American International University-Bangladesh (AIUB)
Department of Computer Science
Faculty of Science &Technology (FST)
Summer 22-23

Section: E

Software Quality Assurance and Testing

Vehicle Management System

A Report submitted
By

Serial No	Student Name	Student ID
1	MD. SHAFWAN AHMED DEHAN	19-39302-1
2	ASHESH DEB PRIOM	20-43698-2
3	MD. SHARIAR MAHMUD SACHCHA	20-43772-2
4	NOBONITA NONDE	20-43819-2

Under the supervision of

ABHIJIT BHOWMIK
Associate Professor & Special Assistant [OSA], Computer Science
Faculty of Science and Technology (FST),
American International University-Bangladesh

Software Test Plan

for

Vehicle Management System

Version 1.0 approved.

Prepared by

NOBONITA NONDE

MD. SHARIAR MAHMUD SACHCHA

ASHESH DEB PRIOM

MD. SHAFWAN AHMED DEHAN

American International University-Bangladesh (AIUB)

August 10, 2023

Checked By Industry Personnel

Name:

Designation:

Company:

Sign:

Date:

Table of Contents

Revision History

Revision	Date	Updated by	Update Comments
0.1	2023.08.08	NOBONITA NONDE	First Draft
0.2	2023.08.08	MD. SHARIAR MAHMUD SACHCHA	Second Draft
0.3	2023.08.08	ASHESH DEB PRIOM	Third Draft
0.4	2023.08.09	MD. SHAFWAN AHMED DEHAN	Fourth Draft
0.5	2023.08.09	MD. SHARIAR MAHMUD SACHCHA	Fifth Draft
0.6	2023.08.09	NOBONITA NONDE	Sixth Draft
0.7	2023.08.10	ASHESH DEB PRIOM	Seventh Draft
0.8	2023.08.10	MD. SHAFWAN AHMED DEHAN	Eighth Draft
0.9	2023.08.10	Checked by everyone	Ninth Draft

1. TEST PLAN IDENTIFIER: VMS1.O

2. REFERENCES

- https://iopscience.iop.org/article/10.1088/1757-899X/398/1/012014
- https://www.simplilearn.com/tutorials/selenium-tutorial/selenium-automation-testing
- https://learn.microsoft.com/en-us/dotnet/core/testing/

3. INTRODUCTION

Background to the Problem

In today's fast-paced world, where mobility and convenience have become paramount, the efficient management of vehicles has emerged as a significant challenge. The surge in vehicle ownership, coupled with diverse consumer preferences, has led to a complex landscape where individuals struggle to seamlessly access and understand vehicle information. This lack of clarity often results in wasted time, misinformed decisions, and even safety concerns.

The root cause of this problem lies in the absence of a comprehensive and user-friendly system that empowers consumers with instant access to detailed vehicle specifications, ownership information, and the ability to quickly secure qualified drivers. Additionally, during critical situations such as accidents, the current lack of a streamlined communication mechanism with emergency services can lead to delayed response times and potentially worsened outcomes. It has become evident that a well-designed solution is crucial to address these challenges and provide consumers with a more transparent and secure vehicle ownership experience.

Solution to the Problem

Our proposed solution tackles the multifaceted challenges associated with vehicle management by offering a robust and intuitive platform that caters to the diverse needs of vehicle owners. The VMS offers a range of features designed to enhance the vehicle ownership experience, streamline operations, and ensure safety. Comprehensive Vehicle Information, the VMS empowers consumers with access to detailed specifications of their registered vehicles. This includes essential details such as make, model, year, engine specifications, and maintenance history. This solution enables owners to make informed decisions about their vehicles' maintenance and upgrades. Effortless Vehicle Registration, the VMS provides a user-friendly interface for consumers to effortlessly register their vehicles. By simplifying the registration process, we aim to save time and eliminate administrative hassles for vehicle owners, ensuring a seamless onboarding experience. Personalized Driver Services, recognizing the evolving needs of consumers, our VMS introduces an innovative feature that enables users to hire professional drivers for their vehicles. This service ensures that vehicle owners can delegate the task of driving, creating a convenient and safe transportation option. Emergency SOS Assistance, in critical situations such as accidents or emergencies, the VMS incorporates an SOS button that triggers an immediate alert to nearby active rescue teams. This real-time communication facilitates swift assistance, potentially saving lives and minimizing the impact of unforeseen events.

Software Description and Purpose, the Vehicle Management System (VMS) is a innovative software application designed to revolutionize the way individuals manage and interact with their vehicles. With a user-friendly interface accessible through web and mobile platforms, the VMS aims to offer the benefits of, Enhanced vehicle ownership experience, the VMS empowers consumers with a centralized platform to access their vehicle information, maintenance schedules, and driver services, creating a more streamlined and organized ownership experience. Time and Resource Efficiency By digitizing and automating vehicle registration, maintenance tracking, and driver hiring processes, the VMS saves users valuable time and minimizes administrative efforts. Safety and Security, the VMS prioritizes user safety by incorporating the SOS feature, which enables prompt and efficient responses from nearby rescue teams during emergencies, ensuring timely assistance when it matters most.

Existing Studies and Solutions, while several software solutions address various aspects of vehicle management, none have seamlessly integrated the breadth of features offered by our VMS. Existing applications often focus on singular aspects such as maintenance tracking or emergency services, without providing a comprehensive platform that caters to the holistic needs of vehicle owners. In conclusion, the Vehicle Management System presents a ground-breaking solution to the challenges faced by modern vehicle owners. By combining comprehensive vehicle information, effortless registration, personalized driver services, and emergency SOS assistance, the VMS stands as a transformative tool that enhances the ownership experience, streamlines operations, and ensures safety in an increasingly dynamic automotive landscape.

4. REQUEIREMNT SPECIFICATION

4.1 System Features

1. System Login

Functional Requirements

1.1 Vehicle Management System allows users to login to the system with their registered mobile number/Email and password.

1.2 If the user put wrong username or password the system returns wrong username or password. Users can login to their home only given proper information.

1.3 By clicking the 'remember me' selection user can save their information for one day.

1.4 After putting in the proper information (Username, Password) correctly user's need to click login button and then they can enter their home page.

Priority Level: High

Precondition: user have valid user id and password.

2. Registration/Signup.

Functional Requirements

2.1 Users can register for this system at any time. There is no time limitation for registration.

2.2 For create a new account user must provide their first name, last name, gender, Email/Phone number, their desired password, NID, which type of user and their present address.

2.3 After putting all the information properly, the user needs to click the Register button.

2.4 If any information is not available in the system, then the system will show the missing point to fill it up.

2.5 If the registration is completed properly then the system will redirect the user to the login page.

Priority Level: High

Precondition: User must use Bangladeshi phone number; no information can be missing, and Password must be 8 digits.

3. Forget password.

Functional Requirements

3.1 If the user forgets their password, they can reset their password by clicking forget password link.

3.2 After clicking forget password a new interface will appear where user put their phone number/email then the system searches their profile from the system database. If it is founded system show their account information, it will show there is no account matching.

3.3 If the account is matched with the given information, then user can see a button "Send OTP".

3.4 Clicking "Send OTP" an OTP message will be sent to the registered phone number.

3.5 Giving the OTP in the OTP section properly user can set new password. Putting wrong OTP system will show "Wrong OTP".

Priority Level: Medium
Precondition: OTP must be the same as the message which is sent to the user phone number.

4. Logout.
Functional Requirements

4.1 The "Logout" functionality should be initiated by the user through an easily accessible interface element (e.g., a "Logout" button) prominently displayed within the user interface.

4.2 Upon initiating the logout process, the software application must terminate the user's current session and revoke their access to any protected resources.

4.3 After initiating the logout process, the user should be presented with a confirmation dialog box to confirm their intention to log out. This confirmation step helps prevent accidental logouts.

4.4 If the user is inactive within 5 minutes the system will auto log out the user from the system.

Priority Level: Low
Precondition: Dialog box must be selected by the user.

5. Profile.
Functional Requirements

5.1 Every user can add and update their profile information by clicking the profile button.

5.2 First time users must fill-up their information properly by registration after that they can browse the system properly.

5.3 Users can also a modify button after first time adding their information. By clicking the Modify button users can update their given information.

5.4 Users can add their profile picture by this section, but it is not mandatory.

Priority Level: Medium
Precondition: None.

6. Vehicle Information (Driver).
Functional Requirements

6.1 By selecting vehicle information, the user can add vehicle information, modify information and picture. First time adding vehicle is mandatory for the driver.

6.2 After clicking vehicle information 2 new buttons will appear. One is Add vehicle and another in modify vehicle.

6.3 Clicking add vehicle user can add their vehicle by giving License no., Registration date of the vehicle, send scan copy of their driving license and vehicle license.

6.4 Clicking modify vehicle users update their vehicle information.

Priority Level: High
Precondition: None.

7. Hire Driver.

Functional Requirements

7.1 General users can hire a driver by clicking the hire driver button.

7.2 After clicking “Hire Driver” button user can see a list of drivers.

7.3 After selecting any driver, the information of the will appears to the user. They can Hire them by pressing Confirm button or back to the driver list by clicking Cancel button.

Priority Level: High

Precondition: User must have a driving license.

8. Search Vehicle.

Functional Requirements

8.1 Users can search for drivers by giving their vehicle number.

8.2 After putting the vehicle number in the search section, the system will appear the driver information for safety.

8.3 If the number does not match with any vehicle number, the system will show that the number does not match with any vehicle.

Priority Level: Medium

Precondition: None.

9. Emergency Response Button.

Functional Requirements

9.1 There is an SOS button in this system. By clicking SOS button, a message will send the verified number to help the user.

9.2 After clicking if a pop-up message will appear in the interface where have a dialog box, selecting the dialog box then user need to confirm it by “Confirm” button then, a message will send the verified number to help the user.

Priority Level: Medium

Precondition: User must login to their account.

10. Complaint.

Functional Requirements

10.1 In this system there is a complain button if any unnecessary event occurred user can complain.

10.2 Selecting complain button a message box will appear to the user where user can write query about the accidental incident then click send button.

10.3 Only admin can see the other user’s message from the complaint section. Without admin no one can see other’s message.

Priority Level: Medium

Precondition: User must login to their account.

4.2 System Quality Attributes

Usability: The usability of the Vehicle Management System is an important aspect of its design. The user interface has been carefully crafted to be simple and user-friendly, ensuring that anyone can easily navigate and perform tasks. Users can easily view vehicle details, maintenance history, and generate reports. The searching option is accurate, ensuring that users can quickly find the information they need. The system loading time is fast, ensuring that users do not experience any delays while using the application. Additionally, the administrator has full access to the system, allowing them to manage and control the system with ease. Overall, the usability of the Vehicle Management System is optimized to provide users with a hassle-free experience.

Maintainability: The vehicle management system is built with maintainability in mind, ensuring that the maintenance team can easily perform their tasks. The system is designed to be stable, allowing for smooth operations without interruptions. Additionally, the system is easily adaptable, making it simple to make changes and updates as required. The team can quickly analyze the system to identify and fix any issues that may arise, ensuring that it always runs efficiently. Overall, the system's maintainability ensures that it can provide reliable service to users and be easily managed by the maintenance team.

