Day	100	200	120	8	200
5	5	Car	Nissan	FB 15	Grey	Over 200km Per Day	100	300	120	12	200

Figure 3. 22 Stored table values in "Details_OneDay_Hire_Table"

3.4.5 Create SQL table to bill Vehicle One Day Hires details

	Column Name	Data Type	Allow Nulls
🔑	Bill_ID	int	☐
	Driver_Name	varchar(50)	☐
	Date	varchar(50)	☐
	Bill_Amount	int	☐
			☐

Figure 3. 23 "Bill_OneDay_Hire_Table" Table to bill Vehicle One Day Hires details

	Bill_ID	Driver_Name	Date	Bill_Amount
1	1	Michelle	1/16/2022 12:00:00 AM	5000
2	2	Ryan	2/3/2022 12:00:00 AM	57290
3	3	Ryan	2/3/2022 12:00:00 AM	96640
4	4	Ryan	2/3/2022 12:00:00 AM	249240

Figure 3. 24 Stored table values in "Bill_OneDay_Hire_Table"

3.4.6 Test Cases for vehicle One Day Hires

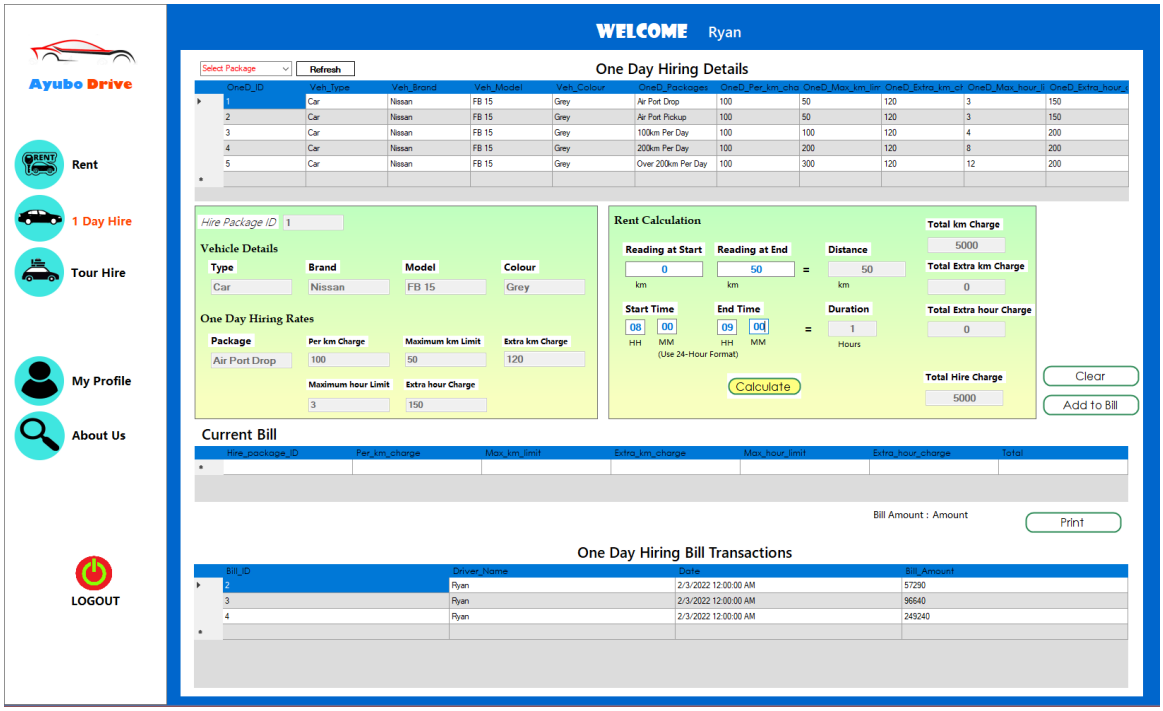


Figure 3. 25 One Day Hire vehicle for less than time and distance limit

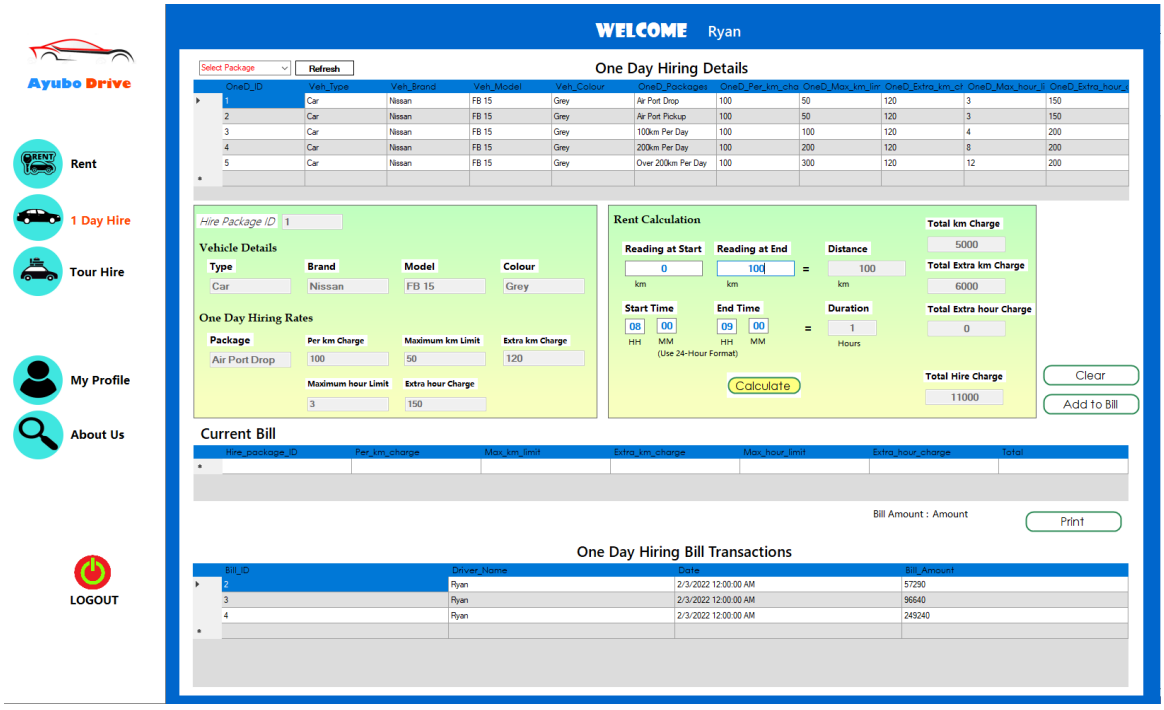


Figure 3. 26 One Day Hire vehicle for less than time limit but over the distance limit

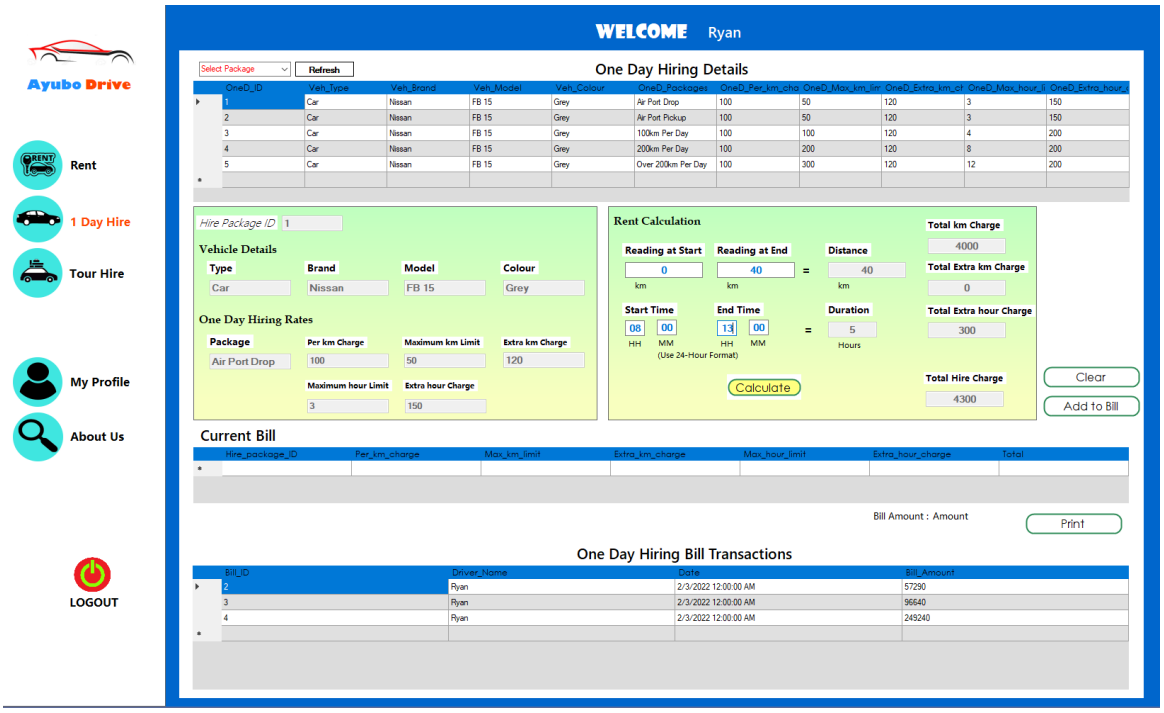


Figure 3. 27 One Day Hire vehicle for over than time limit but less the distance limit

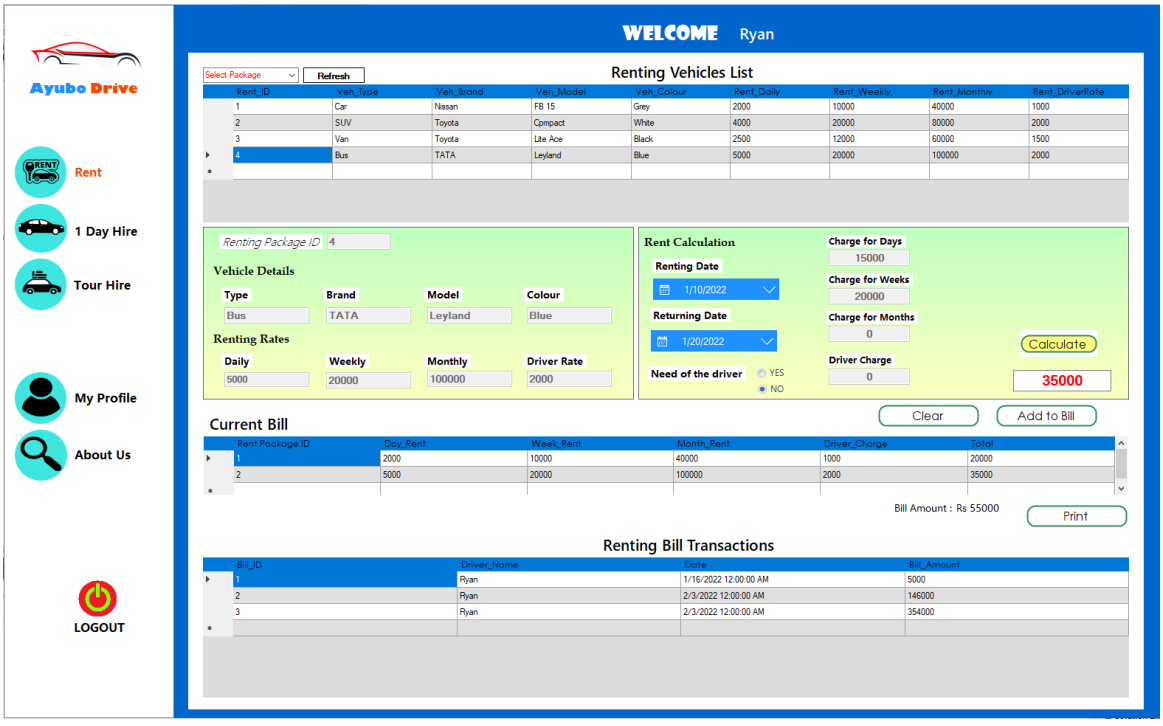


Figure 3. 28 One Day Hire vehicle for over than time and distance limit

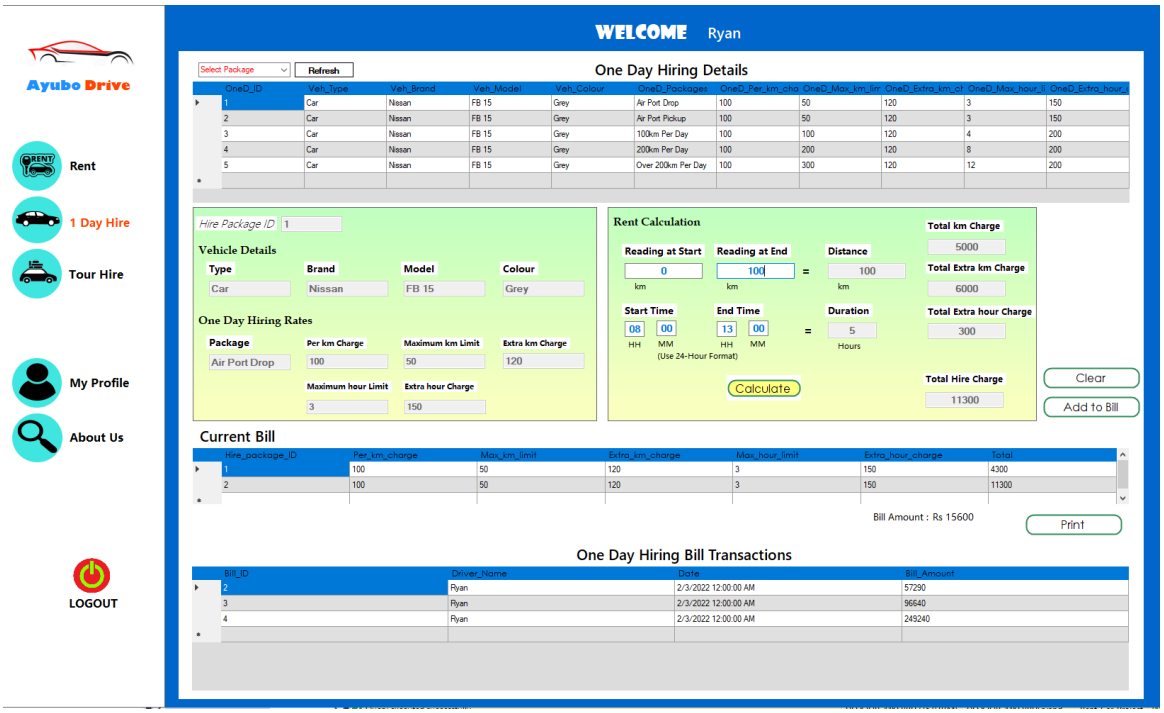


Figure 3. 29 Vehicle One Day Hire values adding to temporary bill table

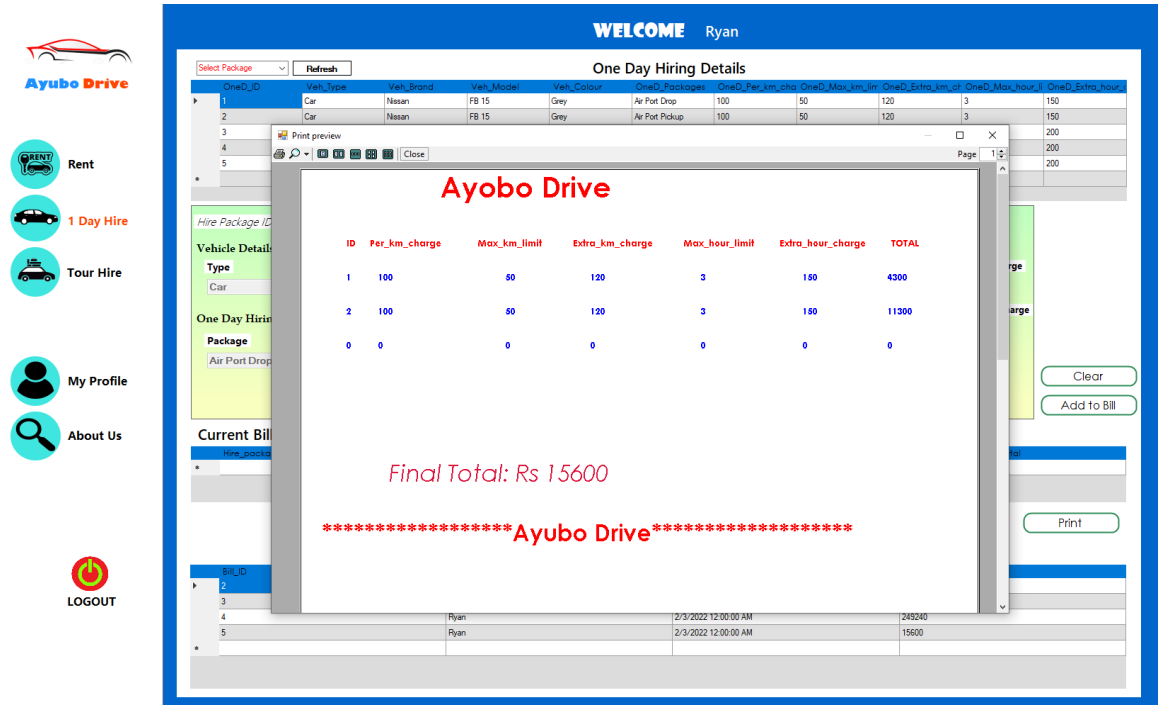


Figure 3. 30 Vehicle One Day Hire values adding to One Day Hire bill table and printing the bill

3.5 Algorithms and design system for vehicle tariff calculation for Long Tour Hires

3.5.1 Flow Chart for Vehicle tariff calculation for Long Tour Hires

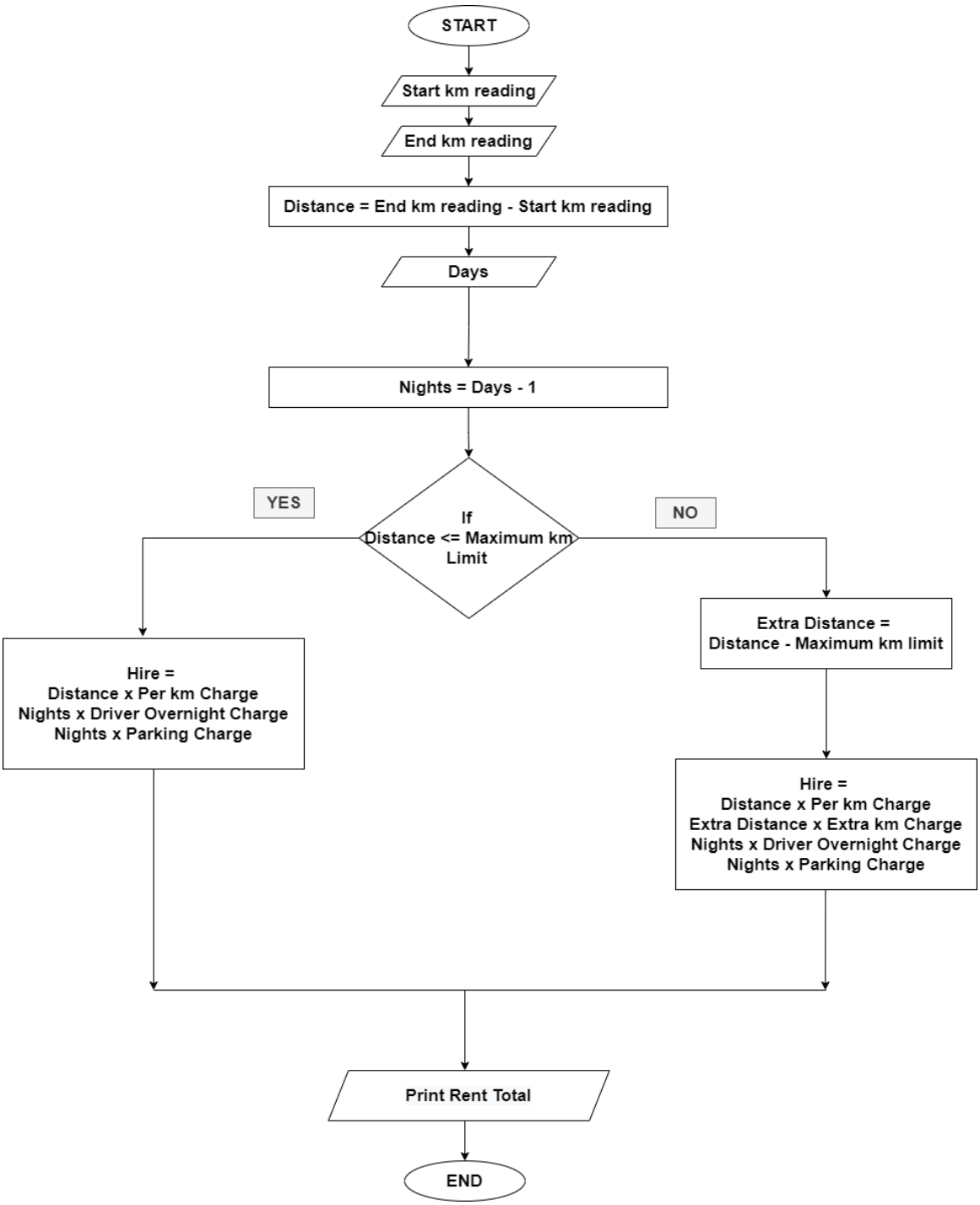


Figure 3. 31 Flow Chart for Vehicle tariff calculation for Long Tour Hires

3.5.2 Visual Studio IDE interface to calculate vehicle Long Tour Hires

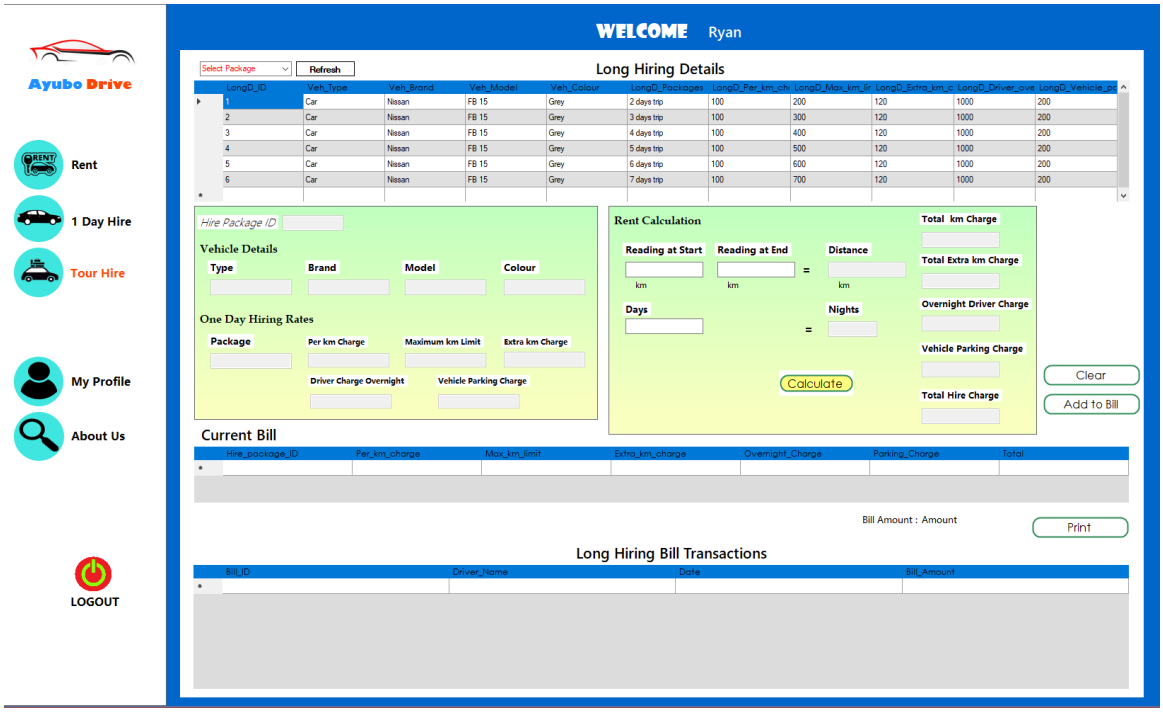


Figure 3. 32 Visual Studio IDE interface to calculate vehicle Long Tour Hires

3.5.3 Visual Studio IDE coding to calculate vehicle Long Tour Hires

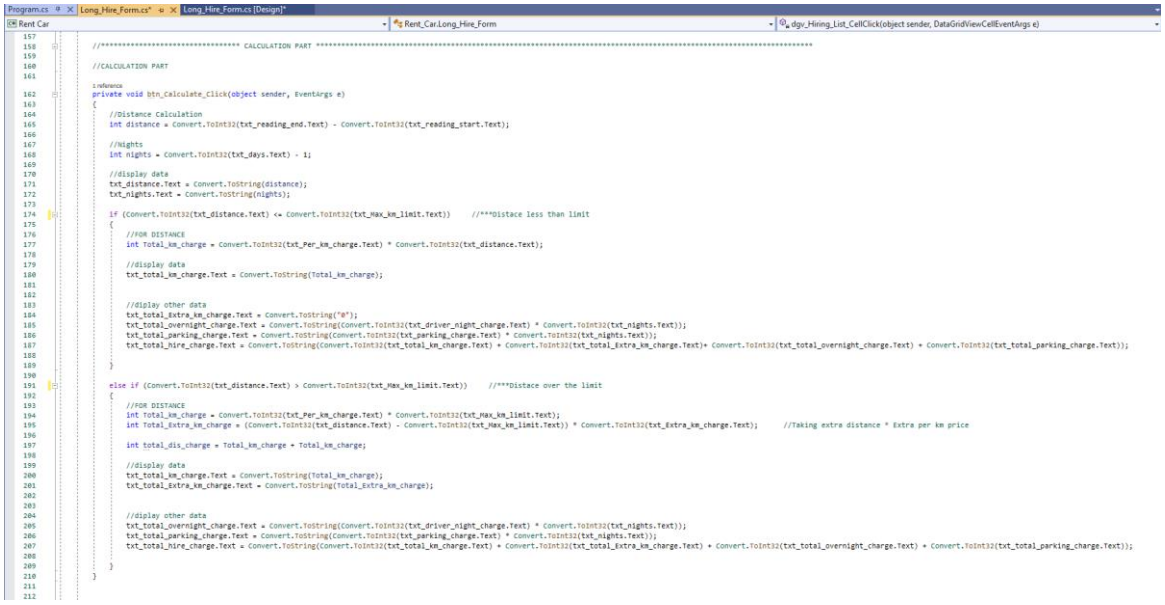


Figure 3. 33 Visual Studio IDE coding to calculate vehicle Long Tour Hires - part 1

3.5.4 Create SQL table to store Vehicle Long Tour Hires details

Column Name	Data Type	Allow Nulls
LongD_ID	int	☐
Veh_Type	varchar(50)	☐
Veh_Brand	varchar(50)	☐
Veh_Model	varchar(50)	☐
Veh_Colour	varchar(50)	☐
LongD_Packages	varchar(50)	☐
LongD_Per_km_charge	int	☐
LongD_Max_km_limit	int	☐
LongD_Extra_km_charge	int	☐
LongD_Driver_overnight_...	int	☐
LongD_Vehicle_parking_...	int	☐
		☐

Figure 3. 34 "Details_LongTour_Hire_Table" Table to store Vehicle Long Tour Hires details

	LongD_ID	Veh_Type	Veh_Brand	Veh_Model	Veh_Colour	LongD_Packages	LongD_Per_km_charge	LongD_Max_km_limit	LongD_Extra_km_charge	LongD_Driver_overnight_charge	LongD_Vehicle_parking_charge
1	1	Car	Nissan	FB 15	Grey	2 days trip	100	200	120	1000	200
2	2	Car	Nissan	FB 15	Grey	3 days trip	100	300	120	1000	200
3	3	Car	Nissan	FB 15	Grey	4 days trip	100	400	120	1000	200
4	4	Car	Nissan	FB 15	Grey	5 days trip	100	500	120	1000	200
5	5	Car	Nissan	FB 15	Grey	6 days trip	100	600	120	1000	200
6	6	Car	Nissan	FB 15	Grey	7 days trip	100	700	120	1000	200

Figure 3. 35 Stored table values in "Details_LongTour_Hire_Table"

3.5.5 Create SQL table to bill Vehicle Long Tour Hires details

	Column Name	Data Type	Allow Nulls
🔑	Bill_ID	int	☐
	Driver_Name	varchar(50)	☐
	Date	varchar(50)	☐
	Bill_Amount	int	☐
			☐

Figure 3. 36 "Bill_LongTour_Hire_Table" Table to bill Vehicle Long Tour Hires details

	Bill_ID	Driver_Name	Date	Bill_Amount
1	1	Michelle	1/16/2022 12:00:00 AM	5000
2	2	Ryan	2/3/2022 12:00:00 AM	57290
3	3	Ryan	2/3/2022 12:00:00 AM	96640
4	4	Ryan	2/3/2022 12:00:00 AM	249240

Figure 3. 37 Stored table values in "Bill_OneDay_Hire_Table"

3.5.6 Test Cases for vehicle Rent

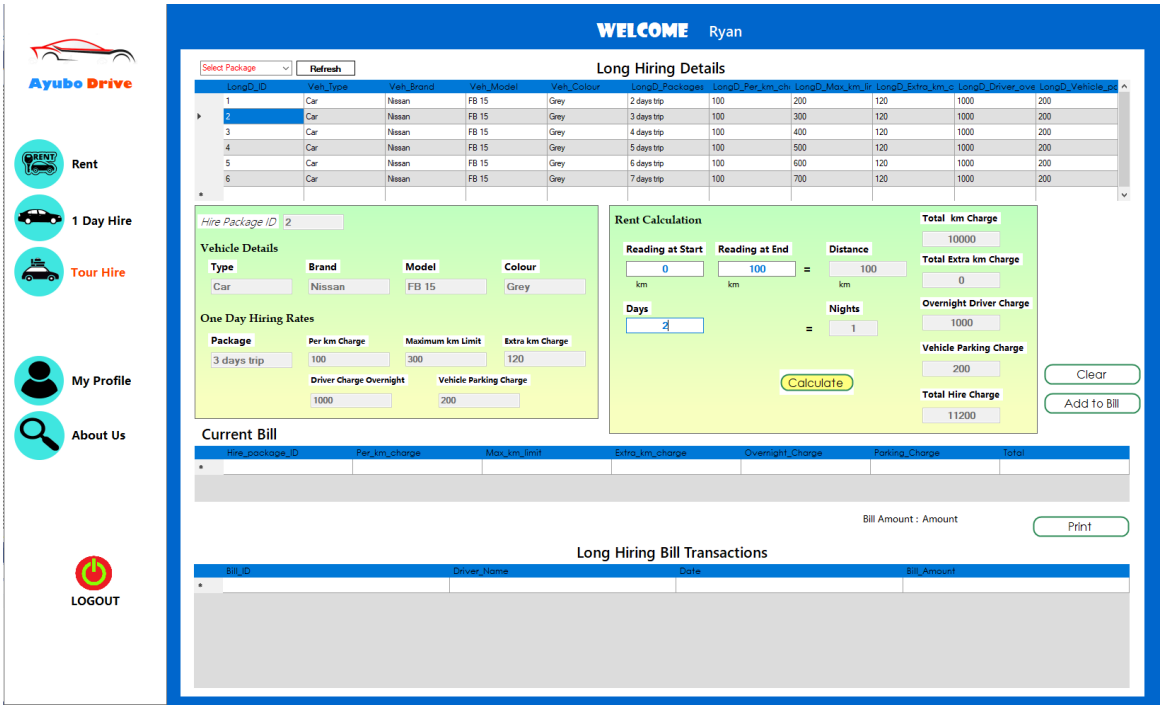


Figure 3. 38 Long Tour Hire vehicle for less than distance limit and for 1 night

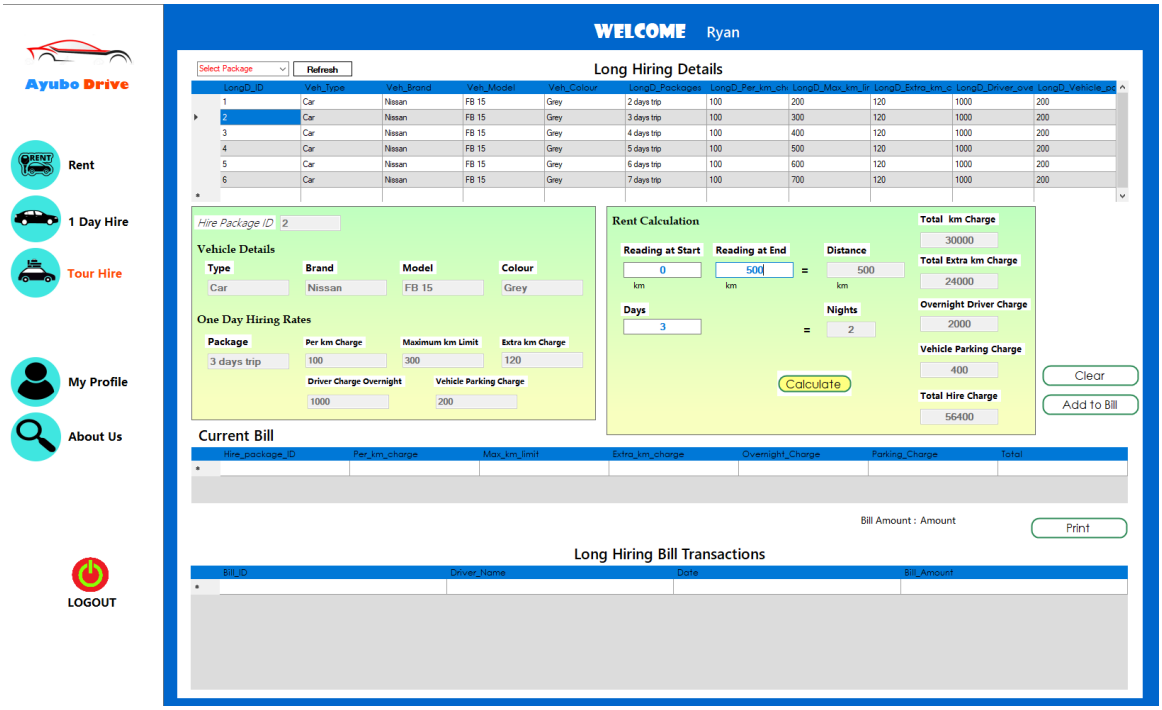


Figure 3. 39 Long Tour Hire vehicle for over the distance limit and for 2 nights

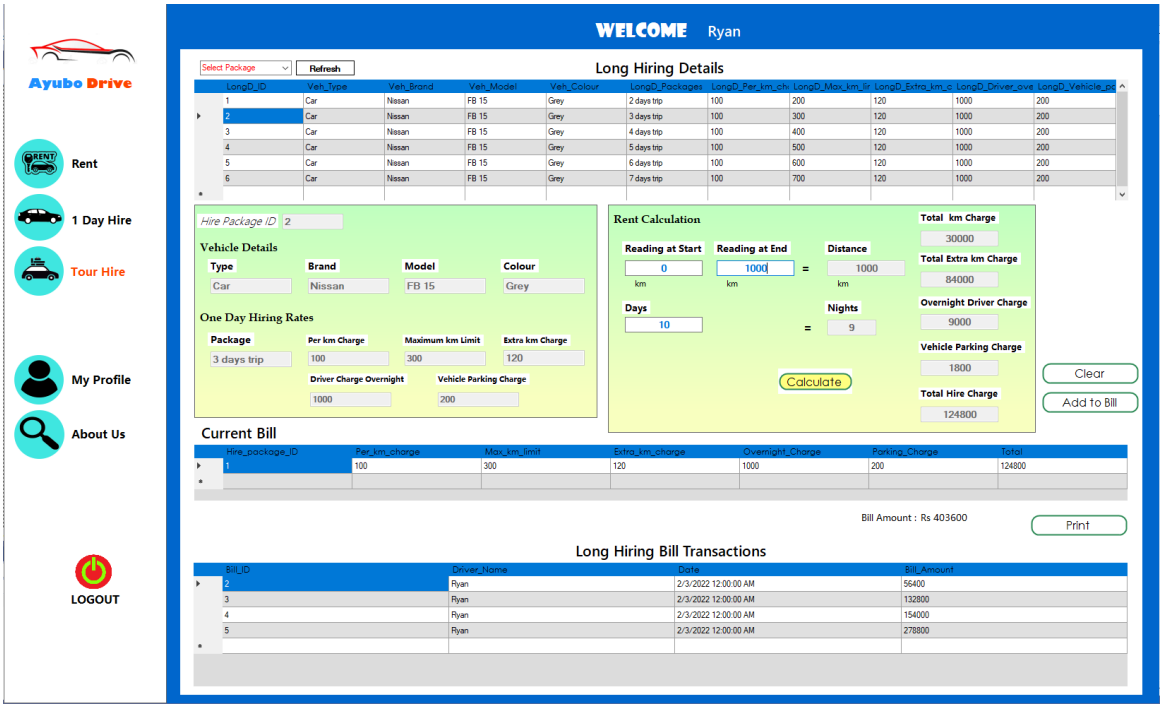


Figure 3. 40 Vehicle Long Tour Hire values adding to temporary bill table

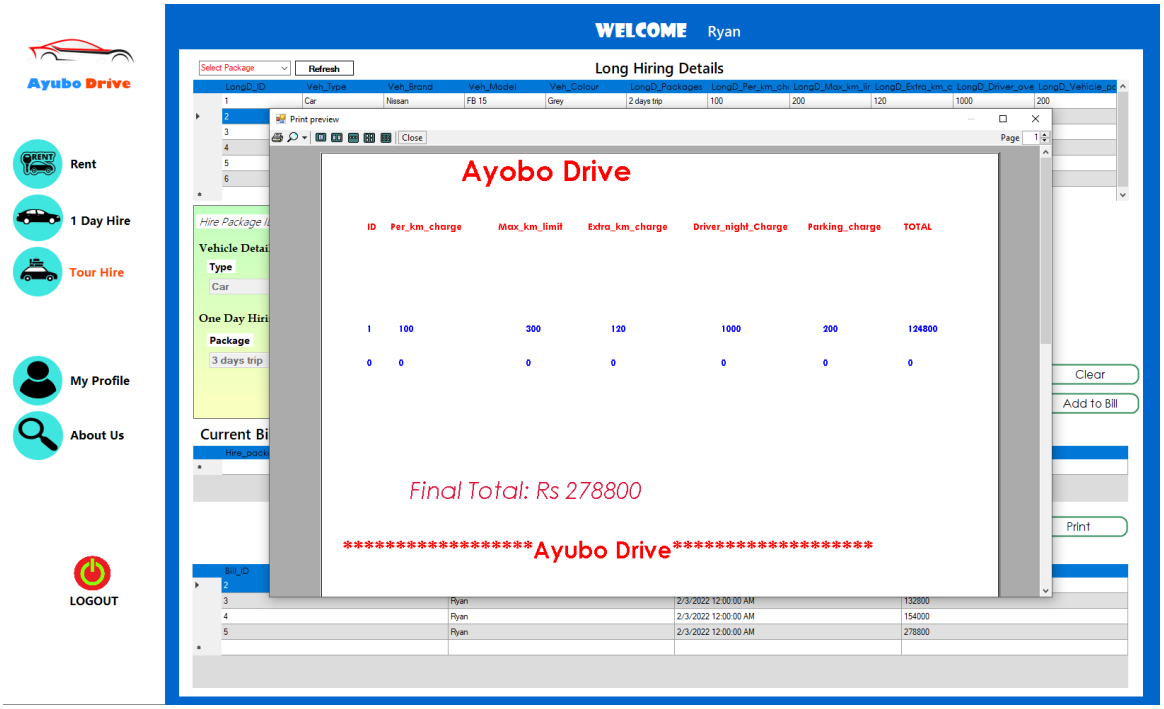


Figure 3. 41 Vehicle Long Tour Hire values adding to Long Tour bill table and printing the bill