Efficiency: The vehicle management system is designed with efficiency in mind. It is lightweight and optimized to consume less memory, allowing for a smooth and hassle-free experience for users. The system loading time is short, providing users with quick access to the information they need. This ensures that users can quickly perform necessary tasks without any delays.

Reliability: The vehicle management system is designed with high reliability in mind, ensuring that it can maintain its service provision under defined conditions for defined periods of time. The system is fault-tolerant, meaning it can withstand component failure and recover from any interruptions quickly. It has a robust database system that stores data securely and ensures that data is always accessible, even if there is a network or server failure. However, any potential downtime caused by server or network issues may lead to a temporary interruption in service.

Functionality: The vehicle management system's functionality is comprehensive and user-friendly. It includes all necessary features for managing vehicles, including adding new vehicles, tracking their location, and generating reports. The system also ensures data security by providing 100% protection for personal information and transactions. The high-security measures in place give users peace of mind and confidence in using the system for their vehicle management needs.

Portability: The vehicle management system is built with portability in mind, allowing it to be used in a variety of settings. It is compatible with different versions of Windows, making it accessible to a wide range of users. The system's adaptability allows it to be easily modified to suit different platforms or environments. Object-oriented design and implementation practices have been used in the system's development, which contributes to its portability. These practices enable the system to be easily transferred from one platform to another without requiring significant changes to its underlying code, thereby saving time and effort in development and maintenance.

Security and Privacy: The vehicle management system deals with sensitive user data, such as personal information and vehicle details. Implementing robust security measures and privacy controls is crucial to safeguard user information and prevent unauthorized access.

Maintenance and Upgrades: A well-structured application design can facilitate future maintenance and upgrades. This attribute contributes to the application's longevity and adaptability to changing user needs. Our system follows these criteria. So, it is easy to maintain and upgrade.

4.3 System Interface

1. Login interface of vehicle management system:

Figure 1: Login Interface.

2. General User interface:

Figure 2: General user interface

3. Search user for forget password:

Figure 3: Search user page for reset password.

4. Vehicle information list page for admin:

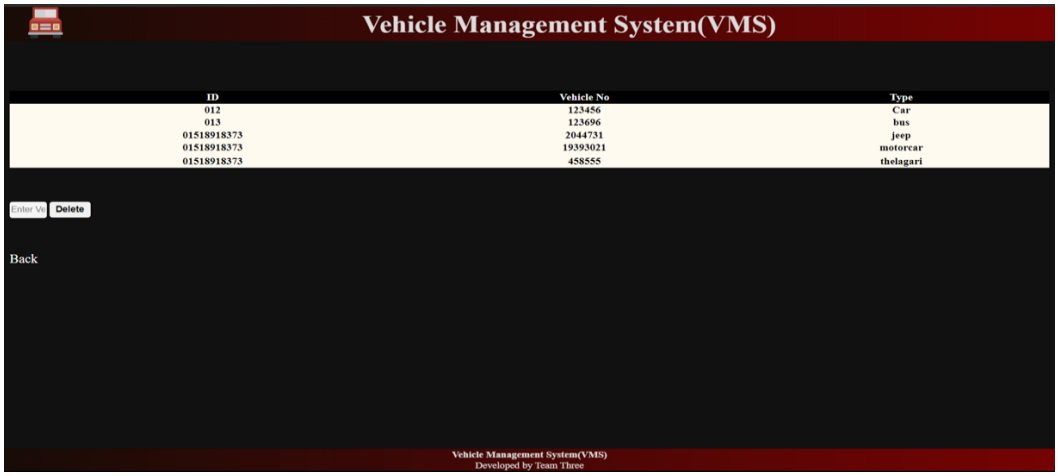


Figure 4: Vehicle list

5. Modify user profile page:

Figure 5: Modify profile.

6. User information page:

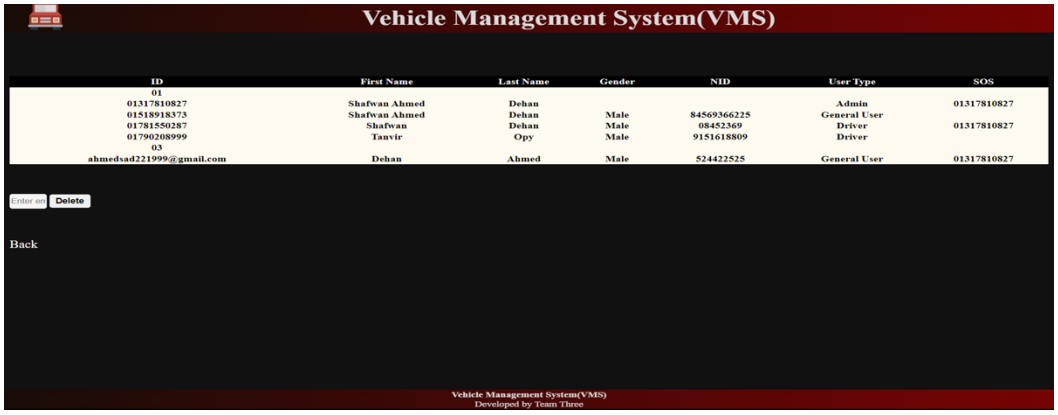


Figure 6: User information

7. Admin Home page:

Figure 7: Admin home interface.

8. Add vehicle page:

Figure 8: Add Vehicle.

9. User Details:

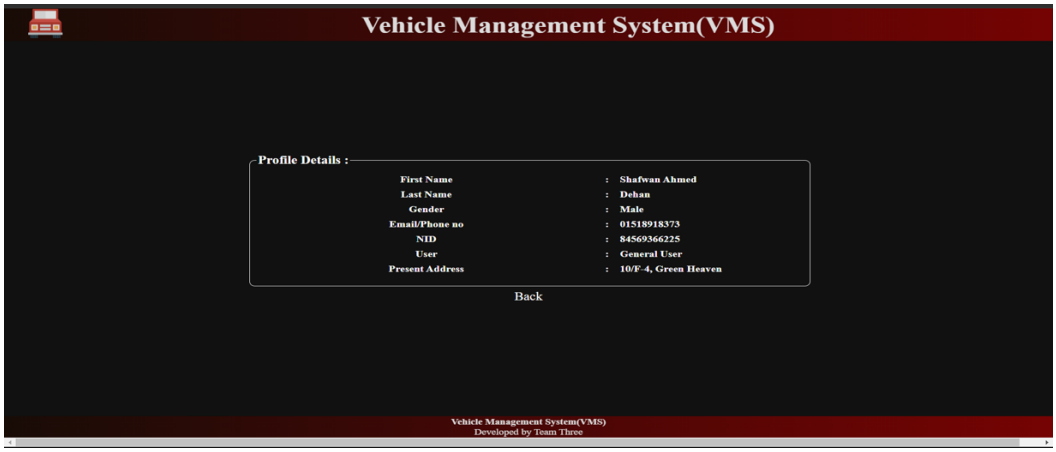


Figure 9: Profile Details.

10. Driver home page:

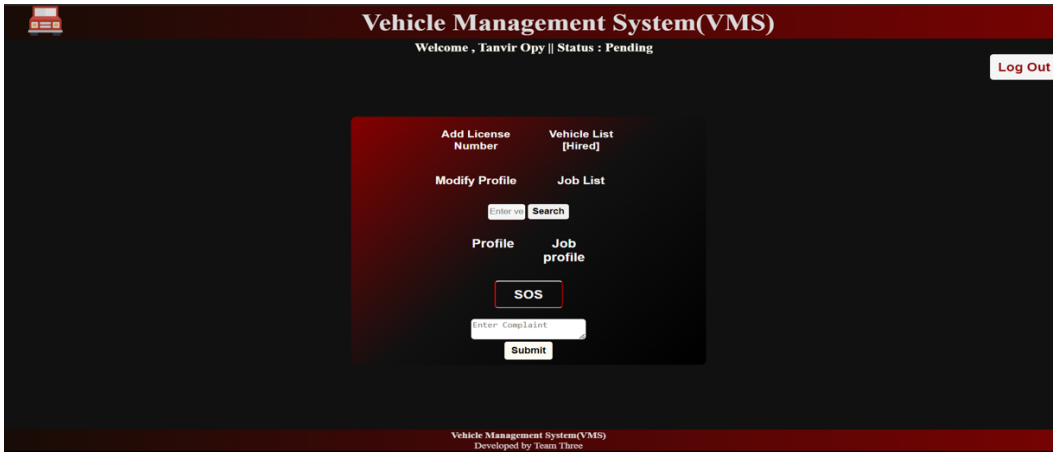


Figure 10: Driver Dashboard.

11. License information update page:

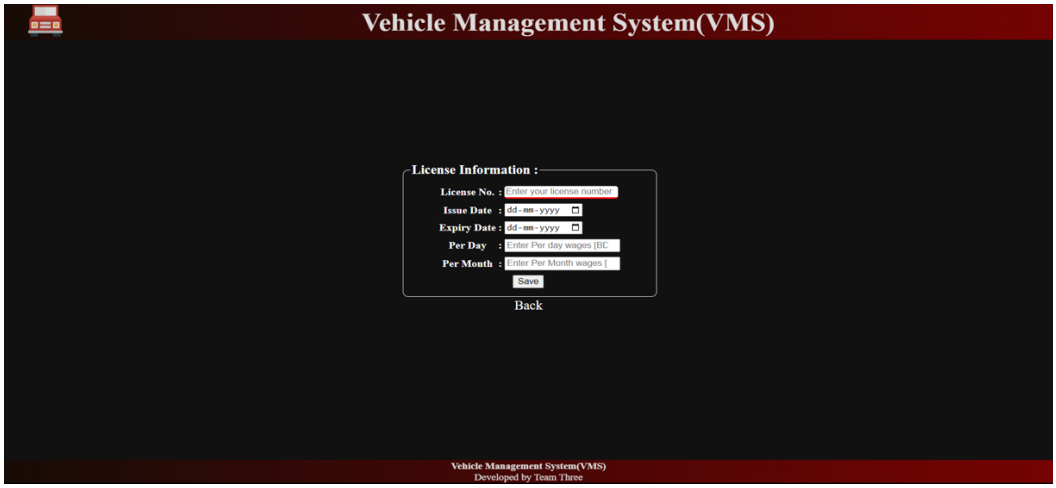


Figure 11: License information.

4.4 Project Requirements

Budget Estimation:

4 developers and engineers working for 3 months on a vehicle management system project:

Duration in weeks = 7 weeks

Office days = 5 days

Working hours = 7 hours

So, per week the working hours is = (5 x 7) hours. = 35 hours

Hence, Total Working hours is = (35 x 12) hours. = 420 hours per developer

Developer hourly salary is = 800 BDT

Total developers Salary = (800*420) BDT

= 3,36,000 BDT

Project Estimation

COCOMO is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality.

So, we will used Constructive Cost Model (COCOMO) to calculate the estimation of our project were,

D = Total time required for project development in Months (M).
KLOC = The size of the code for the project in Kilo lines of code.
a, b, c, d = The constant parameters for a software project.

Consider, project is organic.

For, Organic Software
Project,
a= 2.4, b=1.0, c=2, d=0.38

Consider, KLOC=5000

Effort Estimation Formulas:

Effort, E = (a(KLOC)^b) *EAF
= (2.4 * (5000/1000) ^1.05) * 0.81
= 11.00

Total Development Time, D = c(Effort)^d
= 2.5*(11) ^0.38
= 6.21
= 7 months

Required number of people = Effort/Time
= 11/10
= 1.1
= 2 developers

Cost:
Labor cost = 2 * 35000 = 70,000/- Taka (Approximately)
Service cost = 7000*2 = 14000/- Taka (Approximately) so,
Total cost = 0,000 + 14000 = 75,000/- Taka (Approximately)

5. UML DIAGRAM

5.1 Use Case Diagram

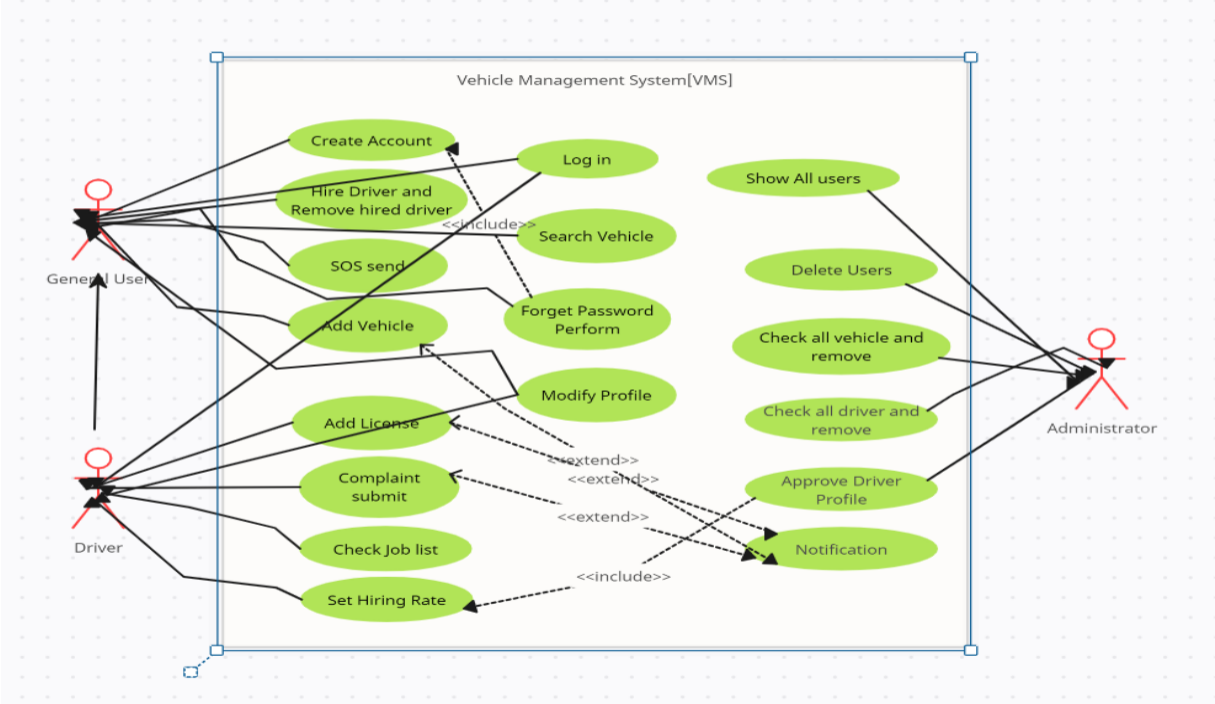


Figure 12: Use Case diagram for Vehicle management System

5.2 ER Diagram:

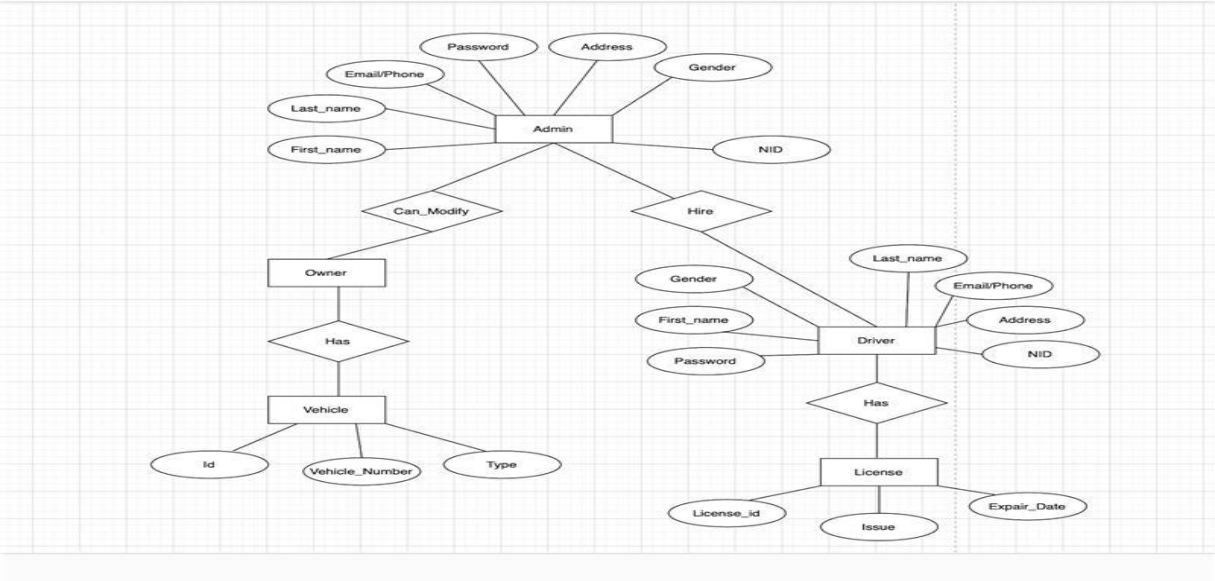


Figure 13: ER diagram for Vehicle management System.

6. FEATURES NOT TO BE TESTED

Automation is most effective for stable, repeatable, and critical aspects of the application, as well as for performance, load, and API testing. However, it's important to assess each feature's specific characteristics to determine whether automation is the right approach.

- If the user interface changes frequently, it can be challenging to maintain automated tests that rely heavily on UI interactions. In this case this type of feature is not to be tested.
- Features that are still in development or undergo frequent changes may not be suitable for automation until they stabilize. Under development features cannot be tested.
- All user information and other all information showing cannot be tested properly that the value is coming from the database, they are correct or not.
- To know the user experience, it is not possible to test usability, understandability etc, factors cannot be tested.

7. TESTING APPROACH

7.1 Testing Levels

- The testing strategy for the upcoming Vehicle Management System (VMS) project is comprised of three testing levels, including Unit, System/Integration (combined), and Acceptance tests. It is anticipated that there will be at least one dedicated full-time independent test person for system/integration testing. However, due to budgetary and time constraints, most of the testing will be conducted by the test manager with active participation from the development teams.

- UNIT Testing will be done by the developer and will be approved by the development team leader. Before Unit Testing is considered acceptable and passed on to the test person, the programmer must provide evidence of unit testing in the form of a test case list, sample output, data printouts, and defect information to the team leader. The test person must also be provided with all unit test information.

- SYSTEM/INTEGRATION Testing will be carried out by the test manager and the development team leader, with assistance from individual developers when needed. No specific test tools have been identified for this project. Programs will enter the System/Integration test phase only after all major defects have been rectified. Programs can have up to two major defects, if they do not hinder testing of the program, and a workaround is available for the error.

- ACCEPTANCE Testing will be performed by the end-users with assistance from the test manager and development team leader. The acceptance test will be conducted in parallel with the current manual process for one month after the completion of the System/Integration test process.

7.2 Test Tools

The only test tools we have used to perform the automated testing is named ‘Selenium’. As well as we have used .NET framework to code. We have Visual Studio 2019 as our IDE to write code and we have used C# as our language.

- Selenium is a popular open-source automation testing framework primarily used for web applications. It provides a suite of tools to automate web browsers, allowing developers and testers to simulate user interactions, perform functional tests, and validate web elements. Selenium's versatility supports multiple programming languages and integrates with various testing frameworks, making it a powerful choice for web testing, including regression and cross-browser testing. Its capabilities empower efficient and reliable testing of web-based software, aiding in identifying issues early in the development process.

- The .NET framework offers various tools and libraries for test description, enabling structured and organized test development. It provides testing frameworks like NUnit, xUnit, and MSTest, which allow developers to define and execute unit tests, integration tests, and more. These frameworks facilitate test description by providing attributes, assertions, and setup/teardown methods, aiding in test case design, execution, and reporting. The .NET framework's test description capabilities contribute to the creation of maintainable, comprehensible, and effective test suites, helping ensure software quality through automated testing practice.

- Visual Studio, a comprehensive integrated development environment (IDE), is a prime choice for testing using the .NET framework. It seamlessly integrates with popular testing frameworks like NUnit, MSTest, and xUnit, streamlining test creation, execution, and analysis. Its robust debugging features, test explorer, and code coverage tools enhance testing efficiency. Visual Studio provides a unified platform for coding, testing, and debugging .NET applications, fostering collaboration among development and testing teams. Its intuitive interface, coupled with built-in testing capabilities, makes it a powerful tool for ensuring the reliability and quality of .NET-based software through efficient testing practice.

- C# is a versatile programming language often used for testing due to its strong typing, object-oriented design, and integration with the .NET framework. It supports test-driven development (TDD) and offers libraries like NUnit, MSTest, and xUnit for creating robust test suites. With C#, developers can write automated tests, perform unit testing, and ensure the functionality, reliability, and performance of software. Its expressive syntax, extensive tooling, and compatibility with popular testing frameworks make C# a compelling choice for testing in various application domains.

7.3 Meetings

The software testing team meeting is a gathering where the team members responsible for ensuring the quality of the software come together. In this meeting, they discuss the progress of ongoing testing efforts, share findings, address challenges, and plan the next steps in the testing process. They review test cases, report bugs, and collaborate to make sure the software meets the desired standards before release. It's a crucial forum for coordination, knowledge sharing, and alignment on testing strategies. The meeting helps the team stay on track, prioritize tasks, and maintain open communication, contributing to delivering reliable, bug-free software to users. Overall, the meeting is a very fundamental thing in the software testing process.