Activity 4

4.1 Visual Studio Forms designs and system to calculate vehicle rent and hire amounts

4.1.1 Car rent system road map

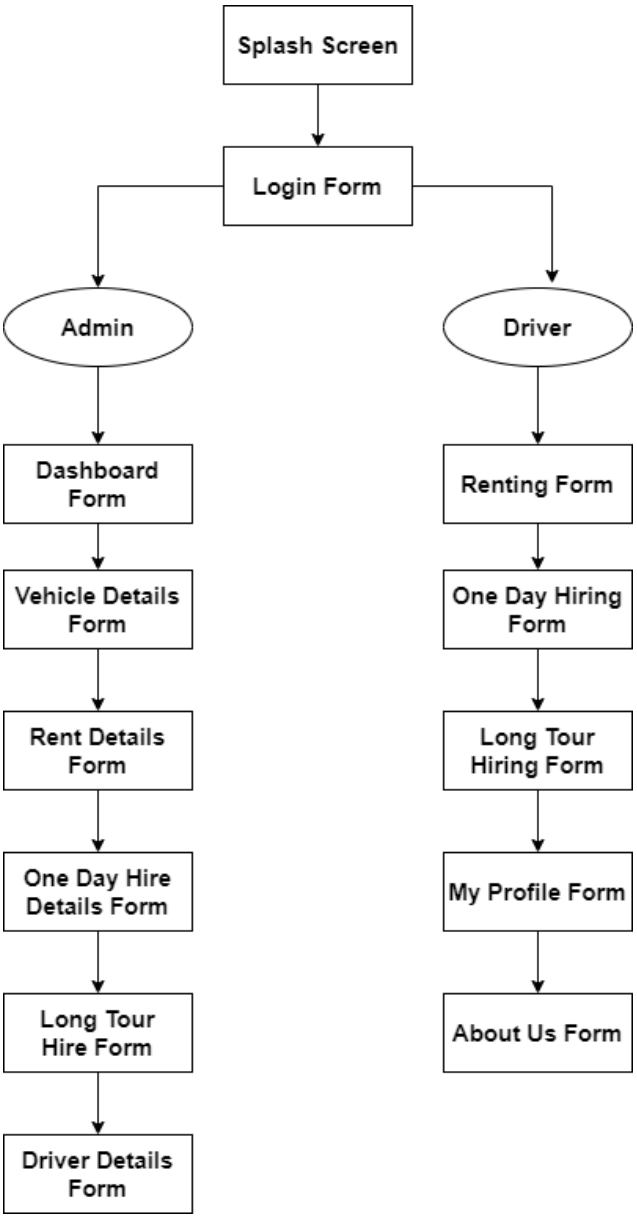


Figure 4. 1 Car rent system road map as a flow chart

4.1.2 Splash Screen Form Design and Code

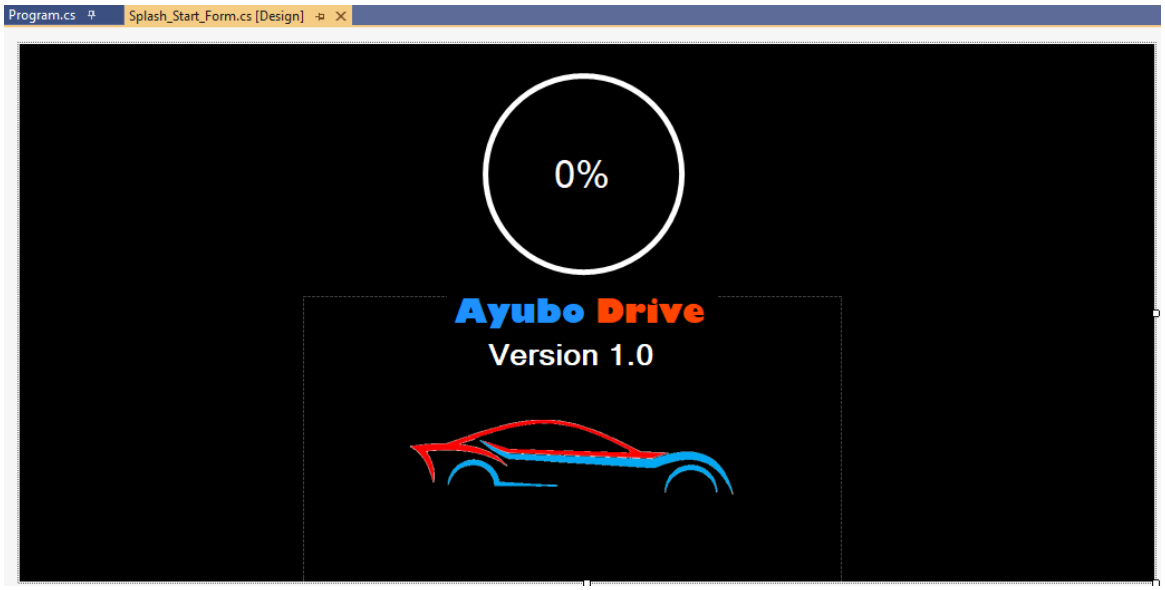


Figure 4. 2 Splash Screen Form Design

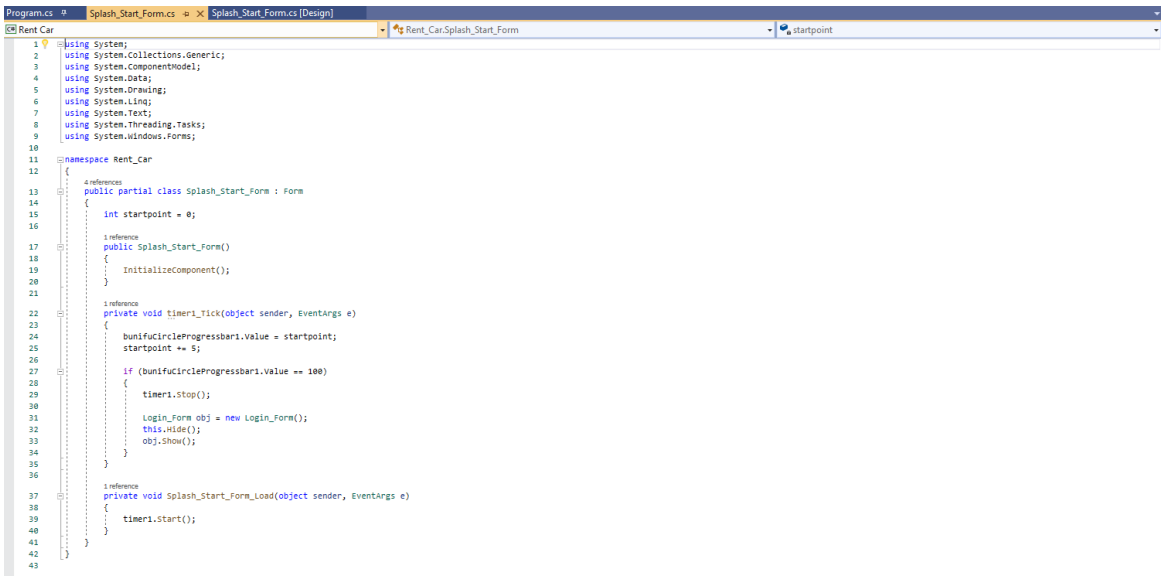


Figure 4. 3 Splash Screen Form Code - part 1

4.1.3 Login Form Design and Code

Figure 4. 4 Login Form Design

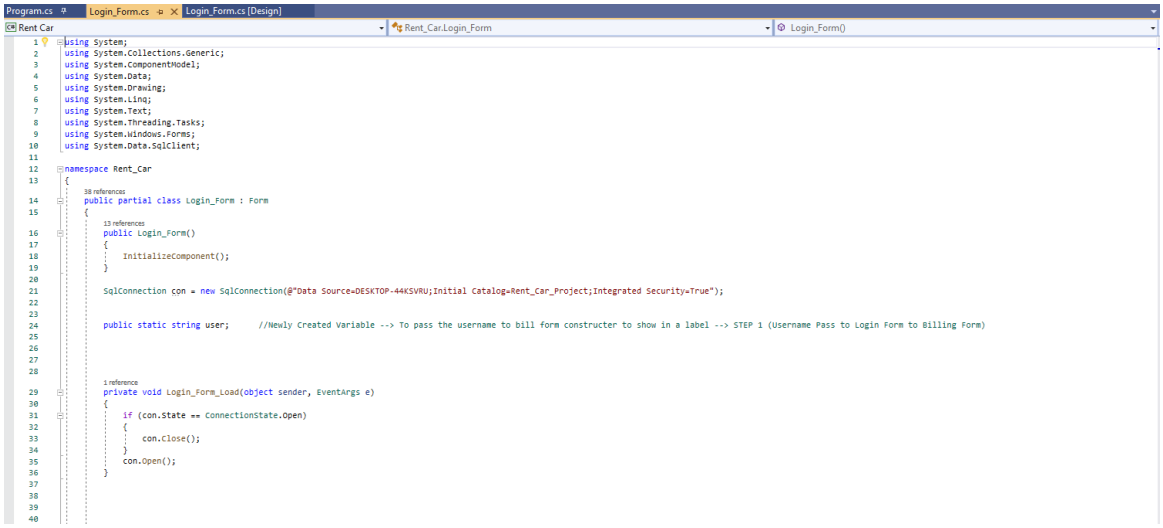


Figure 4. 5 Login Form Code - part 1

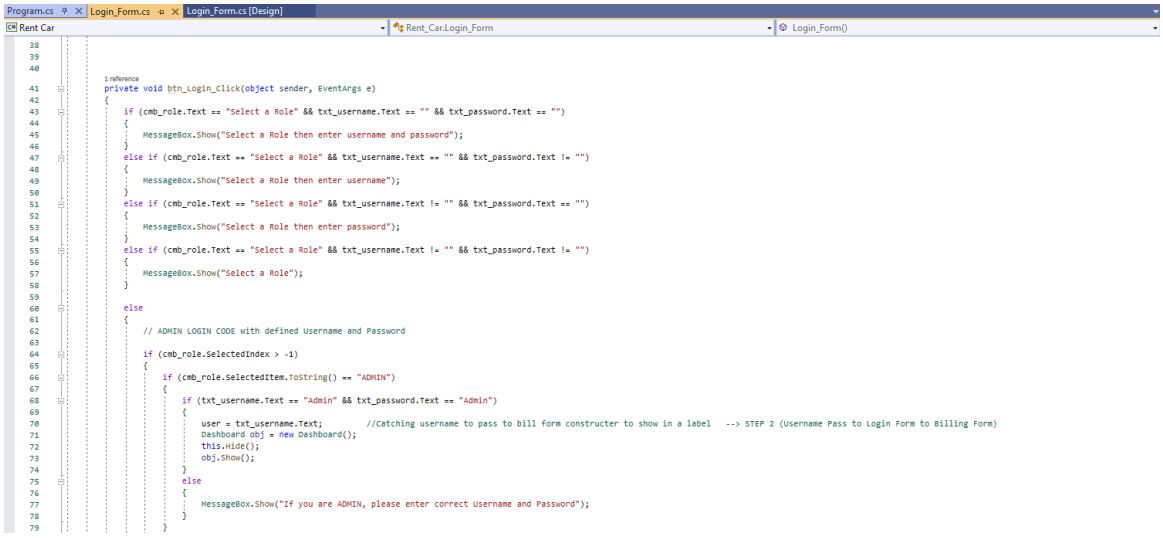


Figure 4. 6 Login Form Code - part 2

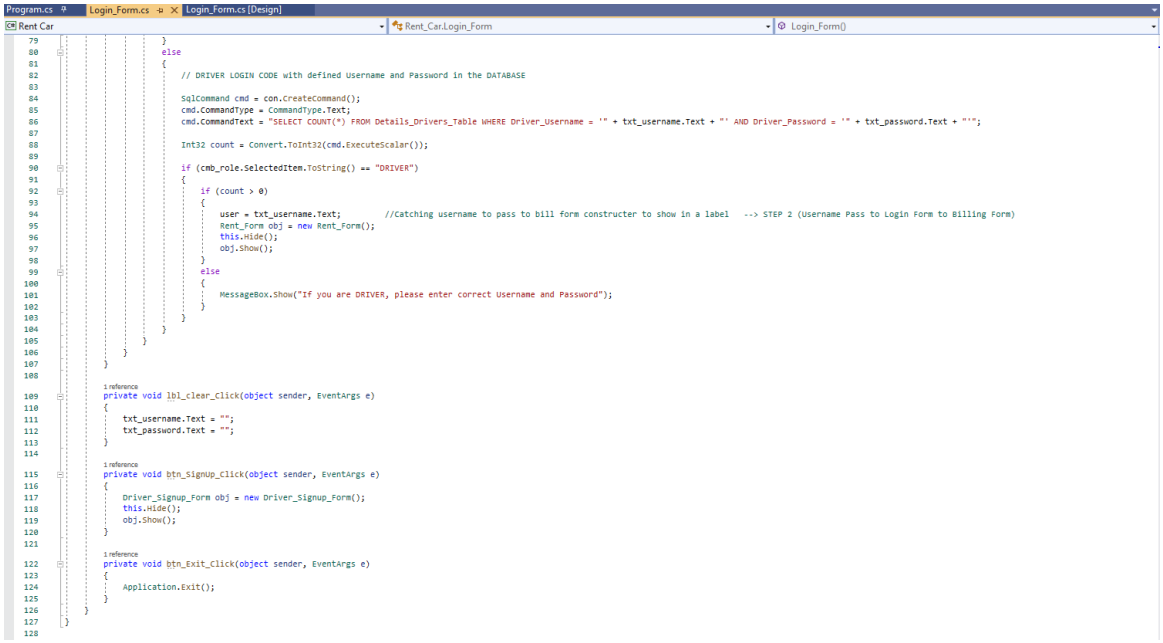


Figure 4. 7 Login Form Code - part 3

4.1.4 Driver Signup Form Design and Code

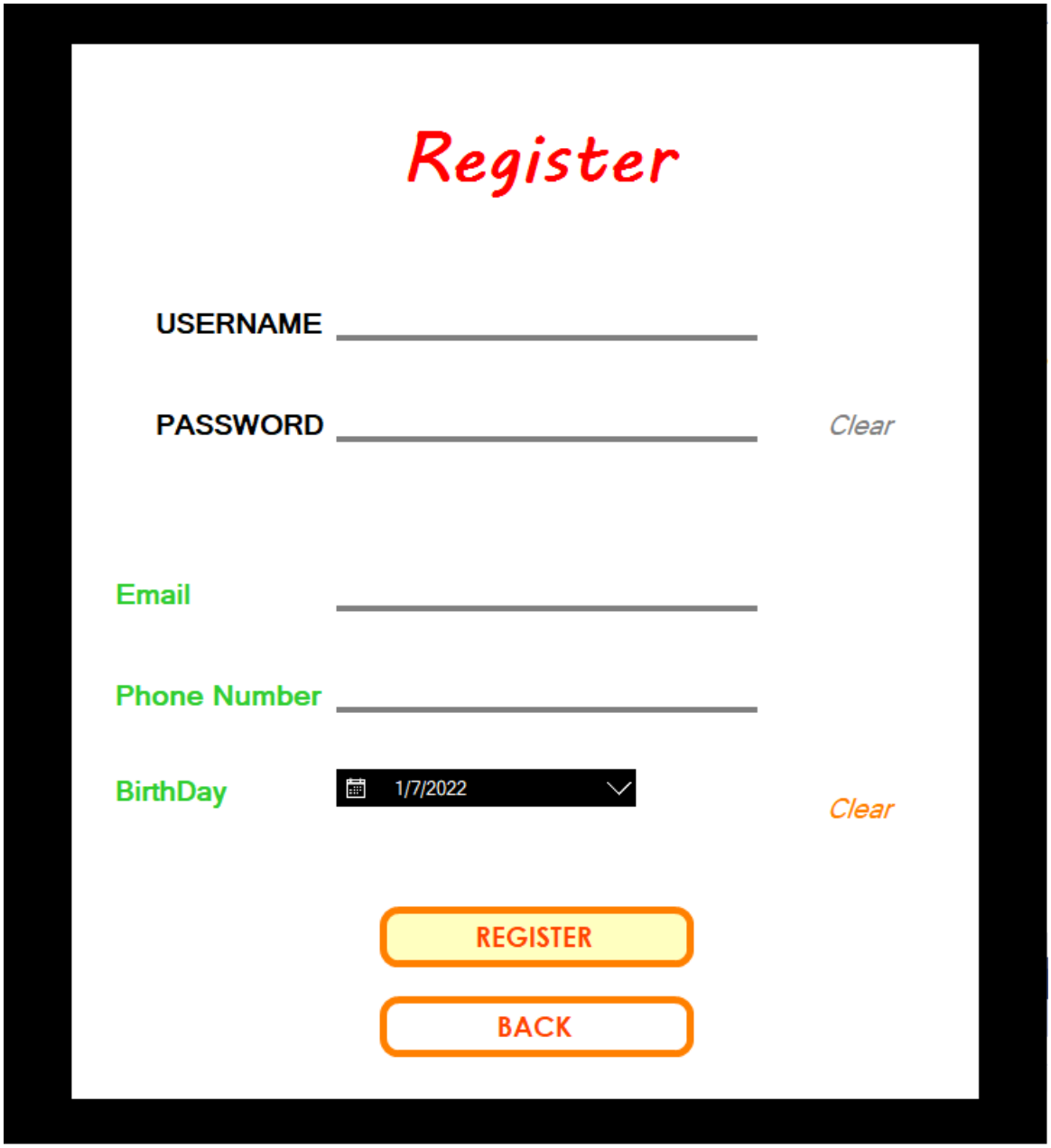
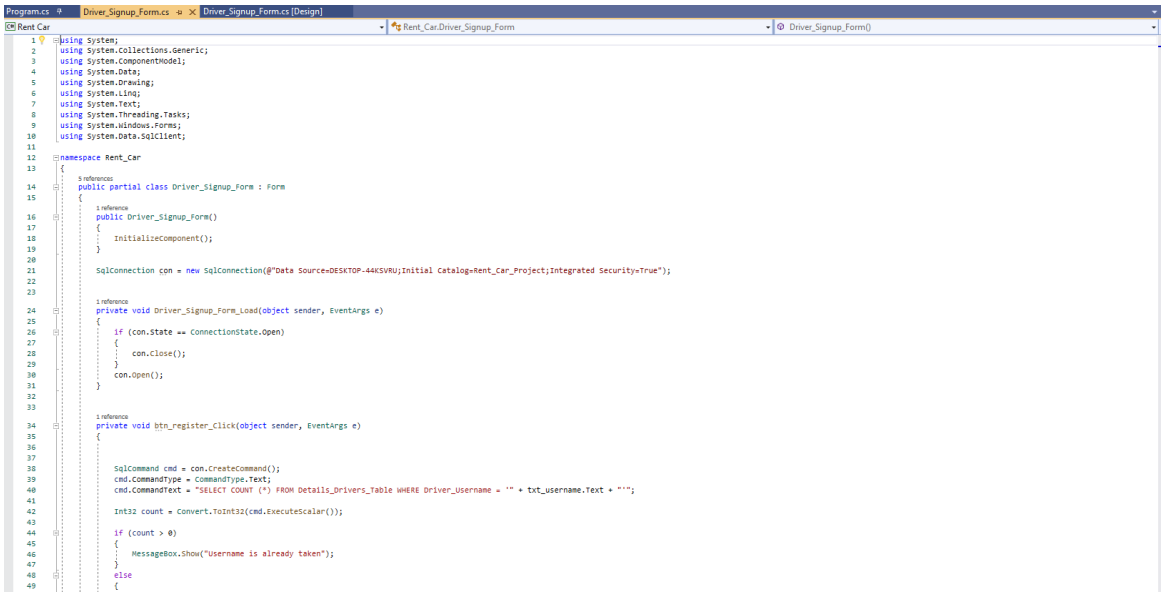
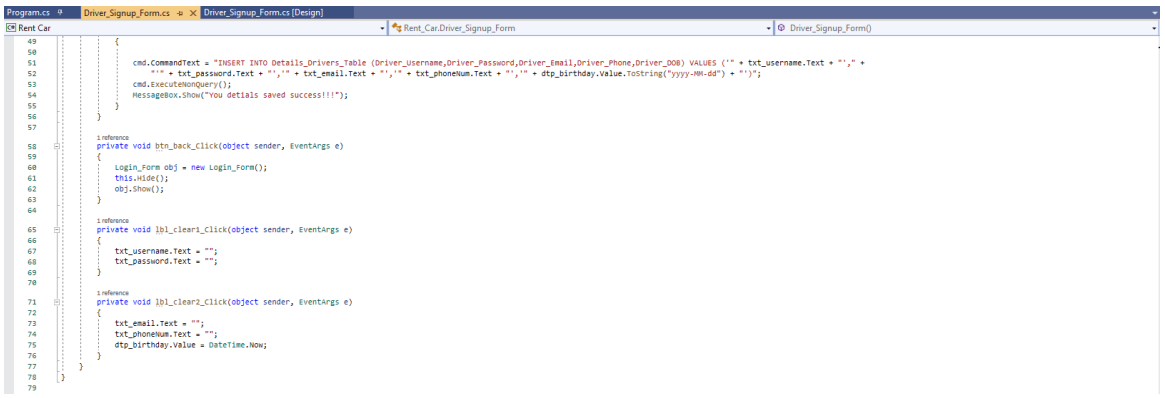


Figure 4. 8 Driver Signup Form Design



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

namespace Rent_Car
{
    public partial class Driver_Signup_Form : Form
    {
        public Driver_Signup_Form()
        {
            InitializeComponent();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");


        private void Driver_Signup_Form_Load(object sender, EventArgs e)
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            con.Open();
        }


        private void btn_register_Click(object sender, EventArgs e)
        {


            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT COUNT (*) FROM Details_Drivers_Table WHERE Driver_Username = '" + txt_username.Text + "'";

            Int32 count = Convert.ToInt32(cmd.ExecuteScalar());

            if (count > 0)
            {
                MessageBox.Show("Username is already taken");
            }
            else
            {
```

Figure 4. 9 Driver Signup Form Code - part 1



```
            {

                cmd.CommandText = "INSERT INTO Details_Drivers_Table (Driver_Username,Driver_Password,Driver_Email,Driver_Phone,Driver_DOB) VALUES ('" + txt_username.Text + "','" +
                    "'" + txt_password.Text + "','" + txt_email.Text + "','" + txt_phoneNum.Text + "','" + dtp_birthday.Value.ToString("yyyy-MM-dd") + "')";
                cmd.ExecuteNonQuery();
                MessageBox.Show("You detials saved success!!!");
            }
        }

        private void btn_back_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            this.Hide();
            obj.Show();
        }

        private void lbl_clear1_Click(object sender, EventArgs e)
        {
            txt_username.Text = "";
            txt_password.Text = "";
        }

        private void lbl_clear2_Click(object sender, EventArgs e)
        {
            txt_email.Text = "";
            txt_phoneNum.Text = "";
            dtp_birthday.Value = DateTime.Now;
        }
    }
}
```

Figure 4. 10 Driver Signup Form Code - part 2

4.1.5 Admin logged Dashboard Form Design and Code

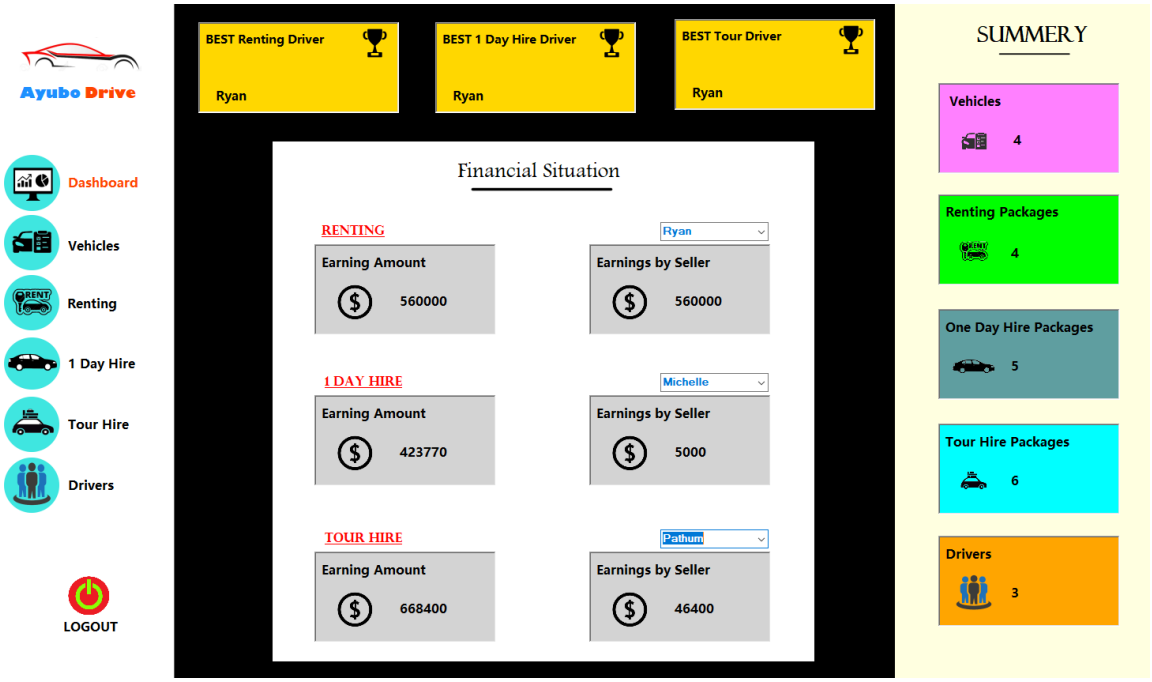
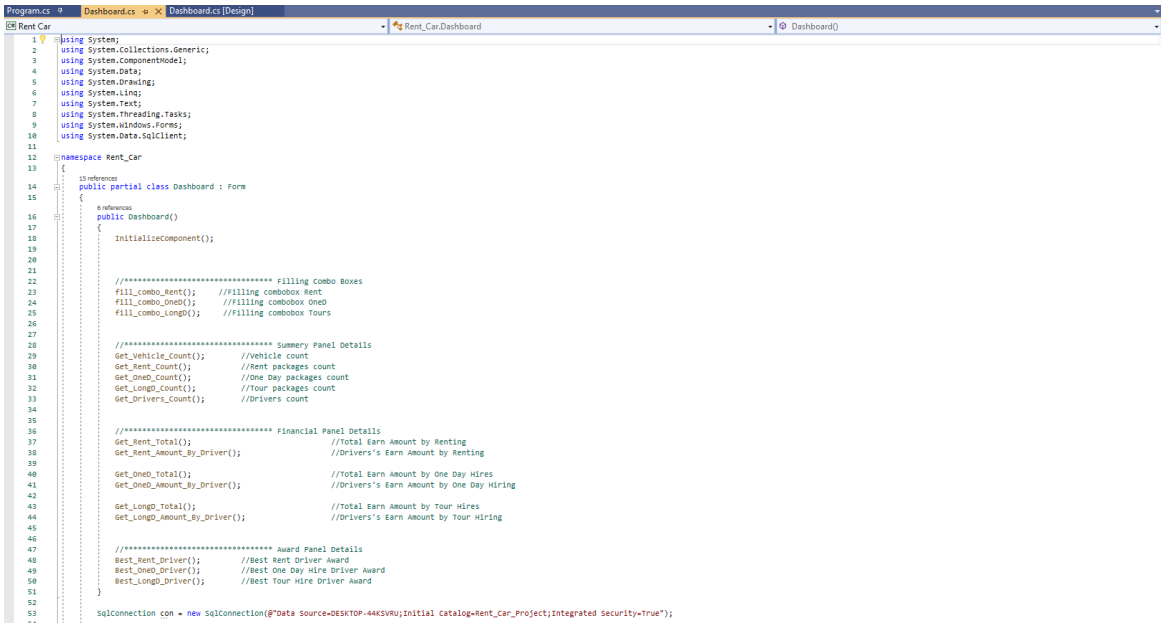
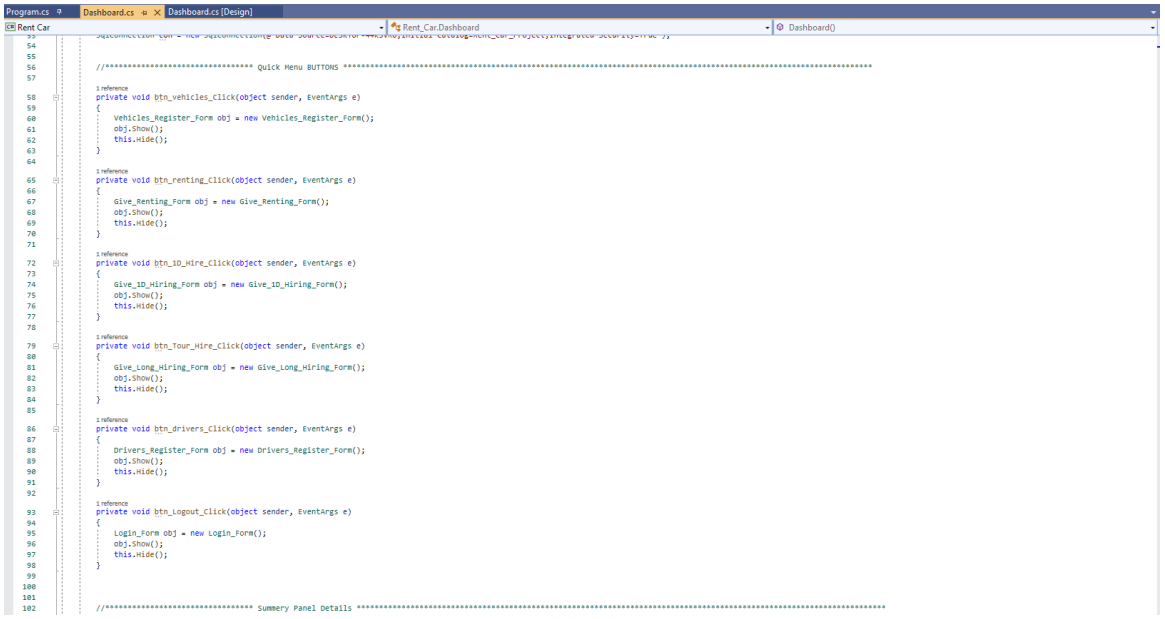


Figure 4. 11 Admin logged Dashboard Form Design



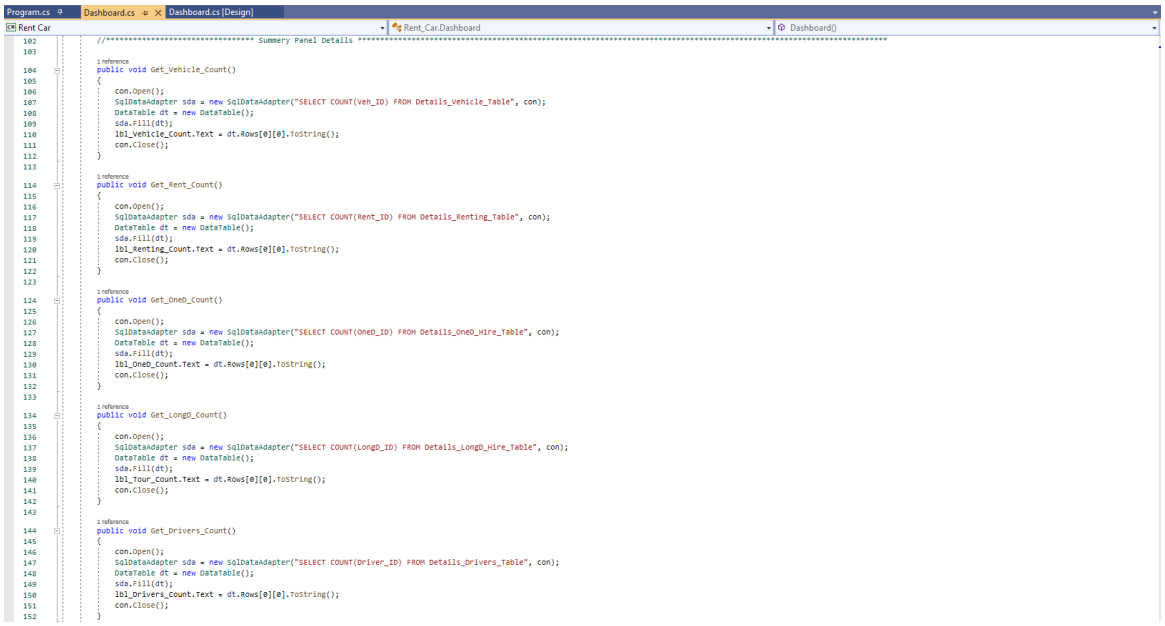
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

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



            //********************************** Filling Combo Boxes
            fill_combo_Rent();     //Filling combobox Rent
            fill_combo_OneD();     //Filling combobox OneD
            fill_combo_LongD();    //Filling combobox Tours


            //********************************** Summery Panel Details
            Get_Vehicle_Count();        //Vehicle count
            Get_Rent_Count();           //Rent packages count
            Get_OneD_Count();           //One Day packages count
            Get_LongD_Count();          //Tour packages count
            Get_Drivers_Count();        //Drivers count


            //********************************** Financial Panel Details
            Get_Rent_Total();                       //Total Earn Amount by Renting
            Get_Rent_Amount_By_Driver();            //Drivers's Earn Amount by Renting

            Get_OneD_Total();                       //Total Earn Amount by One Day Hires
            Get_OneD_Amount_By_Driver();            //Drivers's Earn Amount by One Day Hiring

            Get_LongD_Total();                      //Total Earn Amount by Tour Hires
            Get_LongD_Amount_By_Driver();           //Drivers's Earn Amount by Tour Hiring


            //********************************** Award Panel Details
            Best_Rent_Driver();         //Best Rent Driver Award
            Best_OneD_Driver();         //Best One Day Hire Driver Award
            Best_LongD_Driver();        //Best Tour Hire Driver Award
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
```

Figure 4. 12 Admin logged Dashboard Form Code - part 1



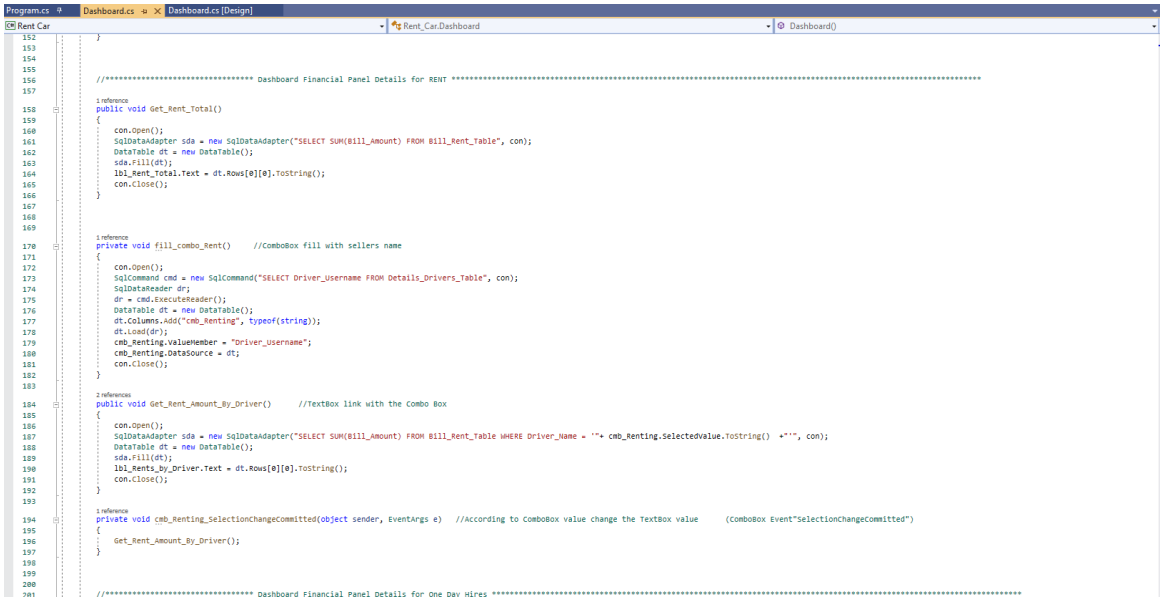
```
        //********************************** Quick Menu BUTTONS ****************************************************************************************************

        1 reference
        private void btn_vehicles_Click(object sender, EventArgs e)
        {
            Vehicles_Register_Form obj = new Vehicles_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_renting_Click(object sender, EventArgs e)
        {
            Give_Renting_Form obj = new Give_Renting_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Give_Long_Hiring_Form obj = new Give_Long_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_drivers_Click(object sender, EventArgs e)
        {
            Drivers_Register_Form obj = new Drivers_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        //********************************** Summery Panel Details ************************************************************************************************
```

Figure 4. 13 Admin logged Dashboard Form Code - part 2



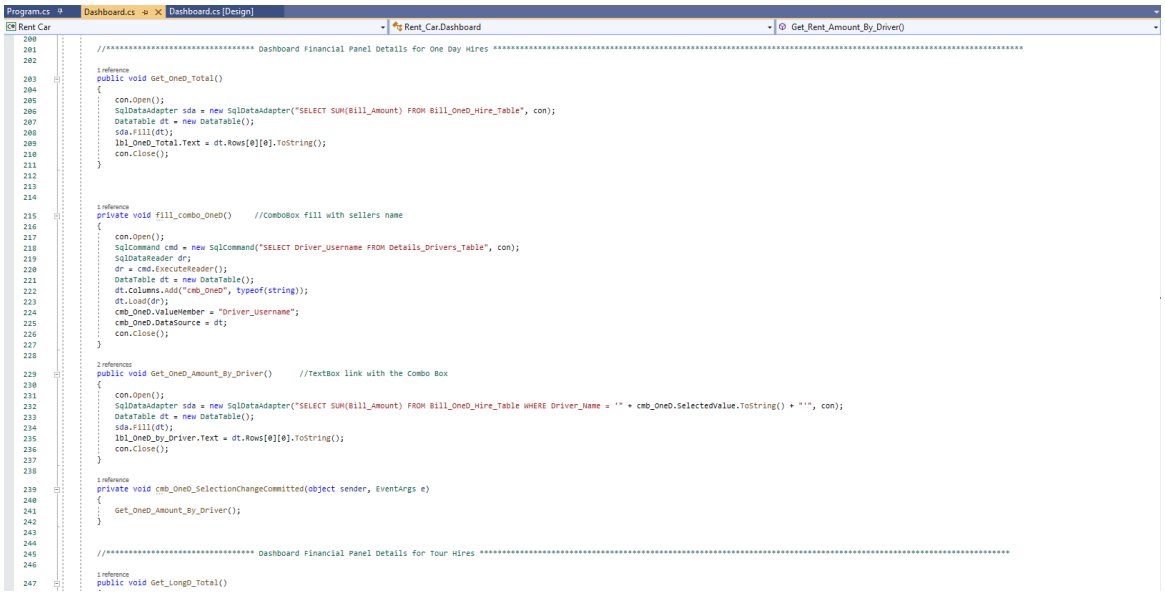
```
Program.cs  Dashboard.cs  Dashboard.cs [Design]
Rent Car  Rent_Car.Dashboard  Dashboard()

//********************************** Summery Panel Details ***********************************************************************************************************

1 reference
public void Get_Vehicle_Count()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(Veh_ID) FROM Details_Vehicle_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Vehicle_Count.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
public void Get_Rent_Count()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(Rent_ID) FROM Details_Renting_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Renting_Count.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
public void Get_OneD_Count()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(OneD_ID) FROM Details_OneD_Hire_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_OneD_Count.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
public void Get_LongD_Count()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(LongD_ID) FROM Details_LongD_Hire_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Tour_Count.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
public void Get_Drivers_Count()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(Driver_ID) FROM Details_Drivers_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Drivers_Count.Text = dt.Rows[0][0].ToString();
    con.Close();
}
```

Figure 4. 14 Admin logged Dashboard Form Code - part 3



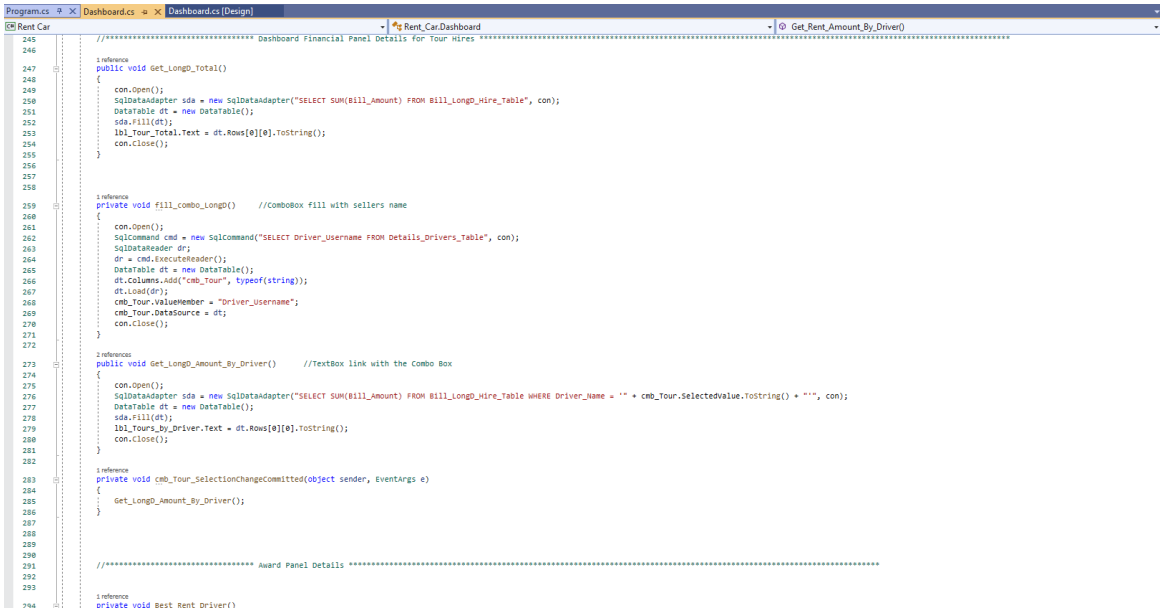
```
Program.cs  Dashboard.cs  Dashboard.cs [Design]
Rent Car  Rent_Car.Dashboard  Dashboard()

}



//********************************** Dashboard Financial Panel Details for RENT ***************************************************************************************

1 reference
public void Get_Rent_Total()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_Rent_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Rent_Total.Text = dt.Rows[0][0].ToString();
    con.Close();
}


1 reference
private void fill_combo_Rent()     //ComboBox fill with sellers name
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Driver_Username FROM Details_Drivers_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_Renting", typeof(string));
    dt.Load(dr);
    cmb_Renting.ValueMember = "Driver_Username";
    cmb_Renting.DataSource = dt;
    con.Close();
}

2 references
public void Get_Rent_Amount_By_Driver()      //TextBox link with the Combo Box
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_Rent_Table WHERE Driver_Name = '"+ cmb_Renting.SelectedValue.ToString()  +"'", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Rents_by_Driver.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
private void cmb_Renting_SelectionChangeCommitted(object sender, EventArgs e)   //According to ComboBox value change the TextBox value      (ComboBox Event"SelectionChangeCommitted")
{
    Get_Rent_Amount_By_Driver();
}



//********************************** Dashboard Financial Panel Details for One Day Hires ******************************************************************************
```

Figure 4. 15 Admin logged Dashboard Form Code - part 4



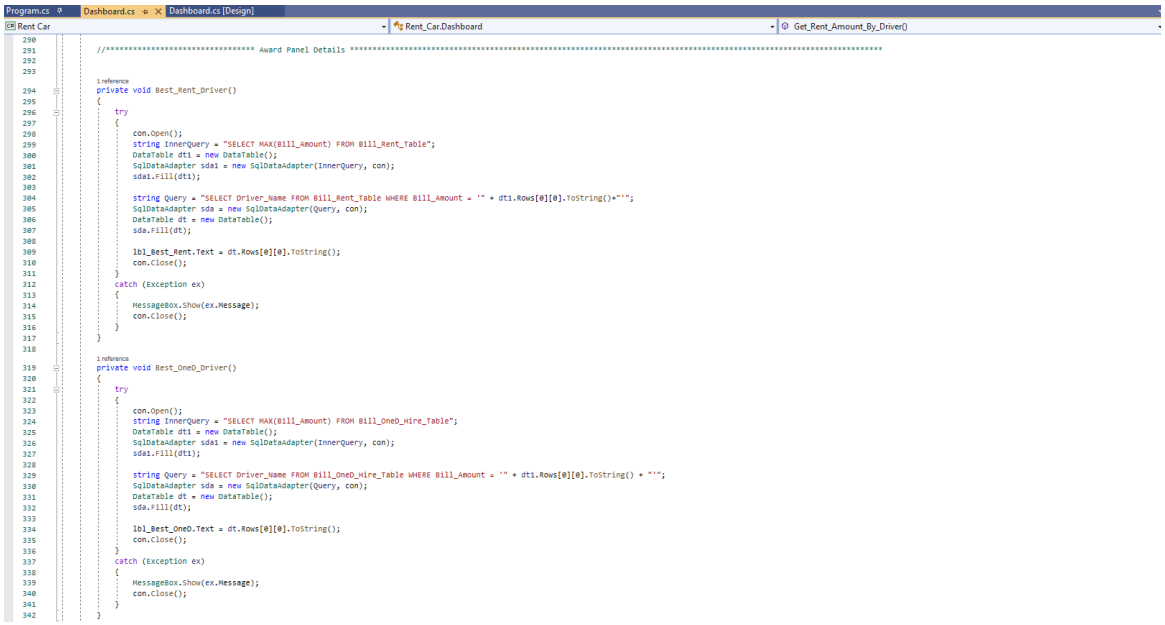
```
//********************************** Dashboard Financial Panel Details for One Day Hires ***********************************************************************************************

1 reference
public void Get_OneD_Total()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_OneD_Hire_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_OneD_Total.Text = dt.Rows[0][0].ToString();
    con.Close();
}



1 reference
private void fill_combo_OneD()     //ComboBox fill with sellers name
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Driver_Username FROM Details_Drivers_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_OneD", typeof(string));
    dt.Load(dr);
    cmb_OneD.ValueMember = "Driver_Username";
    cmb_OneD.DataSource = dt;
    con.Close();
}

2 references
public void Get_OneD_Amount_By_Driver()      //TextBox link with the Combo Box
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_OneD_Hire_Table WHERE Driver_Name = '" + cmb_OneD.SelectedValue.ToString() + "'", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_OneD_by_Driver.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
private void cmb_OneD_SelectionChangeCommitted(object sender, EventArgs e)
{
    Get_OneD_Amount_By_Driver();
}


//********************************** Dashboard Financial Panel Details for Tour Hires ***********************************************************************************************

1 reference
public void Get_LongD_Total()
```

Figure 4. 16 Admin logged Dashboard Form Code - part 5



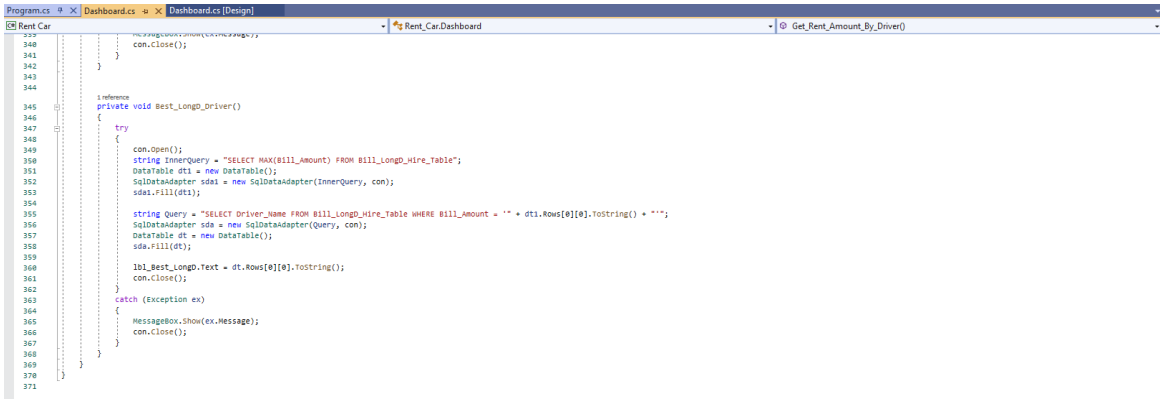
```
//********************************** Dashboard Financial Panel Details for Tour Hires ***********************************************************************************************

1 reference
public void Get_LongD_Total()
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_LongD_Hire_Table", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Tour_Total.Text = dt.Rows[0][0].ToString();
    con.Close();
}



1 reference
private void fill_combo_LongD()     //ComboBox fill with sellers name
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Driver_Username FROM Details_Drivers_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_Tour", typeof(string));
    dt.Load(dr);
    cmb_Tour.ValueMember = "Driver_Username";
    cmb_Tour.DataSource = dt;
    con.Close();
}

2 references
public void Get_LongD_Amount_By_Driver()      //TextBox link with the Combo Box
{
    con.Open();
    SqlDataAdapter sda = new SqlDataAdapter("SELECT SUM(Bill_Amount) FROM Bill_LongD_Hire_Table WHERE Driver_Name = '" + cmb_Tour.SelectedValue.ToString() + "'", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    lbl_Tours_by_Driver.Text = dt.Rows[0][0].ToString();
    con.Close();
}

1 reference
private void cmb_Tour_SelectionChangeCommitted(object sender, EventArgs e)
{
    Get_LongD_Amount_By_Driver();
}



//********************************** Award Panel Details ***********************************************************************************************

1 reference
private void Best_Rent_Driver()
```

Figure 4. 17 Admin logged Dashboard Form Code - part 6




```
//******************************** Award Panel Details ***********************************************************************************************************

1 reference
private void Best_Rent_Driver()
{
    try
    {
        con.Open();
        string InnerQuery = "SELECT MAX(Bill_Amount) FROM Bill_Rent_Table";
        DataTable dt1 = new DataTable();
        SqlDataAdapter sda1 = new SqlDataAdapter(InnerQuery, con);
        sda1.Fill(dt1);

        string Query = "SELECT Driver_Name FROM Bill_Rent_Table WHERE Bill_Amount = '" + dt1.Rows[0][0].ToString()+"'";
        SqlDataAdapter sda = new SqlDataAdapter(Query, con);
        DataTable dt = new DataTable();
        sda.Fill(dt);

        lbl_Best_Rent.Text = dt.Rows[0][0].ToString();
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void Best_OneD_Driver()
{
    try
    {
        con.Open();
        string InnerQuery = "SELECT MAX(Bill_Amount) FROM Bill_OneD_Hire_Table";
        DataTable dt1 = new DataTable();
        SqlDataAdapter sda1 = new SqlDataAdapter(InnerQuery, con);
        sda1.Fill(dt1);

        string Query = "SELECT Driver_Name FROM Bill_OneD_Hire_Table WHERE Bill_Amount = '" + dt1.Rows[0][0].ToString() + "'";
        SqlDataAdapter sda = new SqlDataAdapter(Query, con);
        DataTable dt = new DataTable();
        sda.Fill(dt);

        lbl_Best_OneD.Text = dt.Rows[0][0].ToString();
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}
```

Figure 4. 18 Admin logged Dashboard Form Code - part 7




```
        con.Close();
    }
}

1 reference
private void Best_LongD_Driver()
{
    try
    {
        con.Open();
        string InnerQuery = "SELECT MAX(Bill_Amount) FROM Bill_LongD_Hire_Table";
        DataTable dt1 = new DataTable();
        SqlDataAdapter sda1 = new SqlDataAdapter(InnerQuery, con);
        sda1.Fill(dt1);

        string Query = "SELECT Driver_Name FROM Bill_LongD_Hire_Table WHERE Bill_Amount = '" + dt1.Rows[0][0].ToString() + "'";
        SqlDataAdapter sda = new SqlDataAdapter(Query, con);
        DataTable dt = new DataTable();
        sda.Fill(dt);

        lbl_Best_LongD.Text = dt.Rows[0][0].ToString();
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}
}
}
```

Figure 4. 19 Admin logged Dashboard Form Code - part 8

4.1.6 Admin logged Vehicles Form Design and Code

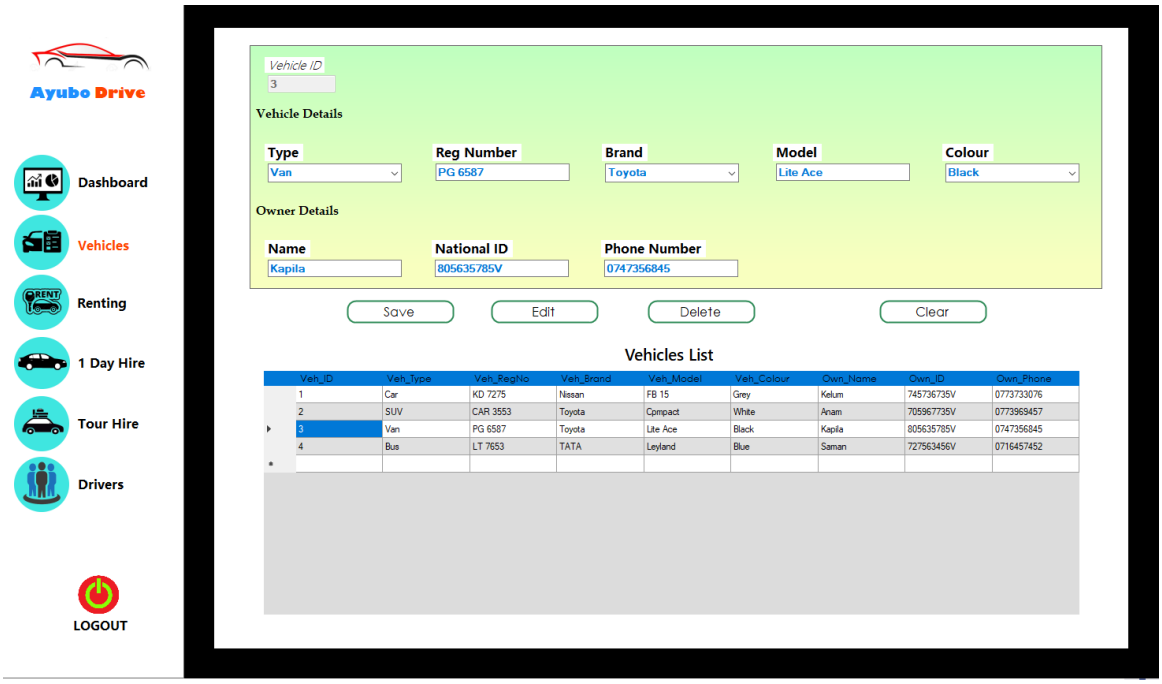
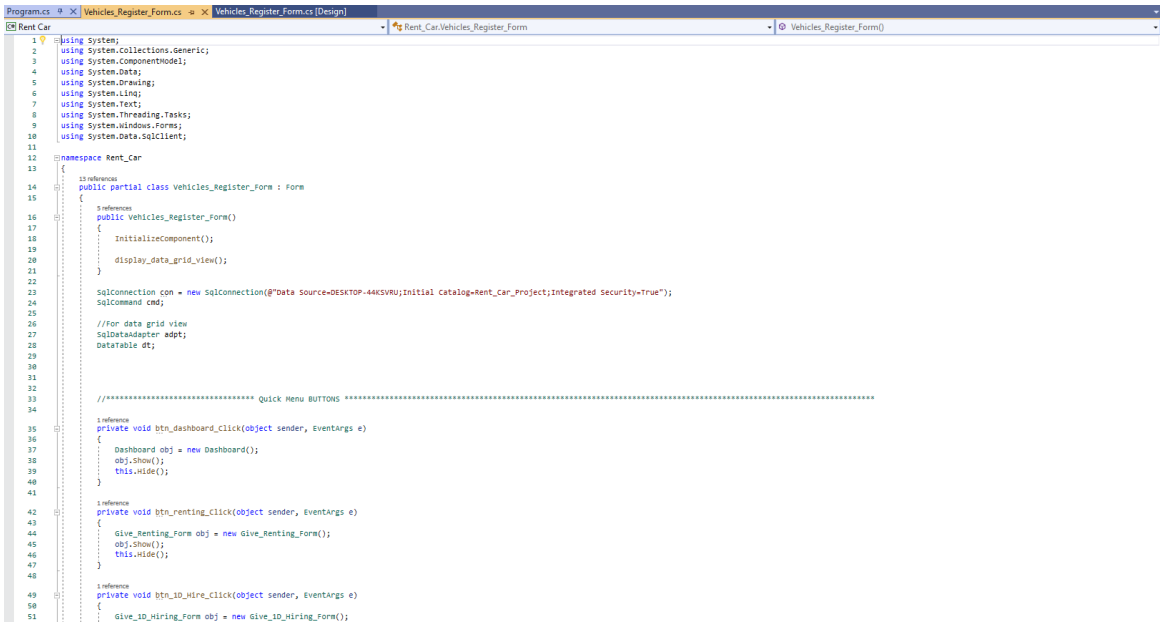


Figure 4. 20 Admin logged Vehicles Form Design

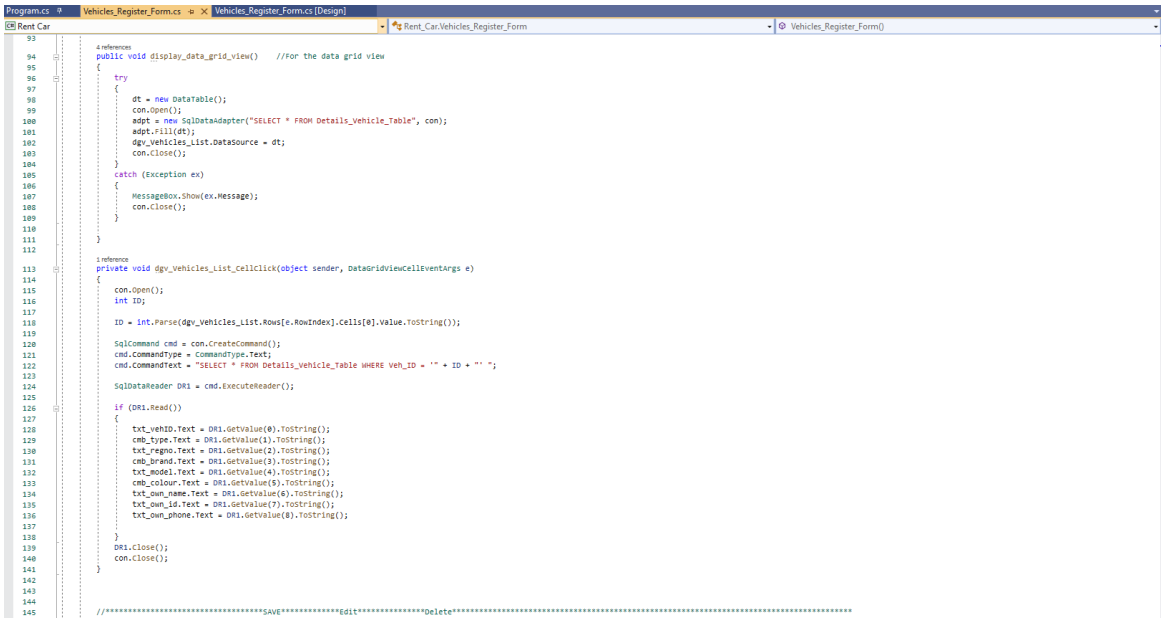


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    13 references
    public partial class Vehicles_Register_Form : Form
    {
        5 references
        public Vehicles_Register_Form()
        {
            InitializeComponent();

            display_data_grid_view();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;

        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;



        //******************************** Quick Menu BUTTONS ********************************************************************************************************

        1 reference
        private void btn_dashboard_Click(object sender, EventArgs e)
        {
            Dashboard obj = new Dashboard();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_renting_Click(object sender, EventArgs e)
        {
            Give_Renting_Form obj = new Give_Renting_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
```

Figure 4. 21 Admin logged Vehicles Form Code - part 1

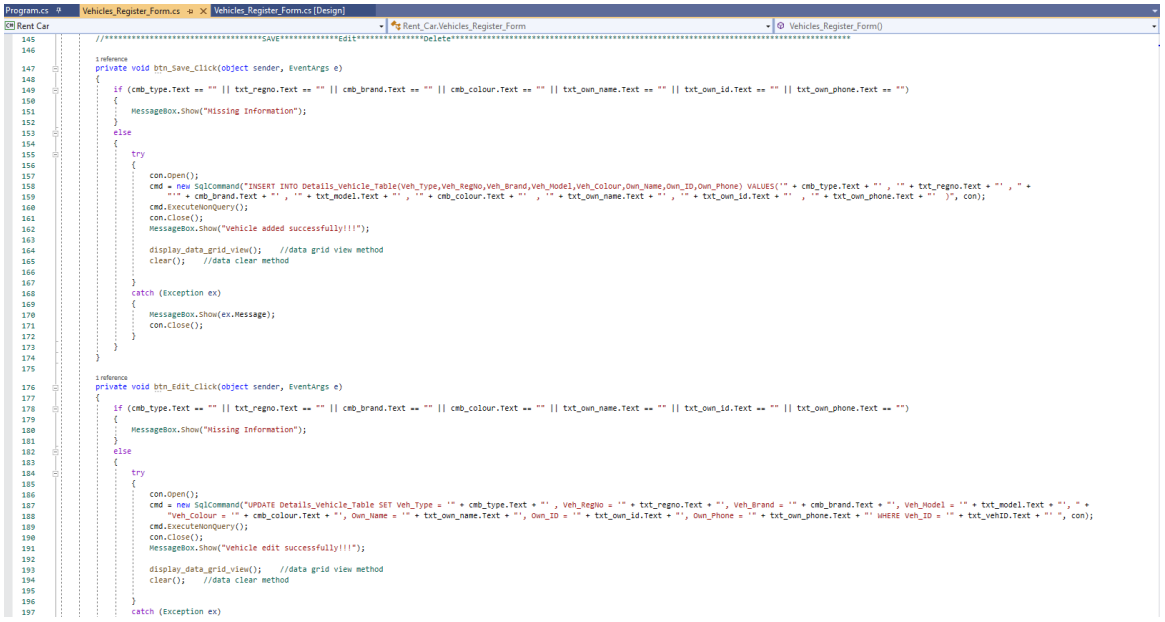
```
        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Give_Long_Hiring_Form obj = new Give_Long_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_drivers_Click(object sender, EventArgs e)
        {
            Drivers_Register_Form obj = new Drivers_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        //******************************** OTHER METHODS ********************************************************************************************************

        3 references
        public void clear()
        {
            txt_vehID.Text = "";
            cmb_type.Text = "";
            txt_regno.Text = "";
            cmb_brand.Text = "";
            txt_model.Text = "";
            cmb_colour.Text = "";
            txt_own_name.Text = "";
            txt_own_id.Text = "";
            txt_own_phone.Text = "";

        }
```

Figure 4. 22 Admin logged Vehicles Form Code - part 2



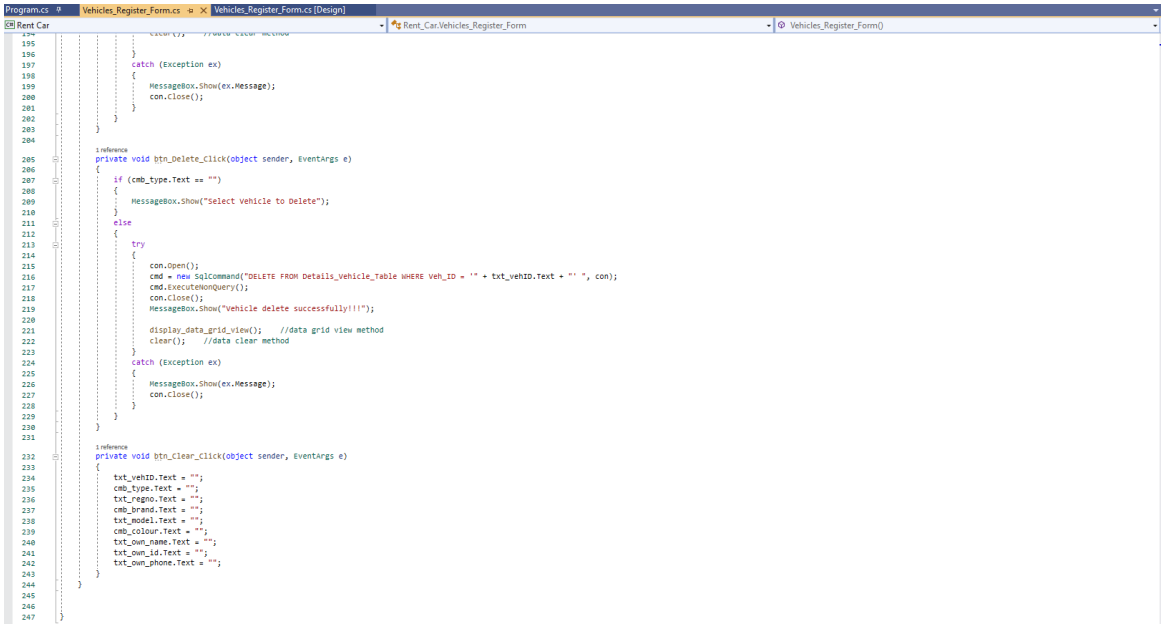
```
Program.cs | Vehicles_Register_Form.cs | Vehicles_Register_Form.cs [Design]
Rent Car | Rent_Car.Vehicles_Register_Form | Vehicles_Register_Form()

4 references
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Details_Vehicle_Table", con);
        adpt.Fill(dt);
        dgv_vehicles_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }

}

1 reference
private void dgv_Vehicles_List_CellClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_Vehicles_List.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Details_Vehicle_Table WHERE Veh_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_vehID.Text = DR1.GetValue(0).ToString();
        cmb_type.Text = DR1.GetValue(1).ToString();
        txt_regno.Text = DR1.GetValue(2).ToString();
        cmb_brand.Text = DR1.GetValue(3).ToString();
        txt_model.Text = DR1.GetValue(4).ToString();
        cmb_colour.Text = DR1.GetValue(5).ToString();
        txt_own_name.Text = DR1.GetValue(6).ToString();
        txt_own_id.Text = DR1.GetValue(7).ToString();
        txt_own_phone.Text = DR1.GetValue(8).ToString();

    }
    DR1.close();
    con.Close();
}


//*************************************SAVE************Edit**************Delete*****************************************************************************************
```

Figure 4. 23 Admin logged Vehicles Form Code - part 3



```
Program.cs | Vehicles_Register_Form.cs | Vehicles_Register_Form.cs [Design]
Rent Car | Rent_Car.Vehicles_Register_Form | Vehicles_Register_Form()

//*************************************SAVE************Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (cmb_type.Text == "" || txt_regno.Text == "" || cmb_brand.Text == "" || cmb_colour.Text == "" || txt_own_name.Text == "" || txt_own_id.Text == "" || txt_own_phone.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Details_Vehicle_Table(Veh_Type,Veh_RegNo,Veh_Brand,Veh_Model,Veh_Colour,Own_Name,Own_ID,Own_Phone) VALUES('" + cmb_type.Text + "' , '" + txt_regno.Text + "' , " +
                "'" + cmb_brand.Text + "' , '" + txt_model.Text + "' , '" + cmb_colour.Text + "'  , '" + txt_own_name.Text + "' , '" + txt_own_id.Text + "'  , '" + txt_own_phone.Text + "'  )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Vehicle added successfully!!!");

            display_data_grid_view();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (cmb_type.Text == "" || txt_regno.Text == "" || cmb_brand.Text == "" || cmb_colour.Text == "" || txt_own_name.Text == "" || txt_own_id.Text == "" || txt_own_phone.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("UPDATE Details_Vehicle_Table SET Veh_Type = '" + cmb_type.Text + "' , Veh_RegNo = '" + txt_regno.Text + "', Veh_Brand = '" + cmb_brand.Text + "', Veh_Model = '" + txt_model.Text + "', " +
                "Veh_Colour = '" + cmb_colour.Text + "', Own_Name = '" + txt_own_name.Text + "', Own_ID = '" + txt_own_id.Text + "', Own_Phone = '" + txt_own_phone.Text + "' WHERE Veh_ID = '" + txt_vehID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Vehicle edit successfully!!!");

            display_data_grid_view();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
```

Figure 4. 24 Admin logged Vehicles Form Code - part 4



```
                clear();    //data clear method
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (cmb_type.Text == "")
            {
                MessageBox.Show("Select Vehicle to Delete");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("DELETE FROM Details_Vehicle_Table WHERE Veh_ID = '" + txt_vehID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Vehicle delete successfully!!!");

                    display_data_grid_view();    //data grid view method
                    clear();    //data clear method
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_vehID.Text = "";
            cmb_type.Text = "";
            txt_regno.Text = "";
            cmb_brand.Text = "";
            txt_model.Text = "";
            cmb_colour.Text = "";
            txt_own_name.Text = "";
            txt_own_id.Text = "";
            txt_own_phone.Text = "";
        }
    }


}
```

Figure 4. 25 Admin logged Vehicles Form Code - part 5

4.1.7 Admin logged Rent Details Form Design and Code

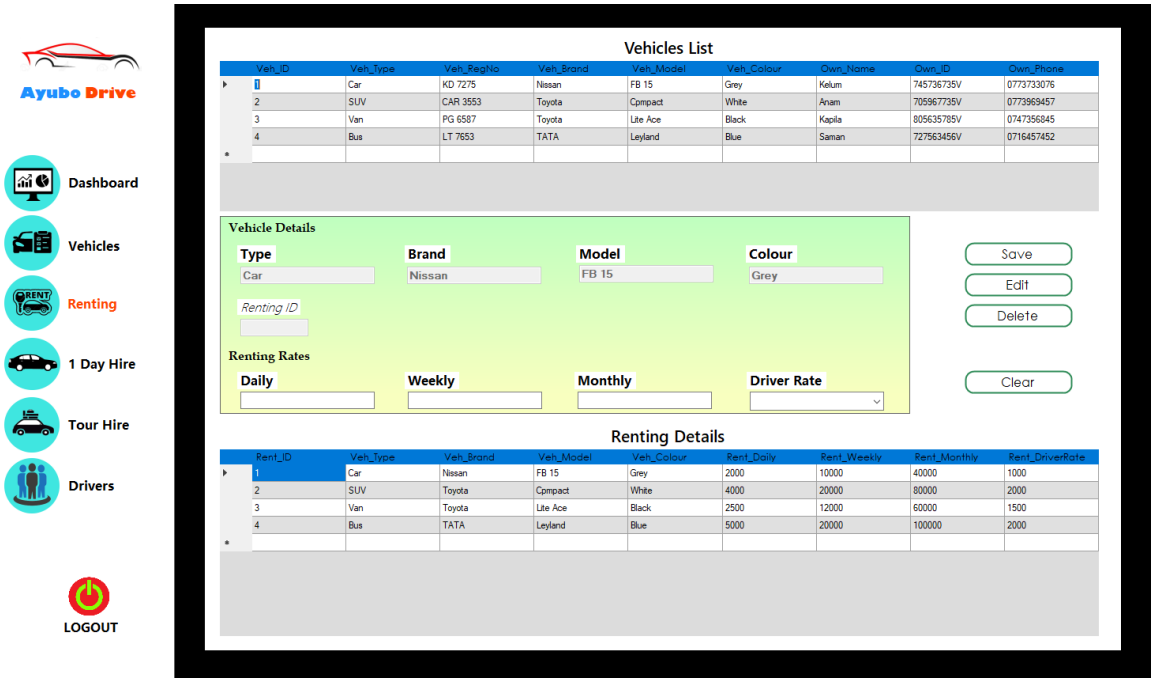
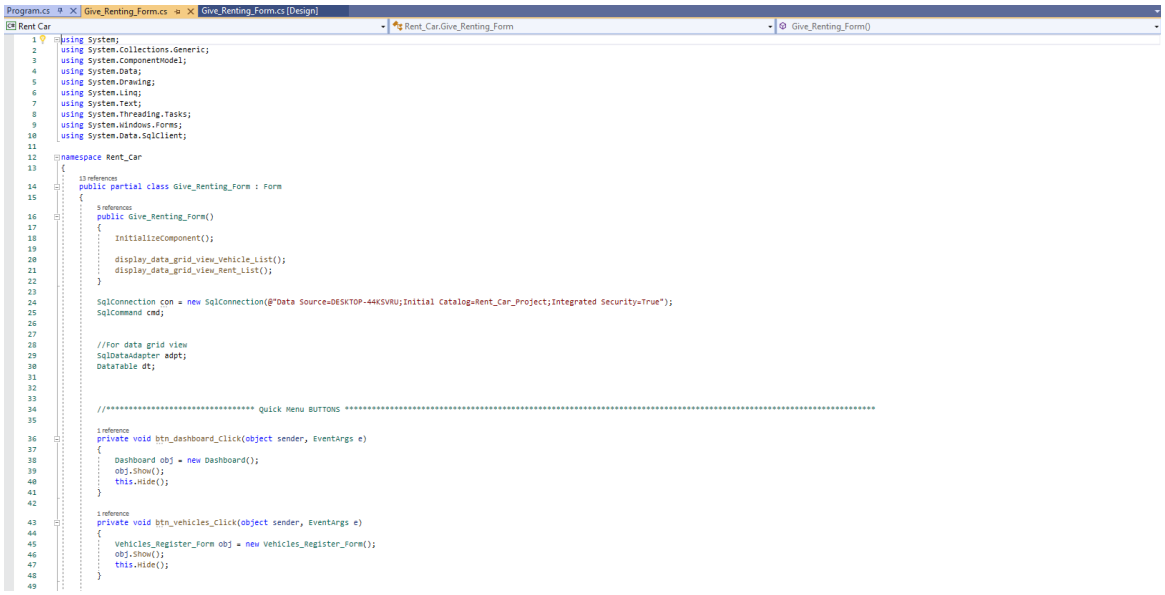
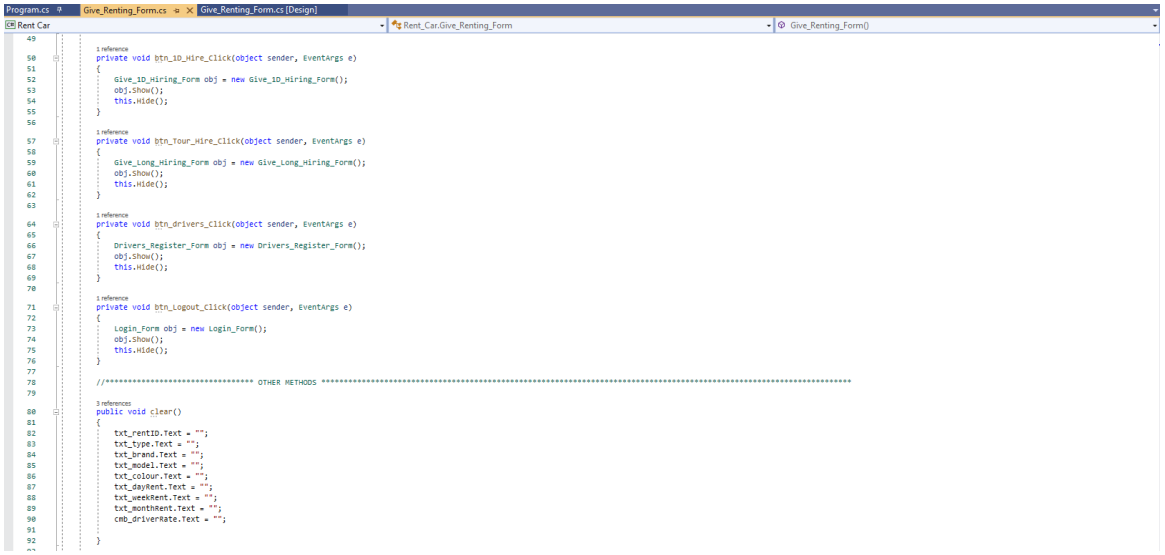


Figure 4. 26 Admin logged Rent Details Form Design



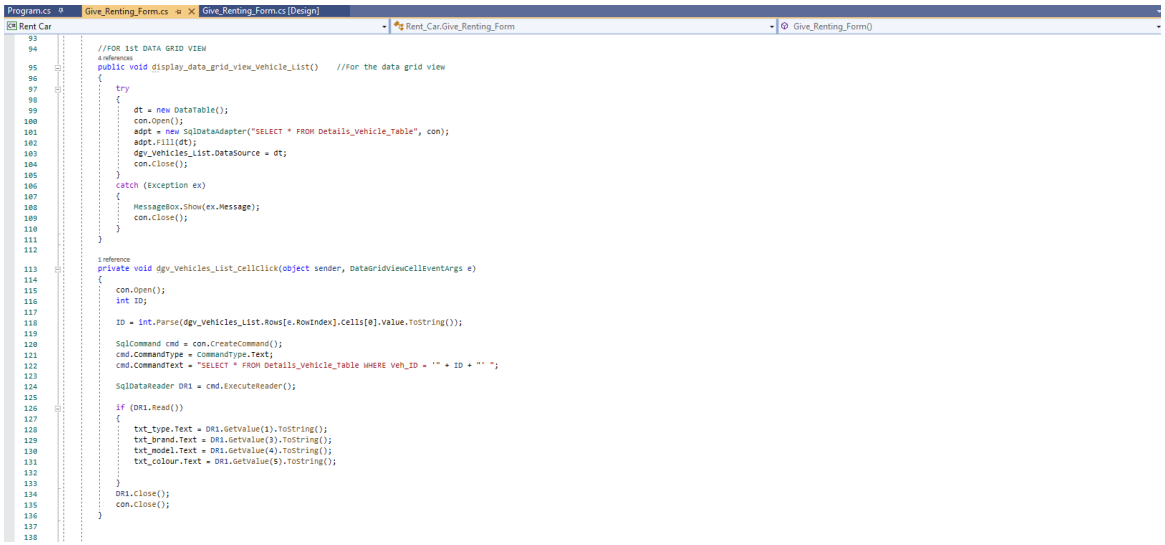
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    13 references
    public partial class Give_Renting_Form : Form
    {
        5 references
        public Give_Renting_Form()
        {
            InitializeComponent();

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_Rent_List();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44K5VRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;



        //********************************** Quick Menu BUTTONS ***************************************************************************************************************

        1 reference
        private void btn_dashboard_Click(object sender, EventArgs e)
        {
            Dashboard obj = new Dashboard();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_vehicles_Click(object sender, EventArgs e)
        {
            Vehicles_Register_Form obj = new Vehicles_Register_Form();
            obj.Show();
            this.Hide();
        }

```

Figure 4. 27 Admin logged Rent Details Form Code - part 1



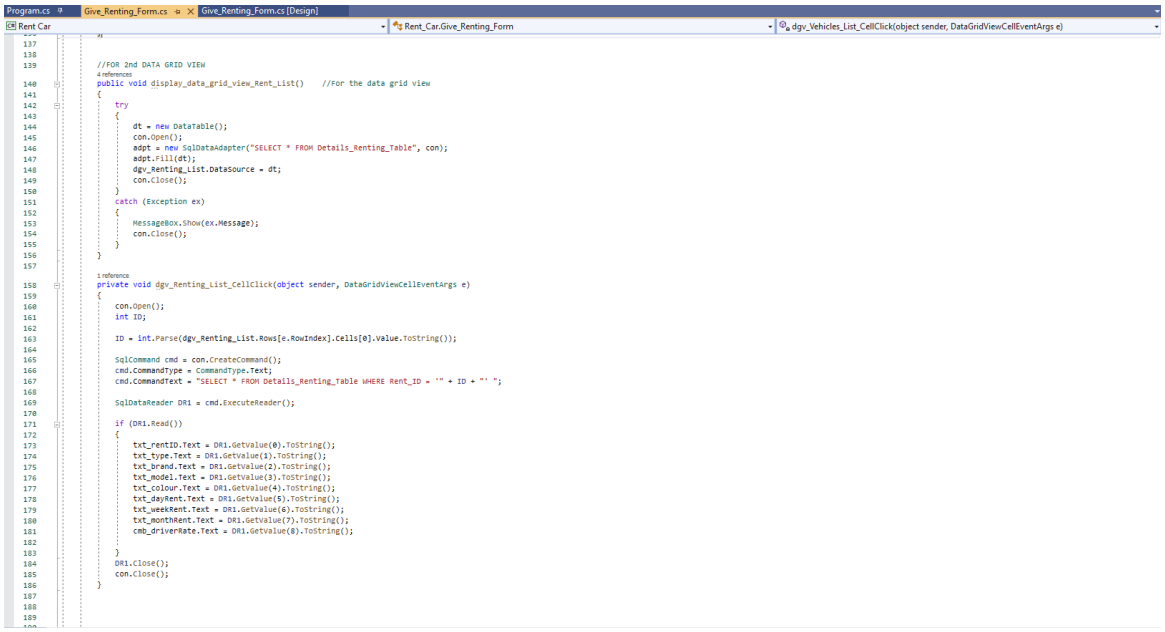
```

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Give_Long_Hiring_Form obj = new Give_Long_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_drivers_Click(object sender, EventArgs e)
        {
            Drivers_Register_Form obj = new Drivers_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        //********************************** OTHER METHODS ********************************************************************************************************************

        3 references
        public void clear()
        {
            txt_rentID.Text = "";
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            txt_dayRent.Text = "";
            txt_weekRent.Text = "";
            txt_monthRent.Text = "";
            cmb_driverRate.Text = "";

        }
```

Figure 4. 28 Admin logged Rent Details Form Code - part 2



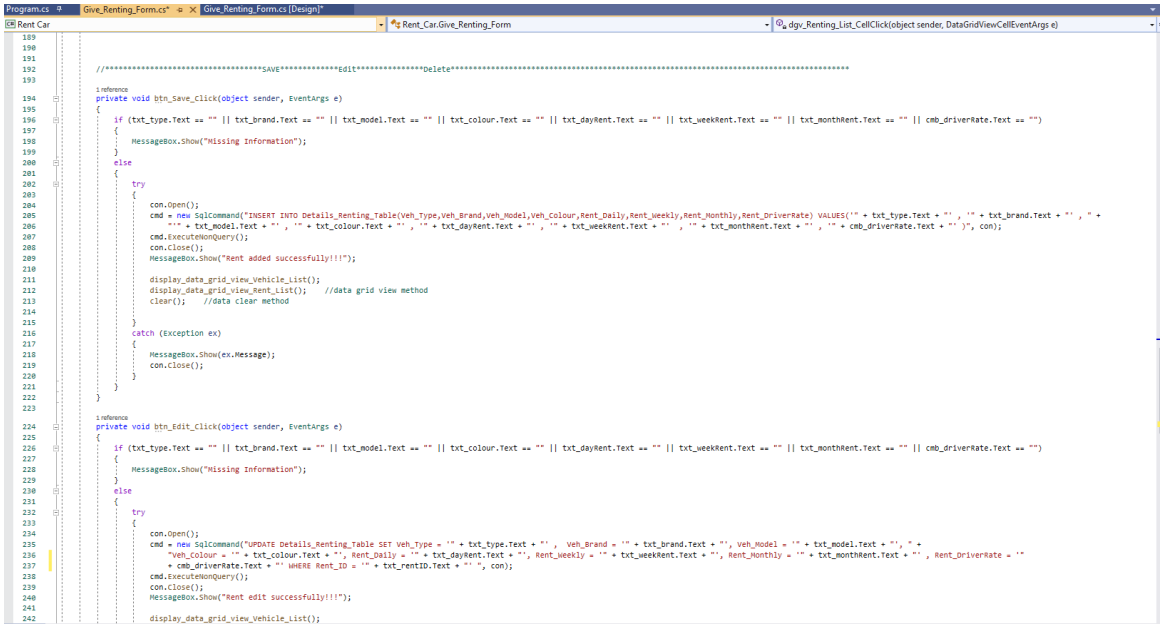
```
//FOR 1st DATA GRID VIEW
4 references
public void display_data_grid_view_Vehicle_List()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Details_Vehicle_Table", con);
        adpt.Fill(dt);
        dgv_Vehicles_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_Vehicles_List_CellClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_Vehicles_List.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Details_Vehicle_Table WHERE Veh_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_type.Text = DR1.GetValue(1).ToString();
        txt_brand.Text = DR1.GetValue(3).ToString();
        txt_model.Text = DR1.GetValue(4).ToString();
        txt_colour.Text = DR1.GetValue(5).ToString();

    }
    DR1.Close();
    con.Close();
}
```

Figure 4. 29 Admin logged Rent Details Form Code - part 3



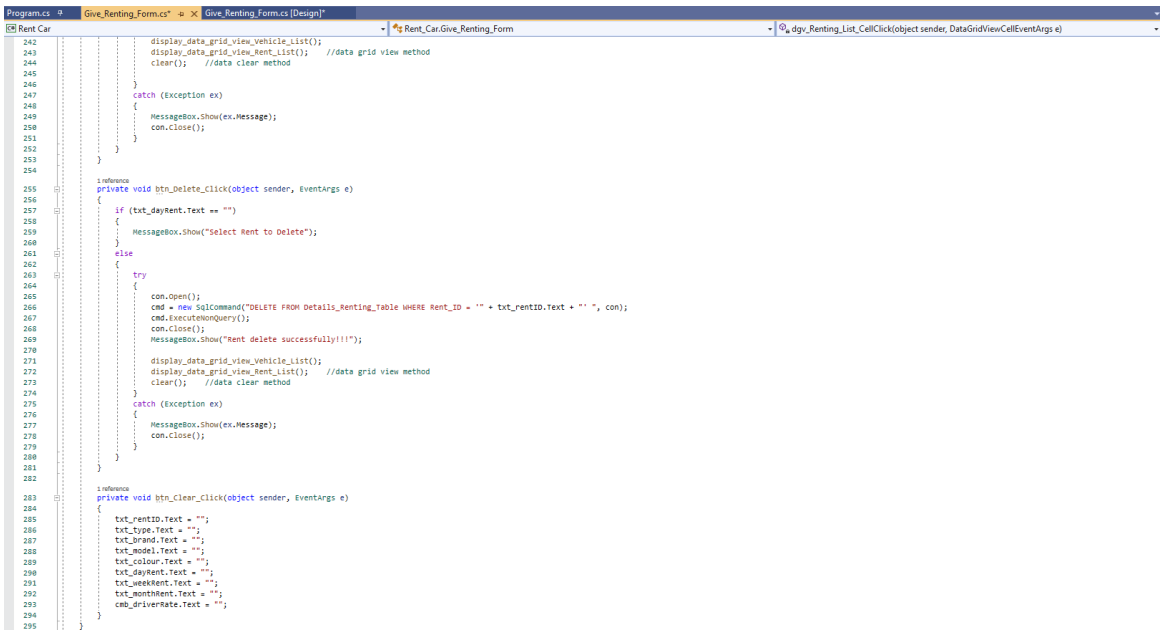
```
//FOR 2nd DATA GRID VIEW
4 references
public void display_data_grid_view_Rent_List()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Details_Renting_Table", con);
        adpt.Fill(dt);
        dgv_Renting_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_Renting_List_CellClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_Renting_List.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Details_Renting_Table WHERE Rent_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_rentID.Text = DR1.GetValue(0).ToString();
        txt_type.Text = DR1.GetValue(1).ToString();
        txt_brand.Text = DR1.GetValue(2).ToString();
        txt_model.Text = DR1.GetValue(3).ToString();
        txt_colour.Text = DR1.GetValue(4).ToString();
        txt_dayRent.Text = DR1.GetValue(5).ToString();
        txt_weekRent.Text = DR1.GetValue(6).ToString();
        txt_monthRent.Text = DR1.GetValue(7).ToString();
        cmb_driverRate.Text = DR1.GetValue(8).ToString();

    }
    DR1.Close();
    con.Close();
}
```

Figure 4. 30 Admin logged Rent Details Form Code - part 4



```
//**********************************SAVE************Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || txt_dayRent.Text == "" || txt_weekRent.Text == "" || txt_monthRent.Text == "" || cmb_driverRate.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Details_Renting_Table(Veh_Type,Veh_Brand,Veh_Model,Veh_Colour,Rent_Daily,Rent_Weekly,Rent_Monthly,Rent_DriverRate) VALUES('" + txt_type.Text + "' , '" + txt_brand.Text + "' , " +
                "'" + txt_model.Text + "' , '" + txt_colour.Text + "' , '" + txt_dayRent.Text + "' , '" + txt_weekRent.Text + "'  , '" + txt_monthRent.Text + "' , '" + cmb_driverRate.Text + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Rent added successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_Rent_List();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || txt_dayRent.Text == "" || txt_weekRent.Text == "" || txt_monthRent.Text == "" || cmb_driverRate.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("UPDATE Details_Renting_Table SET Veh_Type = '" + txt_type.Text + "' ,  Veh_Brand = '" + txt_brand.Text + "', Veh_Model = '" + txt_model.Text + "', " +
                "Veh_Colour = '" + txt_colour.Text + "', Rent_Daily = '" + txt_dayRent.Text + "', Rent_Weekly = '" + txt_weekRent.Text + "', Rent_Monthly = '" + txt_monthRent.Text + "' , Rent_DriverRate = '"
                + cmb_driverRate.Text + "' WHERE Rent_ID = '" + txt_rentID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Rent edit successfully!!!");

            display_data_grid_view_Vehicle_List();
```

Figure 4. 31 Admin logged Rent Details Form Code - part 5



```
            display_data_grid_view_Vehicle_List();
            display_data_grid_view_Rent_List();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_dayRent.Text == "")
    {
        MessageBox.Show("Select Rent to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Details_Renting_Table WHERE Rent_ID = '" + txt_rentID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Rent delete successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_Rent_List();    //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
    txt_rentID.Text = "";
    txt_type.Text = "";
    txt_brand.Text = "";
    txt_model.Text = "";
    txt_colour.Text = "";
    txt_dayRent.Text = "";
    txt_weekRent.Text = "";
    txt_monthRent.Text = "";
    cmb_driverRate.Text = "";
}
}
```

Figure 4. 32 Admin logged Rent Details Form Code - part 6

4.1.8 Admin logged One Day Hire Details Form Design and Code

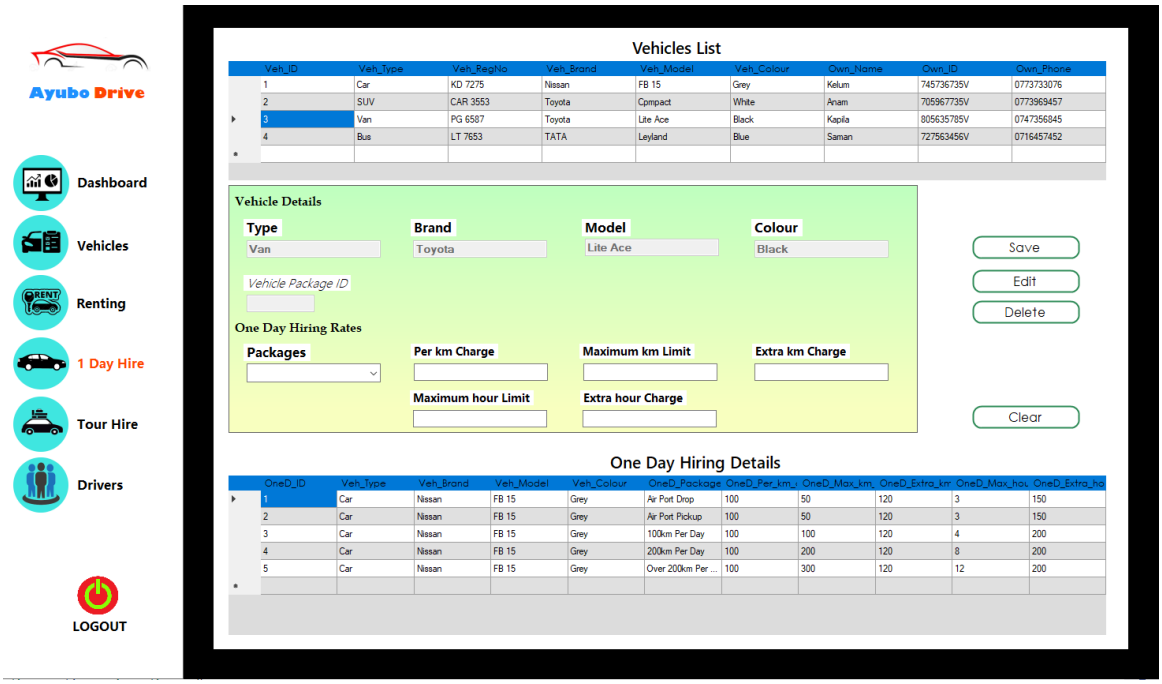
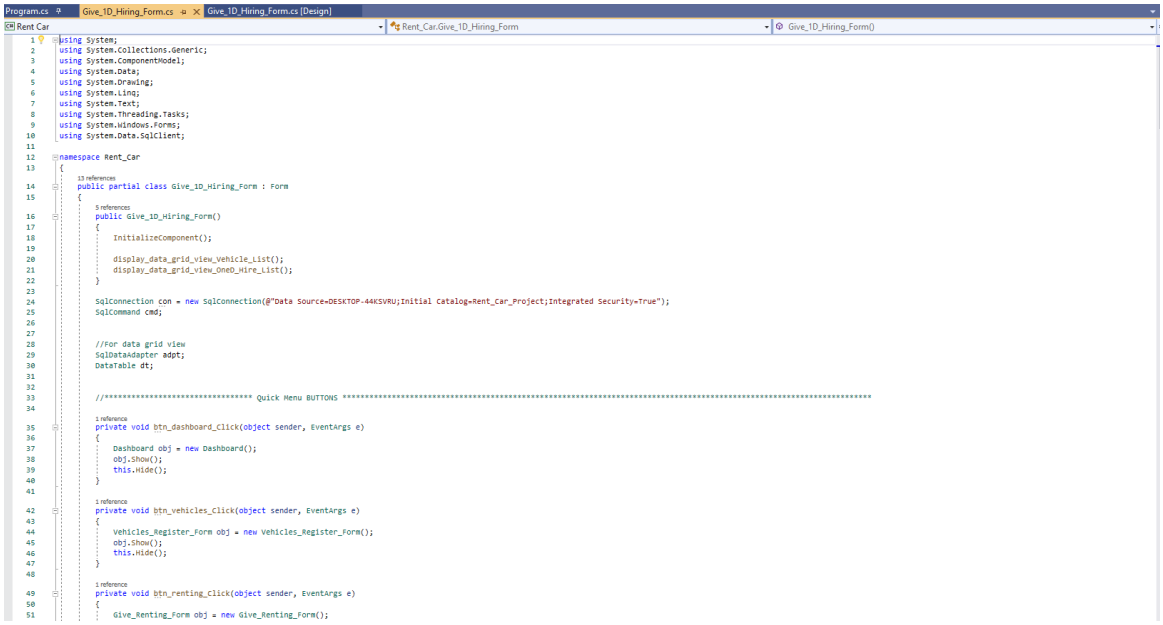
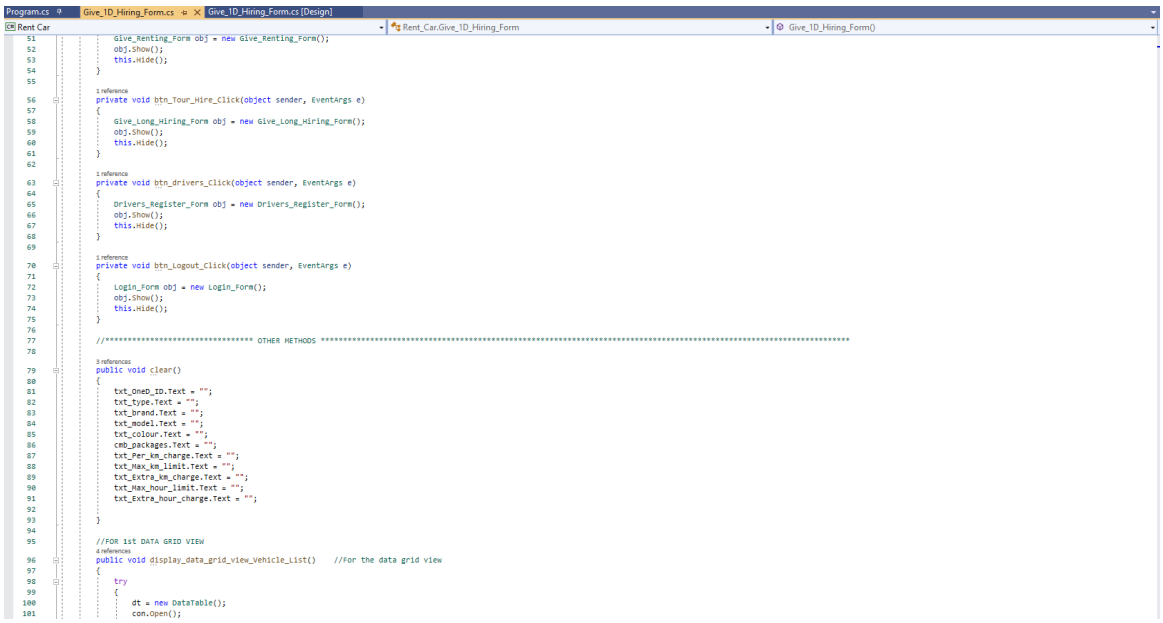


Figure 4. 33 Admin logged One Day Hire Details Form Design



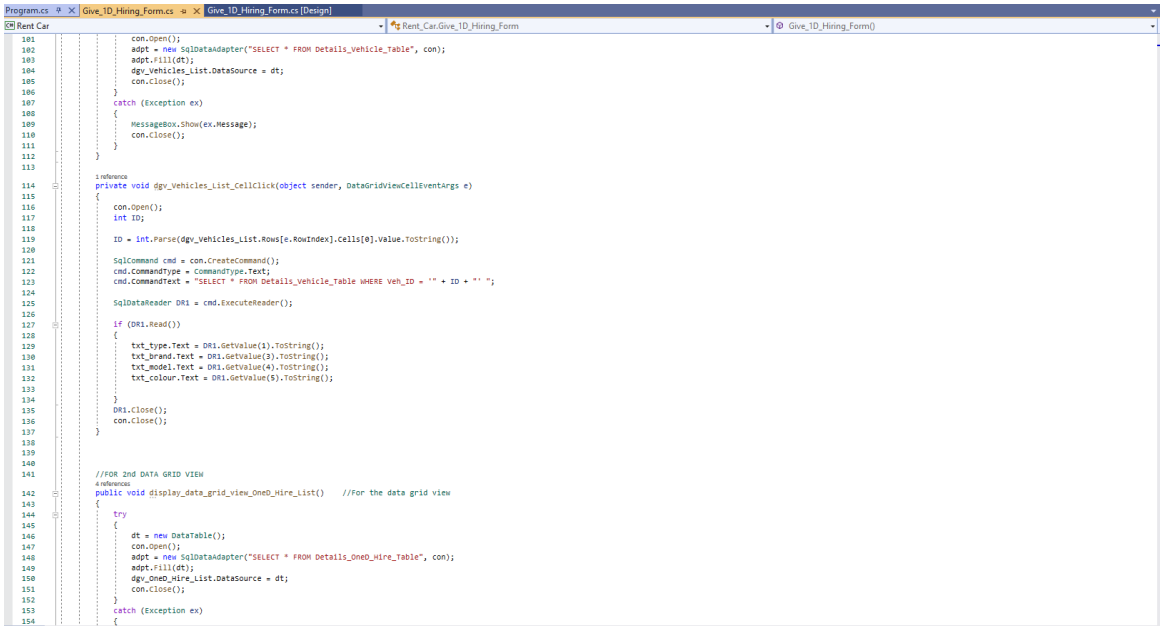
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    13 references
    public partial class Give_1D_Hiring_Form : Form
    {
        5 references
        public Give_1D_Hiring_Form()
        {
            InitializeComponent();

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_OneD_Hire_List();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************************

        1 reference
        private void btn_dashboard_Click(object sender, EventArgs e)
        {
            Dashboard obj = new Dashboard();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_vehicles_Click(object sender, EventArgs e)
        {
            Vehicles_Register_Form obj = new Vehicles_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_renting_Click(object sender, EventArgs e)
        {
            Give_Renting_Form obj = new Give_Renting_Form();
```

Figure 4. 34 Admin logged One Day Hire Details Form Code - part 1



```
            Give_Renting_Form obj = new Give_Renting_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Give_Long_Hiring_Form obj = new Give_Long_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_drivers_Click(object sender, EventArgs e)
        {
            Drivers_Register_Form obj = new Drivers_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        //******************************** OTHER METHODS ********************************************************************************************************************

        3 references
        public void clear()
        {
            txt_OneD_ID.Text = "";
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            cmb_packages.Text = "";
            txt_Per_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            txt_Max_hour_limit.Text = "";
            txt_Extra_hour_charge.Text = "";

        }

        //FOR 1st DATA GRID VIEW
        4 references
        public void display_data_grid_view_Vehicle_List()    //For the data grid view
        {
            try
            {
                dt = new DataTable();
                con.Open();
```

Figure 4. 35 Admin logged One Day Hire Details Form Code - part 2

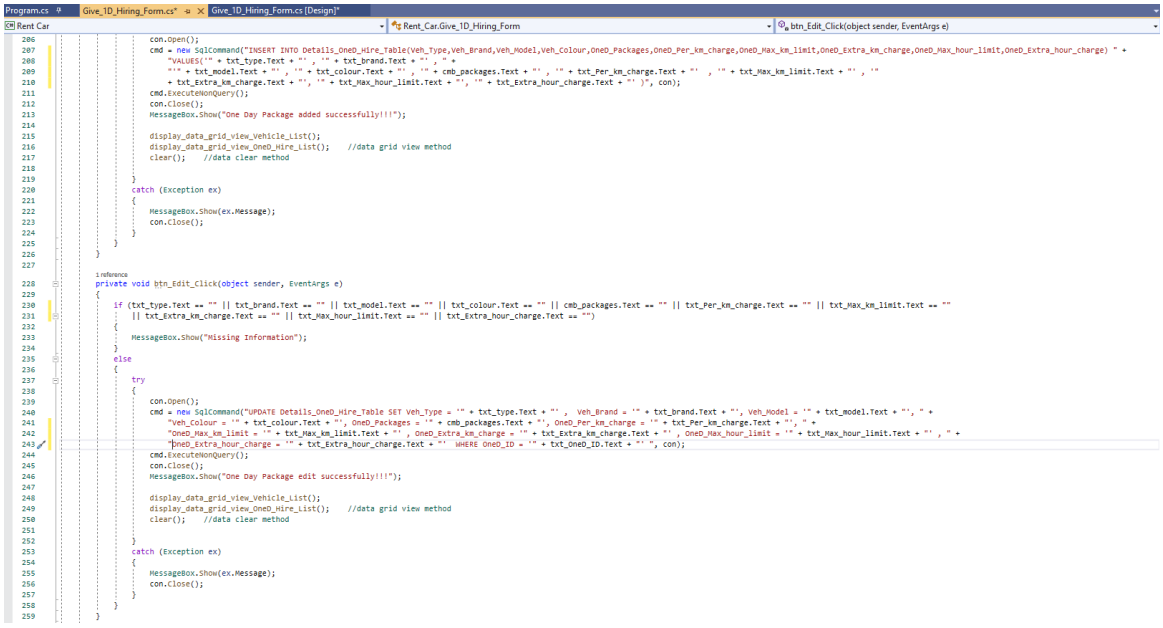


```
Program.cs | Give_1D_Hiring_Form.cs | Give_1D_Hiring_Form.cs [Design]
Rent Car | Rent_Car.Give_1D_Hiring_Form | Give_1D_Hiring_Form()

            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Details_Vehicle_Table", con);
            adpt.Fill(dt);
            dgv_Vehicles_List.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

    1 reference
    private void dgv_Vehicles_List_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_Vehicles_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Details_Vehicle_Table WHERE Veh_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            txt_type.Text = DR1.GetValue(1).ToString();
            txt_brand.Text = DR1.GetValue(2).ToString();
            txt_model.Text = DR1.GetValue(4).ToString();
            txt_colour.Text = DR1.GetValue(5).ToString();