8. TEST CASES/TEST ITEMS

Project Name: Vehicle Management System (VMS)			Test Designed by: Nobonita Nonde	
Test Case ID: VMS_1			Test Designed date: 05-08-23	
Test Priority (Low, Medium, High): High			Test Executed by: Ashesh Deb Priom	
Module Name: Login			Test Execution date: 06-08-23	
Test Title: Verify login with valid Username and Password				
Description: Test web application login page				
The precondition (If any): The user must have valid mail and password				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website or application 2. Enter email/phone 3. Enter the password 4. Click login	Email/phone : 01317810827 Password: dehan22	Users should login to the application.	As expected,	Passed
Post Condition: The user is validated with the database and successfully login to the account. The account session details are logged in the database.				

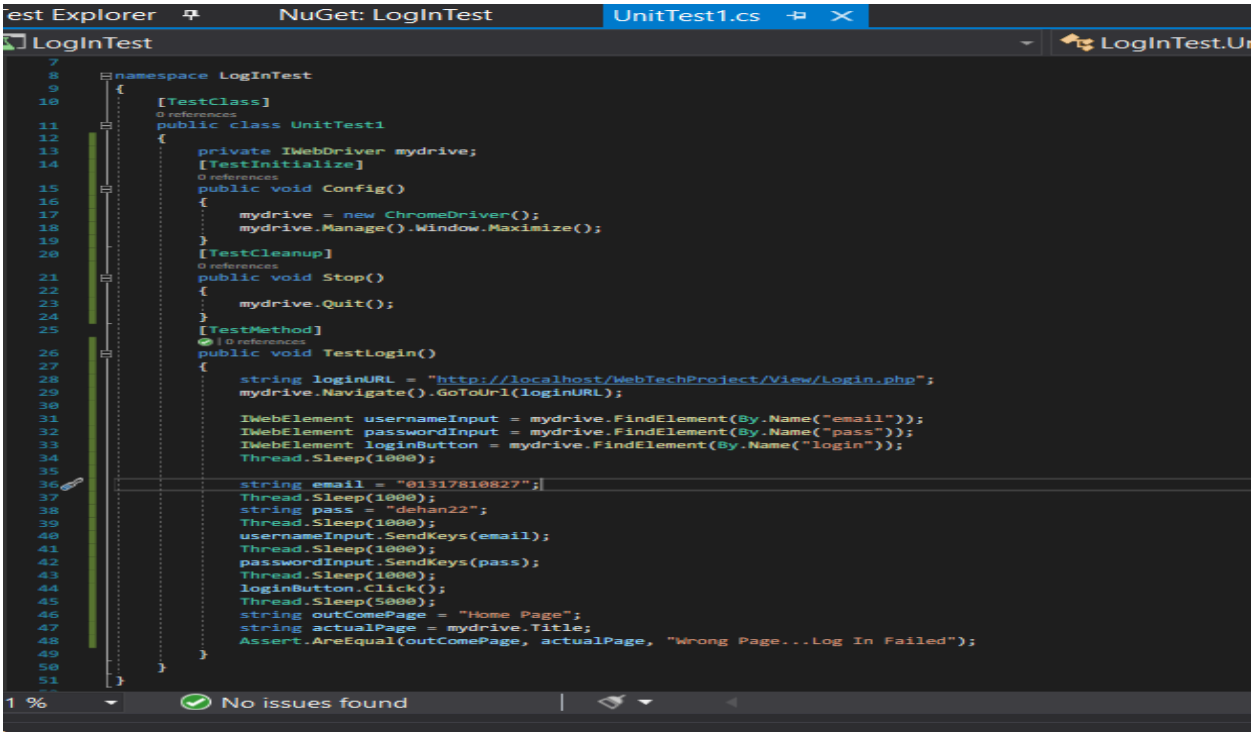


Figure 14: Code for login test

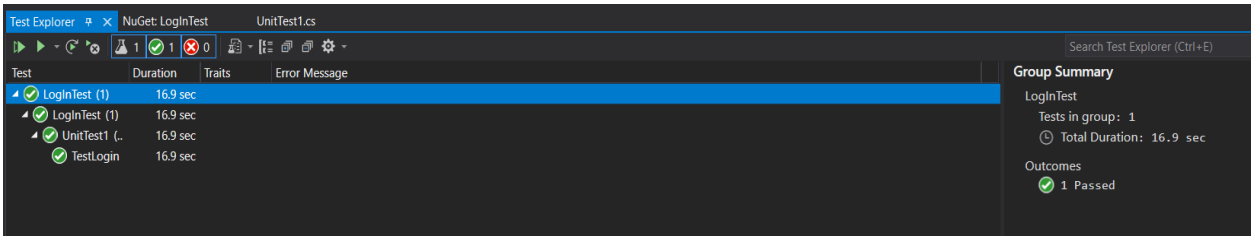


Figure 15: Login test pass.

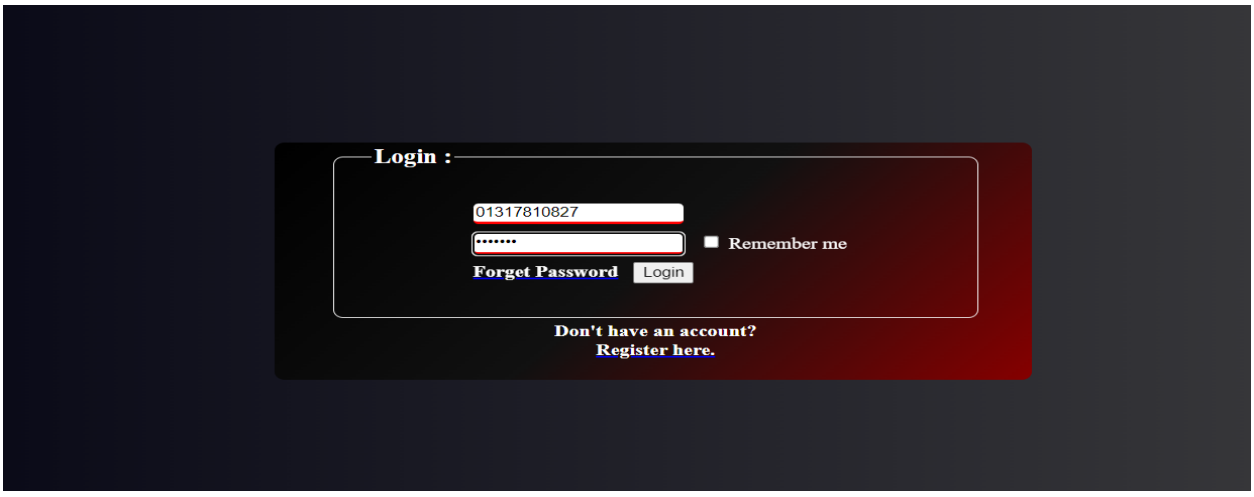


Figure 16: Login page automatic data insertion.

Project Name: Vehicle Management System (VMS)	Test Designed by: MD. SHAFWAN AHMED DEHAN
Test Case ID: VSM_2	Test Designed date: 06-08-23
Test Priority (Low, Medium, High): High	Test Executed by: MD. SHARIAR MAHMUD SACHCHA
Module Name: Registration	Test Execution date: 07-08-23
Test Title: Register user with First name, Last name, valid mail/Phone number, gender, password, nid, present address	
Description: Test web application registration page	
The precondition (If any): User must have valid mail/phone number and password	

Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website tap on the registration page 2. Enter both first and last name 3. Enter valid email/phone number 4. Select gender 5. Make a password 6. Enter NID number 7.enter present address	Enter first name: Safwan Ahmed Last name: Dehan Enter email/phone: 01518918373 Select gender: Male Password: dehan22 NID: 84569366225 Present address: 10/F-4, Green Heaven	If the all information is correct then signup is done.	As expected	Passed
Post Condition: User is validated with database and successfully registration to the account after successful account creation. The account session details are logged in the database.				

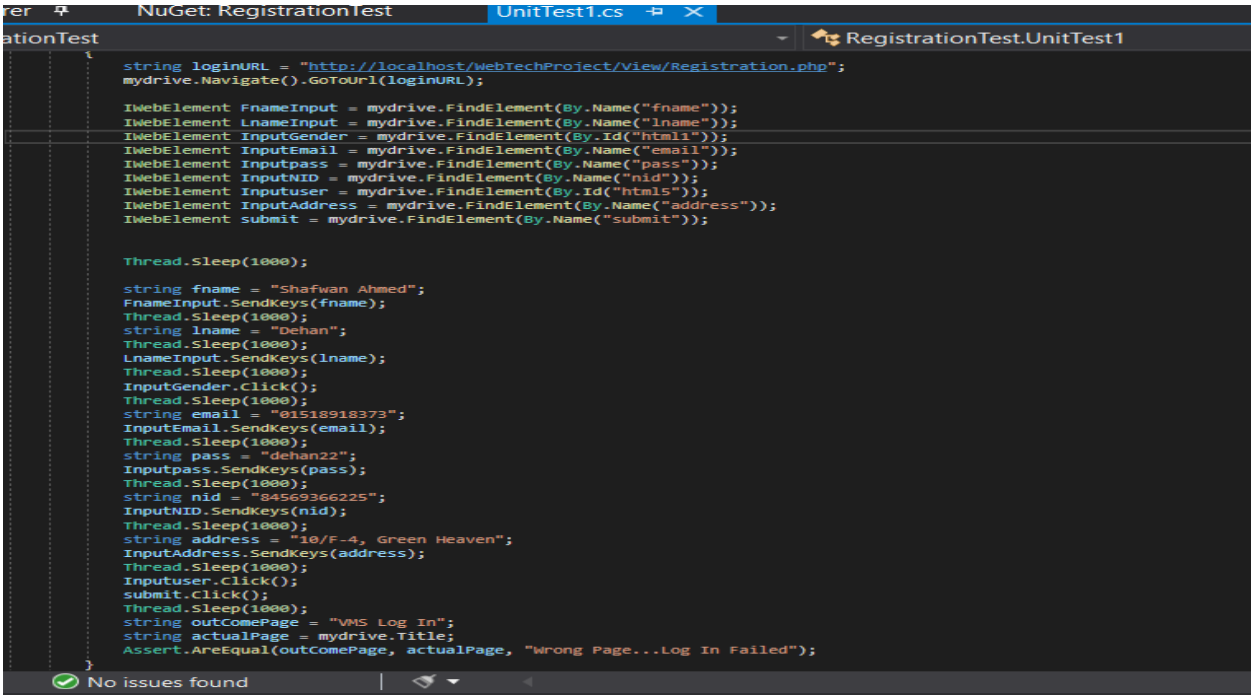


Figure 17: Code for Registration testing.