        }
        DR1.Close();
        con.Close();
    }


    //FOR 2nd DATA GRID VIEW
    4 references
    public void display_data_grid_view_OneD_Hire_List()    //For the data grid view
    {
        try
        {
            dt = new DataTable();
            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Details_OneD_Hire_Table", con);
            adpt.Fill(dt);
            dgv_OneD_Hire_List.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
```

Figure 4. 36 Admin logged One Day Hire Details Form Code - part 3

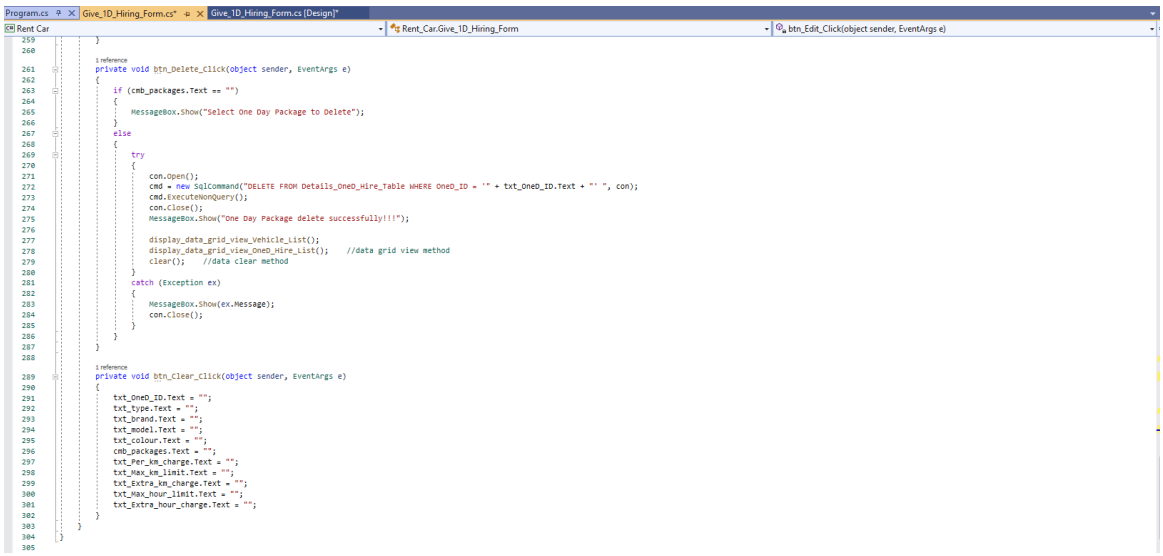
```
Program.cs | Give_1D_Hiring_Form.cs* | Give_1D_Hiring_Form.cs [Design]*
Rent Car | Rent_Car.Give_1D_Hiring_Form | btn_Edit_Click(object sender, EventArgs e)

        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

    1 reference
    private void dgv_OneD_Hire_List_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_OneD_Hire_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Details_OneD_Hire_Table WHERE OneD_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            txt_OneD_ID.Text = DR1.GetValue(0).ToString();
            txt_type.Text = DR1.GetValue(1).ToString();
            txt_brand.Text = DR1.GetValue(2).ToString();
            txt_model.Text = DR1.GetValue(3).ToString();
            txt_colour.Text = DR1.GetValue(4).ToString();
            cmb_packages.Text = DR1.GetValue(5).ToString();
            txt_Per_km_charge.Text = DR1.GetValue(6).ToString();
            txt_Max_km_limit.Text = DR1.GetValue(7).ToString();
            txt_Extra_km_charge.Text = DR1.GetValue(8).ToString();
            txt_Max_hour_limit.Text = DR1.GetValue(9).ToString();
            txt_Extra_hour_charge.Text = DR1.GetValue(10).ToString();

        }
        DR1.Close();
        con.Close();
    }


    //**********************************SAVE************Edit**************Delete*****************************************************************************************

    1 reference
    private void btn_Save_Click(object sender, EventArgs e)
    {
        if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || cmb_packages.Text == "" || txt_Per_km_charge.Text == ""
            || txt_Max_km_limit.Text == "" || txt_Extra_km_charge.Text == "" || txt_Max_hour_limit.Text == "" || txt_Extra_hour_charge.Text == "")
        {
            MessageBox.Show("Missing Information");
        }
        else
        {
            try
            {
                con.Open();
```

Figure 4. 37 Admin logged One Day Hire Details Form Code - part 4



```
            con.Open();
            cmd = new SqlCommand("INSERT INTO Details_OneD_Hire_Table(Veh_Type,Veh_Brand,Veh_Model,Veh_Colour,OneD_Packages,OneD_Per_km_charge,OneD_Max_km_limit,OneD_Extra_km_charge,OneD_Max_hour_limit,OneD_Extra_hour_charge) " +
                "VALUES('" + txt_type.Text + "' , '" + txt_brand.Text + "' , '" +
                "'" + txt_model.Text + "' , '" + txt_colour.Text + "' , '" + cmb_packages.Text + "' , '" + txt_Per_km_charge.Text + "'  , '" + txt_Max_km_limit.Text + "' , '"
                + txt_Extra_km_charge.Text + "', '" + txt_Max_hour_limit.Text + "', '" + txt_Extra_hour_charge.Text + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("One Day Package added successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_OneD_Hire_List();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || cmb_packages.Text == "" || txt_Per_km_charge.Text == "" || txt_Max_km_limit.Text == ""
        || txt_Extra_km_charge.Text == "" || txt_Max_hour_limit.Text == "" || txt_Extra_hour_charge.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("UPDATE Details_OneD_Hire_Table SET Veh_Type = '" + txt_type.Text + "' ,  Veh_Brand = '" + txt_brand.Text + "', Veh_Model = '" + txt_model.Text + "', " +
                "Veh_Colour = '" + txt_colour.Text + "', OneD_Packages = '" + cmb_packages.Text + "', OneD_Per_km_charge = '" + txt_Per_km_charge.Text + "', " +
                "OneD_Max_km_limit = '" + txt_Max_km_limit.Text + "' , OneD_Extra_km_charge = '" + txt_Extra_km_charge.Text + "' , OneD_Max_hour_limit = '" + txt_Max_hour_limit.Text + "' , " +
                "OneD_Extra_hour_charge = '" + txt_Extra_hour_charge.Text + "'  WHERE OneD_ID = '" + txt_OneD_ID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("One Day Package edit successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_OneD_Hire_List();    //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}
```

Figure 4. 38 Admin logged One Day Hire Details Form Code - part 5



```
    }

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (cmb_packages.Text == "")
    {
        MessageBox.Show("Select One Day Package to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Details_OneD_Hire_Table WHERE OneD_ID = '" + txt_OneD_ID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("One Day Package delete successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_OneD_Hire_List();    //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
    txt_OneD_ID.Text = "";
    txt_type.Text = "";
    txt_brand.Text = "";
    txt_model.Text = "";
    txt_colour.Text = "";
    cmb_packages.Text = "";
    txt_Per_km_charge.Text = "";
    txt_Max_km_limit.Text = "";
    txt_Extra_km_charge.Text = "";
    txt_Max_hour_limit.Text = "";
    txt_Extra_hour_charge.Text = "";
}
}
}
```

Figure 4. 39 Admin logged One Day Hire Details Form Code - part 6

4.1.9 Admin logged Long Tour Hire Details Form Design and Code

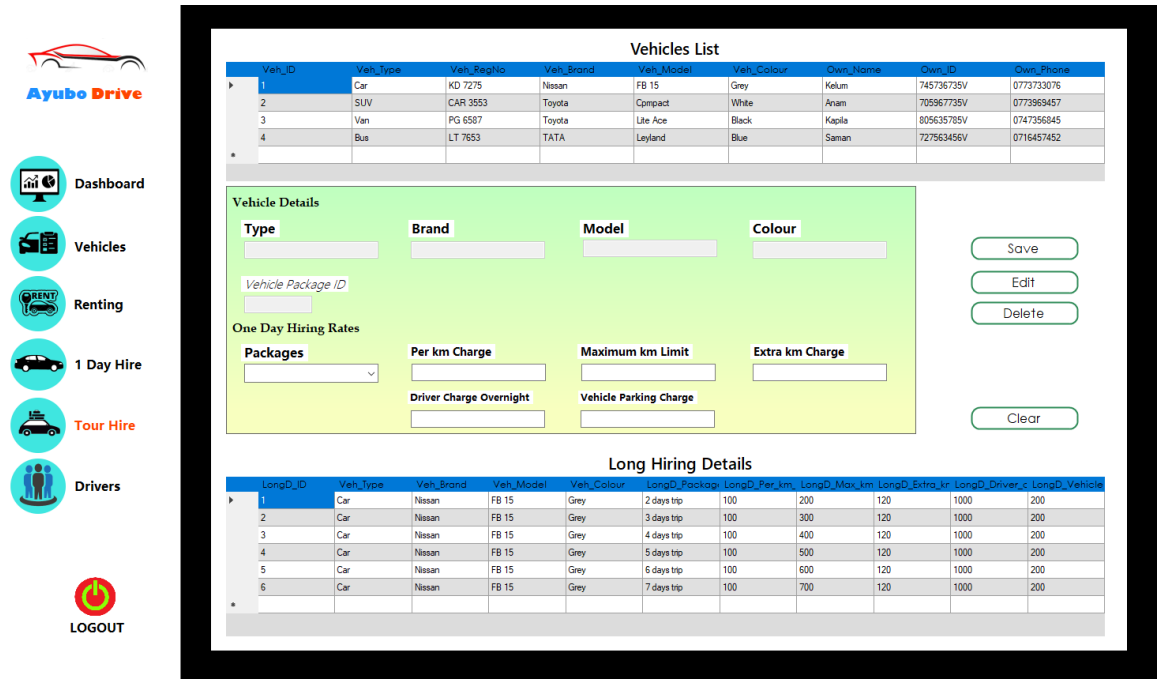
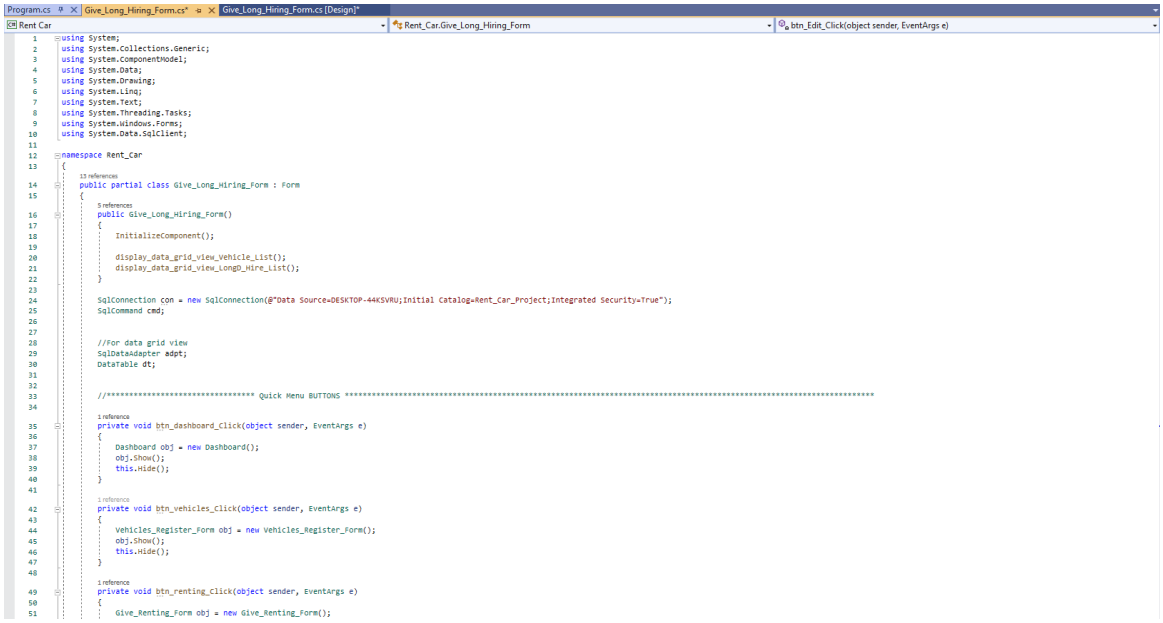
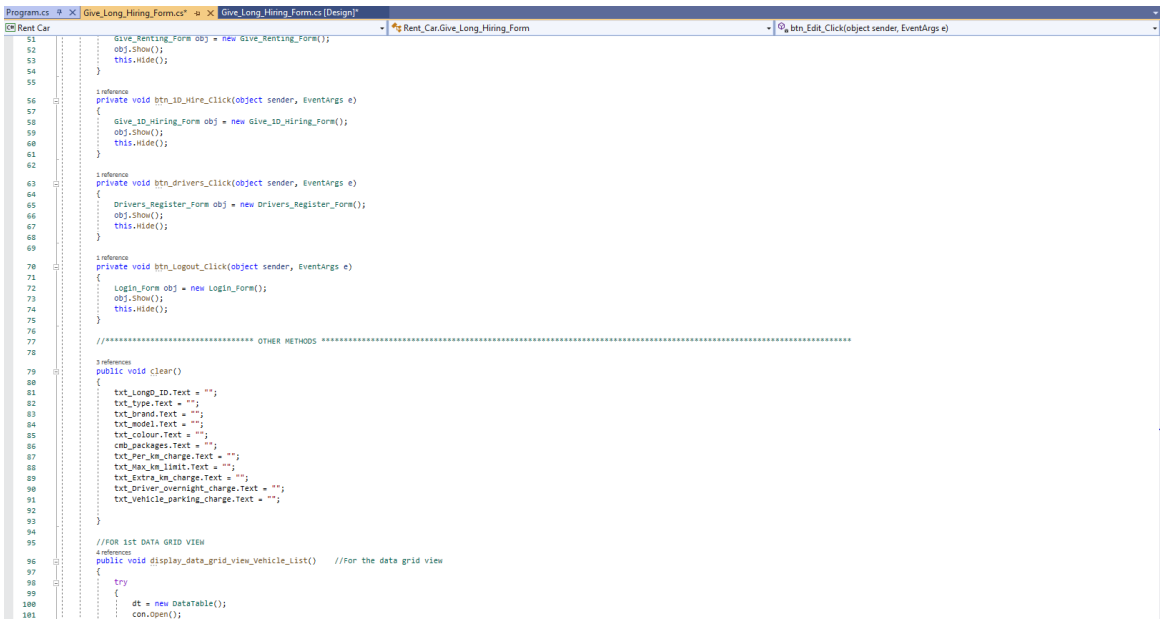


Figure 4. 40 Admin logged Long Tour Hire Details Form Design



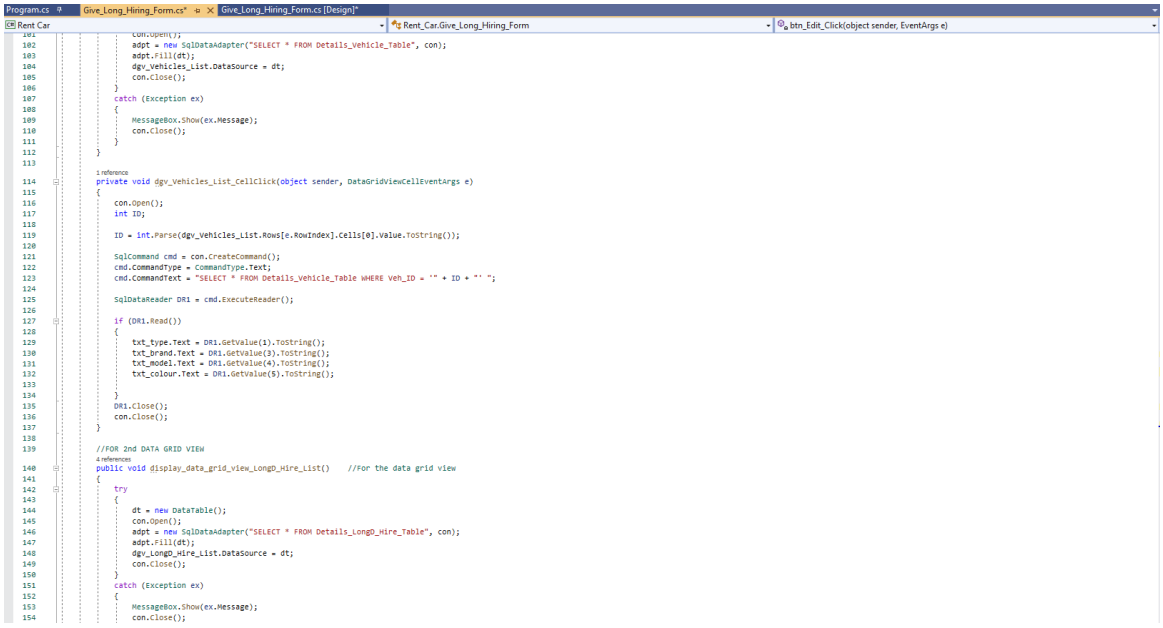
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    13 references
    public partial class Give_Long_Hiring_Form : Form
    {
        5 references
        public Give_Long_Hiring_Form()
        {
            InitializeComponent();

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_LongD_Hire_List();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44K5VRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_dashboard_Click(object sender, EventArgs e)
        {
            Dashboard obj = new Dashboard();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_vehicles_Click(object sender, EventArgs e)
        {
            Vehicles_Register_Form obj = new Vehicles_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_renting_Click(object sender, EventArgs e)
        {
            Give_Renting_Form obj = new Give_Renting_Form();
```

Figure 4. 41 Admin logged Long Tour Hire Details Form Code - part 1



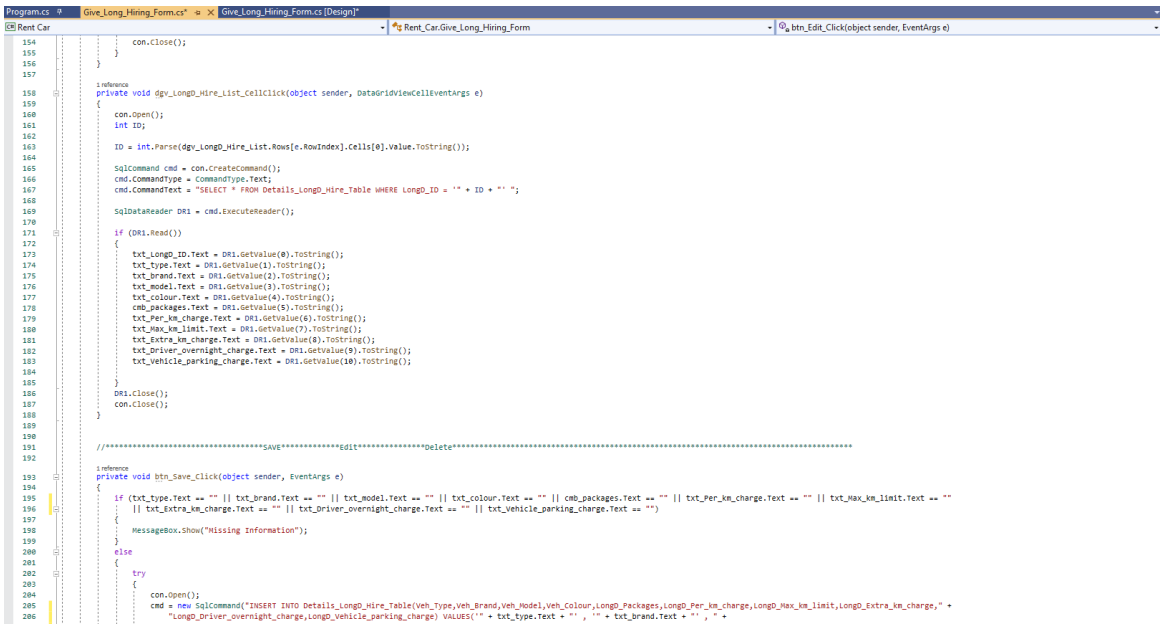
```
            Give_Renting_Form obj = new Give_Renting_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_drivers_Click(object sender, EventArgs e)
        {
            Drivers_Register_Form obj = new Drivers_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            LogIn_Form obj = new LogIn_Form();
            obj.Show();
            this.Hide();
        }

        //******************************** OTHER METHODS ***********************************************************************************************************

        3 references
        public void clear()
        {
            txt_LongD_ID.Text = "";
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            cmb_packages.Text = "";
            txt_Per_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            txt_Driver_overnight_charge.Text = "";
            txt_Vehicle_parking_charge.Text = "";

        }

        //FOR 1st DATA GRID VIEW
        4 references
        public void display_data_grid_view_Vehicle_List()    //For the data grid view
        {
            try
            {
                dt = new DataTable();
                con.Open();
```

Figure 4. 42 Admin logged Long Tour Hire Details Form Code - part 2



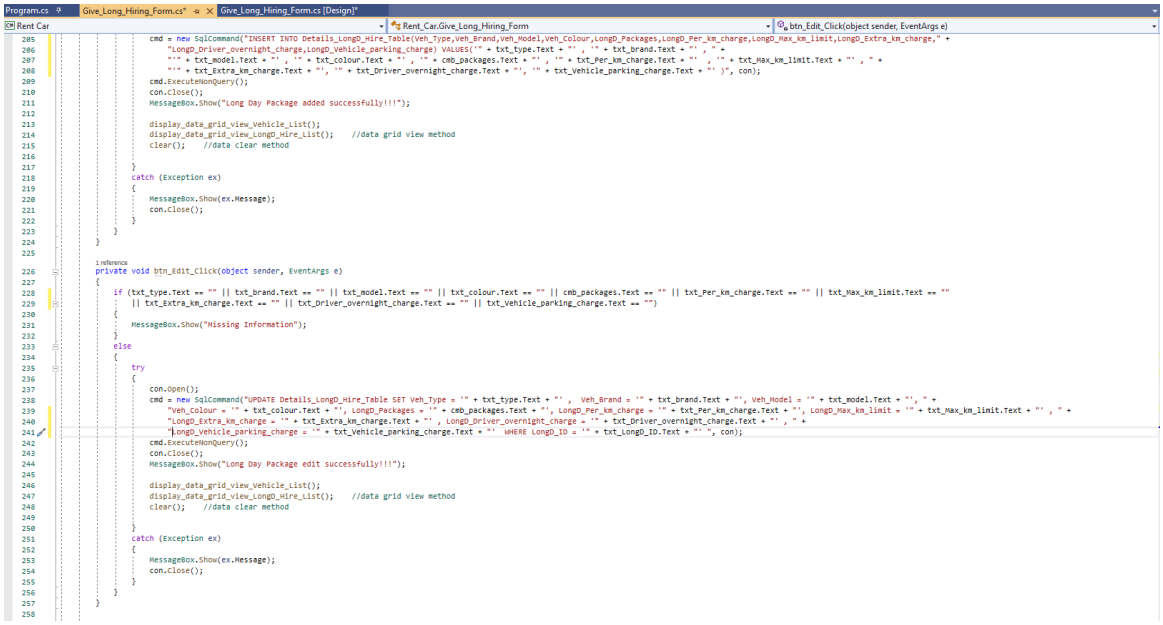
```
Program.cs | Give_Long_Hiring_Form.cs* | Give_Long_Hiring_Form.cs [Design]*
Rent Car | Rent_Car.Give_Long_Hiring_Form | btn_Edit_Click(object sender, EventArgs e)

            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Details_Vehicle_Table", con);
            adpt.Fill(dt);
            dgv_Vehicles_List.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

    1 reference
    private void dgv_Vehicles_List_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_Vehicles_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Details_Vehicle_Table WHERE Veh_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            txt_type.Text = DR1.GetValue(1).ToString();
            txt_brand.Text = DR1.GetValue(3).ToString();
            txt_model.Text = DR1.GetValue(4).ToString();
            txt_colour.Text = DR1.GetValue(5).ToString();

        }
        DR1.Close();
        con.Close();
    }

    //FOR 2nd DATA GRID VIEW
    4 references
    public void display_data_grid_view_LongD_Hire_List()    //For the data grid view
    {
        try
        {
            dt = new DataTable();
            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Details_LongD_Hire_Table", con);
            adpt.Fill(dt);
            dgv_LongD_Hire_List.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
```

Figure 4. 43 Admin logged Long Tour Hire Details Form Code - part 3



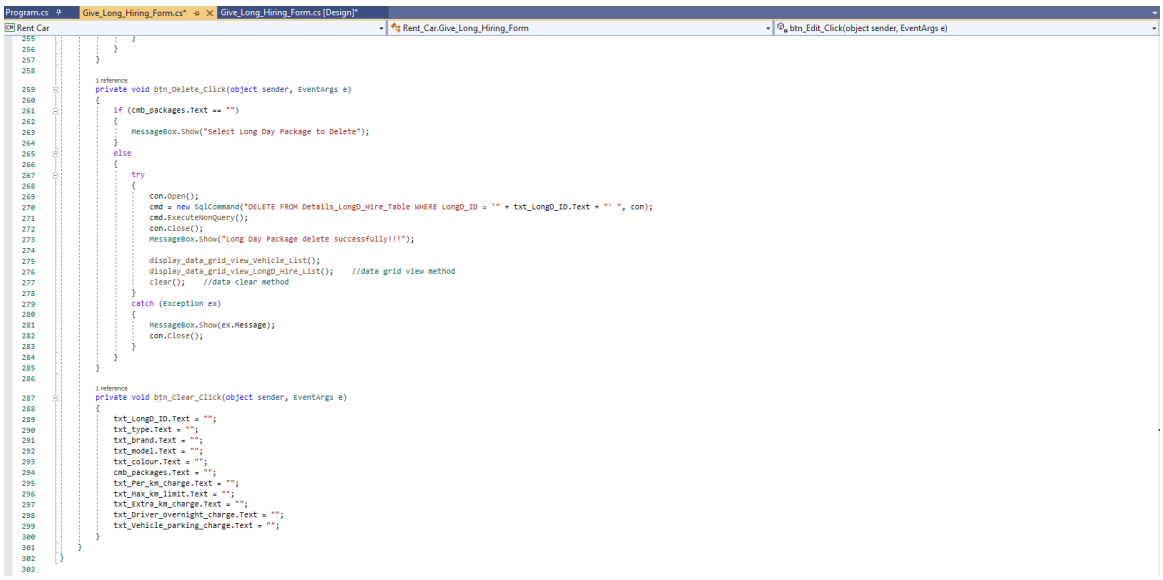
```
Program.cs | Give_Long_Hiring_Form.cs* | Give_Long_Hiring_Form.cs [Design]*
Rent Car | Rent_Car.Give_Long_Hiring_Form | btn_Edit_Click(object sender, EventArgs e)

            con.Close();
        }
    }

    1 reference
    private void dgv_LongD_Hire_List_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_LongD_Hire_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Details_LongD_Hire_Table WHERE LongD_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            txt_LongD_ID.Text = DR1.GetValue(0).ToString();
            txt_type.Text = DR1.GetValue(1).ToString();
            txt_brand.Text = DR1.GetValue(2).ToString();
            txt_model.Text = DR1.GetValue(3).ToString();
            txt_colour.Text = DR1.GetValue(4).ToString();
            cmb_packages.Text = DR1.GetValue(5).ToString();
            txt_Per_km_charge.Text = DR1.GetValue(6).ToString();
            txt_Max_km_limit.Text = DR1.GetValue(7).ToString();
            txt_Extra_km_charge.Text = DR1.GetValue(8).ToString();
            txt_Driver_overnight_charge.Text = DR1.GetValue(9).ToString();
            txt_Vehicle_parking_charge.Text = DR1.GetValue(10).ToString();

        }
        DR1.Close();
        con.Close();
    }

    //**********************************SAVE************Edit**************Delete*****************************************************************************************

    1 reference
    private void btn_Save_Click(object sender, EventArgs e)
    {
        if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || cmb_packages.Text == "" || txt_Per_km_charge.Text == "" || txt_Max_km_limit.Text == ""
            || txt_Extra_km_charge.Text == "" || txt_Driver_overnight_charge.Text == "" || txt_Vehicle_parking_charge.Text == "")
        {
            MessageBox.Show("Missing Information");
        }
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("INSERT INTO Details_LongD_Hire_Table(Veh_Type,Veh_Brand,Veh_Model,Veh_Colour,LongD_Packages,LongD_Per_km_charge,LongD_Max_km_limit,LongD_Extra_km_charge," +
                    "LongD_Driver_overnight_charge,LongD_Vehicle_parking_charge) VALUES('" + txt_type.Text + "' , '" + txt_brand.Text + "' , " +
```

Figure 4. 44 Admin logged Long Tour Hire Details Form Code - part 4



```
            cmd = new SqlCommand("INSERT INTO Details_LongD_Hire_Table(Veh_Type,Veh_Brand,Veh_Model,Veh_Colour,LongD_Packages,LongD_Per_km_charge,LongD_Max_km_limit,LongD_Extra_km_charge," +
                "LongD_Driver_overnight_charge,LongD_Vehicle_parking_charge) VALUES('" + txt_type.Text + "' , '" + txt_brand.Text + "' , " +
                "'" + txt_model.Text + "' , '" + txt_colour.Text + "' , " + cmb_packages.Text + " , '" + txt_Per_km_charge.Text + "' , " + txt_Max_km_limit.Text + " , " +
                "'" + txt_Extra_km_charge.Text + "', '" + txt_Driver_overnight_charge.Text + "', '" + txt_Vehicle_parking_charge.Text + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Long Day Package added successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_LongD_Hire_List();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_type.Text == "" || txt_brand.Text == "" || txt_model.Text == "" || txt_colour.Text == "" || cmb_packages.Text == "" || txt_Per_km_charge.Text == "" || txt_Max_km_limit.Text == ""
        || txt_Extra_km_charge.Text == "" || txt_Driver_overnight_charge.Text == "" || txt_Vehicle_parking_charge.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("UPDATE Details_LongD_Hire_Table SET Veh_Type = '" + txt_type.Text + "' ,  Veh_Brand = '" + txt_brand.Text + "', Veh_Model = '" + txt_model.Text + "', " +
                "Veh_Colour = '" + txt_colour.Text + "', LongD_Packages = '" + cmb_packages.Text + "', LongD_Per_km_charge = '" + txt_Per_km_charge.Text + "', LongD_Max_km_limit = '" + txt_Max_km_limit.Text + "' , " +
                "LongD_Extra_km_charge = '" + txt_Extra_km_charge.Text + "' , LongD_Driver_overnight_charge = '" + txt_Driver_overnight_charge.Text + "' , " +
                "LongD_Vehicle_parking_charge = '" + txt_Vehicle_parking_charge.Text + "'  WHERE LongD_ID = '" + txt_LongD_ID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Long Day Package edit successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_LongD_Hire_List();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}
```

Figure 4. 45 Admin logged Long Tour Hire Details Form Code - part 5



```
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (cmb_packages.Text == "")
    {
        MessageBox.Show("Select Long Day Package to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Details_LongD_Hire_Table WHERE LongD_ID = '" + txt_LongD_ID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Long Day Package delete successfully!!!");

            display_data_grid_view_Vehicle_List();
            display_data_grid_view_LongD_Hire_List();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
    txt_LongD_ID.Text = "";
    txt_type.Text = "";
    txt_brand.Text = "";
    txt_model.Text = "";
    txt_colour.Text = "";
    cmb_packages.Text = "";
    txt_Per_km_charge.Text = "";
    txt_Max_km_limit.Text = "";
    txt_Extra_km_charge.Text = "";
    txt_Driver_overnight_charge.Text = "";
    txt_Vehicle_parking_charge.Text = "";
}
}
}
```

Figure 4. 46 Admin logged Long Tour Hire Details Form Code - part 6

4.1.10 Admin logged Driver Details Form Design and Code

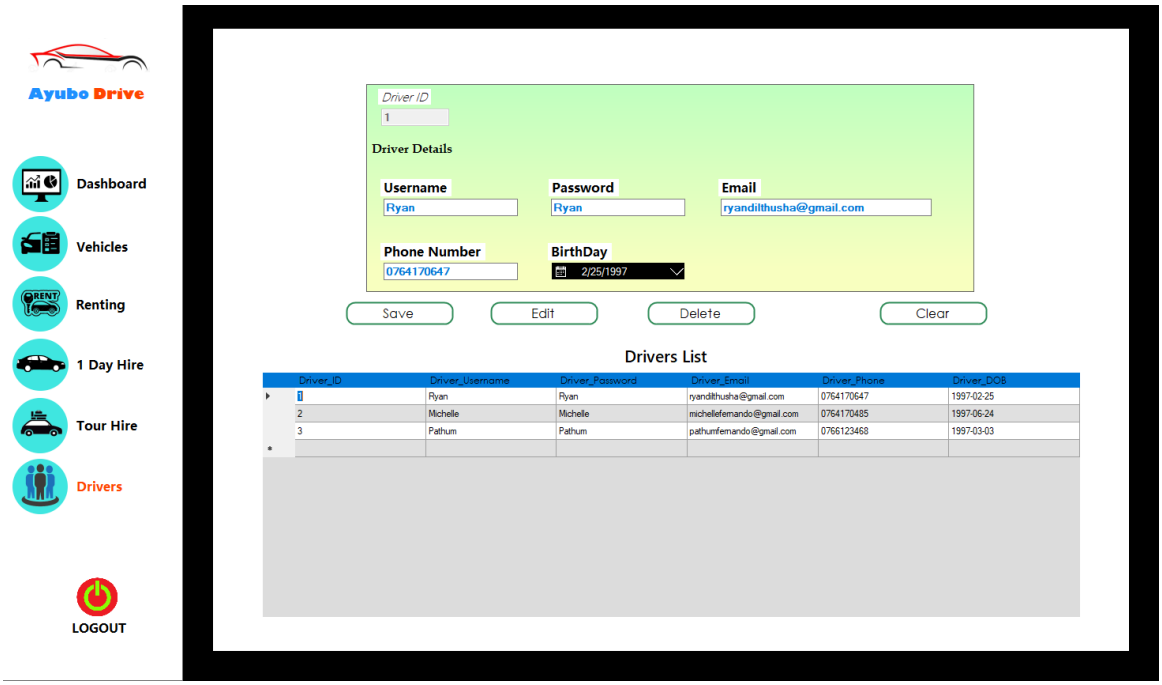
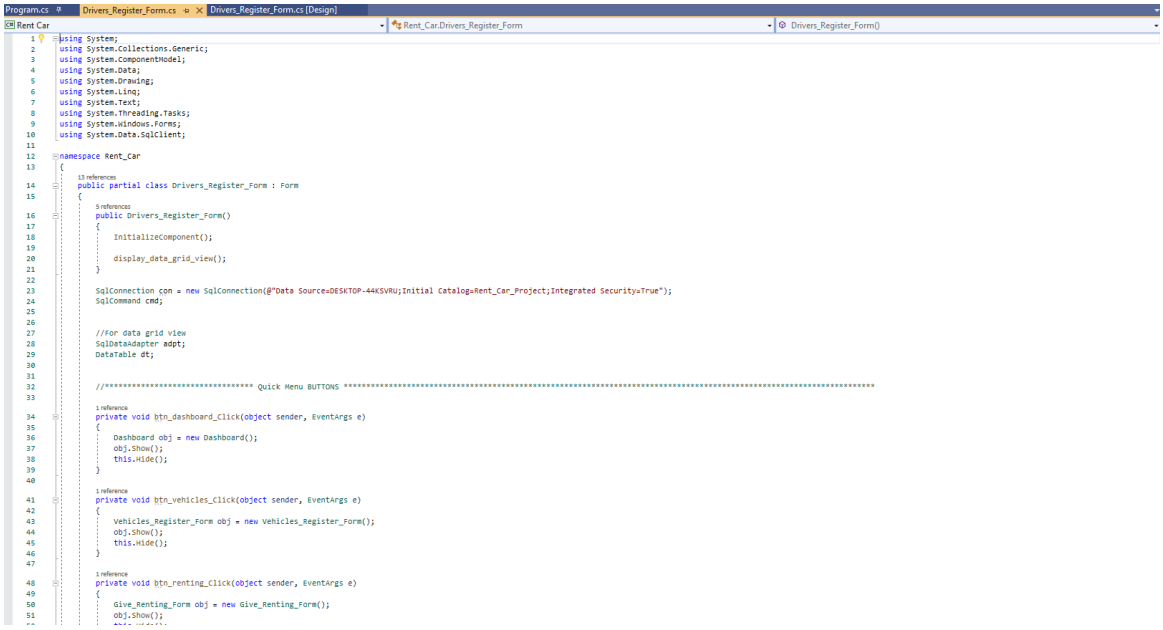
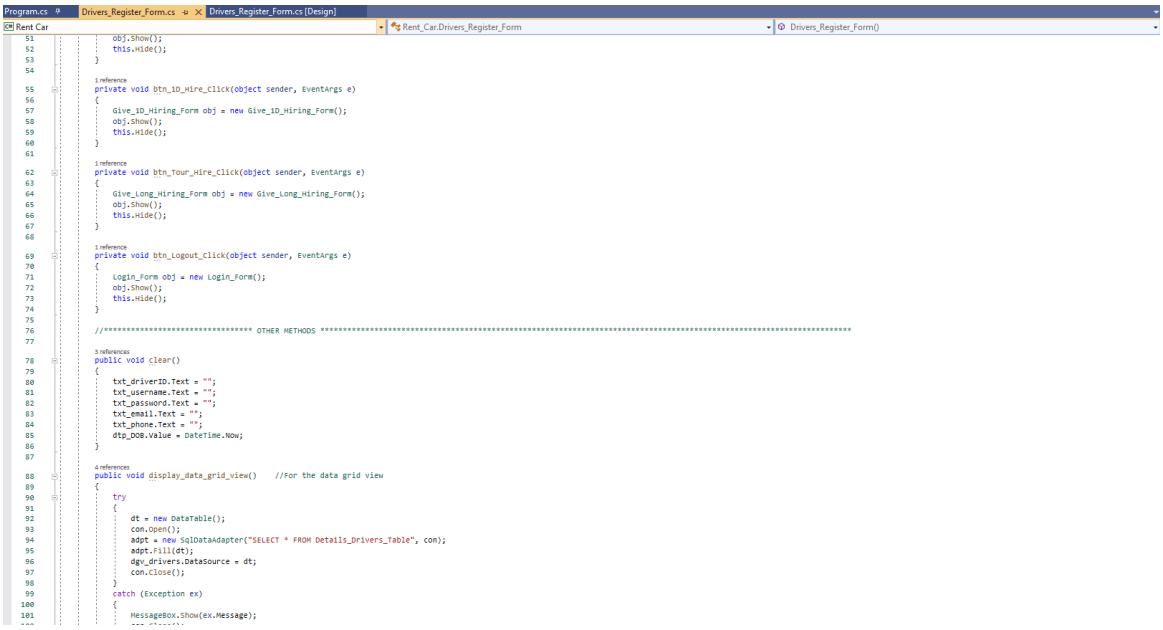


Figure 4. 47 Admin logged Driver Details Form Design



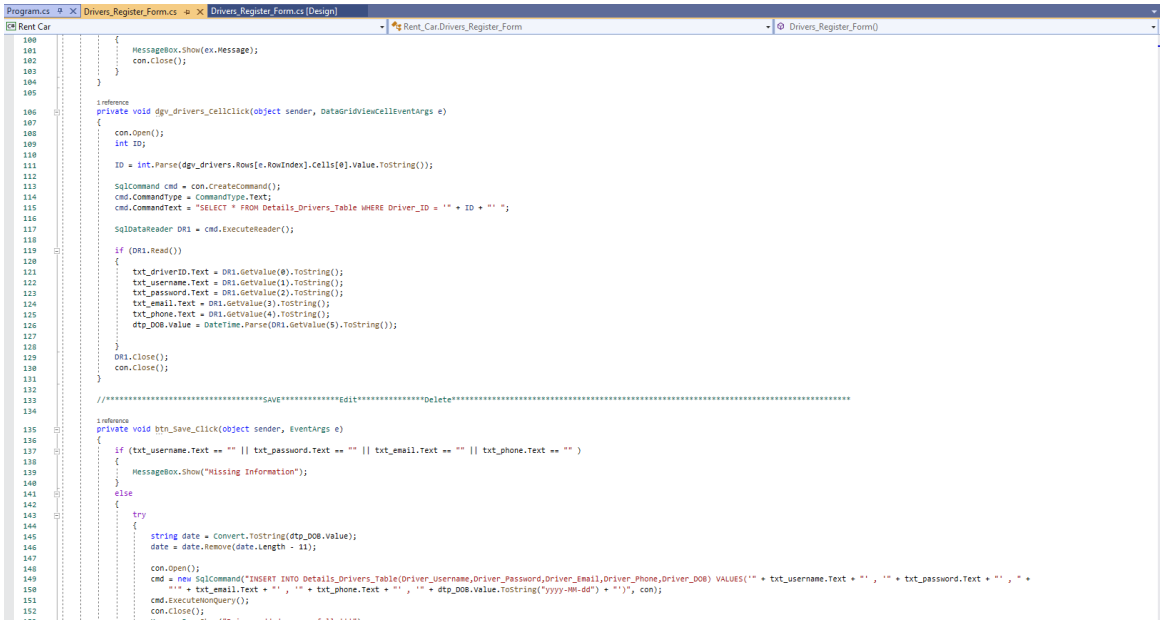
```
Program.cs | Drivers_Register_Form.cs | Drivers_Register_Form.cs [Design]
Rent Car | Rent_Car.Drivers_Register_Form | Drivers_Register_Form()

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

namespace Rent_Car
{
    13 references
    public partial class Drivers_Register_Form : Form
    {
        5 references
        public Drivers_Register_Form()
        {
            InitializeComponent();

            display_data_grid_view();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************************

        1 reference
        private void btn_dashboard_Click(object sender, EventArgs e)
        {
            Dashboard obj = new Dashboard();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_vehicles_Click(object sender, EventArgs e)
        {
            Vehicles_Register_Form obj = new Vehicles_Register_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_renting_Click(object sender, EventArgs e)
        {
            Give_Renting_Form obj = new Give_Renting_Form();
            obj.Show();
```

Figure 4. 48 Admin logged Driver Details Form Code - part 1



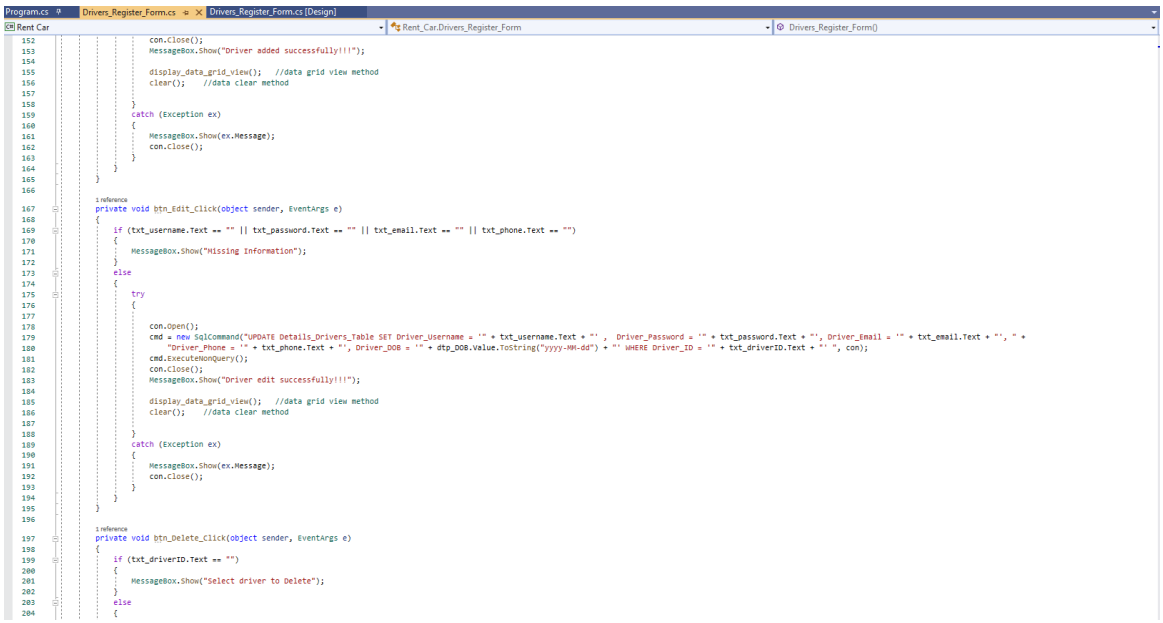
```
Program.cs | Drivers_Register_Form.cs | Drivers_Register_Form.cs [Design]
Rent Car | Rent_Car.Drivers_Register_Form | Drivers_Register_Form()

            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            Give_1D_Hiring_Form obj = new Give_1D_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Give_Long_Hiring_Form obj = new Give_Long_Hiring_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        //******************************** OTHER METHODS ********************************************************************************************************************

        3 references
        public void clear()
        {
            txt_driverID.Text = "";
            txt_username.Text = "";
            txt_password.Text = "";
            txt_email.Text = "";
            txt_phone.Text = "";
            dtp_DOB.Value = DateTime.Now;
        }

        4 references
        public void display_data_grid_view()    //For the data grid view
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Details_Drivers_Table", con);
                adpt.Fill(dt);
                dgv_drivers.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
```

Figure 4. 49 Admin logged Driver Details Form Code - part 2



```
Program.cs | Drivers_Register_Form.cs | Drivers_Register_Form.cs [Design]
Rent Car | Rent_Car.Drivers_Register_Form | Drivers_Register_Form()

            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_drivers_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_drivers.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Details_Drivers_Table WHERE Driver_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                txt_driverID.Text = DR1.GetValue(0).ToString();
                txt_username.Text = DR1.GetValue(1).ToString();
                txt_password.Text = DR1.GetValue(2).ToString();
                txt_email.Text = DR1.GetValue(3).ToString();
                txt_phone.Text = DR1.GetValue(4).ToString();
                dtp_DOB.Value = DateTime.Parse(DR1.GetValue(5).ToString());

            }
            DR1.Close();
            con.Close();
        }

        //**********************************SAVE************Edit**************Delete*****************************************************************************************

        1 reference
        private void btn_Save_Click(object sender, EventArgs e)
        {
            if (txt_username.Text == "" || txt_password.Text == "" || txt_email.Text == "" || txt_phone.Text == "" )
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {
                    string date = Convert.ToString(dtp_DOB.Value);
                    date = date.Remove(date.Length - 11);

                    con.Open();
                    cmd = new SqlCommand("INSERT INTO Details_Drivers_Table(Driver_Username,Driver_Password,Driver_Email,Driver_Phone,Driver_DOB) VALUES('" + txt_username.Text + "' , '" + txt_password.Text + "' , " +
                        "'" + txt_email.Text + "' , '" + txt_phone.Text + "' , '" + dtp_DOB.Value.ToString("yyyy-MM-dd") + "')", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
```

Figure 4. 50 Admin logged Driver Details Form Code - part 3



```
Program.cs | Drivers_Register_Form.cs | Drivers_Register_Form.cs [Design]
Rent Car | Rent_Car.Drivers_Register_Form | Drivers_Register_Form()

                    con.Close();
                    MessageBox.Show("Driver added successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Edit_Click(object sender, EventArgs e)
        {
            if (txt_username.Text == "" || txt_password.Text == "" || txt_email.Text == "" || txt_phone.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Details_Drivers_Table SET Driver_Username = '" + txt_username.Text + "' ,  Driver_Password = '" + txt_password.Text + "', Driver_Email = '" + txt_email.Text + "', " +
                        "Driver_Phone = '" + txt_phone.Text + "', Driver_DOB = '" + dtp_DOB.Value.ToString("yyyy-MM-dd") + "' WHERE Driver_ID = '" + txt_driverID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Driver edit successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_driverID.Text == "")
            {
                MessageBox.Show("Select driver to Delete");
            }
            else
            {
```