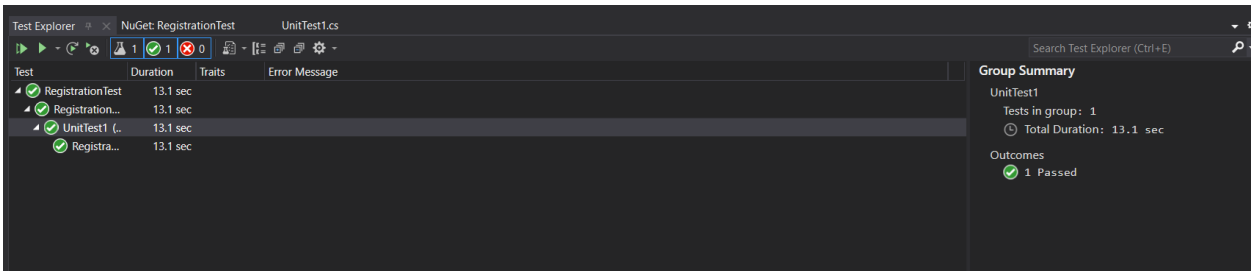


Figure 18: Registration pass.

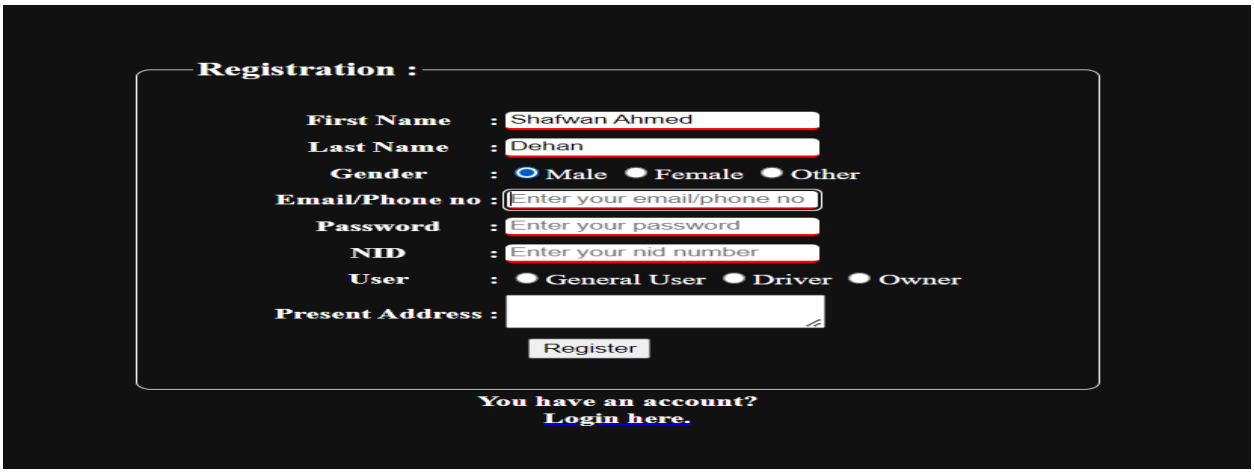


Figure 19: Registration page automatic data insertion.

Project Name: Vehicle Management System			Test Designed by: ASHESH DEB PRIOM	
Test Case ID: VMS_3			Test Designed date: 07-08-23	
Test Priority (Low, Medium, High): Low			Test Executed by: NOBONITA NONDE	
Module Name: SOS			Test Execution date: 07-08-23	
Test Title: Send SMS to emergency contact of the user				
Description: By using this function Updates all the information for the users and adds new info for the users				
The precondition (If any): The user must have a logged in the system, as well as have a emergency contact number.				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website 2. Log in to the system 3. Press on SOS button	Press the SOS button	Which number is set for emergency contact will get a SMS.	As expected	Passed
Post Condition: None				

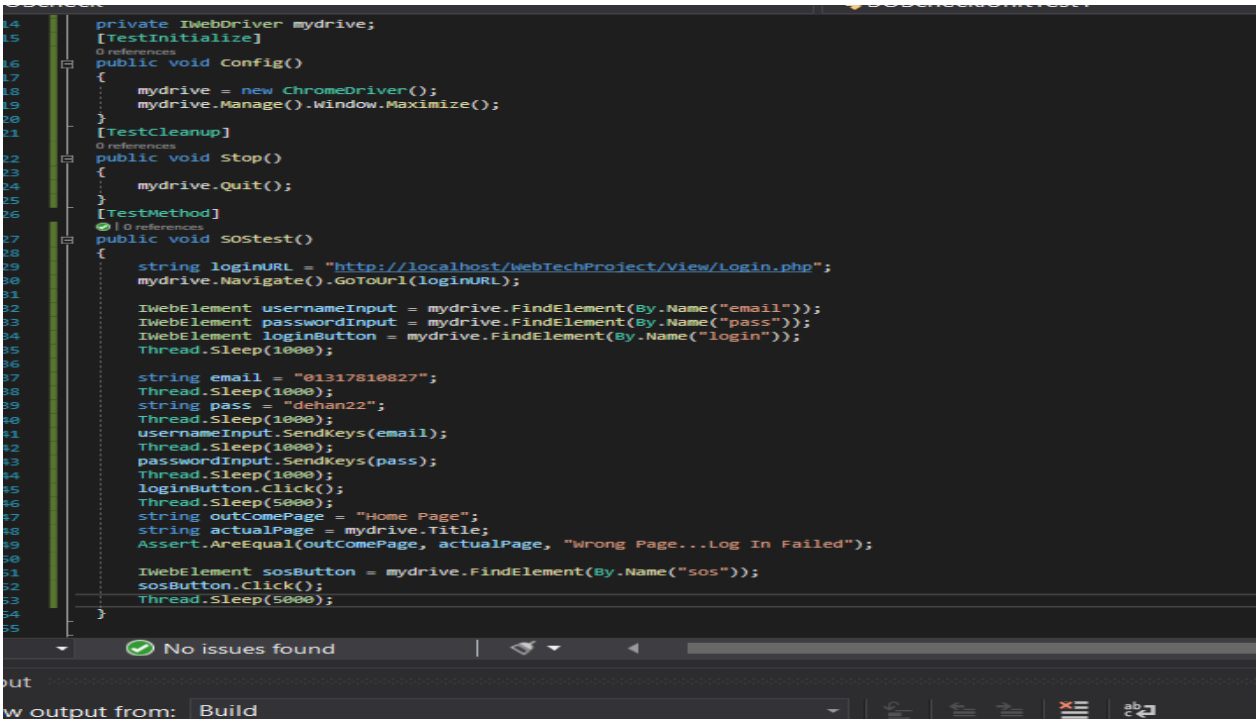


Figure 20: Code for SOS testing.

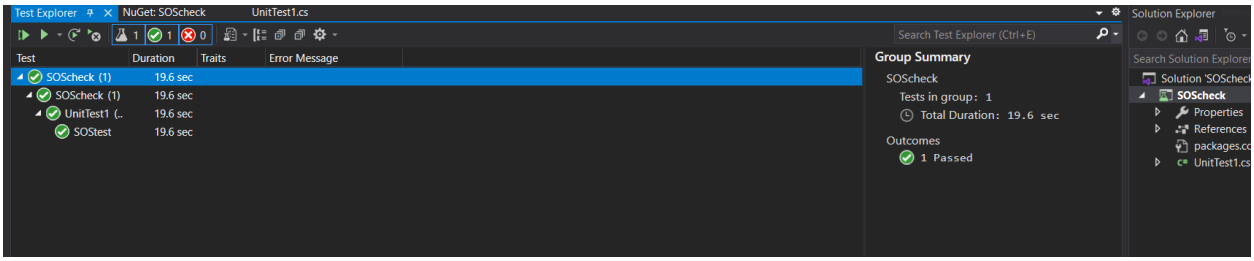


Figure 21: SOS testing pass

Figure 22: SOS page automatic data insertion.

Project Name: Vehicle Management System			Test Designed by: MD. SHAFWAN AHMED DEHAN	
Test Case ID: VMS_4			Test Designed date: 08-08-23	
Test Priority (Low, Medium, High): Medium			Test Executed by: ASHESH DEB PRIOM	
Module Name: Forget Password			Test Execution date: 08-08-23	
Test Title: Profile search for forget password				
Description: To set new user password				
The precondition (If any): The user should be registered previously.				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Click on forget password. 2. In profile search page click on email/number 3. Fill email/number 4. Send OTP button click.	Email/Number: 01317810827	At first login to the system. For adding the vehicles or remove the vehicles must give the medicines name	As expected	Passed
Post Condition: None				

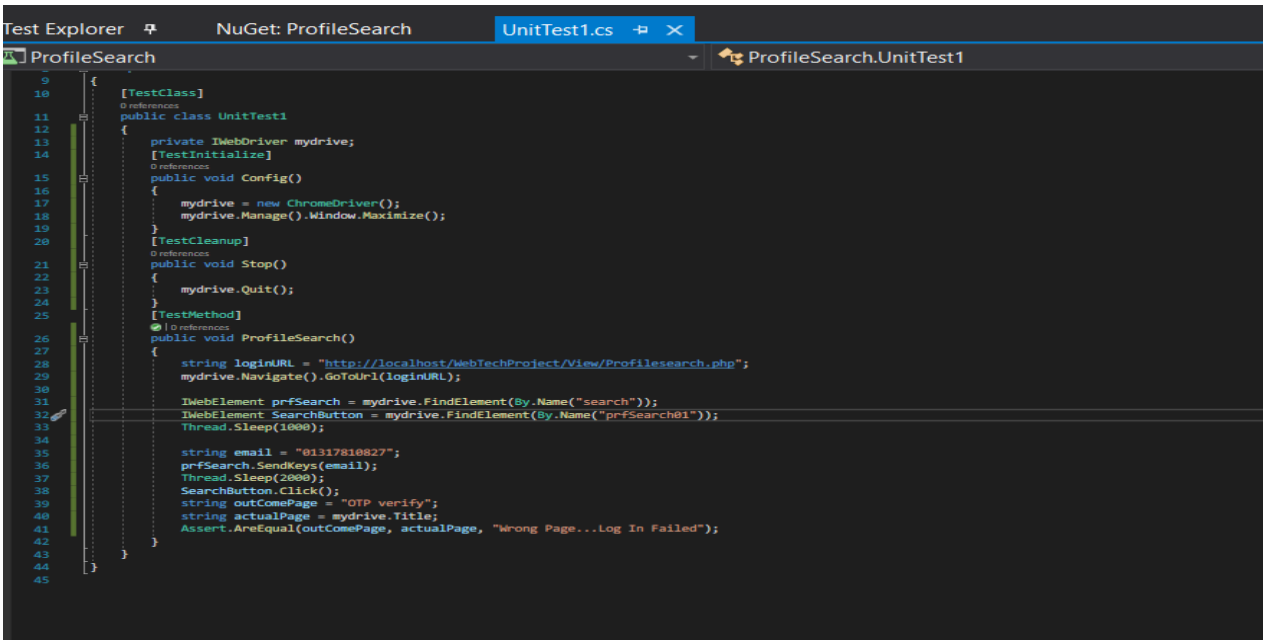


Figure 23: Code for Forget password testing.

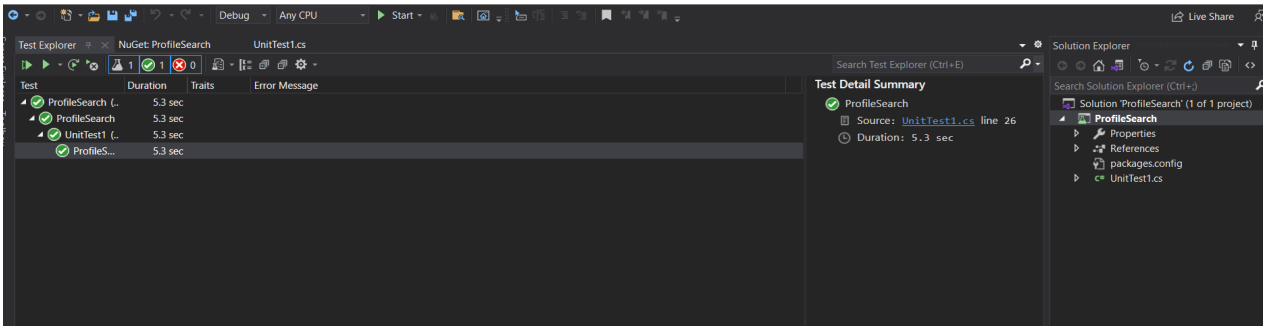


Figure 24: Forget password pass.

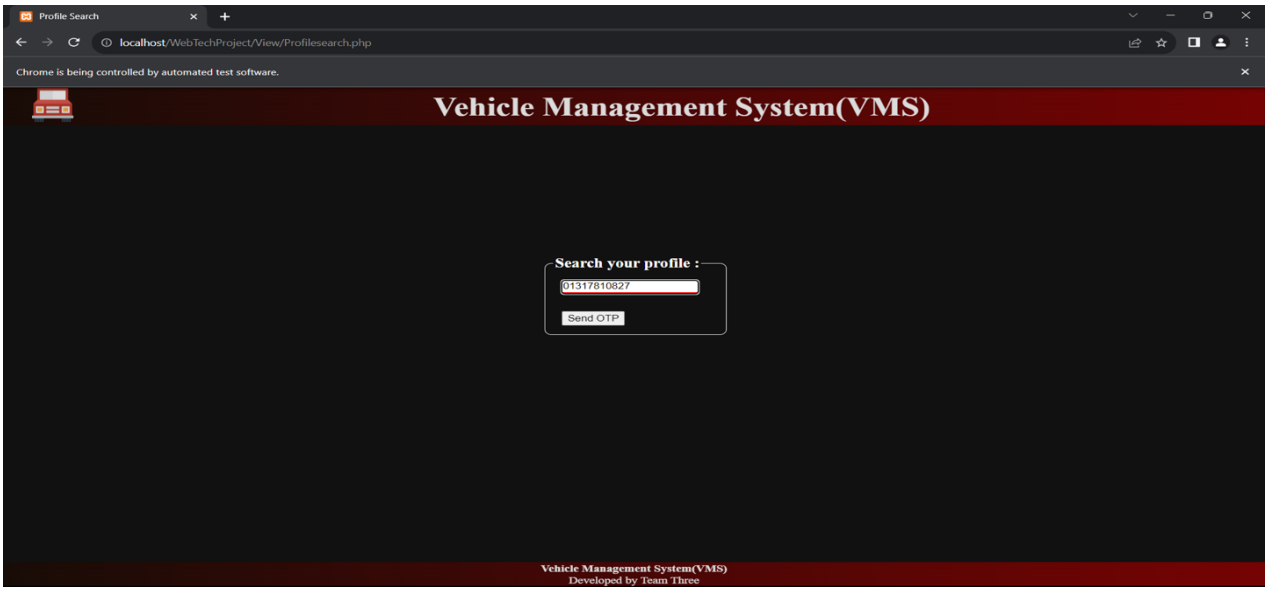


Figure 25: Search profile for forget password page automatic data insertion.

Project Name: Vehicle Management System			Test Designed by: MD. SHARIAR MAHMUD SACHCHA	
Test Case ID: VMS_5			Test Designed date: 08-08-23	
Test Priority (Low, Medium, High): High			Test Executed by: NOBONITA NONDE	
Module Name: Search vehicle			Test Execution date: 09-08-23	
Test Title: Search to find a vehicle				
Description: Search the vehicles to find the description and other details.				
The precondition (If any): The user must have a log in the system, the name/id of searching must be on the database system				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Log in on the system 2. Insert name/id on Search bar 3. Click on search bar	Search: 19393021	At first login to the system. Then it will show the vehicle type, vehicle id, owner name and nid Number.	As expected.	Passed
Post Condition: None				

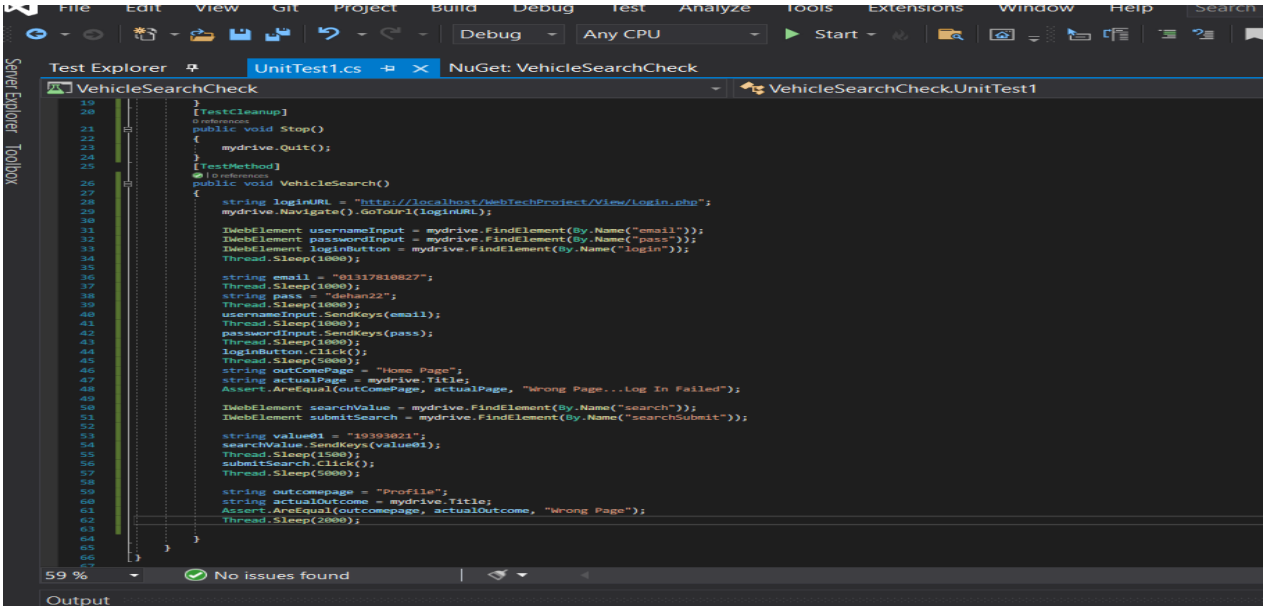


Figure 26: Code for Search vehicle testing.

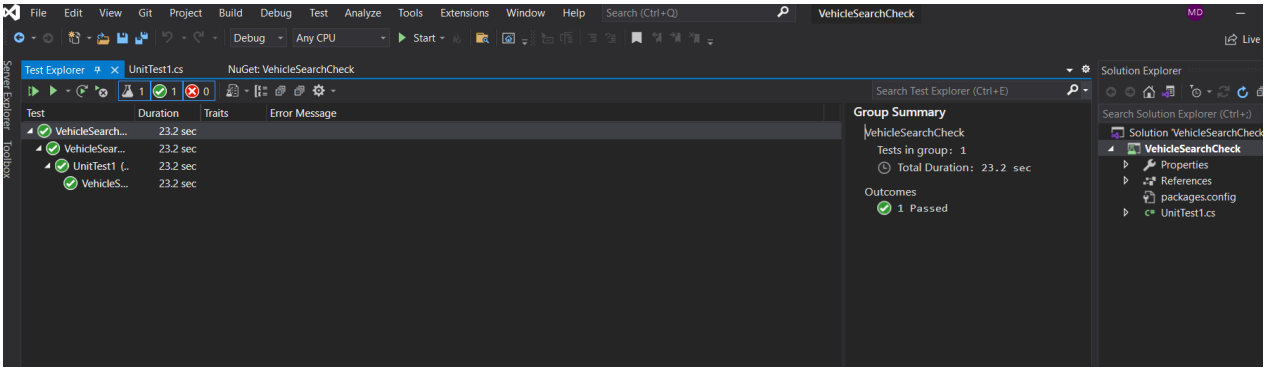
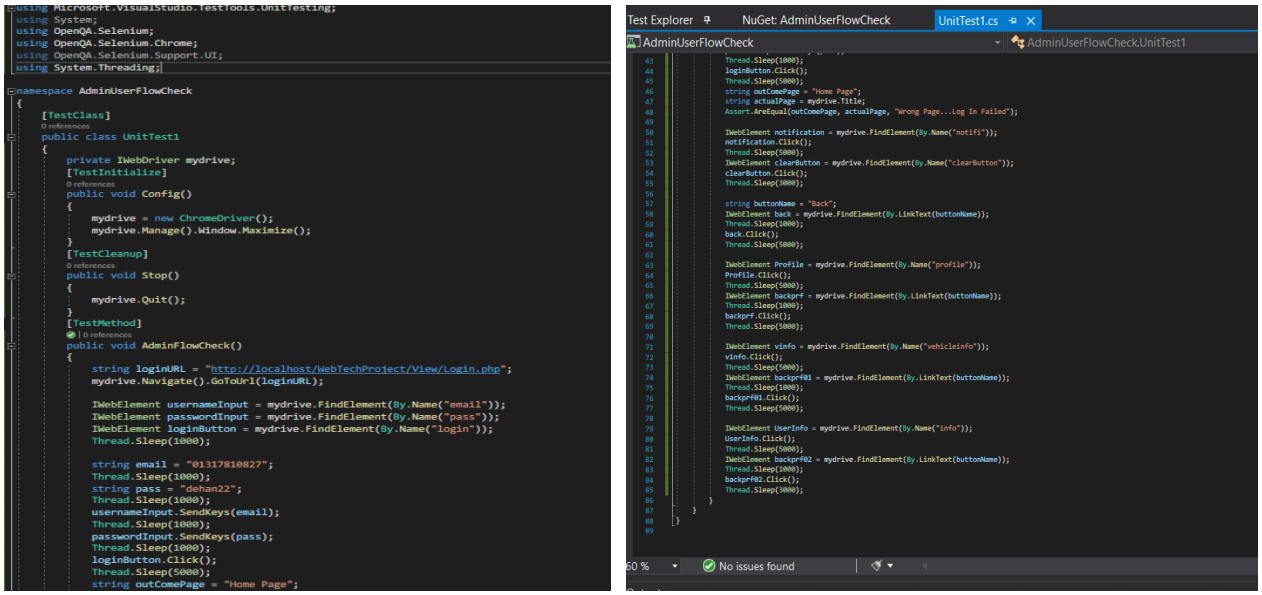


Figure 27: Search vehicle testing pass.