Figure 4. 51 Admin logged Driver Details Form Code - part 4

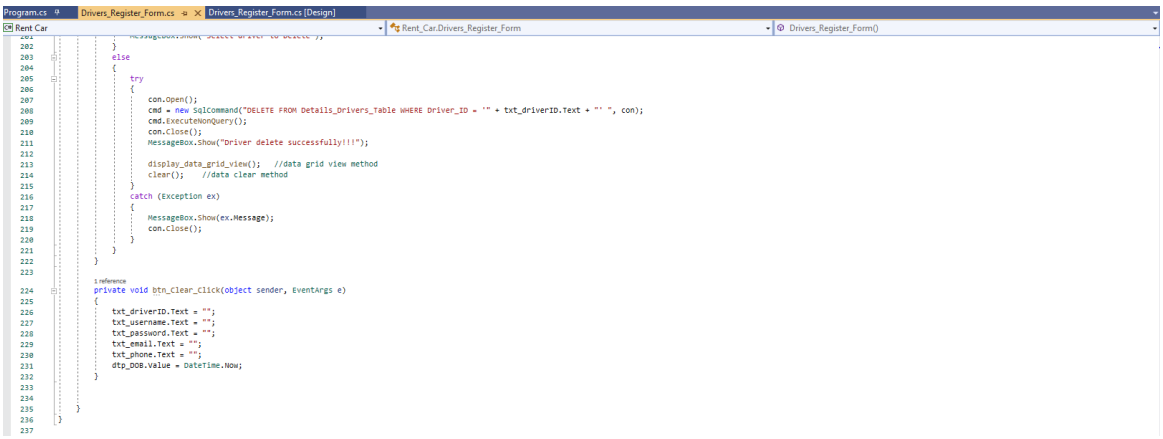


Figure 4. 52 Admin logged Driver Details Form Code - part 5

4.1.11 Driver logged Rent Vehicle Form Design and Code

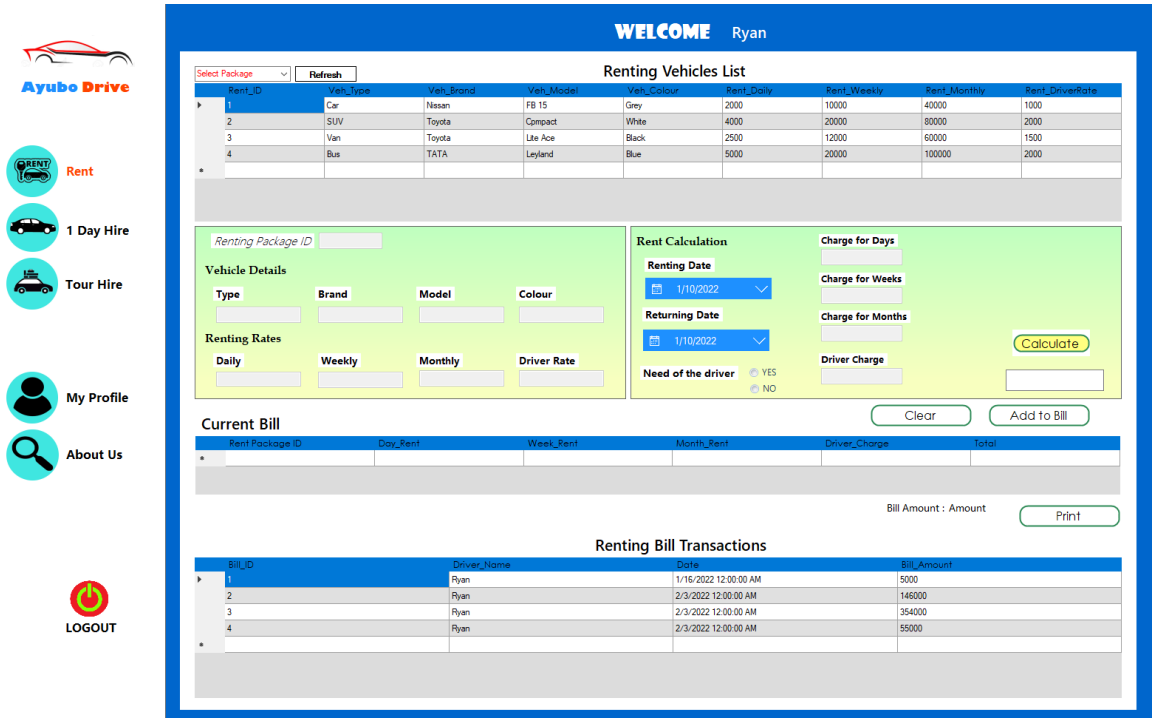
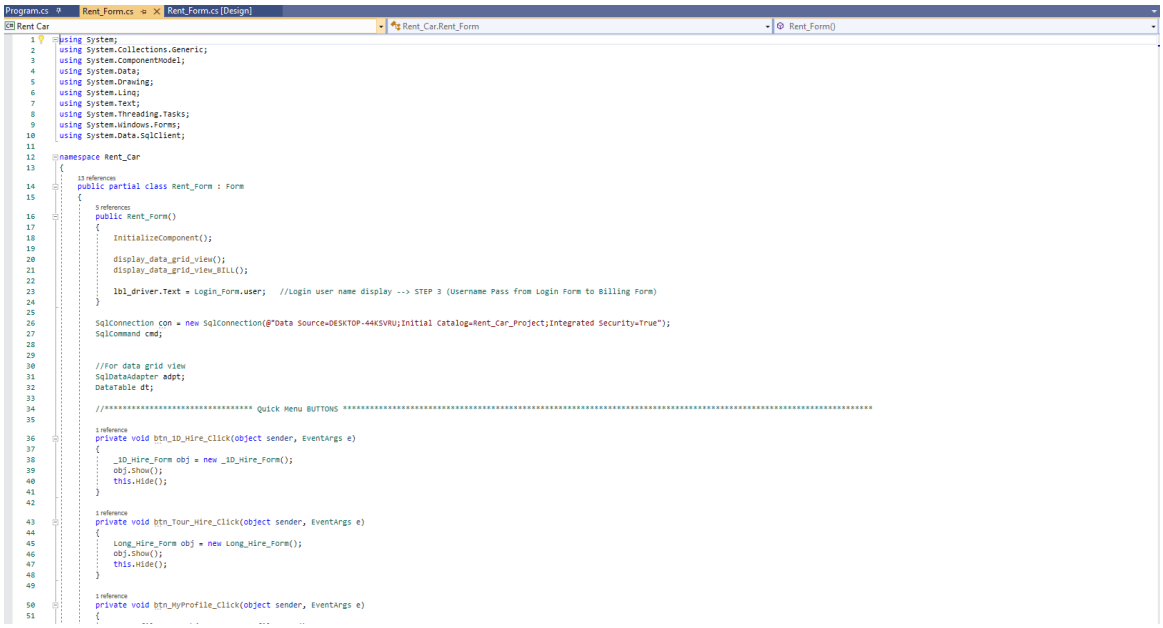
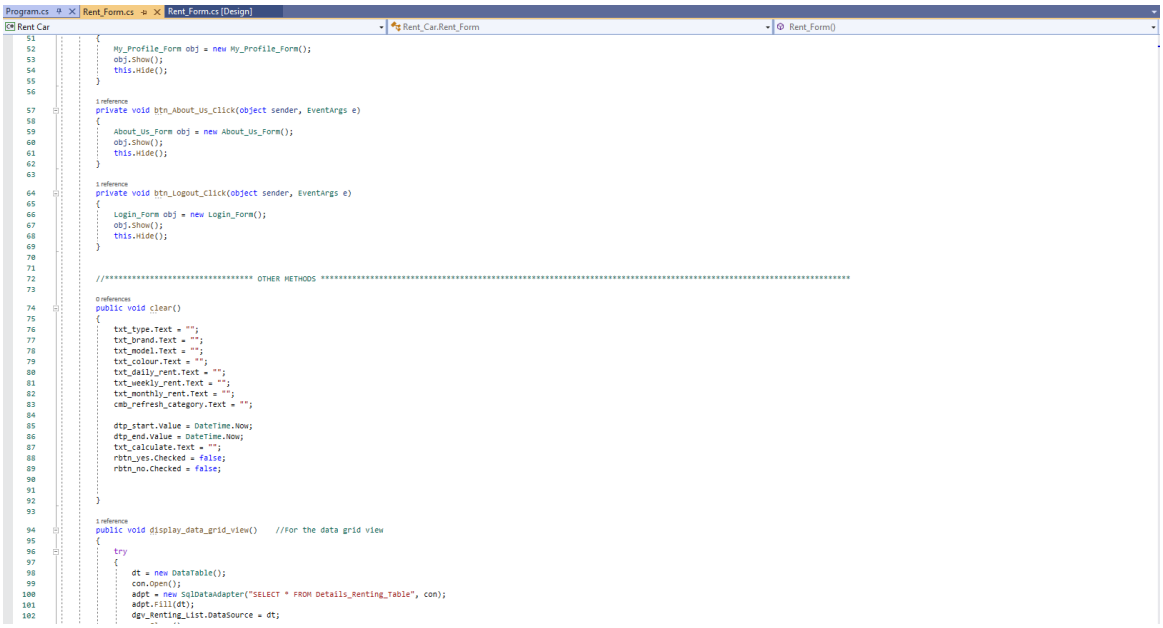


Figure 4. 53 Driver logged Rent Vehicle Form Design



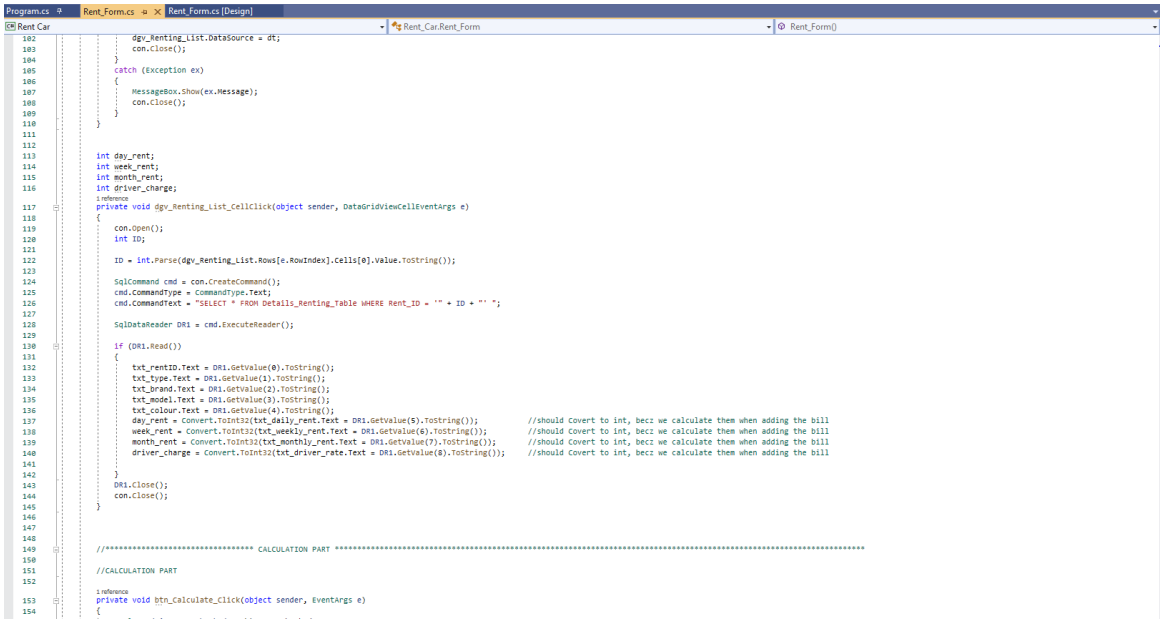
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    13 references
    public partial class Rent_Form : Form
    {
        5 references
        public Rent_Form()
        {
            InitializeComponent();

            display_data_grid_view();
            display_data_grid_view_BILL();

            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;

        //******************************** Quick Menu BUTTONS ********************************************************************************************************************

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            _1D_Hire_Form obj = new _1D_Hire_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Long_Hire_Form obj = new Long_Hire_Form();
            obj.Show();
            this.Hide();
        }

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

Figure 4. 54 Driver logged Rent Vehicle Form Code - part 1



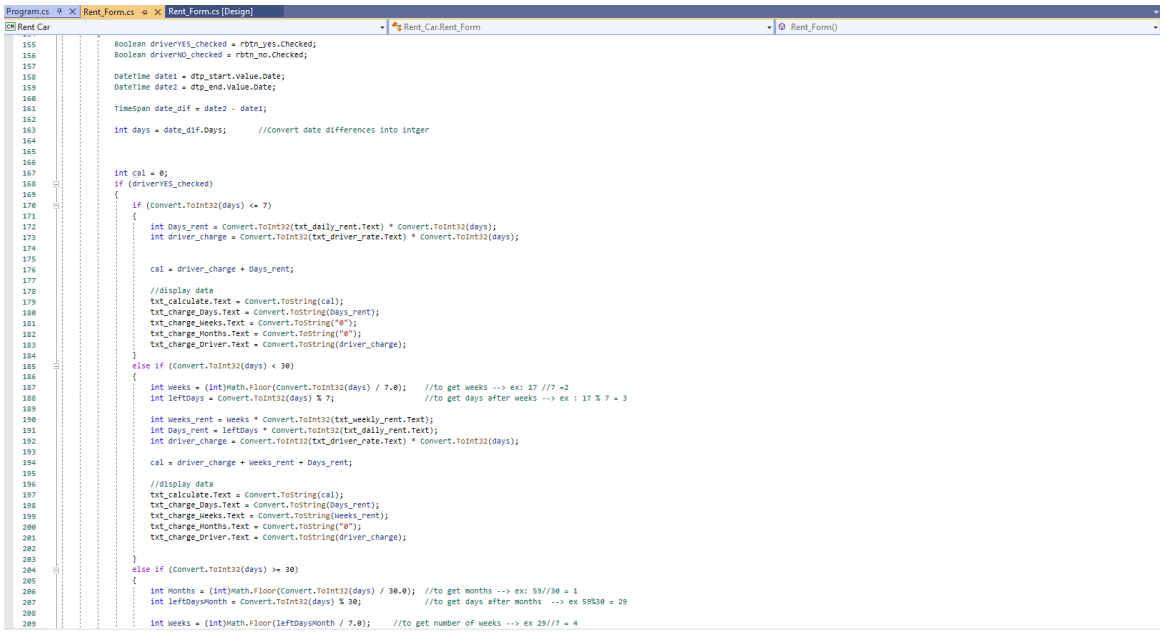
```
        {
            My_Profile_Form obj = new My_Profile_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_About_Us_Click(object sender, EventArgs e)
        {
            About_Us_Form obj = new About_Us_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        //******************************** OTHER METHODS ********************************************************************************************************************

        0 references
        public void clear()
        {
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            txt_daily_rent.Text = "";
            txt_weekly_rent.Text = "";
            txt_monthly_rent.Text = "";
            cmb_refresh_category.Text = "";

            dtp_start.Value = DateTime.Now;
            dtp_end.Value = DateTime.Now;
            txt_calculate.Text = "";
            rbtn_yes.Checked = false;
            rbtn_no.Checked = false;


        }

        1 reference
        public void display_data_grid_view()    //For the data grid view
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Details_Renting_Table", con);
                adpt.Fill(dt);
                dgv_Renting_List.DataSource = dt;
```

Figure 4. 55 Driver logged Rent Vehicle Form Code - part 2



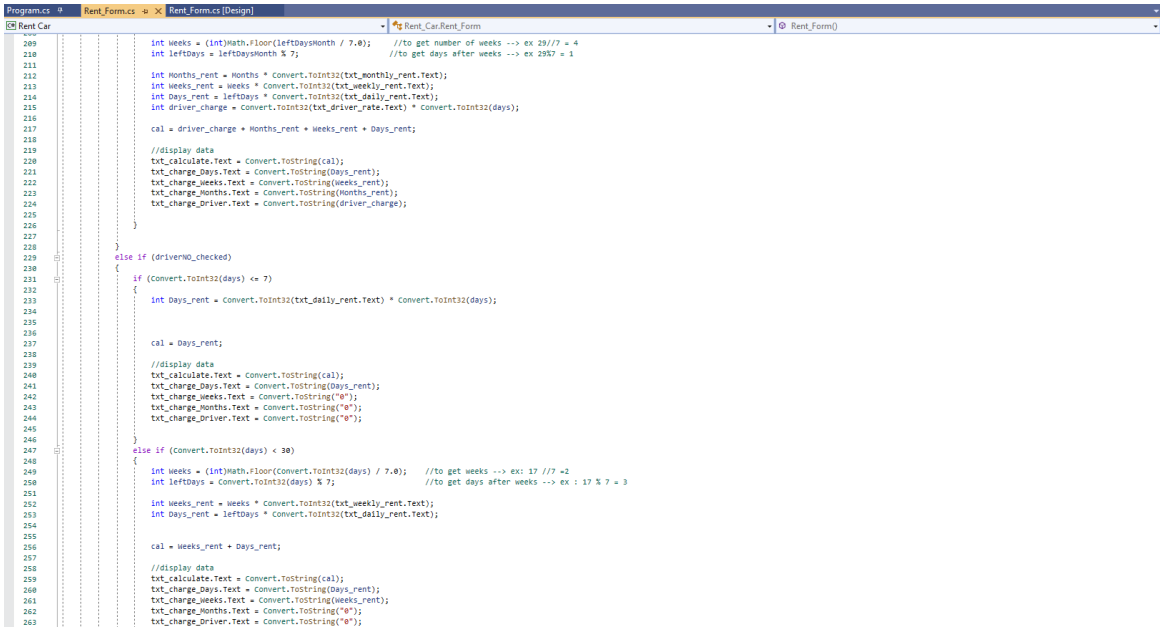
```
            dgv_Renting_List.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }


    int day_rent;
    int week_rent;
    int month_rent;
    int driver_charge;
    1 reference
    private void dgv_Renting_List_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_Renting_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Details_Renting_Table WHERE Rent_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            txt_rentID.Text = DR1.GetValue(0).ToString();
            txt_type.Text = DR1.GetValue(1).ToString();
            txt_brand.Text = DR1.GetValue(2).ToString();
            txt_model.Text = DR1.GetValue(3).ToString();
            txt_colour.Text = DR1.GetValue(4).ToString();
            day_rent = Convert.ToInt32(txt_daily_rent.Text = DR1.GetValue(5).ToString());          //should Covert to int, becz we calculate them when adding the bill
            week_rent = Convert.ToInt32(txt_weekly_rent.Text = DR1.GetValue(6).ToString());        //should Covert to int, becz we calculate them when adding the bill
            month_rent = Convert.ToInt32(txt_monthly_rent.Text = DR1.GetValue(7).ToString());      //should Covert to int, becz we calculate them when adding the bill
            driver_charge = Convert.ToInt32(txt_driver_rate.Text = DR1.GetValue(8).ToString());    //should Covert to int, becz we calculate them when adding the bill

        }
        DR1.Close();
        con.Close();
    }


    //********************************* CALCULATION PART ***********************************************************************************************************************

    //CALCULATION PART

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

Figure 4. 56 Driver logged Rent Vehicle Form Code - part 3



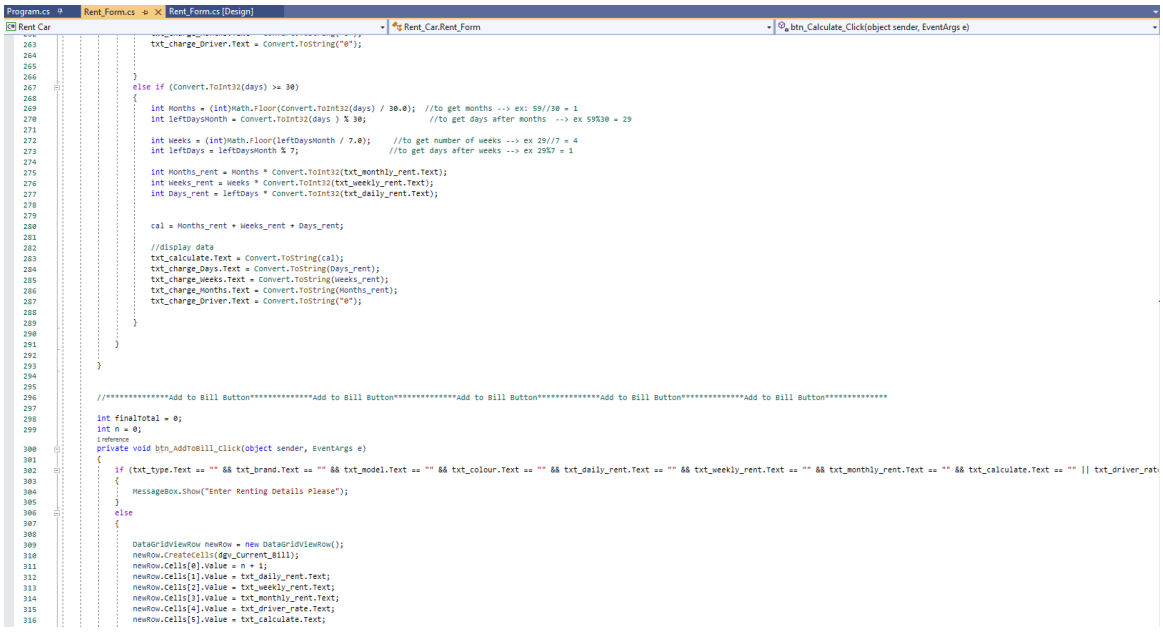
```
        Boolean driverYES_checked = rbtn_yes.Checked;
        Boolean driverNO_checked = rbtn_no.Checked;

        DateTime date1 = dtp_start.Value.Date;
        DateTime date2 = dtp_end.Value.Date;

        TimeSpan date_dif = date2 - date1;

        int days = date_dif.Days;       //Convert date differences into intger


        int cal = 0;
        if (driverYES_checked)
        {
            if (Convert.ToInt32(days) <= 7)
            {
                int Days_rent = Convert.ToInt32(txt_daily_rent.Text) * Convert.ToInt32(days);
                int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);


                cal = driver_charge + Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString("0");
                txt_charge_Months.Text = Convert.ToString("0");
                txt_charge_Driver.Text = Convert.ToString(driver_charge);
            }
            else if (Convert.ToInt32(days) < 30)
            {
                int Weeks = (int)Math.Floor(Convert.ToInt32(days) / 7.0);    //to get weeks --> ex: 17 //7 =2
                int leftDays = Convert.ToInt32(days) % 7;                    //to get days after weeks --> ex : 17 % 7 = 3

                int weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
                int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
                int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);

                cal = driver_charge + Weeks_rent + Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
                txt_charge_Months.Text = Convert.ToString("0");
                txt_charge_Driver.Text = Convert.ToString(driver_charge);

            }
            else if (Convert.ToInt32(days) >= 30)
            {
                int Months = (int)Math.Floor(Convert.ToInt32(days) / 30.0);  //to get months --> ex: 59//30 = 1
                int leftDaysMonth = Convert.ToInt32(days) % 30;              //to get days after months  --> ex 59%30 = 29

                int Weeks = (int)Math.Floor(leftDaysMonth / 7.0);     //to get number of weeks --> ex 29//7 = 4
```

Figure 4. 57 Driver logged Rent Vehicle Form Code - part 4



```
                int Weeks = (int)Math.Floor(leftDaysMonth / 7.0);     //to get number of weeks --> ex 29//7 = 4
                int leftDays = leftDaysMonth % 7;                    //to get days after weeks --> ex 29%7 = 1

                int Months_rent = Months * Convert.ToInt32(txt_monthly_rent.Text);
                int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
                int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);
                int driver_charge = Convert.ToInt32(txt_driver_rate.Text) * Convert.ToInt32(days);

                cal = driver_charge + Months_rent + Weeks_rent + Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
                txt_charge_Months.Text = Convert.ToString(Months_rent);
                txt_charge_Driver.Text = Convert.ToString(driver_charge);

            }

        }
        else if (driverNO_checked)
        {
            if (Convert.ToInt32(days) <= 7)
            {
                int Days_rent = Convert.ToInt32(txt_daily_rent.Text) * Convert.ToInt32(days);



                cal = Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString("0");
                txt_charge_Months.Text = Convert.ToString("0");
                txt_charge_Driver.Text = Convert.ToString("0");

            }
            else if (Convert.ToInt32(days) < 30)
            {
                int Weeks = (int)Math.Floor(Convert.ToInt32(days) / 7.0);    //to get weeks --> ex: 17 //7 =2
                int leftDays = Convert.ToInt32(days) % 7;                    //to get days after weeks --> ex : 17 % 7 = 3

                int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
                int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);


                cal = Weeks_rent + Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
                txt_charge_Months.Text = Convert.ToString("0");
                txt_charge_Driver.Text = Convert.ToString("0");
```

Figure 4. 58 Driver logged Rent Vehicle Form Code - part 5



```
                txt_charge_Driver.Text = Convert.ToString("0");


            }
            else if (Convert.ToInt32(days) >= 30)
            {
                int Months = (int)Math.Floor(Convert.ToInt32(days) / 30.0);  //to get months --> ex: 59//30 = 1
                int leftDaysMonth = Convert.ToInt32(days ) % 30;                    //to get days after months  --> ex 59%30 = 29

                int Weeks = (int)Math.Floor(leftDaysMonth / 7.0);     //to get number of weeks --> ex 29//7 = 4
                int leftDays = leftDaysMonth % 7;                    //to get days after weeks --> ex 29%7 = 1

                int Months_rent = Months * Convert.ToInt32(txt_monthly_rent.Text);
                int Weeks_rent = Weeks * Convert.ToInt32(txt_weekly_rent.Text);
                int Days_rent = leftDays * Convert.ToInt32(txt_daily_rent.Text);


                cal = Months_rent + Weeks_rent + Days_rent;

                //display data
                txt_calculate.Text = Convert.ToString(cal);
                txt_charge_Days.Text = Convert.ToString(Days_rent);
                txt_charge_Weeks.Text = Convert.ToString(Weeks_rent);
                txt_charge_Months.Text = Convert.ToString(Months_rent);
                txt_charge_Driver.Text = Convert.ToString("0");

            }

        }

    }

    //*************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************

    int finalTotal = 0;
    int n = 0;
    1 reference
    private void btn_AddToBill_Click(object sender, EventArgs e)
    {
        if (txt_type.Text == "" && txt_brand.Text == "" && txt_model.Text == "" && txt_colour.Text == "" && txt_daily_rent.Text == "" && txt_weekly_rent.Text == "" && txt_monthly_rent.Text == "" && txt_calculate.Text == "" || txt_driver_rat
        {
            MessageBox.Show("Enter Renting Details Please");
        }
        else
        {

            DataGridViewRow newRow = new DataGridViewRow();
            newRow.CreateCells(dgv_current_Bill);
            newRow.Cells[0].Value = n + 1;
            newRow.Cells[1].Value = txt_daily_rent.Text;
            newRow.Cells[2].Value = txt_weekly_rent.Text;
            newRow.Cells[3].Value = txt_monthly_rent.Text;
            newRow.Cells[4].Value = txt_driver_rate.Text;
            newRow.Cells[5].Value = txt_calculate.Text;
```

Figure 4. 59 Driver logged Rent Vehicle Form Code - part 6

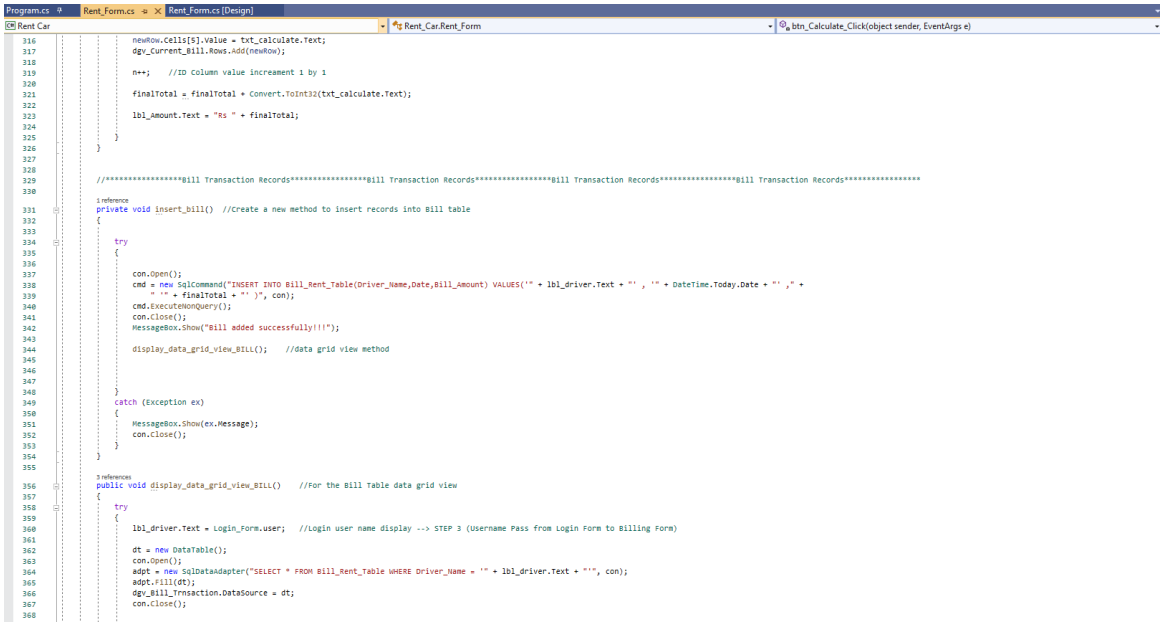


Figure 4. 60 Driver logged Rent Vehicle Form Code - part 7

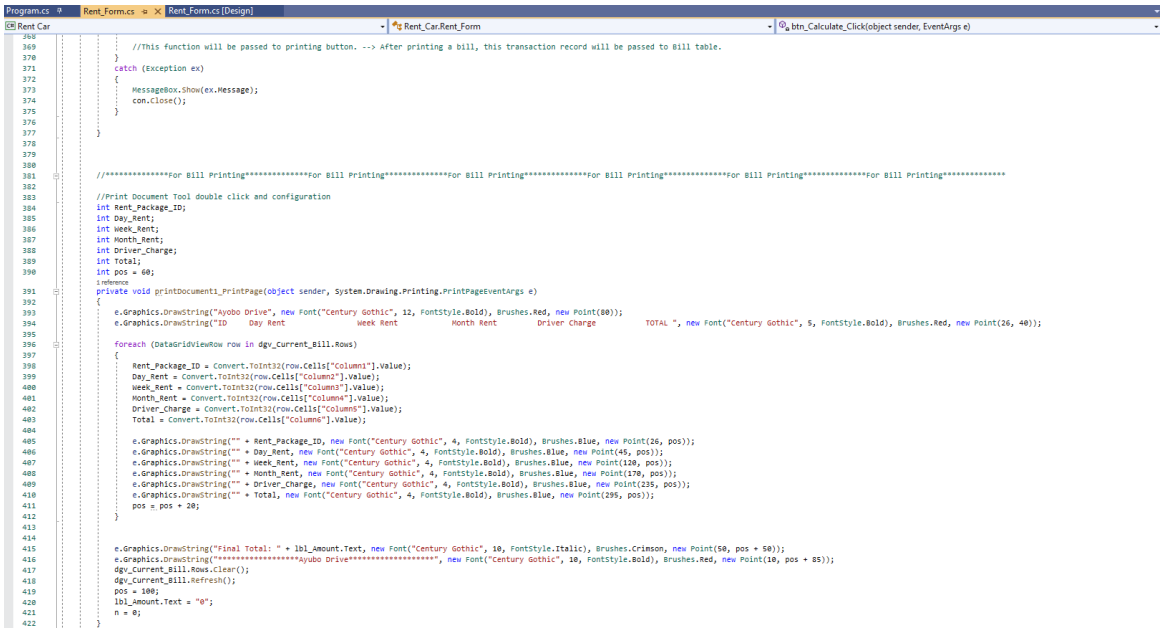


Figure 4. 61 Driver logged Rent Vehicle Form Code - part 8

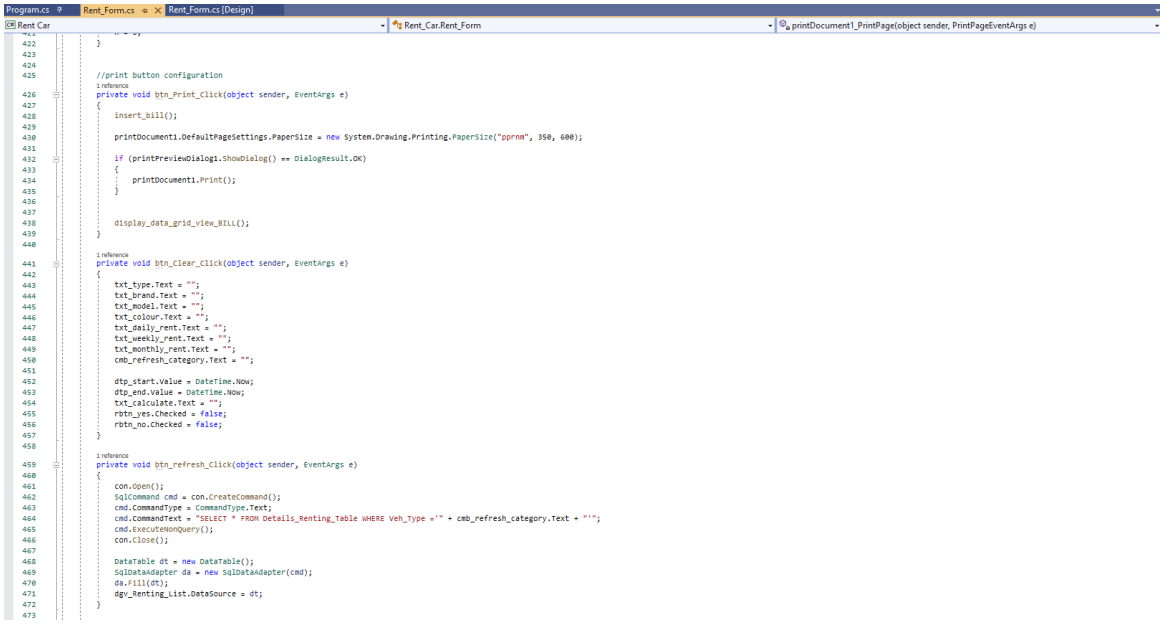


Figure 4. 62 Driver logged Rent Vehicle Form Code - part 9

4.1.12 Driver logged One Day Hire Vehicle Form Design and Code

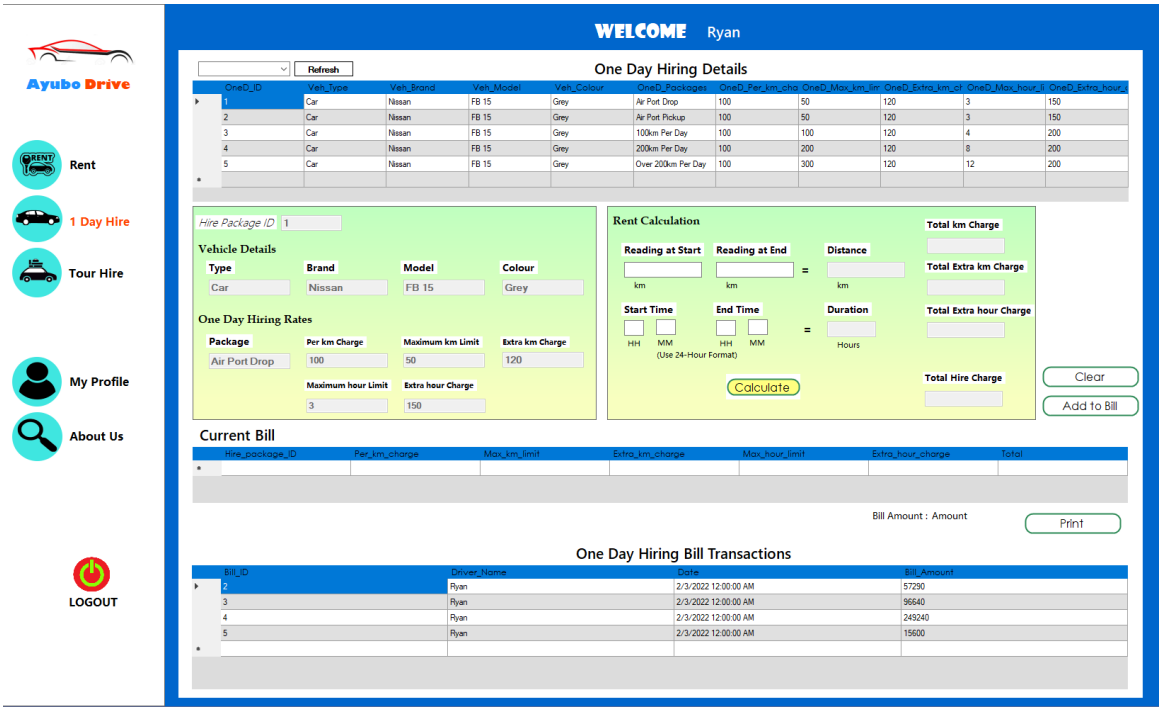
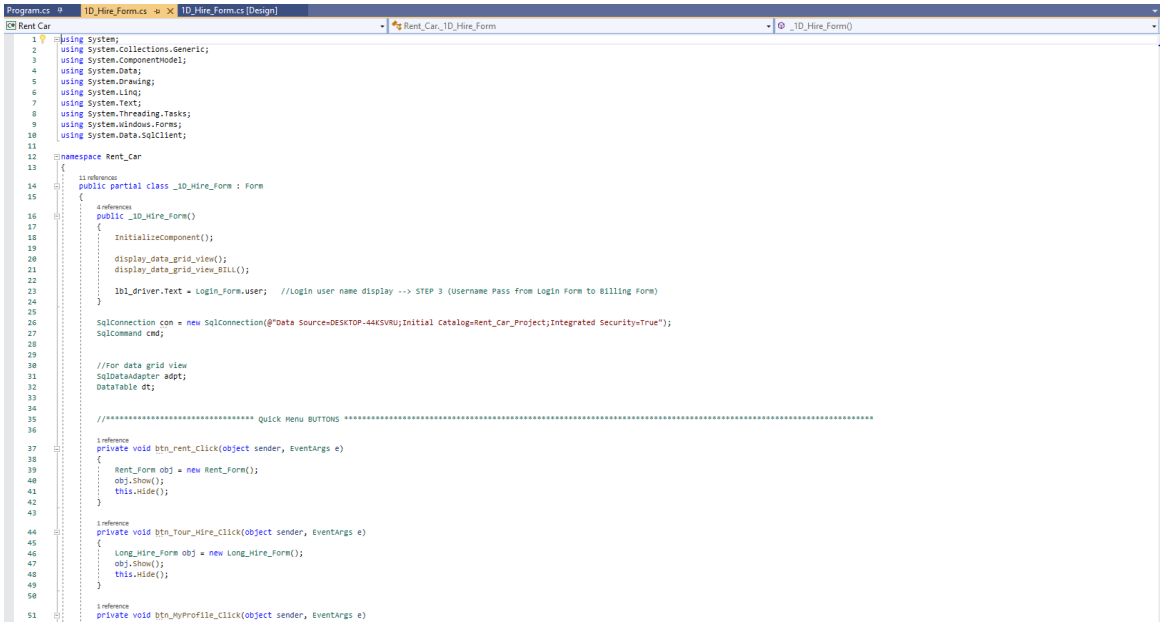
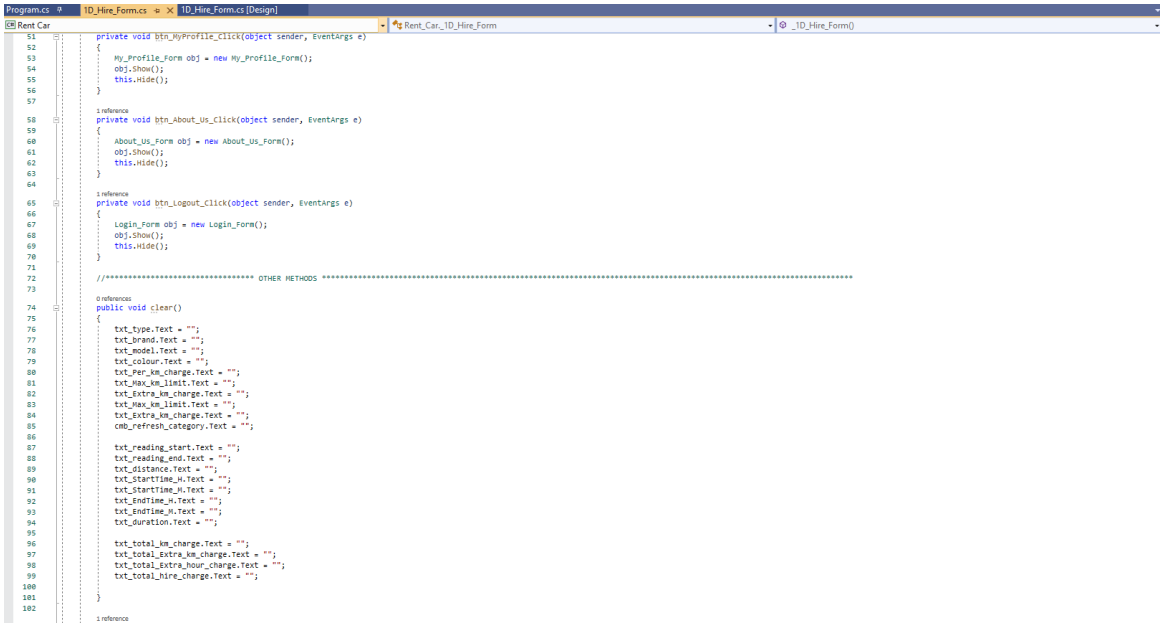


Figure 4. 63 Driver logged One Day Hire Vehicle Form Design



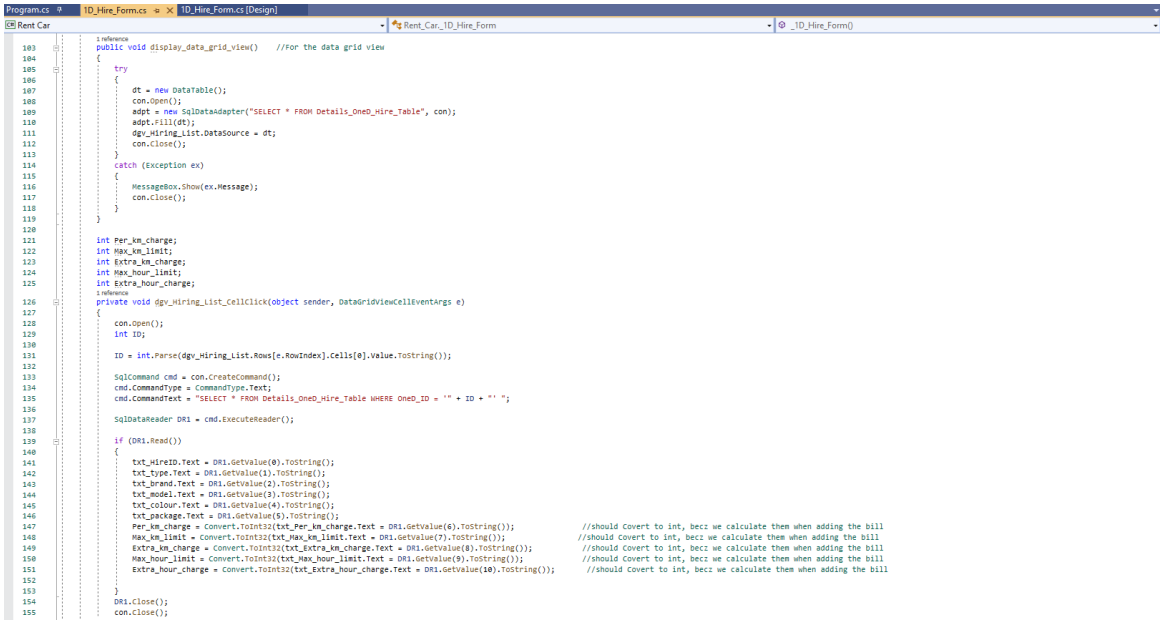
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    11 references
    public partial class _1D_Hire_Form : Form
    {
        4 references
        public _1D_Hire_Form()
        {
            InitializeComponent();

            display_data_grid_view();
            display_data_grid_view_BILL();

            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************** Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_rent_Click(object sender, EventArgs e)
        {
            Rent_Form obj = new Rent_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Long_Hire_Form obj = new Long_Hire_Form();
            obj.Show();
            this.Hide();
        }

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

Figure 4. 64 Driver logged One Day Hire Vehicle Form Code - part 1



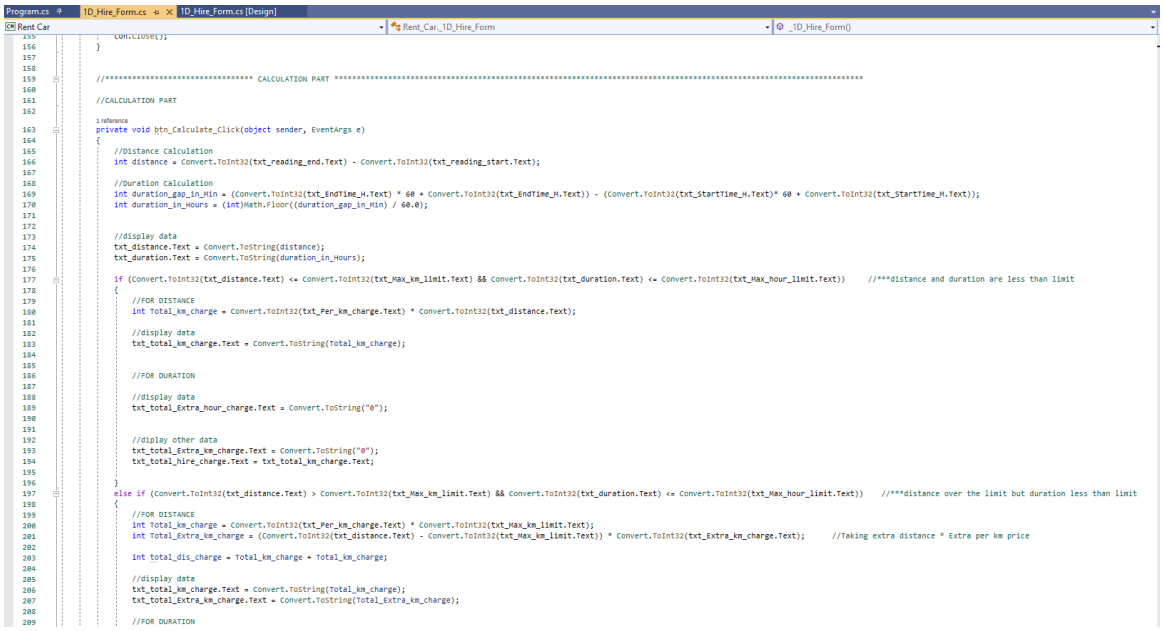
```
        private void btn_MyProfile_Click(object sender, EventArgs e)
        {
            My_Profile_Form obj = new My_Profile_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_About_Us_Click(object sender, EventArgs e)
        {
            About_Us_Form obj = new About_Us_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        //********************************** OTHER METHODS ***********************************************************************************************************

        0 references
        public void clear()
        {
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            txt_Per_Km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            cmb_refresh_category.Text = "";

            txt_reading_start.Text = "";
            txt_reading_end.Text = "";
            txt_distance.Text = "";
            txt_StartTime_H.Text = "";
            txt_StartTime_M.Text = "";
            txt_EndTime_H.Text = "";
            txt_EndTime_M.Text = "";
            txt_duration.Text = "";

            txt_total_km_charge.Text = "";
            txt_total_Extra_km_charge.Text = "";
            txt_total_Extra_hour_charge.Text = "";
            txt_total_hire_charge.Text = "";