Figure 28: Search vehicle page automatic data insertion.

Project Name: Vehicle Management System (VMS)			Test Designed by: Ashesh Deb Priom	
Test Case ID: VMS_6			Test Designed date: 09-08-23	
Test Priority (Low, Medium, High): Medium			Test Executed by: MD. SHARIAR MAHMUD SACHCHA	
Module Name: Flow Checking			Test Execution date: 10-08-23	
Test Title: Flow check of system				
Description: Check the flow of the full system if the system is working.				
The precondition (If any): The user must have valid mail and password				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Login into the system. 2. Click on every button.	Email/phone: 01317810827 Password: dehan22	All the buttons of page will work properly.	As expected	Passed
Post Condition: None				



```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using System.Threading;

namespace AdminUserFlowCheck
{
    [TestClass]
    public class UnitTest1
    {
        private IWebDriver mydrive;
        [TestInitialize]
        public void Config()
        {
            mydrive = new ChromeDriver();
            mydrive.Manage().Window.Maximize();
        }
        [TestCleanup]
        public void Stop()
        {
            mydrive.Quit();
        }
        [TestMethod]
        public void AdminFlowCheck()
        {
            string loginURL = "http://localhost/WebTechProject/View/Login.php";
            mydrive.Navigate().GoToUrl(loginURL);

            IWebElement usernameInput = mydrive.FindElement(By.Name("email"));
            IWebElement passwordInput = mydrive.FindElement(By.Name("pass"));
            IWebElement loginButton = mydrive.FindElement(By.Name("login"));
            Thread.Sleep(1000);

            string email = "01317810827";
            Thread.Sleep(1000);
            string pass = "dehan22";
            Thread.Sleep(1000);
            usernameInput.SendKeys(email);
            Thread.Sleep(1000);
            passwordInput.SendKeys(pass);
            Thread.Sleep(1000);
            loginButton.Click();
            Thread.Sleep(5000);
            string outComePage = "Home Page";
```

```
Test Explorer   NuGet: AdminUserFlowCheck   UnitTest1.cs
AdminUserFlowCheck                          AdminUserFlowCheck.UnitTest1
            Thread.Sleep(1000);
            loginButton.Click();
            Thread.Sleep(5000);
            string outComePage = "Home Page";
            string actualPage = mydrive.Title;
            Assert.AreEqual(outComePage, actualPage, "Wrong Page...Log In Failed");

            IWebElement notification = mydrive.FindElement(By.Name("notifi"));
            notification.Click();
            Thread.Sleep(5000);
            IWebElement clearButton = mydrive.FindElement(By.Name("clearButton"));
            clearButton.Click();
            Thread.Sleep(3000);

            string buttonName = "Back";
            IWebElement back = mydrive.FindElement(By.LinkText(buttonName));
            Thread.Sleep(1000);
            back.Click();
            Thread.Sleep(5000);

            IWebElement Profile = mydrive.FindElement(By.Name("profile"));
            Profile.Click();
            Thread.Sleep(5000);
            IWebElement backprf = mydrive.FindElement(By.LinkText(buttonName));
            Thread.Sleep(1000);
            backprf.Click();
            Thread.Sleep(5000);

            IWebElement vinfo = mydrive.FindElement(By.Name("vehicleinfo"));
            vinfo.Click();
            Thread.Sleep(5000);
            IWebElement backprf01 = mydrive.FindElement(By.LinkText(buttonName));
            Thread.Sleep(1000);
            backprf01.Click();
            Thread.Sleep(5000);

            IWebElement UserInfo = mydrive.FindElement(By.Name("info"));
            UserInfo.Click();
            Thread.Sleep(5000);
            IWebElement backprf02 = mydrive.FindElement(By.LinkText(buttonName));
            Thread.Sleep(1000);
            backprf02.Click();
            Thread.Sleep(3000);
        }
    }
}
60 %   No issues found
```

Figure 29: Code for Admin all feature testing.

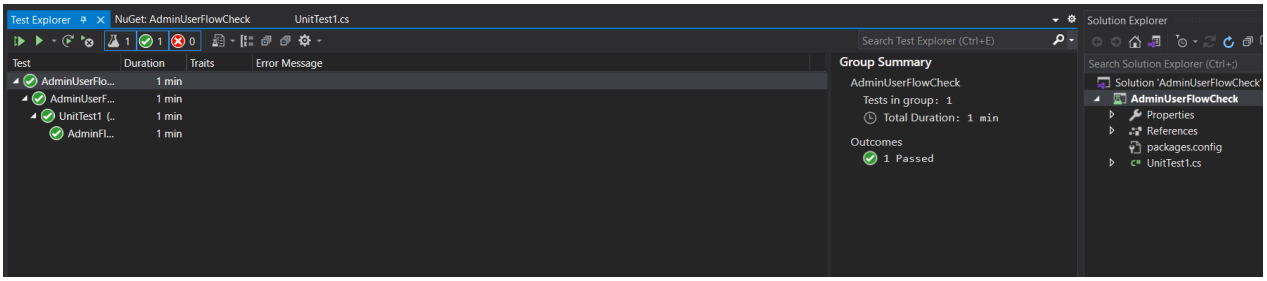


Figure 30: Search vehicle testing pass.

Figure 31: Automatic login Admin profile.

Figure 32: Automatic perform SOS.

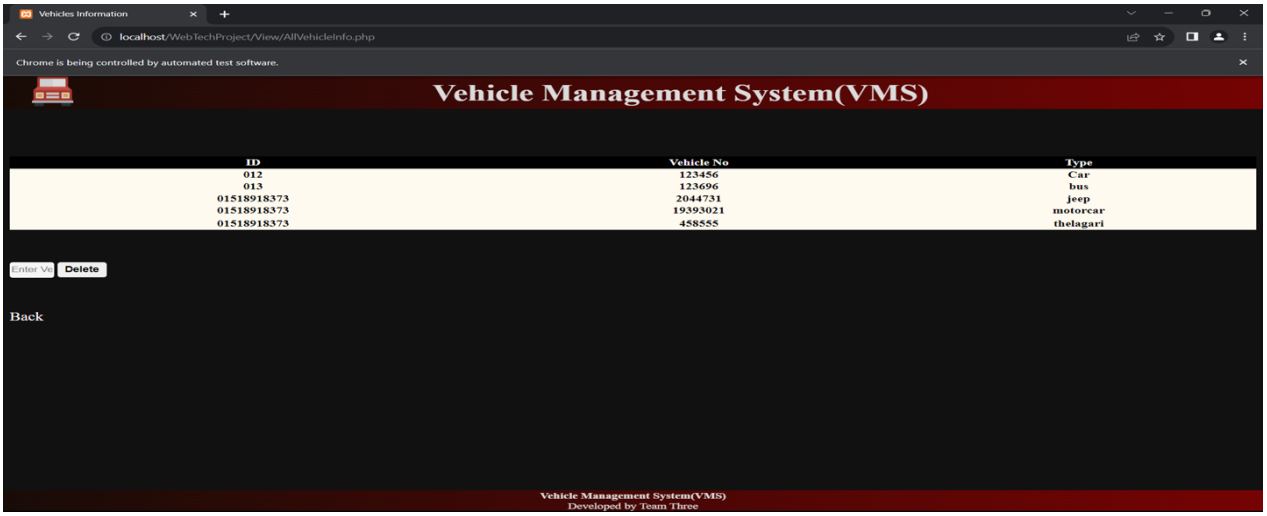


Figure 33: Automatic click Vehicle Information.

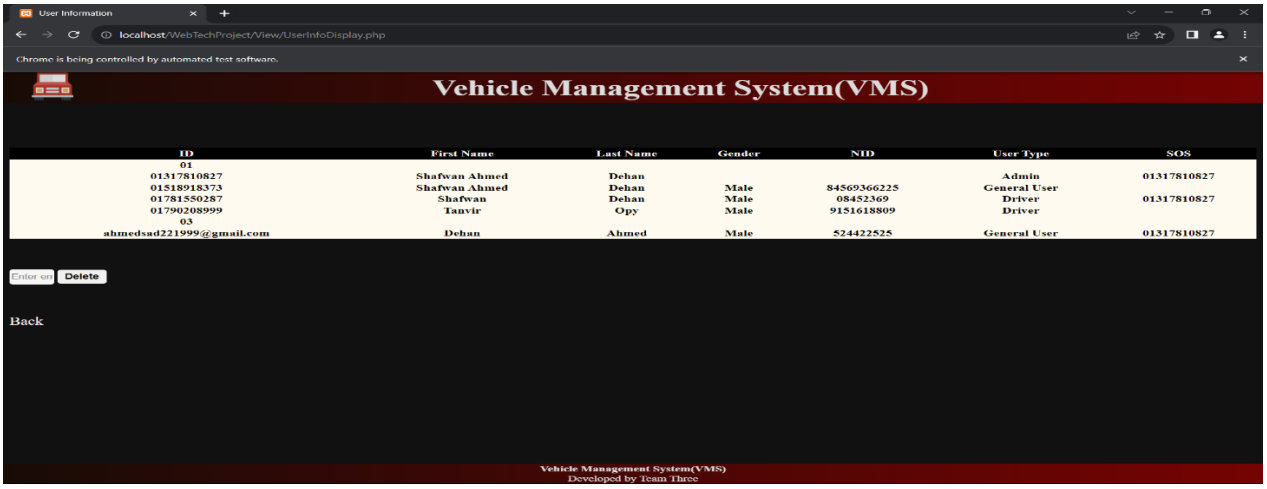
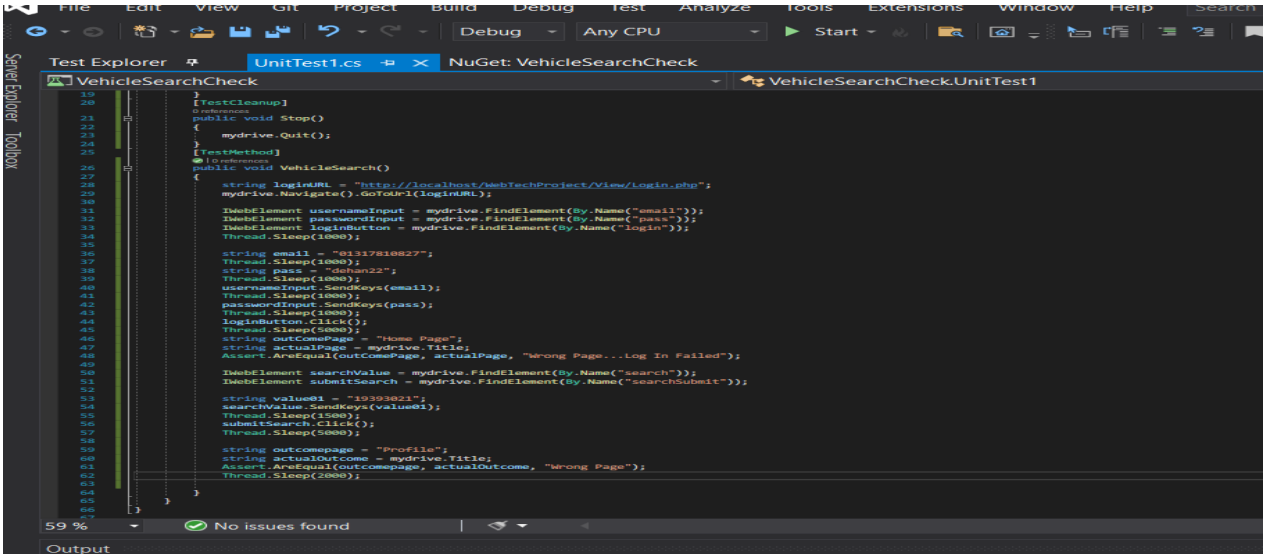


Figure 34: Automatic perform User details

Figure 35: Automatic perform Notification.