        }

        1 reference
```

Figure 4. 65 Driver logged One Day Hire Vehicle Form Code - part 2



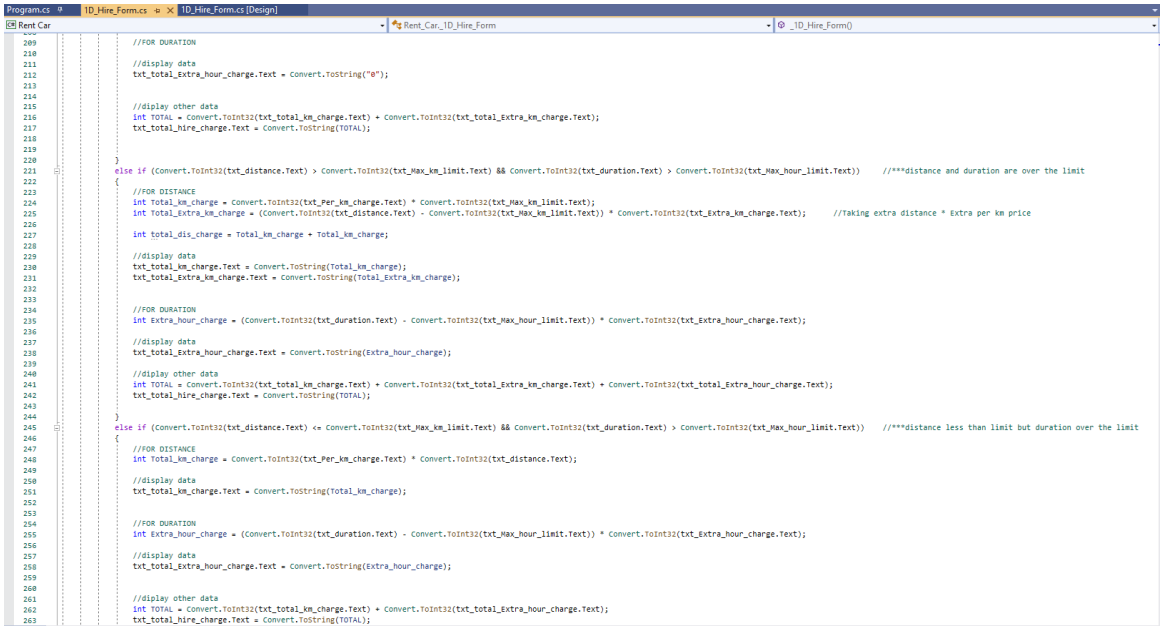
```
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Details_OneD_Hire_Table", con);
        adpt.Fill(dt);
        dgv_Hiring_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

int Per_km_charge;
int Max_km_limit;
int Extra_km_charge;
int Max_hour_limit;
int Extra_hour_charge;
private void dgv_Hiring_List_CellClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_Hiring_List.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Details_OneD_Hire_Table WHERE OneD_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_HireID.Text = DR1.GetValue(0).ToString();
        txt_type.Text = DR1.GetValue(1).ToString();
        txt_brand.Text = DR1.GetValue(2).ToString();
        txt_model.Text = DR1.GetValue(3).ToString();
        txt_colour.Text = DR1.GetValue(4).ToString();
        txt_package.Text = DR1.GetValue(5).ToString();
        Per_km_charge = Convert.ToInt32(txt_Per_km_charge.Text = DR1.GetValue(6).ToString());              //should covert to int, becz we calculate them when adding the bill
        Max_km_limit = Convert.ToInt32(txt_Max_km_limit.Text = DR1.GetValue(7).ToString());               //should covert to int, becz we calculate them when adding the bill
        Extra_km_charge = Convert.ToInt32(txt_Extra_km_charge.Text = DR1.GetValue(8).ToString());         //should covert to int, becz we calculate them when adding the bill
        Max_hour_limit = Convert.ToInt32(txt_Max_hour_limit.Text = DR1.GetValue(9).ToString());           //should covert to int, becz we calculate them when adding the bill
        Extra_hour_charge = Convert.ToInt32(txt_Extra_hour_charge.Text = DR1.GetValue(10).ToString());     //should covert to int, becz we calculate them when adding the bill

    }
    DR1.Close();
    con.Close();
```

Figure 4. 66 Driver logged One Day Hire Vehicle Form Code - part 3



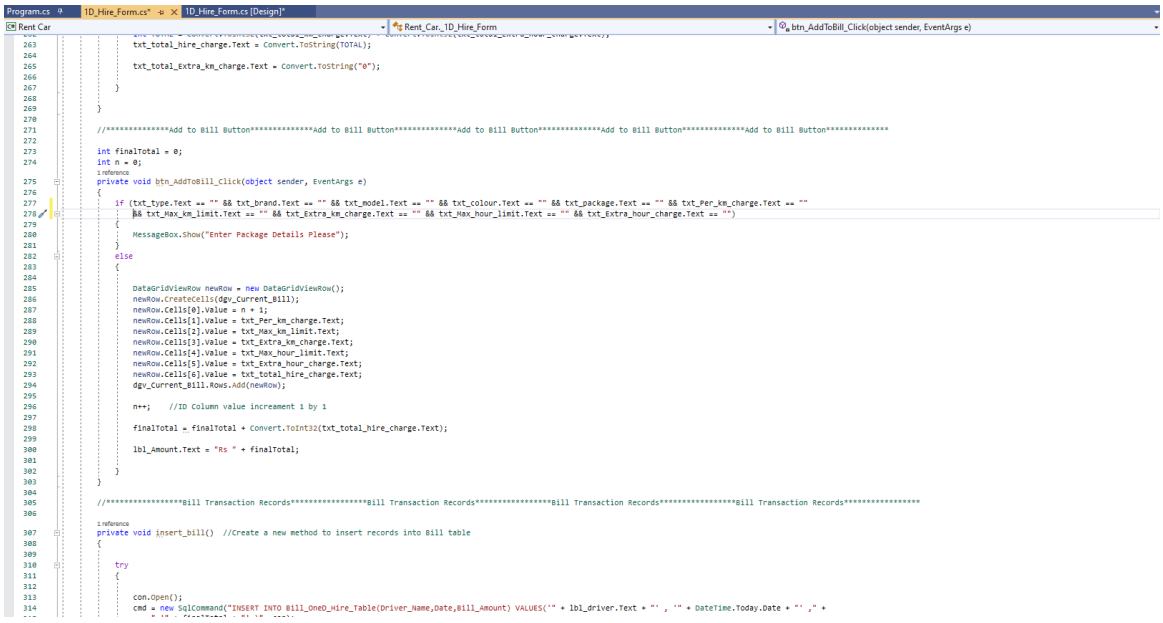
```
    con.Close();
}


//****************************** CALCULATION PART ***********************************************************************************************************

//CALCULATION PART

1 reference
private void btn_Calculate_Click(object sender, EventArgs e)
{
    //Distance Calculation
    int distance = Convert.ToInt32(txt_reading_end.Text) - Convert.ToInt32(txt_reading_start.Text);

    //Duration Calculation
    int duration_gap_in_Min = (Convert.ToInt32(txt_EndTime_H.Text) * 60 + Convert.ToInt32(txt_EndTime_M.Text)) - (Convert.ToInt32(txt_StartTime_H.Text)* 60 + Convert.ToInt32(txt_StartTime_M.Text));
    int duration_in_Hours = (int)Math.Floor((duration_gap_in_Min) / 60.0);


    //display data
    txt_distance.Text = Convert.ToString(distance);
    txt_duration.Text = Convert.ToString(duration_in_Hours);

    if (Convert.ToInt32(txt_distance.Text) <= Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) <= Convert.ToInt32(txt_Max_hour_limit.Text))     //***distance and duration are less than limit
    {
        //FOR DISTANCE
        int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_distance.Text);

        //display data
        txt_total_km_charge.Text = Convert.ToString(Total_km_charge);


        //FOR DURATION

        //display data
        txt_total_Extra_hour_charge.Text = Convert.ToString("0");


        //diplay other data
        txt_total_Extra_km_charge.Text = Convert.ToString("0");
        txt_total_hire_charge.Text = txt_total_km_charge.Text;

    }
    else if (Convert.ToInt32(txt_distance.Text) > Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) <= Convert.ToInt32(txt_Max_hour_limit.Text))    //***distance over the limit but duration less than limit
    {
        //FOR DISTANCE
        int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_Max_km_limit.Text);
        int Total_Extra_km_charge = (Convert.ToInt32(txt_distance.Text) - Convert.ToInt32(txt_Max_km_limit.Text)) * Convert.ToInt32(txt_Extra_km_charge.Text);     //Taking extra distance * Extra per km price

        int total_dis_charge = Total_km_charge + Total_km_charge;

        //display data
        txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
        txt_total_Extra_km_charge.Text = Convert.ToString(Total_Extra_km_charge);

        //FOR DURATION
```

Figure 4. 67 Driver logged One Day Hire Vehicle Form Code - part 4



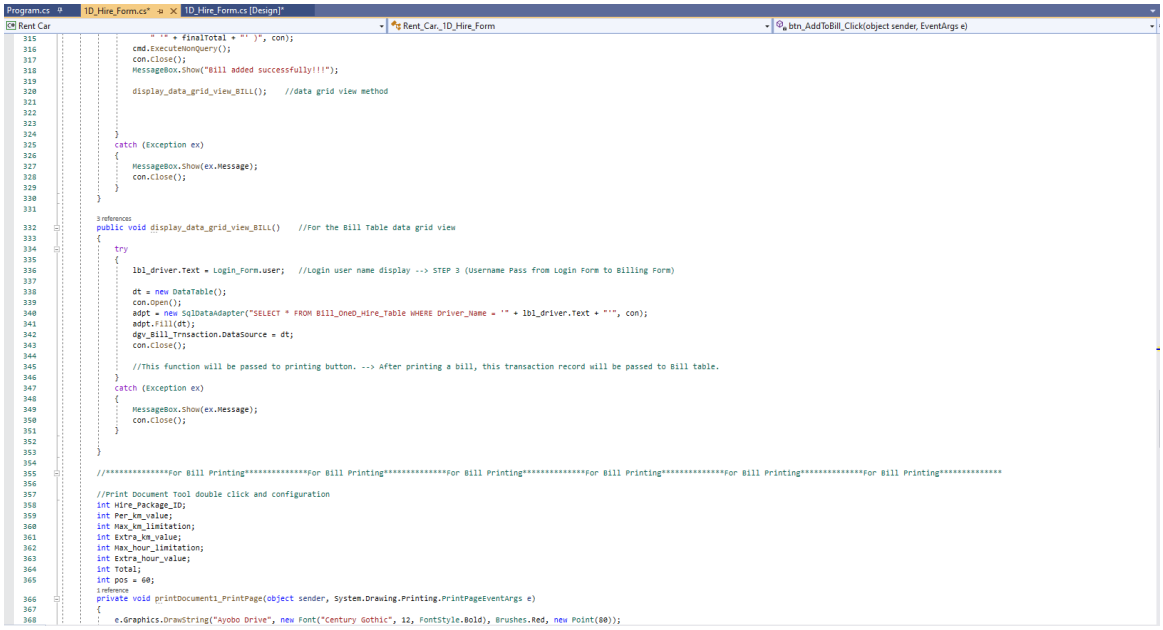
```
            //FOR DURATION

            //display data
            txt_total_Extra_hour_charge.Text = Convert.ToString("0");


            //diplay other data
            int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text);
            txt_total_hire_charge.Text = Convert.ToString(TOTAL);


        }
        else if (Convert.ToInt32(txt_distance.Text) > Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) > Convert.ToInt32(txt_Max_hour_limit.Text))     //***distance and duration are over the limit
        {
            //FOR DISTANCE
            int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_Max_km_limit.Text);
            int Total_Extra_km_charge = (Convert.ToInt32(txt_distance.Text) - Convert.ToInt32(txt_Max_km_limit.Text)) * Convert.ToInt32(txt_Extra_km_charge.Text);      //Taking extra distance * Extra per km price

            int total_dis_charge = Total_km_charge + Total_km_charge;

            //display data
            txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
            txt_total_Extra_km_charge.Text = Convert.ToString(Total_Extra_km_charge);


            //FOR DURATION
            int Extra_hour_charge = (Convert.ToInt32(txt_duration.Text) - Convert.ToInt32(txt_Max_hour_limit.Text)) * Convert.ToInt32(txt_Extra_hour_charge.Text);

            //display data
            txt_total_Extra_hour_charge.Text = Convert.ToString(Extra_hour_charge);

            //diplay other data
            int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text) + Convert.ToInt32(txt_total_Extra_hour_charge.Text);
            txt_total_hire_charge.Text = Convert.ToString(TOTAL);

        }
        else if (Convert.ToInt32(txt_distance.Text) <= Convert.ToInt32(txt_Max_km_limit.Text) && Convert.ToInt32(txt_duration.Text) > Convert.ToInt32(txt_Max_hour_limit.Text))    //***distance less than limit but duration over the limit
        {
            //FOR DISTANCE
            int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_distance.Text);

            //display data
            txt_total_km_charge.Text = Convert.ToString(Total_km_charge);


            //FOR DURATION
            int Extra_hour_charge = (Convert.ToInt32(txt_duration.Text) - Convert.ToInt32(txt_Max_hour_limit.Text)) * Convert.ToInt32(txt_Extra_hour_charge.Text);

            //display data
            txt_total_Extra_hour_charge.Text = Convert.ToString(Extra_hour_charge);


            //diplay other data
            int TOTAL = Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_hour_charge.Text);
            txt_total_hire_charge.Text = Convert.ToString(TOTAL);
```

Figure 4. 68 Driver logged One Day Hire Vehicle Form Code - part 5



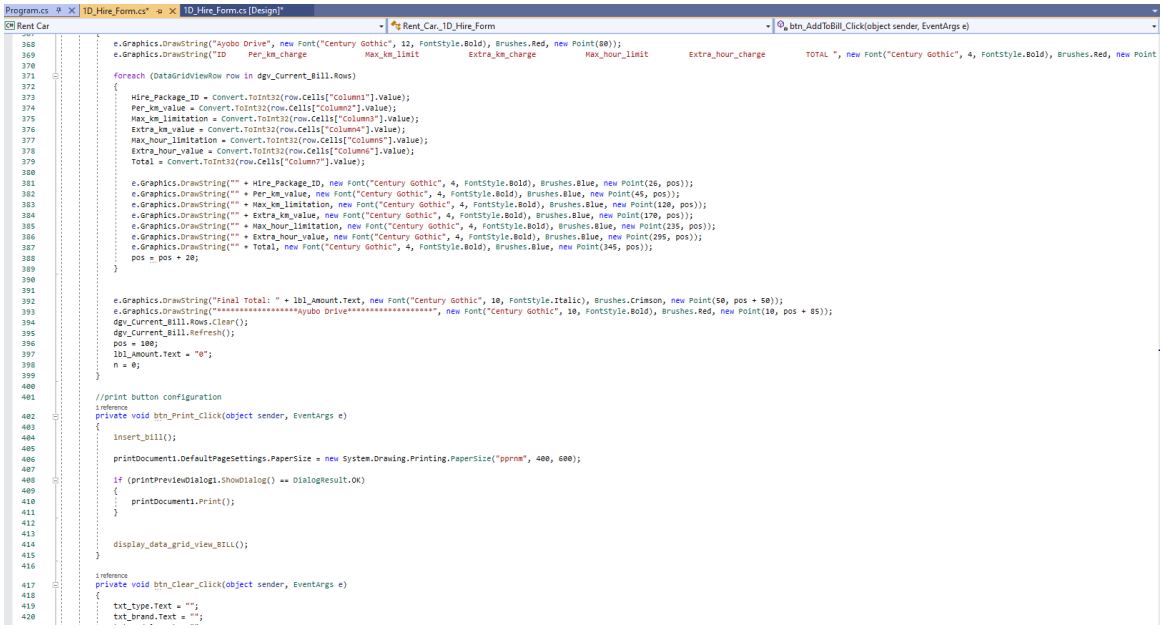
```
            txt_total_hire_charge.Text = Convert.ToString(TOTAL);

            txt_total_Extra_km_charge.Text = Convert.ToString("0");

        }
    }

    //**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************

    int finalTotal = 0;
    int n = 0;
    1 reference
    private void btn_AddToBill_Click(object sender, EventArgs e)
    {
        if (txt_type.Text == "" && txt_brand.Text == "" && txt_model.Text == "" && txt_colour.Text == "" && txt_package.Text == "" && txt_Per_km_charge.Text == ""
            && txt_Max_km_limit.Text == "" && txt_Extra_km_charge.Text == "" && txt_Max_hour_limit.Text == "" && txt_Extra_hour_charge.Text == "")
        {
            MessageBox.Show("Enter Package Details Please");
        }
        else
        {

            DataGridViewRow newRow = new DataGridViewRow();
            newRow.CreateCells(dgv_Current_Bill);
            newRow.Cells[0].Value = n + 1;
            newRow.Cells[1].Value = txt_Per_km_charge.Text;
            newRow.Cells[2].Value = txt_Max_km_limit.Text;
            newRow.Cells[3].Value = txt_Extra_km_charge.Text;
            newRow.Cells[4].Value = txt_Max_hour_limit.Text;
            newRow.Cells[5].Value = txt_Extra_hour_charge.Text;
            newRow.Cells[6].Value = txt_total_hire_charge.Text;
            dgv_Current_Bill.Rows.Add(newRow);

            n++;    //ID Column value increament 1 by 1

            finalTotal = finalTotal + Convert.ToInt32(txt_total_hire_charge.Text);

            lbl_Amount.Text = "Rs " + finalTotal;

        }
    }

    //****************Bill Transaction Records****************Bill Transaction Records****************Bill Transaction Records****************Bill Transaction Records****************

    1 reference
    private void insert_bill()  //Create a new method to insert records into Bill table
    {

        try
        {

            con.Open();
            cmd = new SqlCommand("INSERT INTO Bill_OneD_Hire_Table(Driver_Name,Date,Bill_Amount) VALUES('" + lbl_driver.Text + "' , '" + DateTime.Today.Date + "' ," +
```

Figure 4. 69 Driver logged One Day Hire Vehicle Form Code - part 6



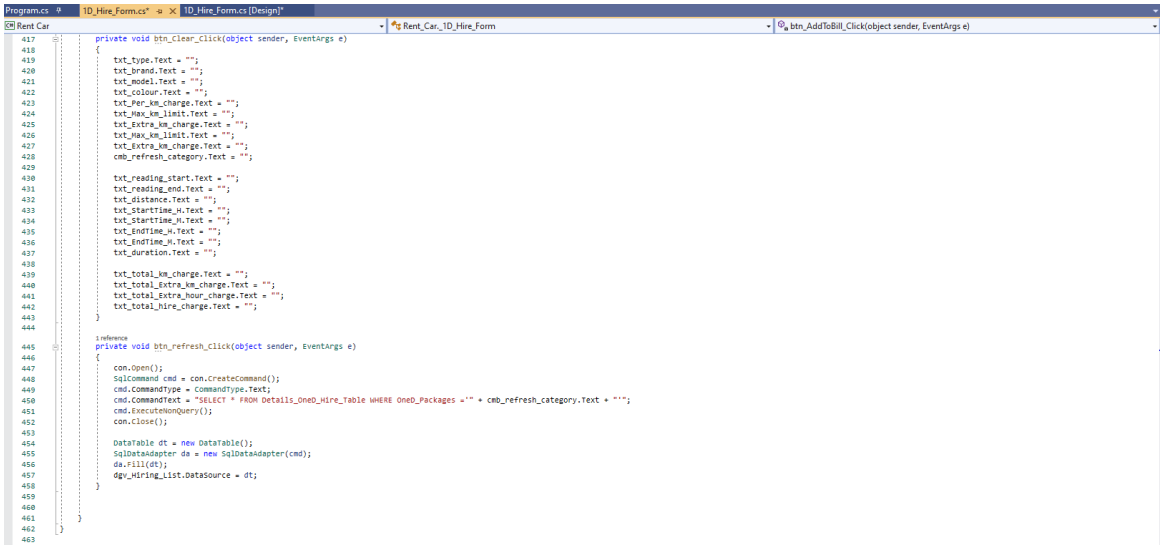
```
                " '" + finalTotal + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Bill added successfully!!!");

            display_data_grid_view_BILL();    //data grid view method



        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

    3 references
    public void display_data_grid_view_BILL()    //For the Bill Table data grid view
    {
        try
        {
            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)

            dt = new DataTable();
            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Bill_OneD_Hire_Table WHERE Driver_Name = '" + lbl_driver.Text + "'", con);
            adpt.Fill(dt);
            dgv_Bill_Trnsaction.DataSource = dt;
            con.Close();

            //This function will be passed to printing button. --> After printing a bill, this transaction record will be passed to Bill table.
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }

    }

    //**************For Bill Printing**************For Bill Printing**************For Bill Printing**************For Bill Printing**************For Bill Printing**************

    //Print Document Tool double click and configuration
    int Hire_Package_ID;
    int Per_km_value;
    int Max_km_limitation;
    int Extra_km_value;
    int Max_hour_limitation;
    int Extra_hour_value;
    int Total;
    int pos = 60;
    1 reference
    private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
    {
        e.Graphics.DrawString("Ayobo Drive", new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Red, new Point(80));
```

Figure 4. 70 Driver logged One Day Hire Vehicle Form Code - part 7



```
        e.Graphics.DrawString("Ayobo Drive", new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Red, new Point(80));
        e.Graphics.DrawString("ID     Per_km_charge            Max_km_limit          Extra_km_charge          Max_hour_limit        Extra_hour_charge        TOTAL ", new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Red, new Point

        foreach (DataGridViewRow row in dgv_Current_Bill.Rows)
        {
            Hire_Package_ID = Convert.ToInt32(row.Cells["Column1"].Value);
            Per_km_value = Convert.ToInt32(row.Cells["Column2"].Value);
            Max_km_limitation = Convert.ToInt32(row.Cells["Column3"].Value);
            Extra_km_value = Convert.ToInt32(row.Cells["Column4"].Value);
            Max_hour_limitation = Convert.ToInt32(row.Cells["Column5"].Value);
            Extra_hour_value = Convert.ToInt32(row.Cells["Column6"].Value);
            Total = Convert.ToInt32(row.Cells["Column7"].Value);

            e.Graphics.DrawString("" + Hire_Package_ID, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(26, pos));
            e.Graphics.DrawString("" + Per_km_value, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(45, pos));
            e.Graphics.DrawString("" + Max_km_limitation, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(120, pos));
            e.Graphics.DrawString("" + Extra_km_value, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(170, pos));
            e.Graphics.DrawString("" + Max_hour_limitation, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(235, pos));
            e.Graphics.DrawString("" + Extra_hour_value, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(295, pos));
            e.Graphics.DrawString("" + Total, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(345, pos));
            pos = pos + 20;
        }


        e.Graphics.DrawString("Final Total: " + lbl_Amount.Text, new Font("Century Gothic", 10, FontStyle.Italic), Brushes.Crimson, new Point(50, pos + 50));
        e.Graphics.DrawString("*****************Ayubo Drive*****************", new Font("Century Gothic", 10, FontStyle.Bold), Brushes.Red, new Point(10, pos + 85));
        dgv_Current_Bill.Rows.Clear();
        dgv_Current_Bill.Refresh();
        pos = 100;
        lbl_Amount.Text = "0";
        n = 0;
    }

    //print button configuration
    1 reference
    private void btn_Print_Click(object sender, EventArgs e)
    {
        insert_bill();

        printDocument1.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm", 400, 600);

        if (printPreviewDialog1.ShowDialog() == DialogResult.OK)
        {
            printDocument1.Print();
        }


        display_data_grid_view_BILL();
    }

    1 reference
    private void btn_Clear_Click(object sender, EventArgs e)
    {
        txt_type.Text = "";
        txt_brand.Text = "";
```

Figure 4. 71 Driver logged One Day Hire Vehicle Form Code - part 8



```
Program.cs | 1D_Hire_Form.cs* | 1D_Hire_Form.cs [Design]*
Rent Car | Rent_Car_1D_Hire_Form | btn_AddToBill_Click(object sender, EventArgs e)

417     private void btn_Clear_Click(object sender, EventArgs e)
418     {
419         txt_type.Text = "";
420         txt_brand.Text = "";
421         txt_model.Text = "";
422         txt_colour.Text = "";
423         txt_Per_km_charge.Text = "";
424         txt_Max_km_limit.Text = "";
425         txt_Extra_km_charge.Text = "";
426         txt_Max_km_limit.Text = "";
427         txt_Extra_km_charge.Text = "";
428         cmb_refresh_category.Text = "";
429
430         txt_reading_start.Text = "";
431         txt_reading_end.Text = "";
432         txt_distance.Text = "";
433         txt_starttime_H.Text = "";
434         txt_starttime_M.Text = "";
435         txt_EndTime_H.Text = "";
436         txt_EndTime_M.Text = "";
437         txt_duration.Text = "";
438
439         txt_total_km_charge.Text = "";
440         txt_total_Extra_km_charge.Text = "";
441         txt_total_Extra_hour_charge.Text = "";
442         txt_total_hire_charge.Text = "";
443     }
444
        1 reference
445     private void btn_refresh_Click(object sender, EventArgs e)
446     {
447         con.Open();
448         SqlCommand cmd = con.CreateCommand();
449         cmd.CommandType = CommandType.Text;
450         cmd.CommandText = "SELECT * FROM Details_OneD_Hire_Table WHERE OneD_Packages ='" + cmb_refresh_category.Text + "'";
451         cmd.ExecuteNonQuery();
452         con.Close();
453
454         DataTable dt = new DataTable();
455         SqlDataAdapter da = new SqlDataAdapter(cmd);
456         da.Fill(dt);
457         dgv_Hiring_List.DataSource = dt;
458     }
459
460
461 }
462
463
```

Figure 4. 72 Driver logged One Day Hire Vehicle Form Code - part 9

4.1.13 Driver logged Long Tour Hire Vehicle Form Design and Code

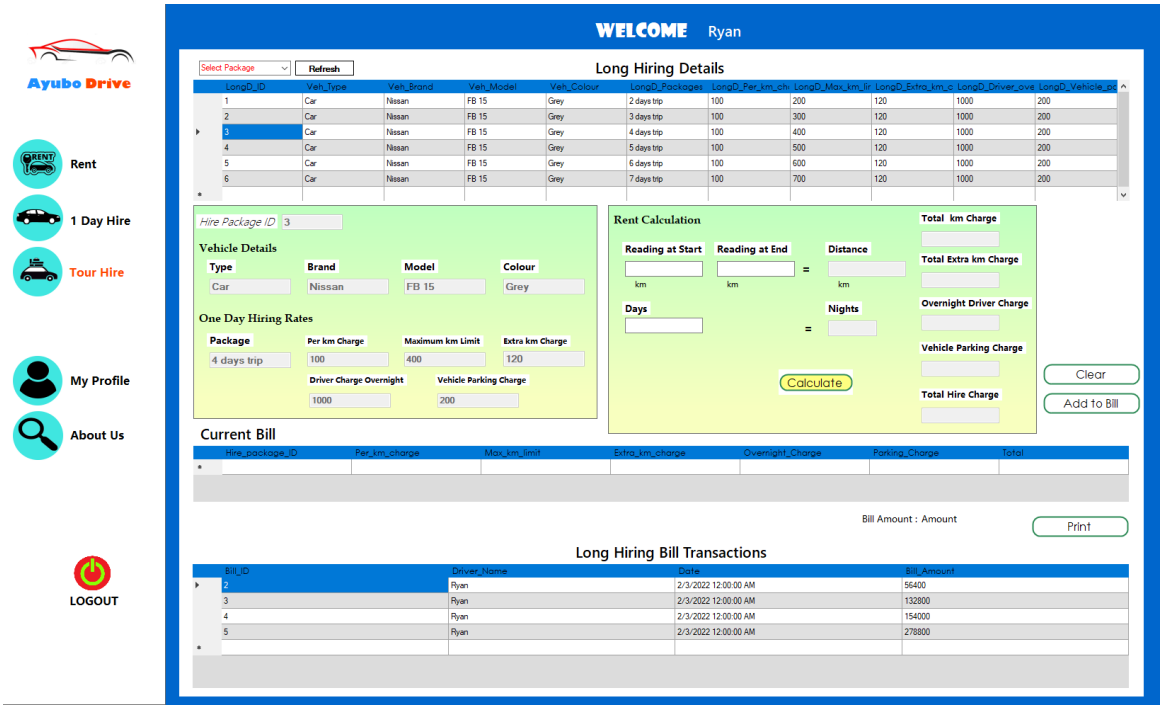
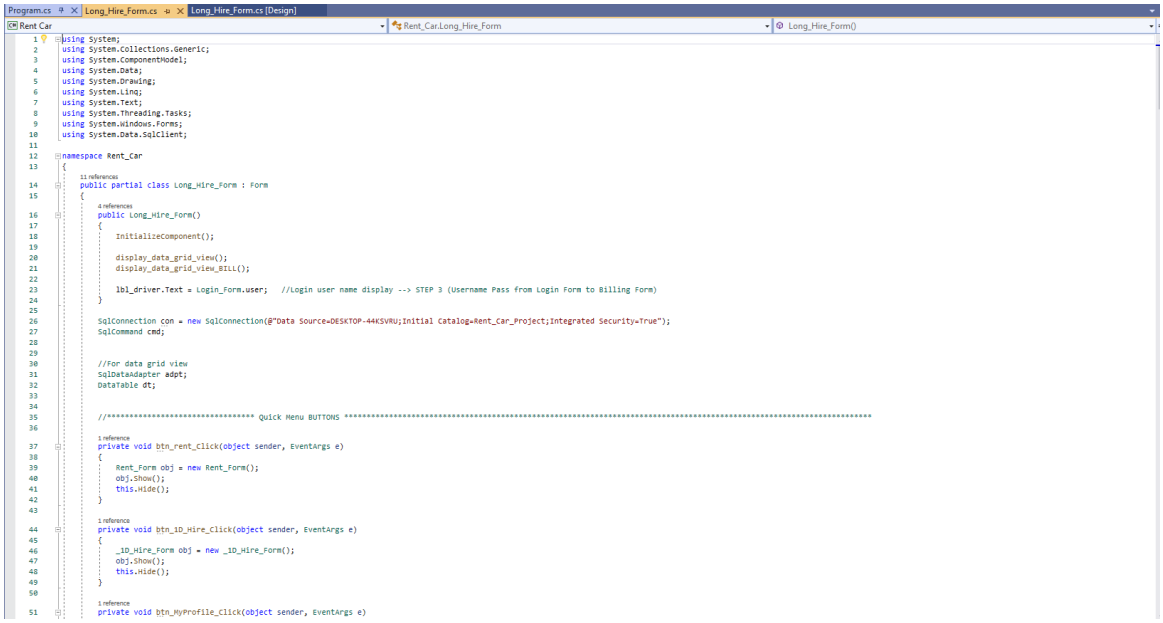
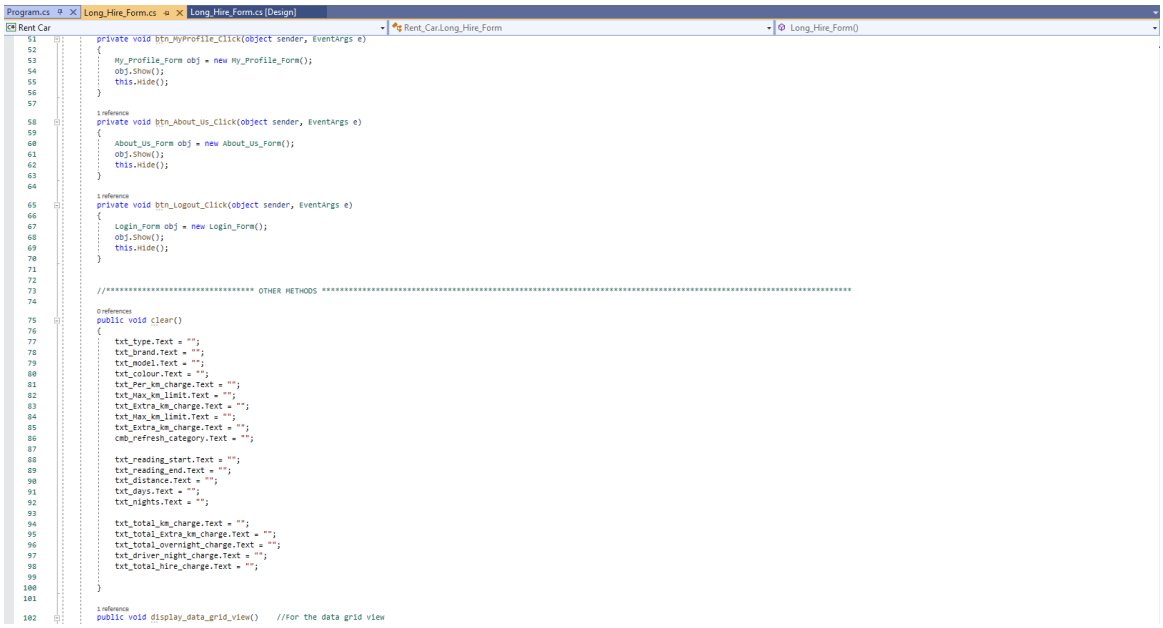


Figure 4. 73 Driver logged Long Tour Hire Vehicle Form Design



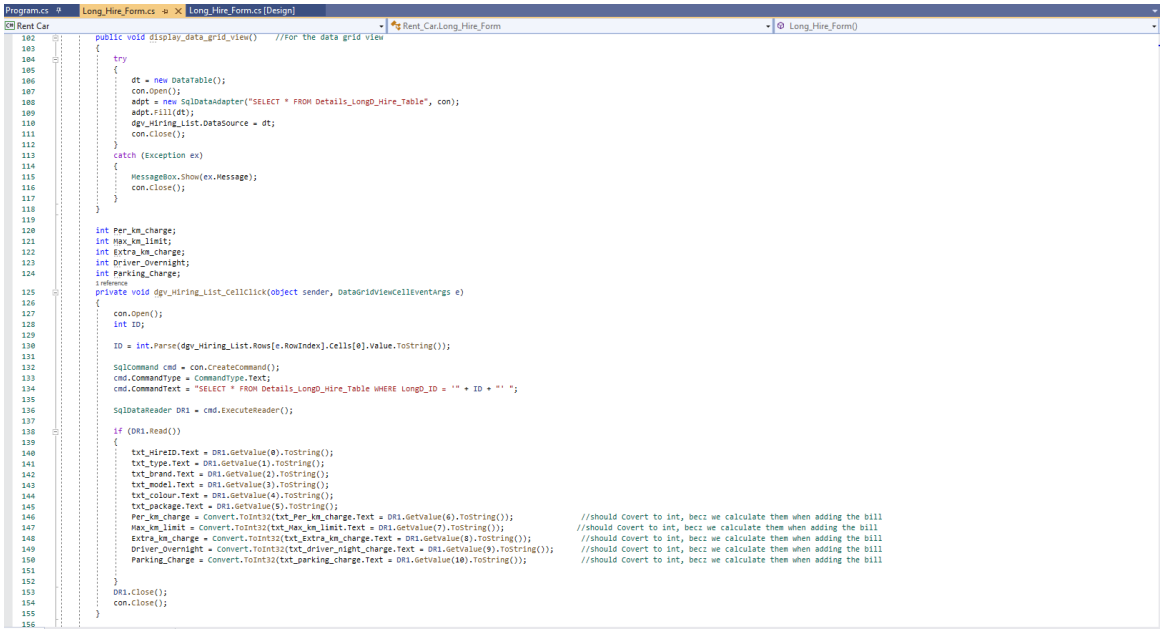
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    11 references
    public partial class Long_Hire_Form : Form
    {
        4 references
        public Long_Hire_Form()
        {
            InitializeComponent();


            display_data_grid_view();
            display_data_grid_view_BILL();

            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***************************************************************************************************************

        1 reference
        private void btn_rent_Click(object sender, EventArgs e)
        {
            Rent_Form obj = new Rent_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            _1D_Hire_Form obj = new _1D_Hire_Form();
            obj.Show();
            this.Hide();
        }

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

Figure 4. 74 Driver logged Long Tour Hire Vehicle Form Code - part 1



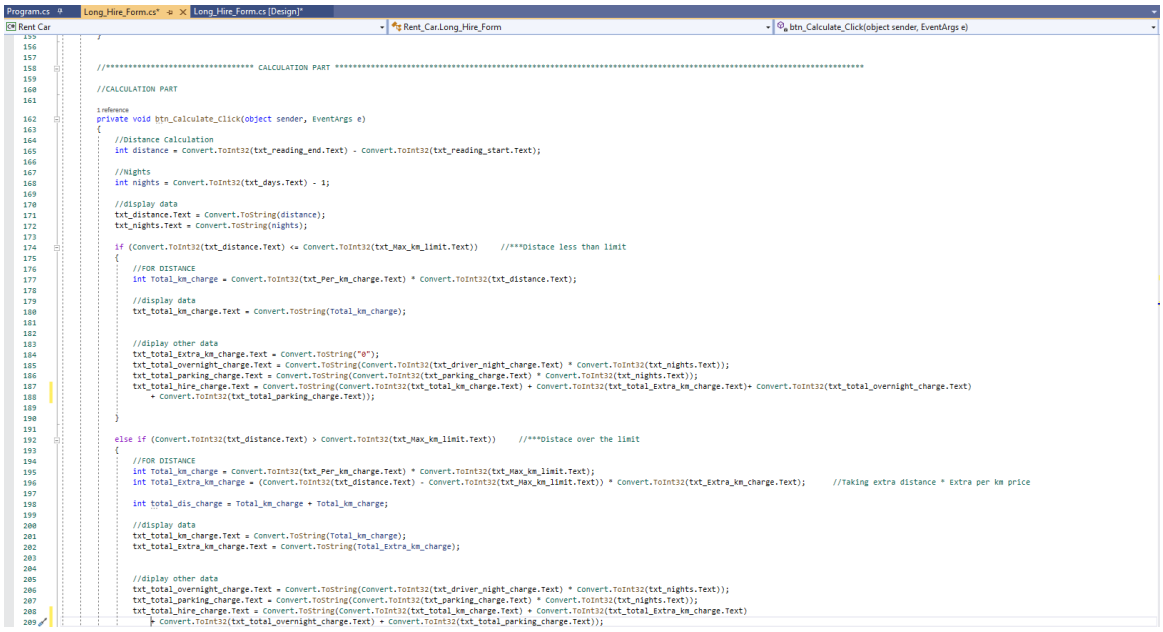
```
        private void btn_MyProfile_Click(object sender, EventArgs e)
        {
            My_Profile_Form obj = new My_Profile_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_About_Us_Click(object sender, EventArgs e)
        {
            About_Us_Form obj = new About_Us_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        //********************************* OTHER METHODS ********************************************************************************************************************

        0 references
        public void clear()
        {
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            txt_Per_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            cmb_refresh_category.Text = "";

            txt_reading_start.Text = "";
            txt_reading_end.Text = "";
            txt_distance.Text = "";
            txt_days.Text = "";
            txt_nights.Text = "";

            txt_total_km_charge.Text = "";
            txt_total_Extra_km_charge.Text = "";
            txt_total_overnight_charge.Text = "";
            txt_driver_night_charge.Text = "";
            txt_total_hire_charge.Text = "";

        }

        1 reference
        public void display_data_grid_view()    //For the data grid view
```

Figure 4. 75 Driver logged Long Tour Hire Vehicle Form Code - part 2



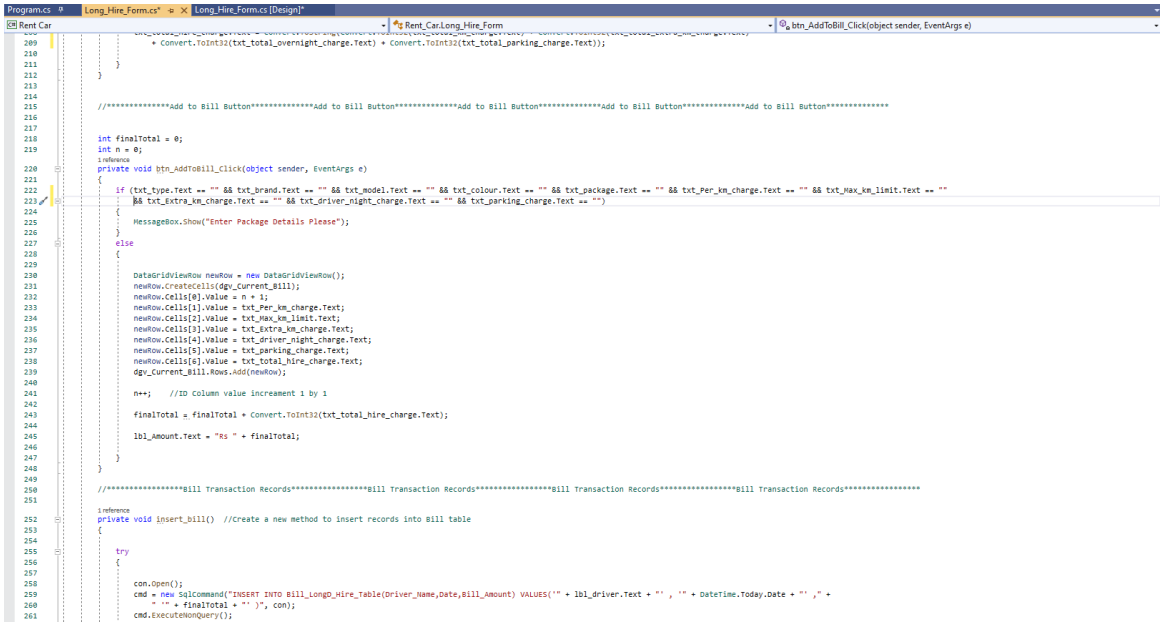
```
Program.cs | Long_Hire_Form.cs | Long_Hire_Form.cs [Design]
Rent Car | Rent_Car.Long_Hire_Form | Long_Hire_Form()

public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Details_LongD_Hire_Table", con);
        adpt.Fill(dt);
        dgv_Hiring_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

int Per_km_charge;
int Max_km_limit;
int Extra_km_charge;
int Driver_Overnight;
int Parking_Charge;
1 reference
private void dgv_Hiring_List_CellClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_Hiring_List.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Details_LongD_Hire_Table WHERE LongD_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_HireID.Text = DR1.GetValue(0).ToString();
        txt_type.Text = DR1.GetValue(1).ToString();
        txt_brand.Text = DR1.GetValue(2).ToString();
        txt_model.Text = DR1.GetValue(3).ToString();
        txt_colour.Text = DR1.GetValue(4).ToString();
        txt_package.Text = DR1.GetValue(5).ToString();
        Per_km_charge = Convert.ToInt32(txt_Per_km_charge.Text = DR1.GetValue(6).ToString());              //should Covert to int, becz we calculate them when adding the bill
        Max_km_limit = Convert.ToInt32(txt_Max_km_limit.Text = DR1.GetValue(7).ToString());              //should Covert to int, becz we calculate them when adding the bill
        Extra_km_charge = Convert.ToInt32(txt_Extra_km_charge.Text = DR1.GetValue(8).ToString());        //should Covert to int, becz we calculate them when adding the bill
        Driver_Overnight = Convert.ToInt32(txt_driver_night_charge.Text = DR1.GetValue(9).ToString());   //should Covert to int, becz we calculate them when adding the bill
        Parking_Charge = Convert.ToInt32(txt_parking_charge.Text = DR1.GetValue(10).ToString());         //should Covert to int, becz we calculate them when adding the bill

    }
    DR1.Close();
    con.Close();
}
```

Figure 4. 76 Driver logged Long Tour Hire Vehicle Form Code - part 3



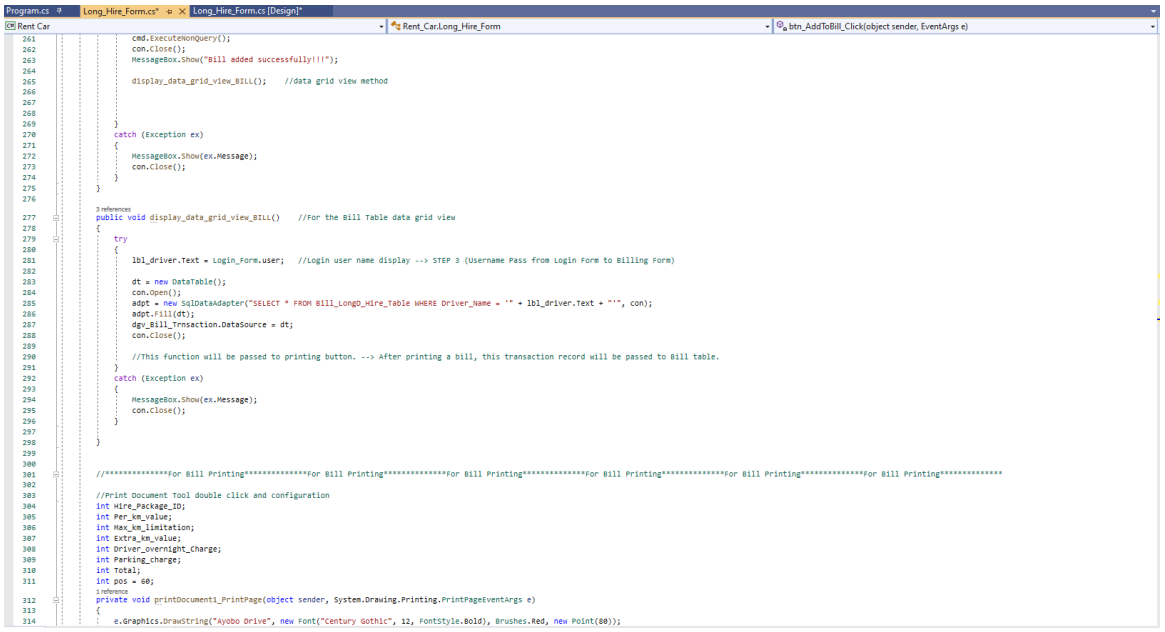
```
Program.cs | Long_Hire_Form.cs* | Long_Hire_Form.cs [Design]*
Rent Car | Rent_Car.Long_Hire_Form | btn_Calculate_Click(object sender, EventArgs e)

}


//********************************** CALCULATION PART ***********************************************************************************************************

//CALCULATION PART

1 reference
private void btn_Calculate_Click(object sender, EventArgs e)
{
    //Distance Calculation
    int distance = Convert.ToInt32(txt_reading_end.Text) - Convert.ToInt32(txt_reading_start.Text);

    //Nights
    int nights = Convert.ToInt32(txt_days.Text) - 1;

    //display data
    txt_distance.Text = Convert.ToString(distance);
    txt_nights.Text = Convert.ToString(nights);

    if (Convert.ToInt32(txt_distance.Text) <= Convert.ToInt32(txt_Max_km_limit.Text))     //***Distace less than limit
    {
        //FOR DISTANCE
        int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_distance.Text);

        //display data
        txt_total_km_charge.Text = Convert.ToString(Total_km_charge);


        //diplay other data
        txt_total_Extra_km_charge.Text = Convert.ToString("0");
        txt_total_overnight_charge.Text = Convert.ToString(Convert.ToInt32(txt_driver_night_charge.Text) * Convert.ToInt32(txt_nights.Text));
        txt_total_parking_charge.Text = Convert.ToString(Convert.ToInt32(txt_parking_charge.Text) * Convert.ToInt32(txt_nights.Text));
        txt_total_hire_charge.Text = Convert.ToString(Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text)+ Convert.ToInt32(txt_total_overnight_charge.Text)
            + Convert.ToInt32(txt_total_parking_charge.Text));

    }

    else if (Convert.ToInt32(txt_distance.Text) > Convert.ToInt32(txt_Max_km_limit.Text))     //***Distace over the limit
    {
        //FOR DISTANCE
        int Total_km_charge = Convert.ToInt32(txt_Per_km_charge.Text) * Convert.ToInt32(txt_Max_km_limit.Text);
        int Total_Extra_km_charge = (Convert.ToInt32(txt_distance.Text) - Convert.ToInt32(txt_Max_km_limit.Text)) * Convert.ToInt32(txt_Extra_km_charge.Text);      //Taking extra distance * Extra per km price

        int total_dis_charge = Total_km_charge + Total_km_charge;

        //display data
        txt_total_km_charge.Text = Convert.ToString(Total_km_charge);
        txt_total_Extra_km_charge.Text = Convert.ToString(Total_Extra_km_charge);


        //diplay other data
        txt_total_overnight_charge.Text = Convert.ToString(Convert.ToInt32(txt_driver_night_charge.Text) * Convert.ToInt32(txt_nights.Text));
        txt_total_parking_charge.Text = Convert.ToString(Convert.ToInt32(txt_parking_charge.Text) * Convert.ToInt32(txt_nights.Text));
        txt_total_hire_charge.Text = (Convert.ToString(Convert.ToInt32(txt_total_km_charge.Text) + Convert.ToInt32(txt_total_Extra_km_charge.Text)
            + Convert.ToInt32(txt_total_overnight_charge.Text) + Convert.ToInt32(txt_total_parking_charge.Text));
```

Figure 4. 77 Driver logged Long Tour Hire Vehicle Form Code - part 4



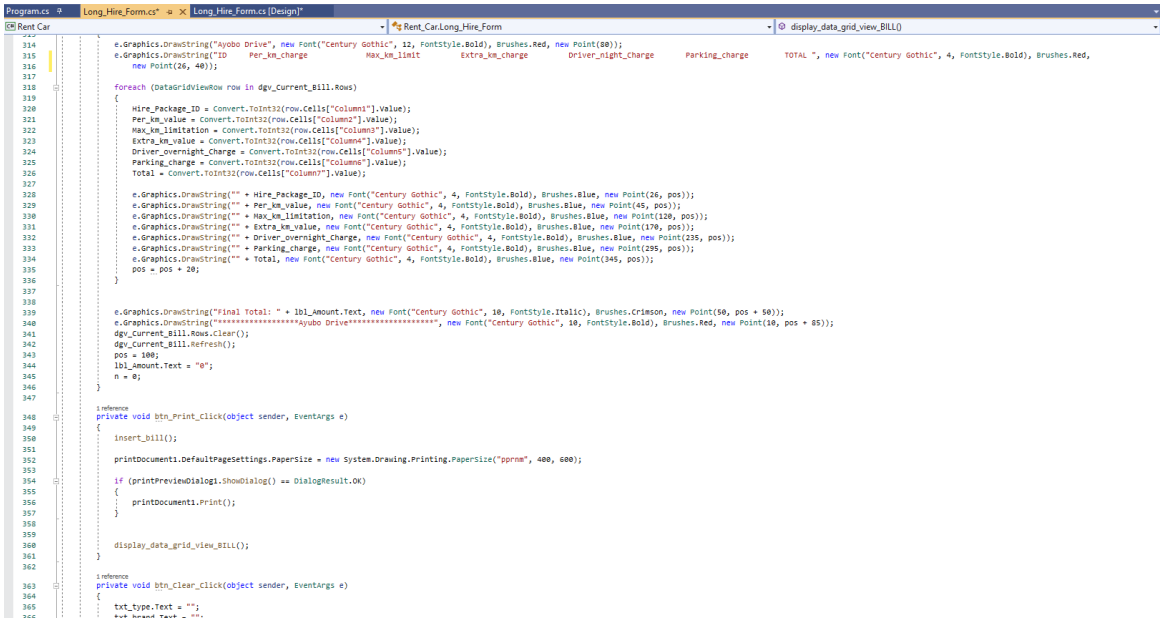
```
Program.cs   Long_Hire_Form.cs*   Long_Hire_Form.cs [Design]*
Rent Car   Rent_Car.Long_Hire_Form   btn_AddToBill_Click(object sender, EventArgs e)
209             + Convert.ToInt32(txt_total_overnight_charge.Text) + Convert.ToInt32(txt_total_parking_charge.Text));
210
211         }
212     }
213
214
215     //**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************Add to Bill Button**************
216
217
218     int finalTotal = 0;
219     int n = 0;
        1 reference
220     private void btn_AddToBill_Click(object sender, EventArgs e)
221     {
222         if (txt_type.Text == "" && txt_brand.Text == "" && txt_model.Text == "" && txt_colour.Text == "" && txt_package.Text == "" && txt_Per_km_charge.Text == "" && txt_Max_km_limit.Text == ""
223             && txt_Extra_km_charge.Text == "" && txt_driver_night_charge.Text == "" && txt_parking_charge.Text == "")
224         {
225             MessageBox.Show("Enter Package Details Please");
226         }
227         else
228         {
229
230             DataGridViewRow newRow = new DataGridViewRow();
231             newRow.CreateCells(dgv_Current_Bill);
232             newRow.Cells[0].Value = n + 1;
233             newRow.Cells[1].Value = txt_Per_km_charge.Text;
234             newRow.Cells[2].Value = txt_Max_km_limit.Text;
235             newRow.Cells[3].Value = txt_Extra_km_charge.Text;
236             newRow.Cells[4].Value = txt_driver_night_charge.Text;
237             newRow.Cells[5].Value = txt_parking_charge.Text;
238             newRow.Cells[6].Value = txt_total_hire_charge.Text;
239             dgv_Current_Bill.Rows.Add(newRow);
240
241             n++;    //ID Column value increament 1 by 1
242
243             finalTotal = finalTotal + Convert.ToInt32(txt_total_hire_charge.Text);
244
245             lbl_Amount.Text = "Rs " + finalTotal;
246
247         }
248     }
249
250     //****************Bill Transaction Records****************Bill Transaction Records****************Bill Transaction Records****************Bill Transaction Records****************
251
        1 reference
252     private void insert_bill()  //Create a new method to insert records into Bill table
253     {
254
255         try
256         {
257
258             con.Open();
259             cmd = new SqlCommand("INSERT INTO Bill_LongD_Hire_Table(Driver_Name,Date,Bill_Amount) VALUES('" + lbl_driver.Text + "' , '" + DateTime.Today.Date + "' ," +
260                 " '" + finalTotal + "' )", con);
261             cmd.ExecuteNonQuery();
```

Figure 4. 78 Driver logged Long Tour Hire Vehicle Form Code - part 5



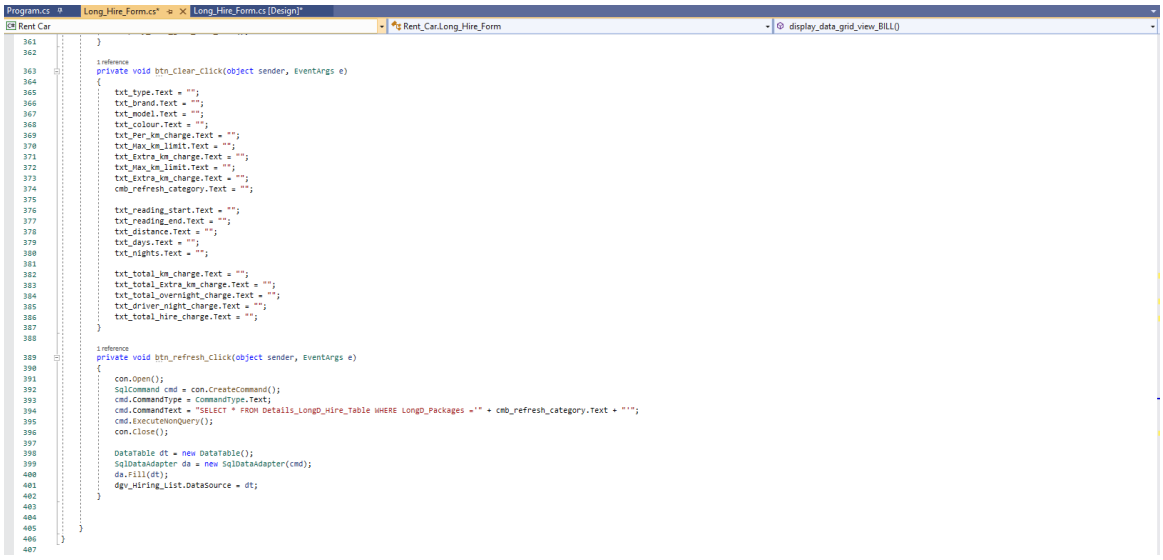
```
Program.cs   Long_Hire_Form.cs*   Long_Hire_Form.cs [Design]*
Rent Car   Rent_Car.Long_Hire_Form   btn_AddToBill_Click(object sender, EventArgs e)
261             cmd.ExecuteNonQuery();
262             con.Close();
263             MessageBox.Show("Bill added successfully!!!");
264
265             display_data_grid_view_BILL();    //data grid view method
266
267
268
269         }
270         catch (Exception ex)
271         {
272             MessageBox.Show(ex.Message);
273             con.Close();
274         }
275     }
276
        3 references
277     public void display_data_grid_view_BILL()    //For the Bill Table data grid view
278     {
279         try
280         {
281             lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
282
283             dt = new DataTable();
284             con.Open();
285             adpt = new SqlDataAdapter("SELECT * FROM Bill_LongD_Hire_Table WHERE Driver_Name = '" + lbl_driver.Text + "'", con);
286             adpt.Fill(dt);
287             dgv_Bill_Trnsaction.DataSource = dt;
288             con.Close();
289
290             //This function will be passed to printing button. --> After printing a bill, this transaction record will be passed to Bill table.
291         }
292         catch (Exception ex)
293         {
294             MessageBox.Show(ex.Message);
295             con.Close();
296         }
297     }
298
299
300
301     //**************For Bill Printing**************For Bill Printing**************For Bill Printing**************For Bill Printing**************For Bill Printing**************For Bill Printing**************
302
303     //Print Document Tool double click and configuration
304     int Hire_Package_ID;
305     int Per_km_value;
306     int Max_km_limitation;
307     int Extra_km_value;
308     int Driver_overnight_Charge;
309     int Parking_charge;
310     int Total;
311     int pos = 60;
        1 reference
312     private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
313     {
314         e.Graphics.DrawString("Ayobo Drive", new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Red, new Point(80));
```

Figure 4. 79 Driver logged Long Tour Hire Vehicle Form Code - part 6



```
Program.cs  Long_Hire_Form.cs*  Long_Hire_Form.cs [Design]*
Rent Car  Rent_Car.Long_Hire_Form  display_data_grid_view_BILL()

            e.Graphics.DrawString("Ayobo Drive", new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Red, new Point(80));
            e.Graphics.DrawString("ID     Per_km_charge          Max_km_limit        Extra_km_charge        Driver_night_Charge      Parking_charge       TOTAL ", new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Red,
                new Point(26, 40));

            foreach (DataGridViewRow row in dgv_current_Bill.Rows)
            {
                Hire_Package_ID = Convert.ToInt32(row.Cells["Column1"].Value);
                Per_km_value = Convert.ToInt32(row.Cells["Column2"].Value);
                Max_km_limitation = Convert.ToInt32(row.Cells["Column3"].Value);
                Extra_km_value = Convert.ToInt32(row.Cells["Column4"].Value);
                Driver_overnight_Charge = Convert.ToInt32(row.Cells["Column5"].Value);
                Parking_charge = Convert.ToInt32(row.Cells["Column6"].Value);
                Total = Convert.ToInt32(row.Cells["Column7"].Value);

                e.Graphics.DrawString("" + Hire_Package_ID, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(26, pos));
                e.Graphics.DrawString("" + Per_km_value, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(45, pos));
                e.Graphics.DrawString("" + Max_km_limitation, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(120, pos));
                e.Graphics.DrawString("" + Extra_km_value, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(170, pos));
                e.Graphics.DrawString("" + Driver_overnight_Charge, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(235, pos));
                e.Graphics.DrawString("" + Parking_charge, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(295, pos));
                e.Graphics.DrawString("" + Total, new Font("Century Gothic", 4, FontStyle.Bold), Brushes.Blue, new Point(345, pos));
                pos = pos + 20;
            }


            e.Graphics.DrawString("Final Total: " + lbl_Amount.Text, new Font("Century Gothic", 10, FontStyle.Italic), Brushes.Crimson, new Point(50, pos + 50));
            e.Graphics.DrawString("*****************Ayubo Drive******************", new Font("Century Gothic", 10, FontStyle.Bold), Brushes.Red, new Point(10, pos + 85));
            dgv_current_Bill.Rows.Clear();
            dgv_current_Bill.Refresh();
            pos = 100;
            lbl_Amount.Text = "0";
            n = 0;
        }

        1 reference
        private void btn_Print_Click(object sender, EventArgs e)
        {
            insert_bill();

            printDocument1.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm", 400, 600);

            if (printPreviewDialog1.ShowDialog() == DialogResult.OK)
            {
                printDocument1.Print();
            }


            display_data_grid_view_BILL();
        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_type.Text = "";
```

Figure 4. 80 Driver logged Long Tour Hire Vehicle Form Code - part 7



```
Program.cs  Long_Hire_Form.cs*  Long_Hire_Form.cs [Design]*
Rent Car  Rent_Car.Long_Hire_Form  display_data_grid_view_BILL()

        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_type.Text = "";
            txt_brand.Text = "";
            txt_model.Text = "";
            txt_colour.Text = "";
            txt_Per_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            txt_Max_km_limit.Text = "";
            txt_Extra_km_charge.Text = "";
            cmb_refresh_category.Text = "";

            txt_reading_start.Text = "";
            txt_reading_end.Text = "";
            txt_distance.Text = "";
            txt_days.Text = "";
            txt_nights.Text = "";

            txt_total_km_charge.Text = "";
            txt_total_Extra_km_charge.Text = "";
            txt_total_overnight_charge.Text = "";
            txt_driver_night_charge.Text = "";
            txt_total_hire_charge.Text = "";
        }

        1 reference
        private void btn_refresh_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Details_LongD_Hire_Table WHERE LongD_Packages ='" + cmb_refresh_category.Text + "'";
            cmd.ExecuteNonQuery();
            con.Close();

            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(dt);
            dgv_Hiring_List.DataSource = dt;
        }


    }
}
```

Figure 4. 81 Driver logged Long Tour Hire Vehicle Form Code - part 8

4.1.14 Driver logged My Profile Form Design and Code

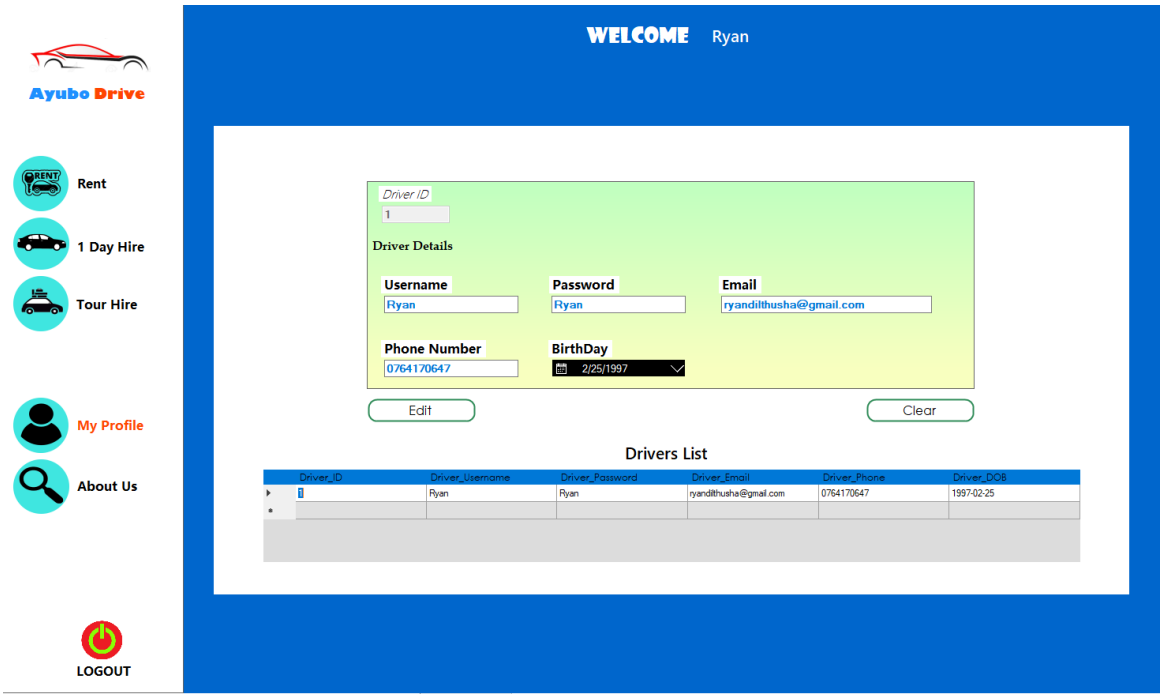
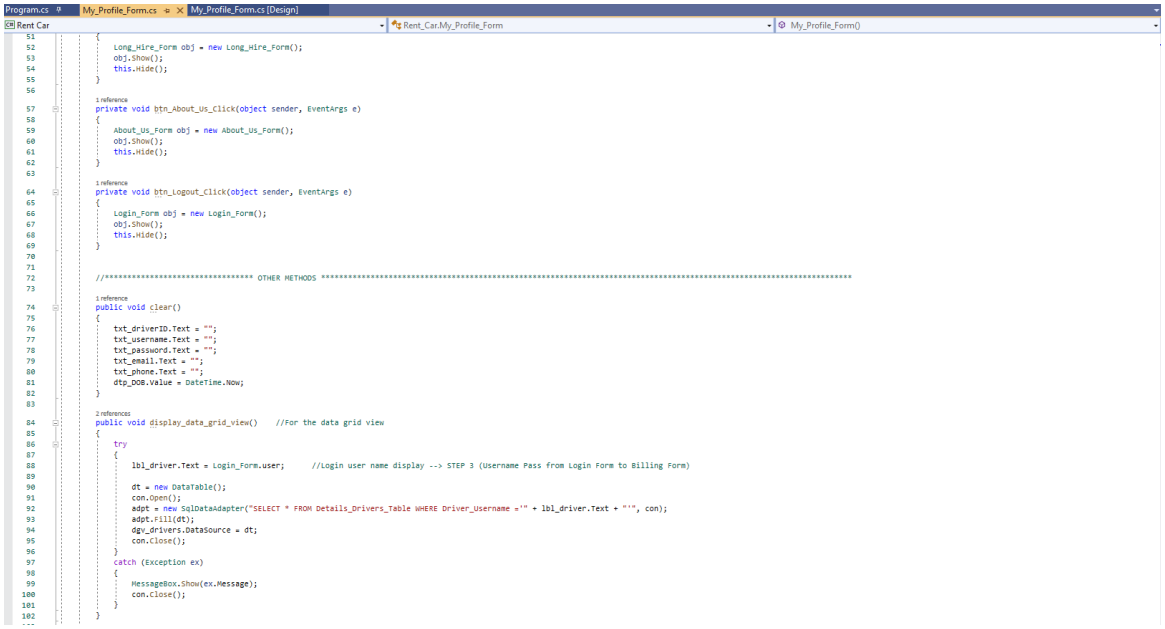


Figure 4. 82 Driver logged My Profile Form Design

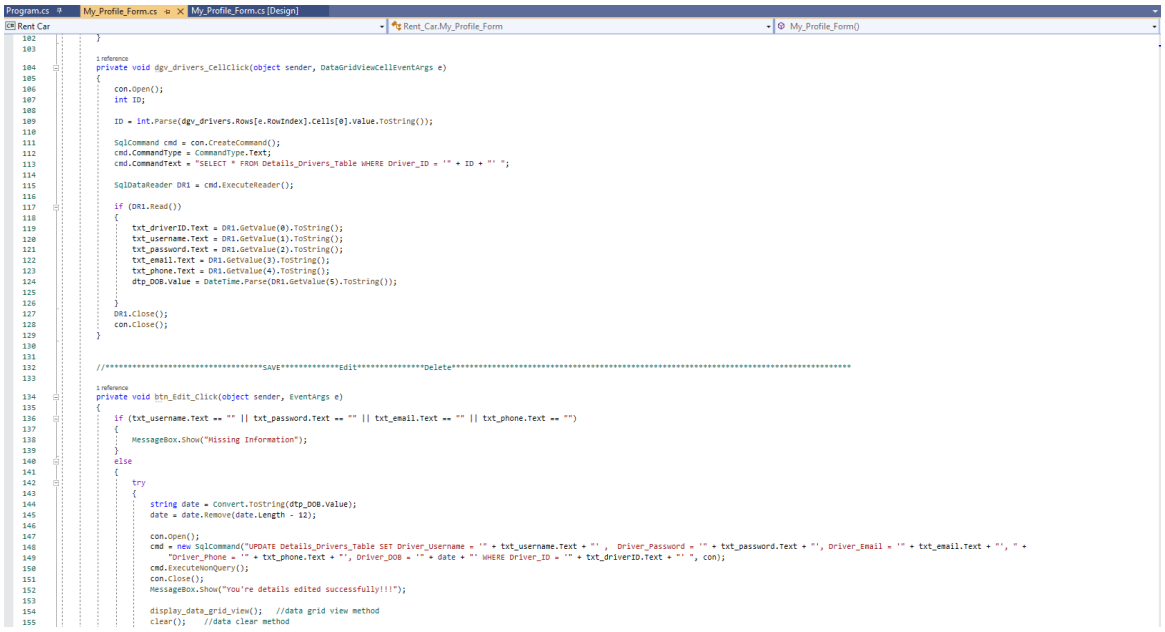
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Rent_Car
{
    11 references
    public partial class My_Profile_Form : Form
    {
        4 references
        public My_Profile_Form()
        {
            InitializeComponent();

            display_data_grid_view();

            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************************

        1 reference
        private void btn_rent_Click(object sender, EventArgs e)
        {
            Rent_Form obj = new Rent_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            _1D_Hire_Form obj = new _1D_Hire_Form();
            obj.Show();
            this.Hide();
        }

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

Figure 4. 83 Driver logged My Profile Form Code - part 1



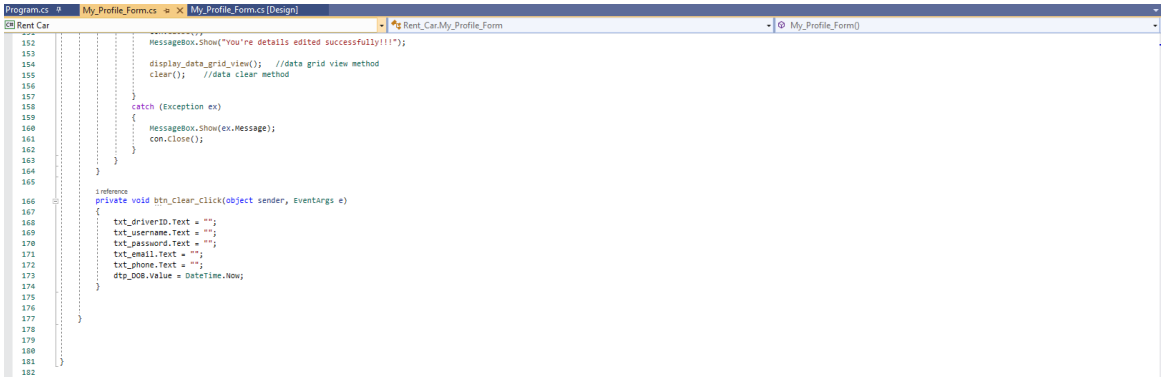
```
        {
            Long_Hire_Form obj = new Long_Hire_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_About_Us_Click(object sender, EventArgs e)
        {
            About_Us_Form obj = new About_Us_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        //******************************** OTHER METHODS ********************************************************************************************************************

        1 reference
        public void clear()
        {
            txt_driverID.Text = "";
            txt_username.Text = "";
            txt_password.Text = "";
            txt_email.Text = "";
            txt_phone.Text = "";
            dtp_DOB.Value = DateTime.Now;
        }

        2 references
        public void display_data_grid_view()    //For the data grid view
        {
            try
            {
                lbl_driver.Text = Login_Form.user;      //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)

                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Details_Drivers_Table WHERE Driver_Username ='" + lbl_driver.Text + "'", con);
                adpt.Fill(dt);
                dgv_drivers.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
```

Figure 4. 84 Driver logged My Profile Form Code - part 2



```
Program.cs | My_Profile_Form.cs | My_Profile_Form.cs [Design]
Rent Car | Rent_Car.My_Profile_Form | My_Profile_Form()
102         }
103
104         1 reference
            private void dgv_drivers_CellClick(object sender, DataGridViewCellEventArgs e)
105         {
106             con.Open();
107             int ID;
108
109             ID = int.Parse(dgv_drivers.Rows[e.RowIndex].Cells[0].Value.ToString());
110
111             SqlCommand cmd = con.CreateCommand();
112             cmd.CommandType = CommandType.Text;
113             cmd.CommandText = "SELECT * FROM Details_Drivers_Table WHERE Driver_ID = '" + ID + "' ";
114
115             SqlDataReader DR1 = cmd.ExecuteReader();
116
117             if (DR1.Read())
118             {
119                 txt_driverID.Text = DR1.GetValue(0).ToString();
120                 txt_username.Text = DR1.GetValue(1).ToString();
121                 txt_password.Text = DR1.GetValue(2).ToString();
122                 txt_email.Text = DR1.GetValue(3).ToString();
123                 txt_phone.Text = DR1.GetValue(4).ToString();
124                 dtp_DOB.Value = DateTime.Parse(DR1.GetValue(5).ToString());
125
126             }
127             DR1.Close();
128             con.Close();
129         }
130
131
132         //************************************SAVE************Edit**************Delete*****************************************************************************************
133
134         1 reference
            private void btn_Edit_Click(object sender, EventArgs e)
135         {
136             if (txt_username.Text == "" || txt_password.Text == "" || txt_email.Text == "" || txt_phone.Text == "")
137             {
138                 MessageBox.Show("Missing Information");
139             }
140             else
141             {
142                 try
143                 {
144                     string date = Convert.ToString(dtp_DOB.Value);
145                     date = date.Remove(date.Length - 12);
146
147                     con.Open();
148                     cmd = new SqlCommand("UPDATE Details_Drivers_Table SET Driver_Username = '" + txt_username.Text + "' ,  Driver_Password = '" + txt_password.Text + "', Driver_Email = '" + txt_email.Text + "', " +
149                         "Driver_Phone = '" + txt_phone.Text + "', Driver_DOB = '" + date + "' WHERE Driver_ID = '" + txt_driverID.Text + "' ", con);
150                     cmd.ExecuteNonQuery();
151                     con.Close();
152                     MessageBox.Show("You're details edited successfully!!!");
153
154                     display_data_grid_view();   //data grid view method
155                     clear();    //data clear method
```

Figure 4. 85 Driver logged My Profile Form Code - part 3



```
Program.cs | My_Profile_Form.cs | My_Profile_Form.cs [Design]
Rent Car | Rent_Car.My_Profile_Form | My_Profile_Form()
152                     MessageBox.Show("You're details edited successfully!!!");
153
154                     display_data_grid_view();   //data grid view method
155                     clear();    //data clear method
156
157                 }
158                 catch (Exception ex)
159                 {
160                     MessageBox.Show(ex.Message);
161                     con.Close();
162                 }
163             }
164         }
165
166         1 reference
            private void btn_Clear_Click(object sender, EventArgs e)
167         {
168             txt_driverID.Text = "";
169             txt_username.Text = "";
170             txt_password.Text = "";
171             txt_email.Text = "";
172             txt_phone.Text = "";
173             dtp_DOB.Value = DateTime.Now;
174         }
175
176
177     }
178
179
180
181 }
182
```

Figure 4. 86 Driver logged My Profile Form Code - part 4

4.1.15 Driver logged About Us Form Design and Code

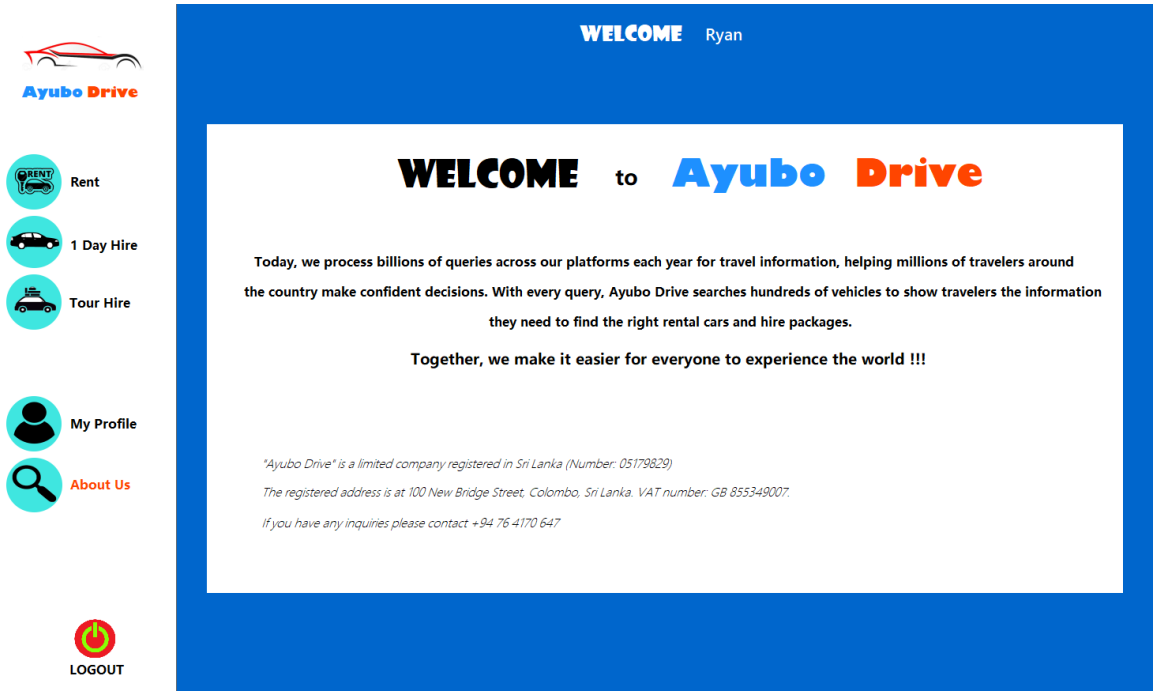
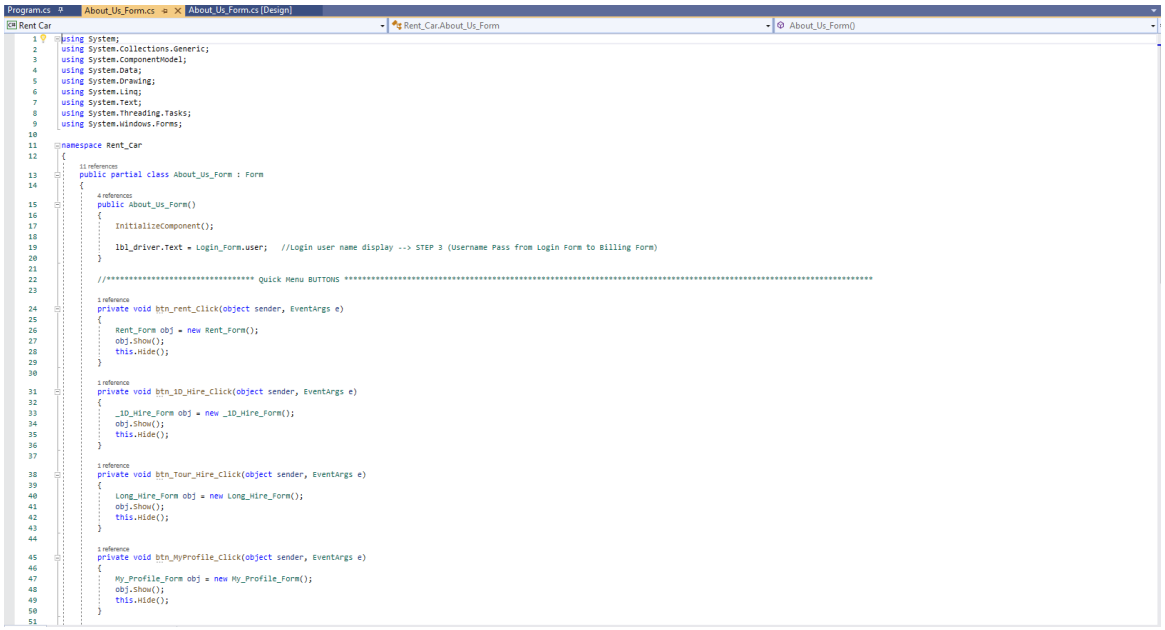
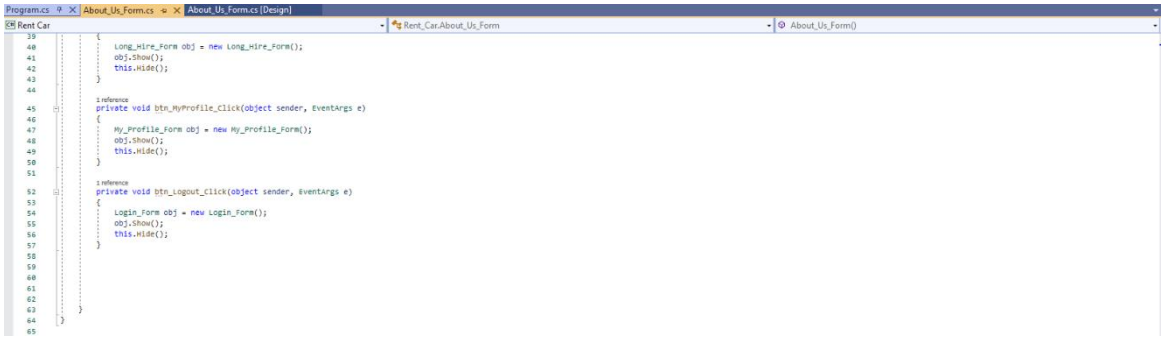


Figure 4. 87 Driver logged About Us Form Design



```
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 Rent_Car
{
    11 references
    public partial class About_Us_Form : Form
    {
        4 references
        public About_Us_Form()
        {
            InitializeComponent();

            lbl_driver.Text = Login_Form.user;   //Login user name display --> STEP 3 (Username Pass from Login Form to Billing Form)
        }

        //********************************** Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_rent_Click(object sender, EventArgs e)
        {
            Rent_Form obj = new Rent_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_1D_Hire_Click(object sender, EventArgs e)
        {
            _1D_Hire_Form obj = new _1D_Hire_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Tour_Hire_Click(object sender, EventArgs e)
        {
            Long_Hire_Form obj = new Long_Hire_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_MyProfile_Click(object sender, EventArgs e)
        {
            My_Profile_Form obj = new My_Profile_Form();
            obj.Show();
            this.Hide();
        }
```

Figure 4. 88 Driver logged About Us Form Code - part 1



```
        {
            Long_Hire_Form obj = new Long_Hire_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_MyProfile_Click(object sender, EventArgs e)
        {
            My_Profile_Form obj = new My_Profile_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_Logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }



    }
}
```

Figure 4. 89 Driver logged About Us Form Code - part 2

4.2 Microsoft SQL table designs and queries

4.2.1 Vehicle Details Table

```
CREATE DATABASE Rent_Car_Project;

CREATE TABLE [dbo].[Details_Vehicle_Table] (
    [Veh_ID]     INT          IDENTITY (1, 1) NOT NULL,
    [Veh_Type]   VARCHAR (50) NOT NULL,
    [Veh_RegNo]  VARCHAR (50) NOT NULL,
    [Veh_Brand]  VARCHAR (50) NOT NULL,
    [Veh_Model]  VARCHAR (50) NOT NULL,
    [Veh_Colour] VARCHAR (50) NOT NULL,
    [Own_Name]   VARCHAR (50) NOT NULL,
    [Own_ID]     VARCHAR (50) NOT NULL,
    [Own_Phone]  VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([Veh_ID] ASC)
);
```

Figure 4. 90 Vehicle Details Table SQL query

DESKTOP-44KSVRU.R...ails_Vehicle_Table | SQLQuery8.sql - DE...4KSVRU\ryand

Column Name	Data Type	Allow Nulls
Veh_ID	int	☐
Veh_Type	varchar(50)	☐
Veh_RegNo	varchar(50)	☐
Veh_Brand	varchar(50)	☐
Veh_Model	varchar(50)	☐
Veh_Colour	varchar(50)	☐
Own_Name	varchar(50)	☐
Own_ID	varchar(50)	☐
Own_Phone	varchar(50)	☐
		☐

Figure 4. 91 Vehicle Details Table SQL table design

4.2.2 Vehicle Renting Details Table

```
CREATE TABLE [dbo].[Details_Renting_Table] (
    [Rent_ID]          INT           IDENTITY (1, 1) NOT NULL,
    [Veh_Type]         VARCHAR (50)  NOT NULL,
    [Veh_Brand]        VARCHAR (50)  NOT NULL,
    [Veh_Model]        VARCHAR (50)  NOT NULL,
    [Veh_Colour]       VARCHAR (50)  NOT NULL,
    [Rent_Daily]       INT           NOT NULL,
    [Rent_Weekly]      INT           NOT NULL,
    [Rent_Monthly]     INT           NOT NULL,
    [Rent_DriverRate] INT           NOT NULL,
    PRIMARY KEY CLUSTERED ([Rent_ID] ASC)
);
```

Figure 4. 92 Vehicle Renting Details Table SQL query

DESKTOP-44KSVRU.R...ails_Renting_Table | Object Explorer Details

Column Name	Data Type	Allow Nulls
Rent_ID	int	☐
Veh_Type	varchar(50)	☐
Veh_Brand	varchar(50)	☐
Veh_Model	varchar(50)	☐
Veh_Colour	varchar(50)	☐
Rent_Daily	int	☐
Rent_Weekly	int	☐
Rent_Monthly	int	☐
Rent_DriverRate	int	☐
		☐

Figure 4. 93 Vehicle Renting Details Table SQL table design

4.2.3 Vehicle One Day Hire Details Table

```
CREATE TABLE [dbo].[Details_OneD_Hire_Table] (
    [OneD_ID]                INT          IDENTITY (1, 1) NOT NULL,
    [Veh_Type]               VARCHAR (50) NOT NULL,
    [Veh_Brand]              VARCHAR (50) NOT NULL,
    [Veh_Model]              VARCHAR (50) NOT NULL,
    [Veh_Colour]             VARCHAR (50) NOT NULL,
    [OneD_Packages]          VARCHAR (50) NOT NULL,
    [OneD_Per_km_charge]     INT          NOT NULL,
    [OneD_Max_km_limit]      INT          NOT NULL,
    [OneD_Extra_km_charge]   INT          NOT NULL,
    [OneD_Max_hour_limit]    INT          NOT NULL,
    [OneD_Extra_hour_charge] INT          NOT NULL,
    PRIMARY KEY CLUSTERED ([OneD_ID] ASC)
);
```

Figure 4. 94 Vehicle One Day Hire Details Table SQL query

DESKTOP-44KSVRU....s_OneD_Hire_Table | DESKTOP-44KSVRU.R...ail

Column Name	Data Type	Allow Nulls
OneD_ID	int	☐
Veh_Type	varchar(50)	☐
Veh_Brand	varchar(50)	☐
Veh_Model	varchar(50)	☐
Veh_Colour	varchar(50)	☐
OneD_Packages	varchar(50)	☐
OneD_Per_km_charge	int	☐
OneD_Max_km_limit	int	☐
OneD_Extra_km_charge	int	☐
OneD_Max_hour_limit	int	☐
OneD_Extra_hour_charge	int	☐
		☐

Figure 4. 95 Vehicle One Day Hire Details Table SQL table design

4.2.4 Vehicle Long Tour Hire Details Table

```
CREATE TABLE [dbo].[Details_LongD_Hire_Table] (
    [LongD_ID]                      INT          IDENTITY (1, 1) NOT NULL,
    [Veh_Type]                      VARCHAR (50) NOT NULL,
    [Veh_Brand]                     VARCHAR (50) NOT NULL,
    [Veh_Model]                     VARCHAR (50) NOT NULL,
    [Veh_Colour]                    VARCHAR (50) NOT NULL,
    [LongD_Packages]                VARCHAR (50) NOT NULL,
    [LongD_Per_km_charge]           INT          NOT NULL,
    [LongD_Max_km_limit]            INT          NOT NULL,
    [LongD_Extra_km_charge]         INT          NOT NULL,
    [LongD_Driver_overnight_charge] INT          NOT NULL,
    [LongD_Vehicle_parking_charge]  INT          NOT NULL,
    PRIMARY KEY CLUSTERED ([LongD_ID] ASC)
);
```

Figure 4. 96 Vehicle One Day Hire Details Table SQL query

DESKTOP-44KSVRU...._LongD_Hire_Table | DESKTOP-44KSVRU....s

Column Name	Data Type	Allow Nulls
LongD_ID	int	☐
Veh_Type	varchar(50)	☐
Veh_Brand	varchar(50)	☐
Veh_Model	varchar(50)	☐
Veh_Colour	varchar(50)	☐
LongD_Packages	varchar(50)	☐
LongD_Per_km_charge	int	☐
LongD_Max_km_limit	int	☐
LongD_Extra_km_charge	int	☐
LongD_Driver_overnight_...	int	☐
LongD_Vehicle_parking_...	int	☐
		☐

Figure 4. 97 Vehicle One Day Hire Details Table SQL table design

4.2.5 Driver Details Table

```
CREATE TABLE [dbo].[Details_Drivers_Table] (
    [Driver_ID]       INT           IDENTITY (1, 1) NOT NULL,
    [Driver_Username] VARCHAR (50) NOT NULL,
    [Driver_Password] VARCHAR (50) NOT NULL,
    [Driver_Email]    VARCHAR (50) NOT NULL,
    [Driver_Phone]    VARCHAR (50) NOT NULL,
    [Driver_DOB]      VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([Driver_ID] ASC)
);
```

Figure 4. 98 Driver Details Table SQL query

DESKTOP-44KSVRU.Re...ails_Drivers_Table | DESKTOP-44KSVRU...._

Column Name	Data Type	Allow Nulls
Driver_ID	int	☐
Driver_Username	varchar(50)	☐
Driver_Password	varchar(50)	☐
Driver_Email	varchar(50)	☐
Driver_Phone	varchar(50)	☐
Driver_DOB	varchar(50)	☐
		☐

Figure 4. 99 Driver Details Table SQL table design

4.2.6 Vehicle Renting Bills Details Table

```
CREATE TABLE [dbo].[Bill_Rent_Table] (
    [Bill_ID]         INT           IDENTITY (1, 1) NOT NULL,
    [Driver_Name]     VARCHAR (50) NOT NULL,
    [Date] VARCHAR(50)           NOT NULL,
    [Bill_Amount]     INT           NOT NULL,
    PRIMARY KEY CLUSTERED ([Bill_ID] ASC)
);
```

Figure 4. 100 Vehicle Renting Bills Details Table SQL query

DESKTOP-44KSVRU.R...bo.Bill_Rent_Table | DESKTOP-44KSVRU.Re...a

Column Name	Data Type	Allow Nulls
Bill_ID	int	☐
Driver_Name	varchar(50)	☐
Date	varchar(50)	☐
Bill_Amount	int	☐
		☐

Figure 4. 101 Vehicle Renting Bills Details Table SQL table design

4.2.7 Vehicle One Day Hire Bills Details Table

```
CREATE TABLE [dbo].[Bill_OneD_Hire_Table] (
    [Bill_ID]         INT          IDENTITY (1, 1) NOT NULL,
    [Driver_Name]     VARCHAR (50) NOT NULL,
    [Date] VARCHAR(50)          NOT NULL,
    [Bill_Amount]     INT          NOT NULL,
    PRIMARY KEY CLUSTERED ([Bill_ID] ASC)
);
```

Figure 4. 102 Vehicle One Day Hire Bills Details Table SQL query

DESKTOP-44KSVRU....l_OneD_Hire_Table | DESKTOP-44KSVRU.R

Column Name	Data Type	Allow Null
Bill_ID	int	☐
Driver_Name	varchar(50)	☐
Date	varchar(50)	☐
Bill_Amount	int	☐
		☐

Figure 4. 103 Vehicle One Day Hire Bills Details Table SQL table design

4.2.8 Vehicle Long Tour Hire Bills Details Table

```
CREATE TABLE [dbo].[Bill_LongD_Hire_Table] (
    [Bill_ID]         INT           IDENTITY (1, 1) NOT NULL,
    [Driver_Name]     VARCHAR (50) NOT NULL,
    [Date] VARCHAR(50)           NOT NULL,
    [Bill_Amount]     INT           NOT NULL,
    PRIMARY KEY CLUSTERED ([Bill_ID] ASC)
);
```

Figure 4. 104 Vehicle Long Tour Hire Bills Details Table SQL query

DESKTOP-44KSVRU...._LongD_Hire_Table | DESKTOP-44KSVRU....I_O

Column Name	Data Type	Allow Nulls
Bill_ID	int	☐
Driver_Name	varchar(50)	☐
Date	varchar(50)	☐
Bill_Amount	int	☐
		☐

Figure 4. 105 Vehicle Long Tour Hire Bills Details Table SQL table design

4.2.9 SQL Database connecting Visual Studio test connection

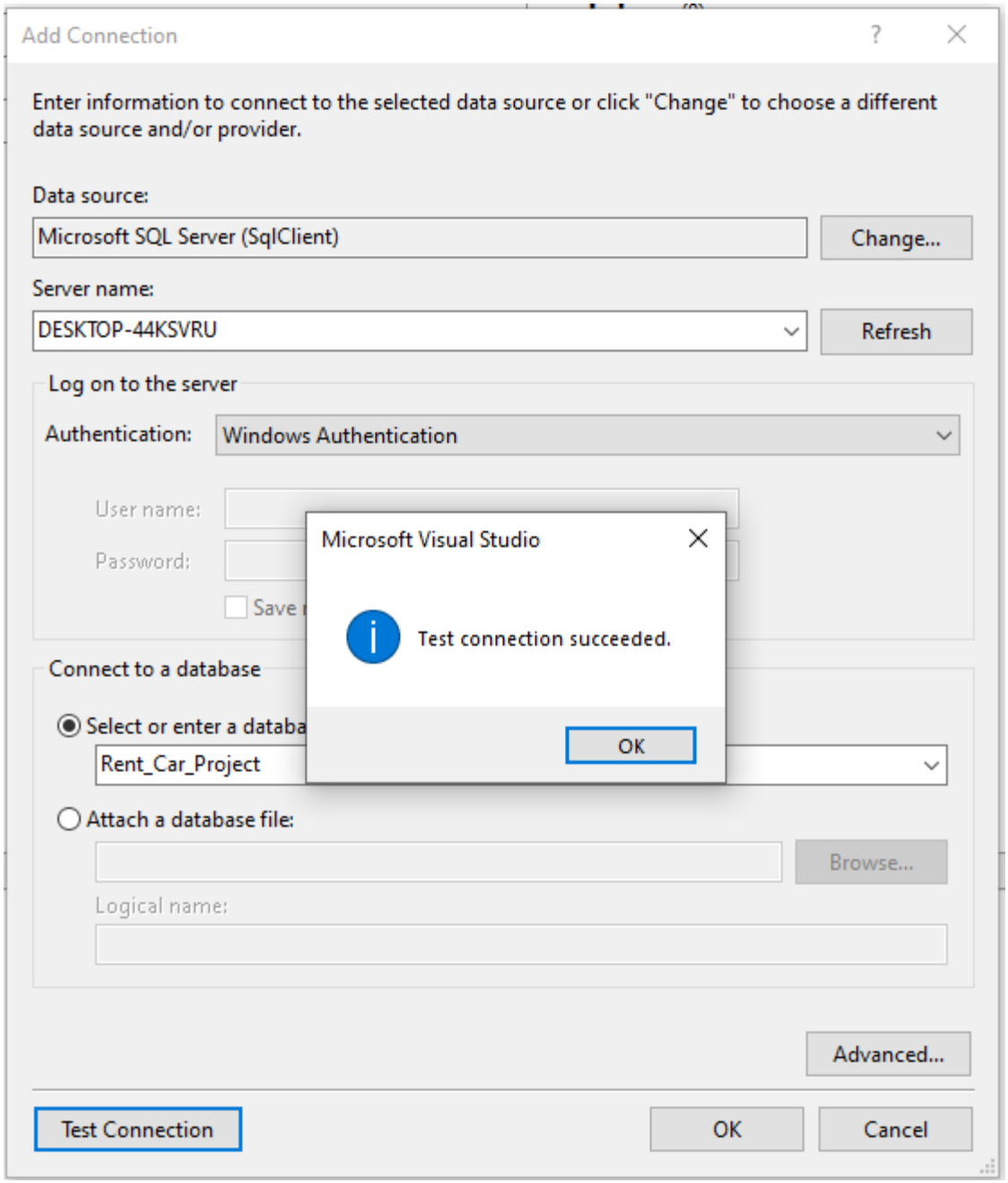


Figure 4. 106 SQL Database connecting Visual Studio test connection

4.3 Debugging

4.3.1 What is debugging in Visual studio IDE

Despite our best efforts, the code we write as software engineers does not always perform as predicted. It can also do something entirely unexpected! When this happens, the next step is to figure out why, and while we may be tempted to just stare at our code for hours. But by using a debugging tool or debugger is helping to resolve those problems far easier and more efficient.

Unfortunately, a debugger will not suddenly expose all of the issues or "bugs" in our code. Debugging is the process of running the code in a debugging tool, such as Visual Studio, step by step to determine where programmers committed a programming error. So, programmers can then see what modifications they need to make to the code, and debugging tools frequently allow them to make temporary changes while the program is still running. Effectively using a debugger is a skill that takes time and experience to master, but it is a must-have for any software engineer.

4.3.2 Why do we need Debugging?

An exception, often known as an error, is an unexpected event that occurs while running code. A debugging tool can lead programmers to the specific spot in the code where the error occurred and help them in coming up with solutions. If code displays some text but the text is incorrect, programmers know that either the data is incorrect or the code that sets the display text has a bug. They can analyze each and every modification to variables by stepping through the code in a debugger to see when and how wrong values are assigned.

When we start an app normally, we only see bugs after the code has completed. A program may also suddenly end without informing us of the reason. When we run an app in a debugger, also known as debugging mode, the debugger actively monitors everything that happens while the program is running. It also enables us to pause the app at any time to inspect its current state, and then move through our code line by line to inspect every detail as it occurs.

4.3.3 Debugging process in Visual studio IDE

We can enter debugging mode in Visual Studio by using following methods.

- F5 key in the keyboard
- Debug tab → Start Debugging menu command
- Start Debugging button in quick buttons bar

If an exception occurs, the Exception Helper in Visual Studio takes us to the exact location where the error happened and provides more details.

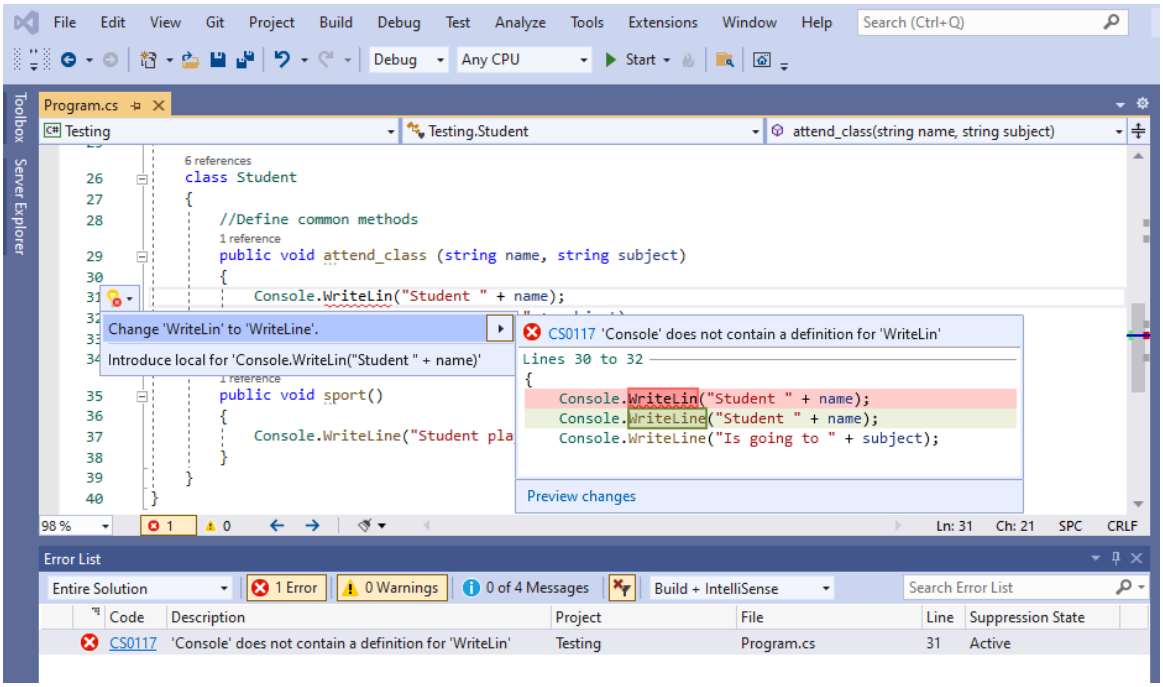


Figure 4. 107 Debugging flaws of the code through Visual Studio IDE

If we didn't get an exception, we should be able to figure out where the flaw is in our code. This is where we use the debugger's breakpoints to give us a chance to look over our code more closely. Breakpoints are the most fundamental and important aspect of a reliable debugging system. A breakpoint is a place in the running code where Visual Studio should pause it so we may examine the values of variables, memory behavior, or the order in which code runs. By clicking in the left margin next to a line of code in Visual Studio, we can immediately set a breakpoint. Alternatively, position mouse cursor on a line and hit F9.

4.3.4 Steps involved in Debugging process

Figure 4. 108 Steps involved in Debugging process

Step 01 : Identify the error →

A faulty error identification can lead to a waste development time. It is common for users to report production mistakes that are difficult to interpret, and the information we receive can occasionally be misleading. It's crucial to figure out what's wrong.

Step 02 : Find the error location →

We need to walk through the code to pinpoint the exact area where the issue is found after accurately detecting the error. At this step, we should concentrate on locating the issue rather than analyzing and understanding it.

Step 03 : Analyze the error →

In the third step, we must evaluate the code using a bottom-up technique starting from the error location. This will support us in understanding the error. Analyzing a bug has two basic goals: finding more faults around the error and ensuring that there are no risks of introducing collateral damage into the solution.

Step 04 : Prove the Analysis→

We need to identify a few more mistakes that may emerge on the program once we've finished examining the original bug. With the use of a test framework, this step involves building automated tests for these areas.

Step 05 : Cover Lateral Damage→

We must build or gather all unit tests for the code where we will make modifications at this step. These unit tests should now all pass if we run them.

Step 06 : Fix and Validate→

The final step is to correct all of the mistakes and run all of the test scripts to ensure that they all succeed.

4.3.5 Strategies to follow when debugging

- To fully comprehend the system, it is necessary to examine it in depth. It supports the debugger in constructing various models of the systems to be debugged.
- Backward analysis of the problem tracks the program backward from the place of the error indication to find the defective code portion. To determine the cause of flaws, we must thoroughly investigate the defect area.
- Forward analysis of a program consists of using breakpoints or print statements at various places throughout the program to track it forward. It's critical to focus on the area where incorrect results are obtained.
- To check for similar problems, we must recall previous experience. The debugger's skill is critical to the success of this strategy.

4.4 How I used the debugging process to develop more secure, robust application with examples

4.4.1 How I used the debugging process

I've built a small program to show the smallest 3 numbers of a list. As in below figure code 1st line defines the list of integers starting with number 1 and ending with number 6. Then in the 2nd line user defined method Get Smallest Number method is being called to catch 3 smallest numbers of the list. Then in the 3rd line is created to iterate these 3 numbers and display 1 by 1 in the console.

But and you can see in the output in the console, this is not the result we've been expected. This is totally opposed of the expected result. This is not the list of smallest numbers. Hence, we have to go deep of to the code first.

```
0 references
class Program
{
    0 references
    public static void Main(string[] args)
    {
        var numbers = new List<int> { 1, 2, 3, 4, 5, 6 };
        var smallest = GetSmallNumbers(numbers, 3);

        foreach (var number in smallest)
            Console.WriteLine(number);
        Console.ReadLine();

    }

    1 reference
    public static List<int> GetSmallNumbers(List<int> list, int count)
    {
        var smallests = new List<int>();

        while (smallests.Count < count)
        {
            var min = GetSmallestNum(list);
            smallests.Add(min);
            list.Remove(min);
        }

        return smallests;
    }

    1 reference
    public static int GetSmallestNum(List<int> list)
    {
        var min = list[0];
        for (var i = 1; i < list.Count; i ++)
        {
            if (list[i] > min)
                min = list[i];
        }
        return min;
    }
}
```

```
C:\Users\ryand\source\repos\Testing\Testing\bin\Debug\Testing.exe
6
5
4
```

Figure 4. 109 C# e code with the having unexpected Console Output.

When we debug the program usually, we do is put 1 or more breakpoints in the program then run the application in debug mode. When we run the application in debug mode, the execution stops at the breakpoints. And there we can inspect the values of different variables to make sure they have right expected value. If not, we know there is a problem with the code that need to be change.

In the figure below there is a breakpoint at the beginning of the code. Breakpoint is represented with a red circle at the left margin. By pressing F9 key, we can insert a breakpoint. And now we can run the application in debug mode with the help of the key F5. Eventually the program execution is stopped at the breakpoint line.

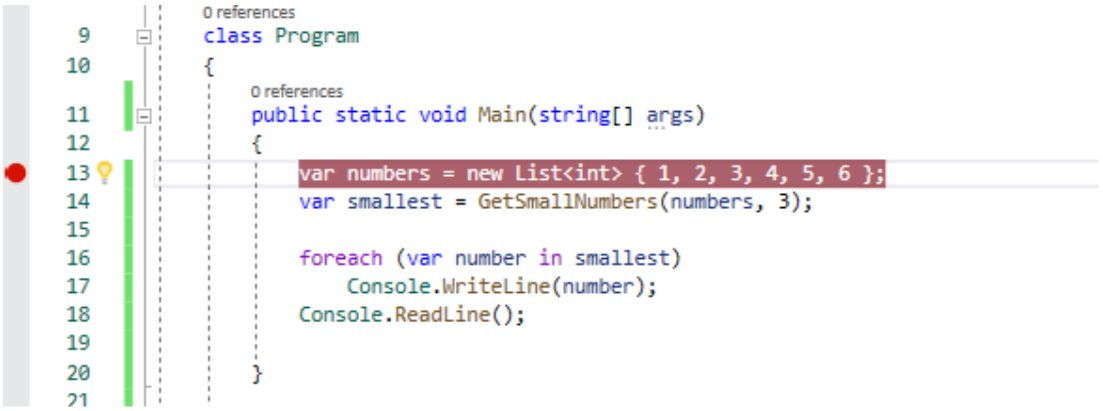


Figure 4. 110 Breakpoint at the 1st line of the code

At this point I can look values of different variables, or I can continue the execution. There are few ways to continue the execution. One of common way is by “step over” with the help of the key F10. This step over is doing is executing 1 line at a time. The yellow arrow at the left margin is representing step over line.

When I move mouse curser on variable of executed lines by step over, we can see the variables stored values at that time. As in below figure the values stored in “smallest” variable is 6,5,4 which is totally different of expected values.

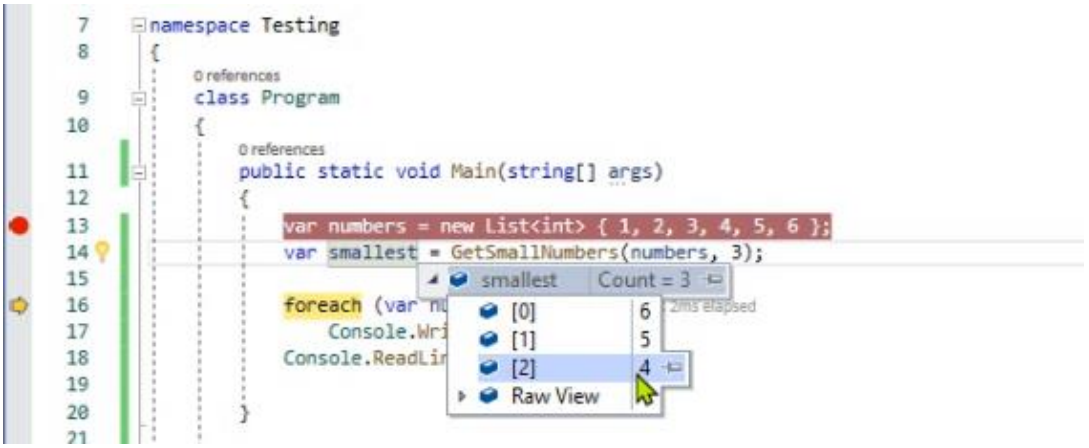


Figure 4. 111 Smallest variable stored values

Then I want to know what is happening in the Get Small Number method. For that I'm going to use debugging tool "step into" with the help of the key F11. In will step into the method Get Small Number. The reason I added a breakpoint here because I don't have to use F10 over and over again to come here. Next time when I run the application in debug mode I can jump here straightway.

When we look into min variable the stored value is 6 which is not expected value. When we look into list, we can see the entire list we've given which is ok here. So, we can determine the problem is with the Get Smallest Num method. So we can step into that method by F11 key and investigate it.

Figure 4. 112 min variable stored values

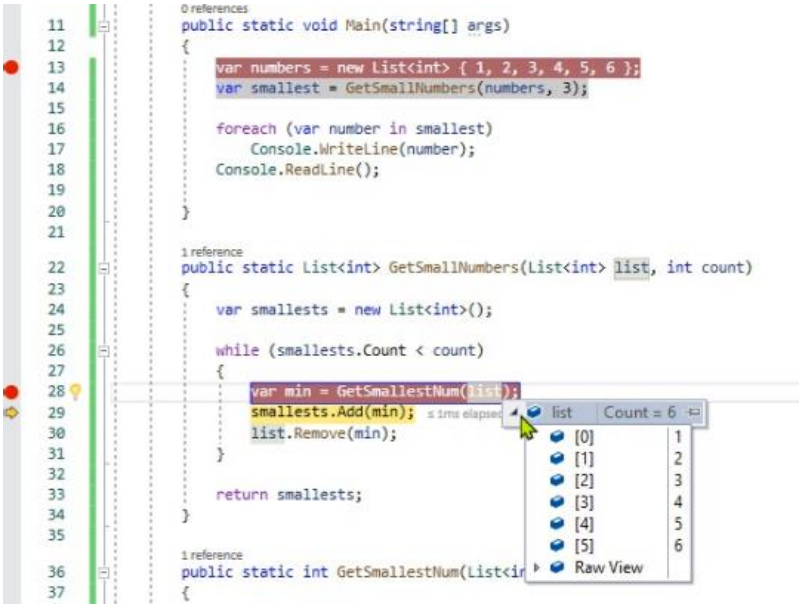


Figure 4. 113 List variable stored values

Then this is a very useful tool for debugger. It can be reach by Debug Tab → Windows → Watch → Watch1.

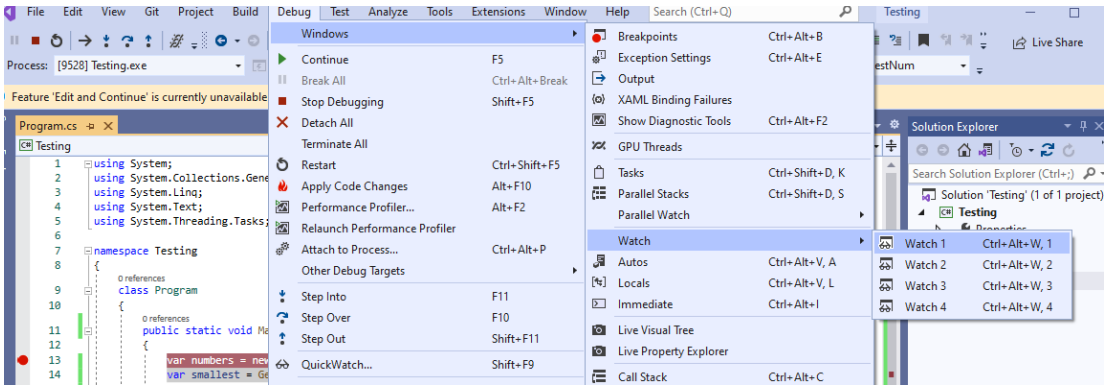


Figure 4. 114 Watch 1 windows tab open by Visual Studio IDE

The use of this is we can put our interested variables into this windows tab and can monitor the values of them. Hence, we don't have to hover our mouse over every variable we want to look up again and again.

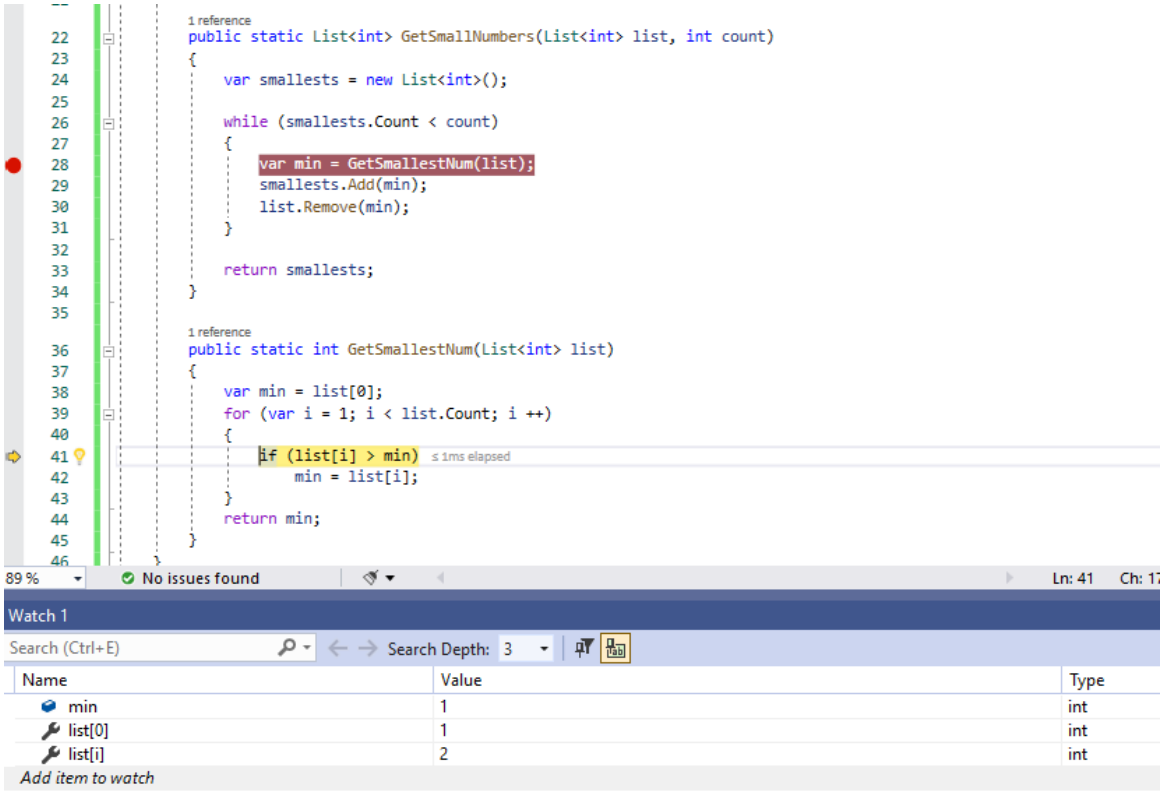


Figure 4. 115 Monitor interested variable values by watch tab.

According to above figure the value of min is 2. But at the execution at the line 41 the value of the list[i] in watch window is 2. The current value of the "i" is 2. We are comparing this with min variable. We are saying here if the list of "i" is greater than min, set min in the list of "i". This proves that we are looking for the larger number here. But the minimum. So, this is the part where the bug has. After fixing this as below figure we can "step out" from the method by shortcut key Shift + F11.

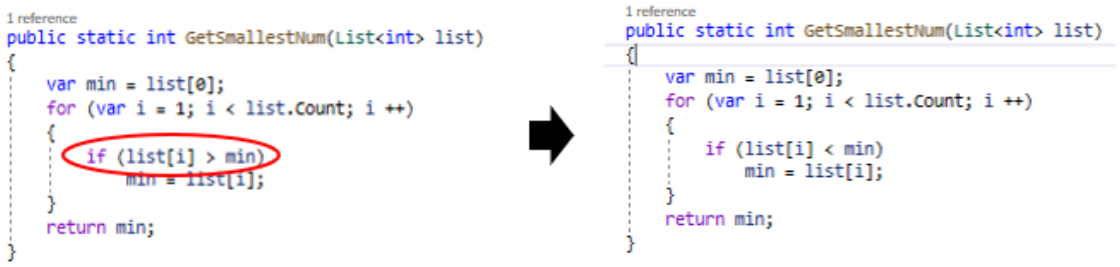


Figure 4. 116 Fixing the bug

Now when we look into the min variable value, it is as we expected as in the below figure.

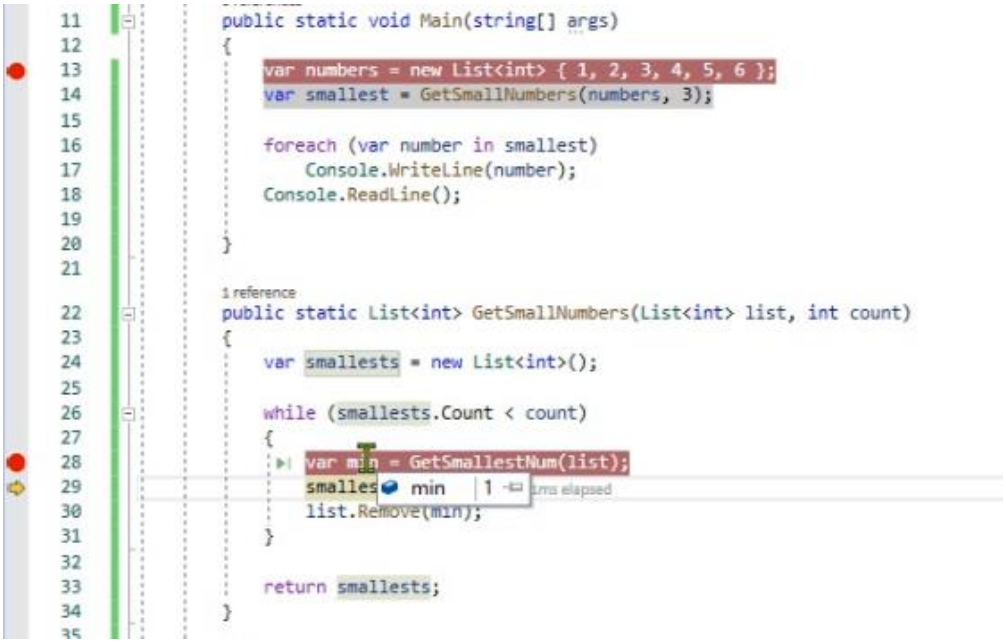


Figure 4. 117 min variable value after debugging

So now we can stop the debugger and we can run the application with normal mode without debugging to see if we get the right results in the console. To stop the debugging, we can use the key F5.

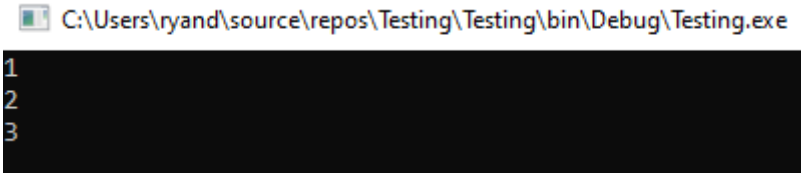


Figure 4. 118 Expected output after debugging

4.4.2 How I handled some little bugs of the program easily

Visual Studio C# throws an exception when something goes wrong. The exception basically informs us that something went wrong in the code that C# couldn't manage. When this happens, the software usually crashes. When I spend too much time on the code, I don't always want an exception to crash the program. I handle exceptions so that the application's usual flow can be maintained even when runtime issues occur.

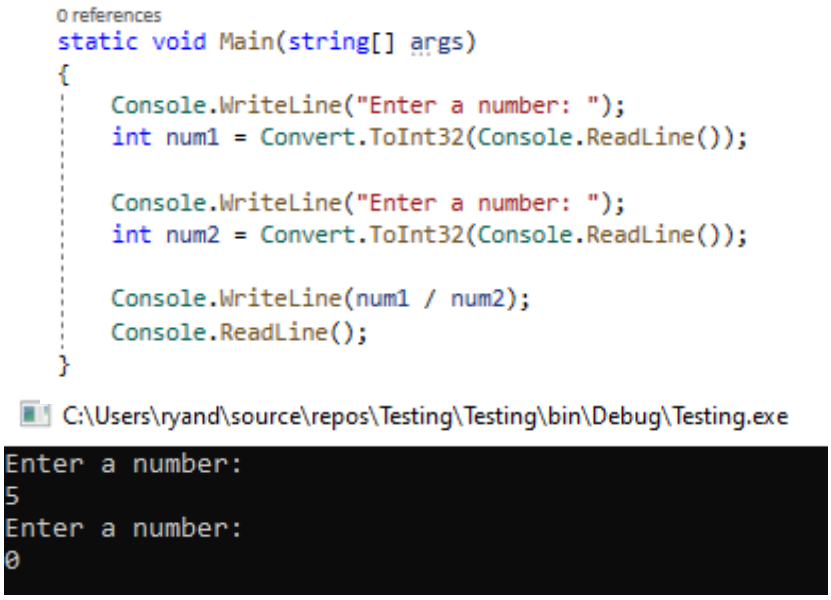


Figure 4. 119 Basic calculator example code with the given parameters for user inputs.

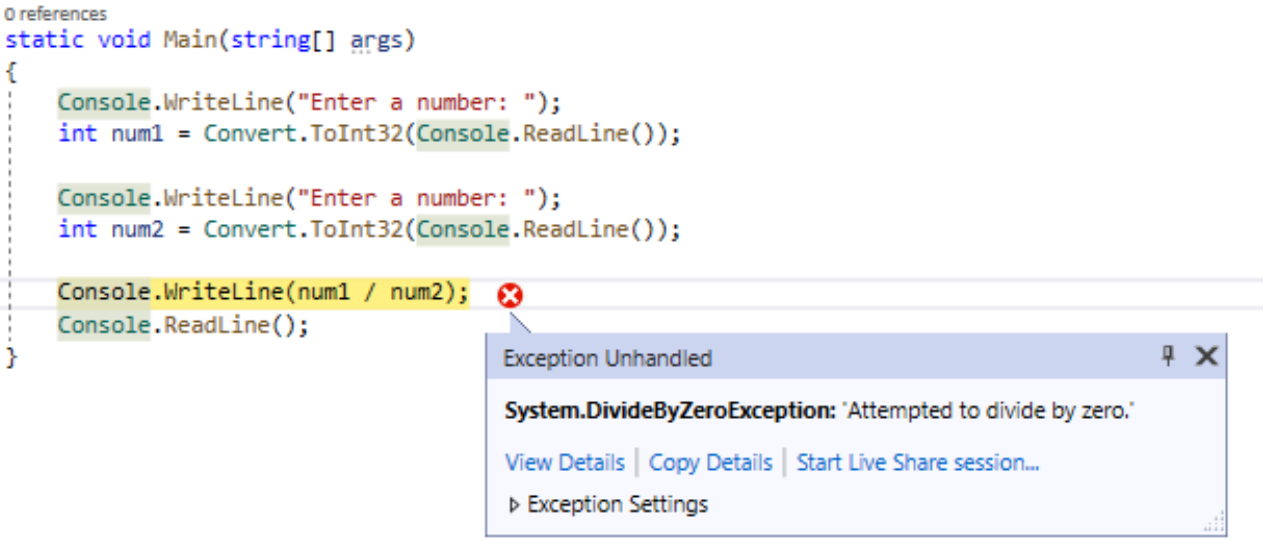


Figure 4. 120 Exception occur when run the program

As in above figure this error occurs because simple mathematics, we can't divide an integer with zero. It's impossible for us to and to computer to do this math.

We don't want the program to crash while dealing with exceptions. This procedure detects exceptions and instructs them on how to manage them without crashing. To do this, we can utilize the try and catch blocks. The "try" and "catch" blocks work in the same way. We can put any code in the try block that we believe will break the application. And if that code crashes the program, it enters the catch block and executes all of the code contained inside. As in below figure we can throw exception in the console without crashing the program.

```
0 references
static void Main(string[] args)
{
    try
    {
        Console.WriteLine("Enter a number: ");
        int num1 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter a number: ");
        int num2 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine(num1 / num2);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }

    Console.ReadLine();
}
```

```
C:\Users\ryand\source\repos\Testing\Testing\bin\Debug\Testing.exe
Enter a number:
5
Enter a number:
0
Attempted to divide by zero.
```

Figure 4. 121 Handle exceptions by try and catch blocks

4.5 Coding Standards

4.5.1 Importance of coding standards

Software developers must follow to coding standards for a variety of reasons. The maintenance of software accounts for 40% to 80% of the entire cost. The original writer of software usually never completely supports it. Coding standards increase software readability by making it easier for developers to understand new code.

4.5.2 Coding standards to follow and how I followed them

1) The code should be simple to understand.

As a result, the developer should try to separate code blocks into paragraphs to define different areas of the code. Indentation should be used to identify the start and end of control structures, as well as a clear indication of where the code is between them.

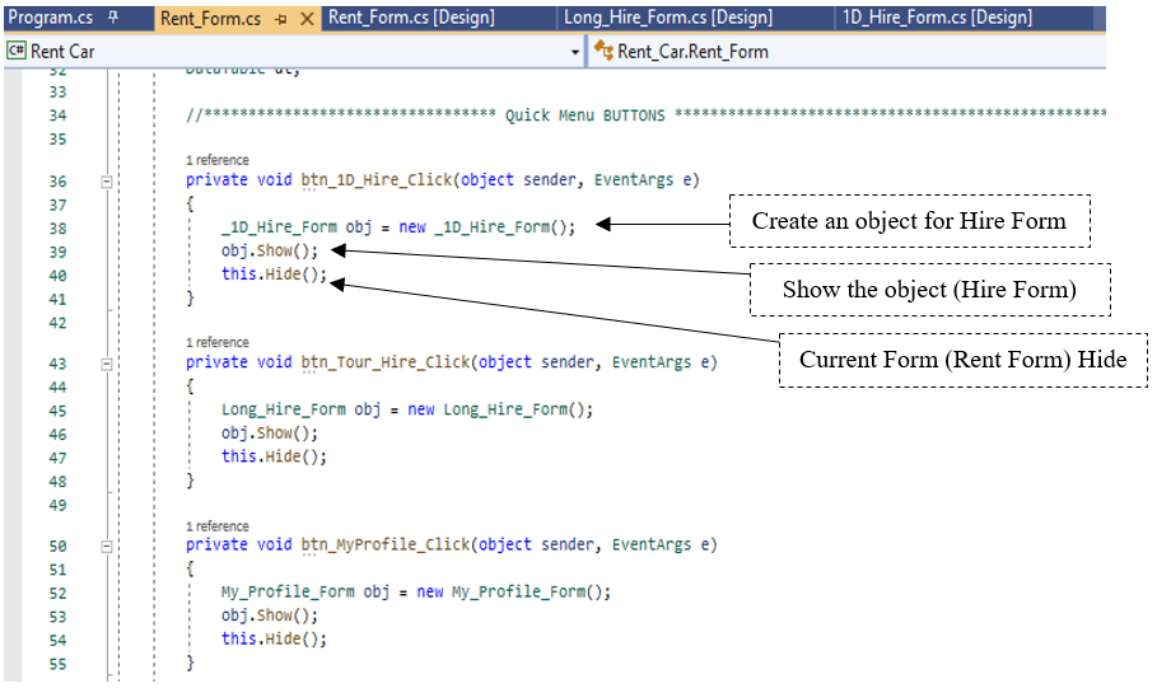


Figure 4. 122 Using simple code to understand – Part of the Ayubo System

2) The naming convention for variables in the code should be consistent throughout. Furthermore, the data in the code should be specified.

There are two popular naming conventions are being used.

I. Camel Case – The first letter of each word is capital except the first word

II. Underscore – Using underscore between words.

In the following figure red circled parts indicate where I've used Underscore naming convention for variables. And green arrow parts indicate where I've used Camel Case naming convention for Methods.

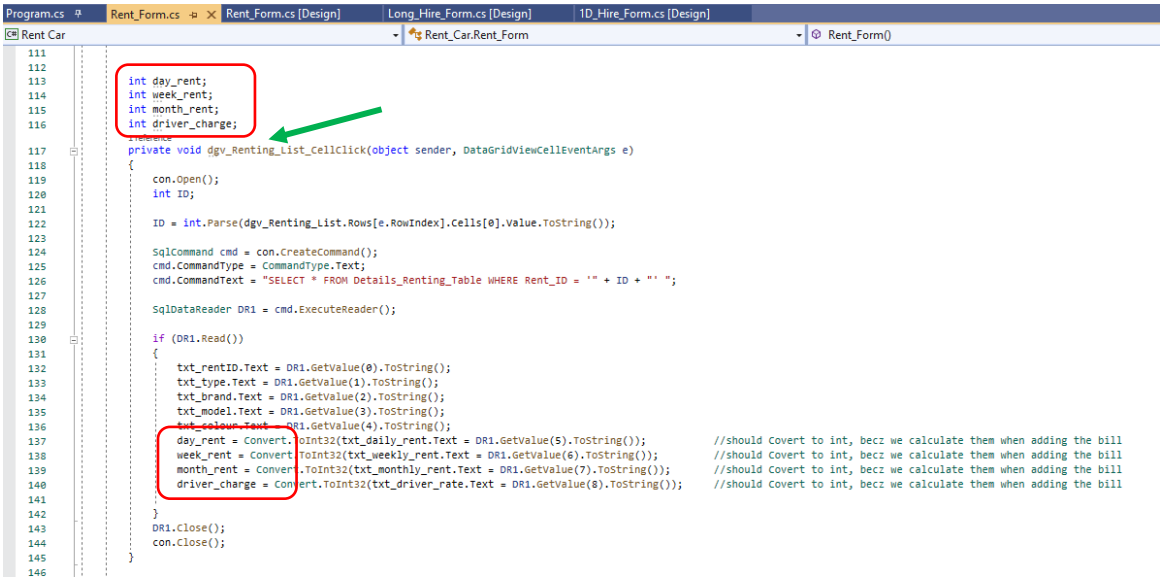


Figure 4. 123 Using consistency of naming convention of the variables – Part of the Ayubo System

3) Should give the functions/methods names based on what they do.


```
71
72      //******************************** OTHER METHODS ***************************************************************
73
            0 references
74      public void clear()
75      {
76          txt_type.Text = "";
77          txt_brand.Text = "";
78          txt_model.Text = "";
79          txt_colour.Text = "";
80          txt_daily_rent.Text = "";
81          txt_weekly_rent.Text = "";
82          txt_monthly_rent.Text = "";
83          cmb_refresh_category.Text = "";
84
85          dtp_start.Value = DateTime.Now;
86          dtp_end.Value = DateTime.Now;
87          txt_calculate.Text = "";
88          rbtn_yes.Checked = false;
89          rbtn_no.Checked = false;
90
91
92      }
93
            1 reference
94      public void display_data_grid_view()    //For the data grid view
95      {
96          try
97          {
98              dt = new DataTable();
99              con.Open();
100             adpt = new SqlDataAdapter("SELECT * FROM Details_Renting_Table", con);
101             adpt.Fill(dt);
102             dgv_Renting_List.DataSource = dt;
103             con.Close();
104         }
105         catch (Exception ex)
106         {
107             MessageBox.Show(ex.Message);
108             con.Close();
109         }
110     }
```

Figure 4. 124 Naming the function according to what they perform – Part of the Ayubo System

4) Use a specific way for making comments on the project. As a result, the developer may be capable of understanding it even after a period of time has passed.

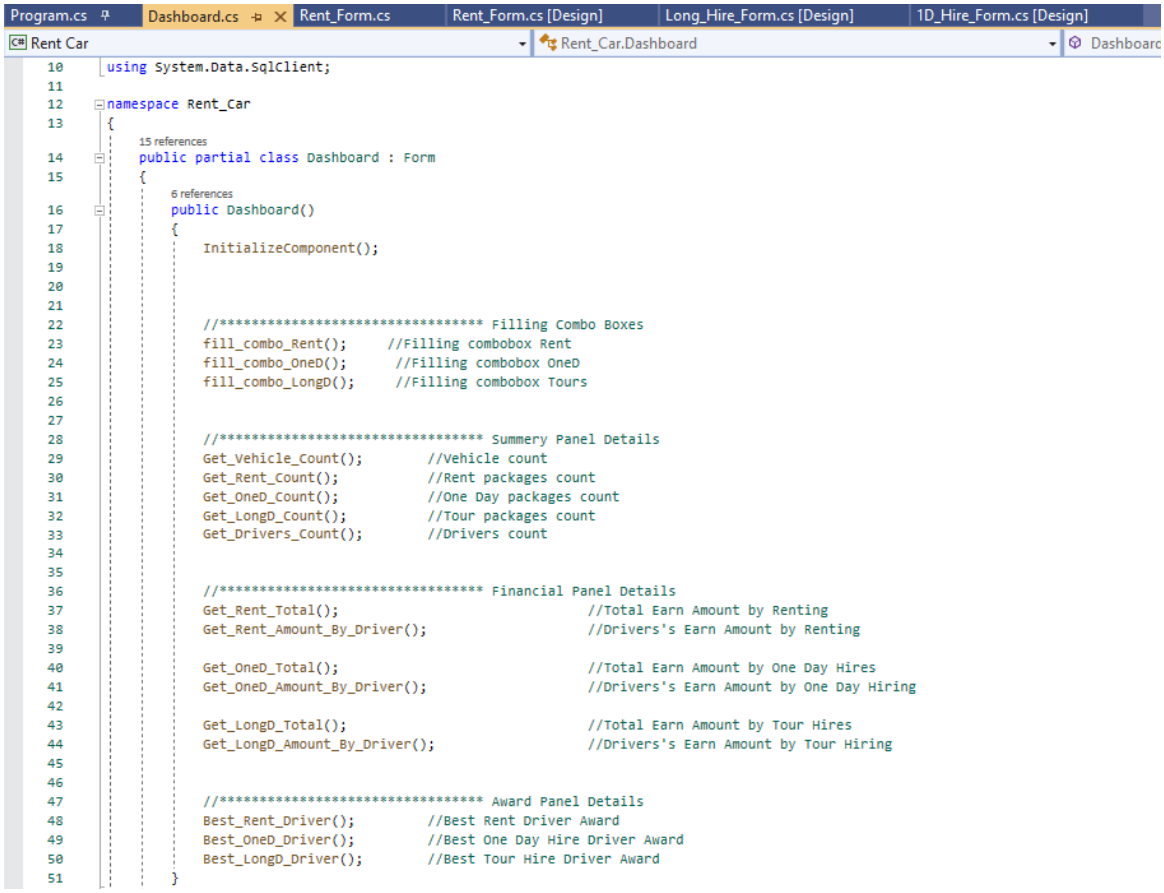


Figure 4. 125 Commenting on the lines of important parts of the project

4.5.3 Benefits of Implementing Coding Standards in Software Development

1) The probability of a project's failure is minimized.

IT projects frequently fail as a result of issues encountered while developing software. Many problems and the probability of project failure are reduced when code quality standards are followed.

2) Improve efficiency

It is frequently observed that software developers spend a lot of time on correcting code quality issues that could have been avoided. Coding standards and best practices would assist the team in detecting problems early on, if not totally preventing them. This will improve productivity throughout the software development process.

3) Cost efficiency

Developers can reuse code that is written in a clear manner whenever they need it. This can significantly minimize the cost of development as well as the time spent on it.

4) Easy to maintain

If coding standards are followed, the code is consistent and easy to maintain. This is due to the fact that anyone may comprehend it and change it at any time.

5) Making the code minimal complexity

Whenever a code is complex, it is more likely to be vulnerable to bugs. Coding standards help in the construction of less complex software programs, which reduces errors.

6) Rectification of bugs

If the source code is written consistently, it becomes much easier to find and fix flaws in the product.

Conclusion

This entire assignment is based on an implementation of a Car rent and hire system design for selected company (Ayubo Drive). The purpose of this assignment is to improve programing skills.

This report includes basic programming principles, creating algorithms, understanding time complexities, creating basic programs by using Python language and C# language, understanding different programming paradigms, working with IDEs, Basic system creation using Microsoft SQL and Visual Studio C# windows forms application, debugging process and coding standards.

References

Investopedia. 2022. Algorithm Definition.

[ONLINE] Available at: https://www.investopedia.com/terms/a/algorithm.asp.

[Accessed 03 February 2022].

Techopedia. 2022. What is an Algorithm? - Definition from Techopedia.

[ONLINE] Available at: https://www.techopedia.com/definition/3739/algorithm.

[Accessed 03 February 2022].

Code with C. 2022. Fibonacci Series Algorithm and Flowchart | Code with C.

[ONLINE] Available at: https://www.codewithc.com/fibonacci-series-algorithm-flowchart/.

[Accessed 03 February 2022].

GeeksforGeeks. 2022. Difference Between Algorithm and Flowchart - GeeksforGeeks.

[ONLINE] Available at: https://www.geeksforgeeks.org/difference-between-algorithm-and-flowchart/.

[Accessed 03 February 2022].

GeeksforGeeks. 2022. Introduction to Algorithms - GeeksforGeeks.

[ONLINE] Available at: https://www.geeksforgeeks.org/introduction-to-algorithms/.

[Accessed 03 February 2022].

Investopedia. 2022. Algorithm Definition.

[ONLINE] Available at: https://www.investopedia.com/terms/a/algorithm.asp.

[Accessed 03 February 2022].

What is an Algorithm?. 2022. What is an Algorithm?.

[ONLINE] Available at: https://www.programiz.com/dsa/algorithm.

[Accessed 03 February 2022].

Khan Academy. 2022. What is an algorithm and why should you care? (video) | Khan Academy.

[ONLINE] Available at: https://www.khanacademy.org/computing/computer-science/algorithms/intro-to-algorithms/v/what-are-algorithms.

[Accessed 03 February 2022].

Shlok Bhatt. 2022. Characteristics of an Algorithm. There are some characteristics which… | by Shlok Bhatt | Medium.

[ONLINE] Available at: https://medium.com/@bhattshlok12/characteristics-of-an-algorithm-49cf4d7bcd9.

[Accessed 03 February 2022].

atechdaily.com. 2022. No page title.

[ONLINE] Available at: https://atechdaily.com/posts/algorithm-for-factorial-number.

[Accessed 03 February 2022].

livescience.com. 2022. What is the Fibonacci sequence? | Live Science.

[ONLINE] Available at: https://www.livescience.com/37470-fibonacci-sequence.html.

[Accessed 03 February 2022].

Fibonacci Sequence. 2022. Fibonacci Sequence.

[ONLINE] Available at: https://www.mathsisfun.com/numbers/fibonacci-sequence.html.

[Accessed 03 February 2022].

Factorial Function !. 2022. Factorial Function !.

[ONLINE] Available at: https://www.mathsisfun.com/numbers/factorial.html.

[Accessed 03 February 2022].

Big-O Notation, Omega Notation and Big-O Notation (Asymptotic Analysis). 2022. Big-O Notation, Omega Notation and Big-O Notation (Asymptotic Analysis).

[ONLINE] Available at: https://www.programiz.com/dsa/asymptotic-notations.

[Accessed 03 February 2022].

Big-O Notation: A simple explanation with examples . 2022. Big-O Notation: A simple explanation with examples .

[ONLINE] Available at: https://www.linkedin.com/pulse/big-o-notation-simple-explanation-examples-pamela-lovett.

[Accessed 03 February 2022].

Developer Insider. 2022. Big-O Notation Explained with Examples.

[ONLINE] Available at: https://developerinsider.co/big-o-notation-explained-with-examples/.

[Accessed 03 February 2022].

Career Karma. 2022. Big O Notation - Time Complexity.

[ONLINE] Available at: https://careerkarma.com/blog/big-o-notation-time/.

[Accessed 03 February 2022].

Maxwell Harvey Croy. 2022. How To Calculate Time Complexity With Big O Notation | by Maxwell Harvey Croy | DataSeries | Medium.

[ONLINE] Available at: https://medium.com/dataseries/how-to-calculate-time-complexity-with-big-o-notation-9afe33aa4c46.

[Accessed 03 February 2022].

freeCodeCamp.org. 2022. What exactly is a programming paradigm?.

[ONLINE] Available at: https://www.freecodecamp.org/news/what-exactly-is-a-programming-paradigm/.

[Accessed 03 February 2022].

Learn Computer Science. 2022. What Is Programming Paradigm ? | Paradigm Types, Features Explained..

[ONLINE] Available at: https://www.learncomputerscienceonline.com/programming-paradigm/.

[Accessed 03 February 2022].

IONOS Digitalguide. 2022. Imperative programming: Advantages & disadvantages of the paradigm - IONOS.

[ONLINE] Available at: https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/.

[Accessed 03 February 2022].

GeeksforGeeks. 2022. Introduction of Programming Paradigms - GeeksforGeeks.

[ONLINE] Available at: https://www.geeksforgeeks.org/introduction-of-programming-paradigms/.

[Accessed 03 February 2022].

Hackr.io. 2022. What is Procedural Programming? [Definition] - Key Features.

[ONLINE] Available at: https://hackr.io/blog/procedural-programming.

[Accessed 03 February 2022].

Object oriented vs procedural vs event driven programming - ppt download. 2022. Object oriented vs procedural vs event driven programming - ppt download.

[ONLINE] Available at: https://slideplayer.com/slide/16031203/.

[Accessed 03 February 2022].

cwells. 2022. Event-driven Programming .

[ONLINE] Available at: https://www.technologyuk.net/computing/software-development/software-design/event-driven-programming.shtml.

[Accessed 03 February 2022].

Techopedia. 2022. What is an Event-Driven Program? - Definition from Techopedia.

[ONLINE] Available at: https://www.techopedia.com/definition/7083/event-driven-program.

[Accessed 03 February 2022].

Bench Partner. 2022. Advantages and Disadvantages of Event-Driven Programming - Bench Partner.

[ONLINE] Available at: https://benchpartner.com/advantages-and-disadvantages-of-event-driven-programming.

[Accessed 03 February 2022].

Characteristics of Object Oriented programming language - oops. 2022. Characteristics of Object Oriented programming language - oops.

[ONLINE] Available at: https://www.careerride.com/oops-characteristics.aspx.

[Accessed 03 February 2022].

Event-Driven Programming Features. 2022. Event-Driven Programming Features.

[ONLINE] Available at: https://www.ukessays.com/essays/computer-science/eventdriven-programming-features-6167.php.

[Accessed 03 February 2022].

Edureka. 2022. What is Debugging? Different Stages of Debugging | Edureka.

[ONLINE] Available at: https://www.edureka.co/blog/what-is-debugging/#:~:text=Debugging%20has%20many%20benefits%20such,structures%20and%20allows%20easy%20interpretation..

[Accessed 03 February 2022].

Mikejo5000. 2022. Debugging code for absolute beginners - Visual Studio (Windows) | Microsoft Docs.

[ONLINE] Available at: https://docs.microsoft.com/en-us/visualstudio/debugger/debugging-absolute-beginners?view=vs-2022&tabs=csharp.

[Accessed 03 February 2022].

Multidots. 2022. Importance of Coding Standard and Code Quality in Software Development.

[ONLINE] Available at: https://www.multidots.com/importance-of-code-quality-and-coding-standard-in-software-development/#:~:text=Coding%20standards%20help%20in%20the,and%20thereby%20reduce%20the%20errors.&text=If%20programming%20standards%20in%20software,at%20any%20point%20in%20time..

[Accessed 03 February 2022].

Difference Between IDE and Code Editor | Difference Between. 2022. Difference Between IDE and Code Editor | Difference Between.

[ONLINE] Available at: http://www.differencebetween.net/technology/difference-between-ide-and-code-editor/.

[Accessed 03 February 2022].

Difference Between IDE and Text Editor | Difference Between. 2022. Difference Between IDE and Text Editor | Difference Between.

[ONLINE] Available at: http://www.differencebetween.net/technology/difference-between-ide-and-text-editor/.

[Accessed 03 February 2022].

Computer Programming. 2022. Computer Programming.

[ONLINE] Available at: https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading13.htm.

[Accessed 03 February 2022].