Project Name: Vehicle Management System			Test Designed by: MD. SHARIAR MAHMUD SACHCHA	
Test Case ID: VMS_7			Test Designed date: 08-08-23	
Test Priority (Low, Medium, High): Medium			Test Executed by: NOBONITA NONDE	
Module Name: Add vehicle			Test Execution date: 09-08-23	
Test Title: Add a vehicle to a customer				
Description: Customers add vehicle to their system.				
The precondition (If any): The user must have a log in the system, the name/id of searching must be on the database system				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Log in on the system 2. Insert vehicle type 3. Insert vehicle number	Vehicle type: jeep ID: 20436982	User will get a message of the vehicle they added and all details of their vehicle.	As expected.	Passed
Post Condition: None				



```
[TestCleanup]
public void Stop()
{
    mydrive.Quit();
}
[TestMethod]
public void VehicleSearch()
{
    string loginURL = "http://localhost/WebTechProject/View/Login.php";
    mydrive.Navigate().GoToUrl(loginURL);

    IWebElement usernameInput = mydrive.FindElement(By.Name("email"));
    IWebElement passwordInput = mydrive.FindElement(By.Name("pass"));
    IWebElement loginButton = mydrive.FindElement(By.Name("login"));
    Thread.Sleep(1000);

    string email = "01317810827";
    Thread.Sleep(1000);
    string pass = "dehan22";
    Thread.Sleep(1000);
    usernameInput.SendKeys(email);
    Thread.Sleep(1000);
    passwordInput.SendKeys(pass);
    Thread.Sleep(1000);
    loginButton.Click();
    Thread.Sleep(5000);
    string outComePage = "Home Page";
    string actualPage = mydrive.Title;
    Assert.AreEqual(outComePage, actualPage, "Wrong Page...Log In Failed");

    IWebElement searchValue = mydrive.FindElement(By.Name("search"));
    IWebElement submitSearch = mydrive.FindElement(By.Name("searchSubmit"));

    string value01 = "19393021";
    searchValue.SendKeys(value01);
    Thread.Sleep(1500);
    submitSearch.Click();
    Thread.Sleep(5000);

    string outcomepage = "Profile";
    string actualOutcome = mydrive.Title;
    Assert.AreEqual(outcomepage, actualOutcome, "Wrong Page");
    Thread.Sleep(2000);

}
}
```

Figure 36: Code for Add Vehicle testing.

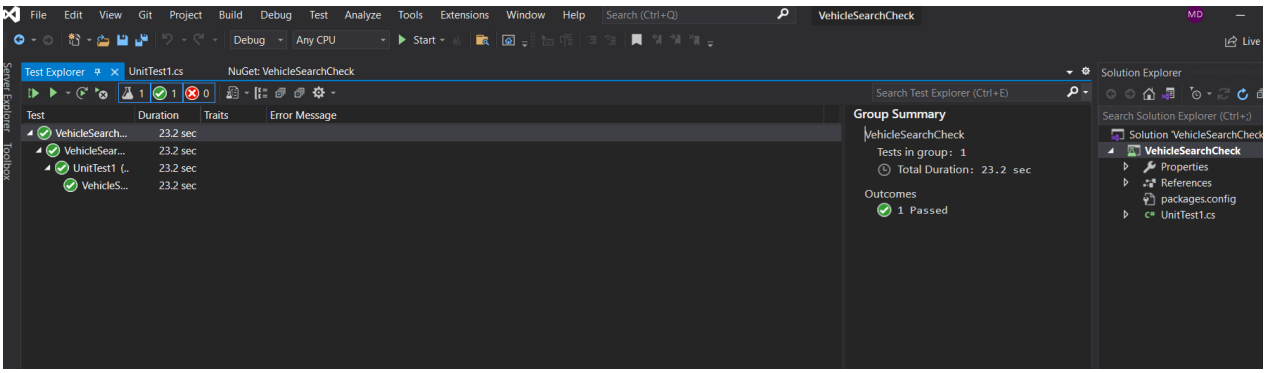


Figure 37: Testing result for Add Vehicle.

Figure 38: Automatic data insertion and perform next page.

Project Name: Vehicle Management System	Test Designed by: MD. SHAFWAN AHMED DEHAN
Test Case ID: VMS_8	Test Designed date: 08-08-23
Test Priority (Low, Medium, High): Medium	Test Executed by: MD. SHARIAR MAHMUD SACHCHA
Module Name: Logout	Test Execution date: 09-08-23
Test Title: Logout from the system.	
Description: Customers add vehicle to their system.	

The precondition (If any): The user must log in to the system, then logout button will appear.

Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Log in on the system 2. Press logout button.	N/A	User will back to the login page and session will be deleted.	As expected.	Passed

Post Condition: None

```
{
    private IWebDriver mydrive;
    [TestInitialize]
    0 references
    public void Config()
    {
        mydrive = new ChromeDriver();
        mydrive.Manage().Window.Maximize();
    }
    [TestCleanup]
    0 references
    public void Stop()
    {
        mydrive.Quit();
    }
    [TestMethod]
    | 0 references
    public void TestMethod1()
    {
        string loginURL = "http://localhost/WebTechProject/View/Login.php";
        mydrive.Navigate().GoToUrl(loginURL);

        IWebElement usernameInput = mydrive.FindElement(By.Name("email"));
        IWebElement passwordInput = mydrive.FindElement(By.Name("pass"));
        IWebElement loginButton = mydrive.FindElement(By.Name("login"));
        Thread.Sleep(1000);

        string email = "01317810827";
        Thread.Sleep(1000);
        string pass = "dehan22";
        Thread.Sleep(1000);
        usernameInput.SendKeys(email);
        Thread.Sleep(1000);
        passwordInput.SendKeys(pass);
        Thread.Sleep(1000);
        loginButton.Click();
        Thread.Sleep(5000);
        string outComePage = "Home Page";
        string actualPage = mydrive.Title;
        Assert.AreEqual(outComePage, actualPage, "Wrong Page...Log In Failed");

        IWebElement logoutButton = mydrive.FindElement(By.Name("logout"));
        logoutButton.Click();
        Thread.Sleep(5000);
        string outputpage = "VMS Log In";
        string actualOutput = mydrive.Title;
        Assert.AreEqual(outputpage, actualOutput, "Wrong Page...");
    }
}
```

No issues found

Figure 39: Code for logout testing.

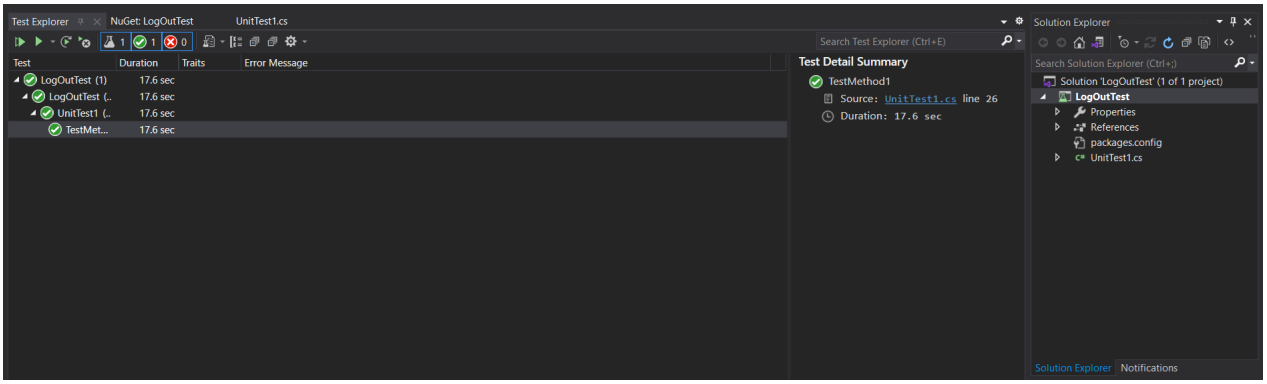


Figure 40: Logout testing result.

9. ITEM PASS/FAIL CRITERIA

The "passed/failed" criteria in software testing are straightforward measures used to determine the success or failure of individual test cases. These criteria serve as a clear indicator of whether the software meets the expected standards and is ready for release.

1. Passed Criteria: When a test case is marked as "passed," it means that the specific functionality or scenario being tested has performed as expected. The software has met the predefined requirements, and no critical defects were detected during the test. Passing criteria often involve ensuring correct results, proper functionality, and adherence to specifications.
2. Failed Criteria: On the other hand, when a test case is marked as "failed," it indicates that the software did not meet the expected outcome. A test case fails when there are discrepancies between the observed behavior and the desired behavior. This can occur due to functional issues, unexpected errors, performance bottlenecks, or other reasons. Failed criteria typically require further investigation, analysis, and debugging to understand the root cause of the failure.

The decision to pass or fail a test case is based on predefined acceptance criteria, which are established before testing begins. These criteria are designed to align with the software's requirements and user expectations. The goal is to ensure that the software is thoroughly evaluated, and any issues are identified and addressed before the software is released to users. Using passed/failed criteria provides a clear and objective assessment of the software's quality, allowing the testing team and stakeholders to make informed decisions about the readiness of the software for the next stages of development or deployment.

10. TEST DELIVERABLES

- **Acceptance test plan:** The acceptance test plan is a document outlining the strategy and criteria for validating that a software product meets user requirements and is ready for deployment. It defines the scope of testing, identifies the test scenarios, and specifies the entry/exit criteria for acceptance testing. This plan ensures that the software aligns with user expectations and business objectives, serving as a guideline for stakeholders to assess whether the product is suitable for Acceptance and ready to move into production.

- **System/Integration test plan:** The system/integration test plan outlines how different components of a software system will be tested together (integration testing) and how the entire system will be tested as a whole It includes the scope, test scenarios, test environment, and criteria for validating that the integrated system functions properly, ensuring that all parts work harmoniously and any potential issues are identified before deployment.

- **Unit test plans/turnover documentation:** Screen prototypes are visual representations of software user interfaces, illustrating the layout, design, and interactions. These mockups serve as a tangible preview for stakeholders, aiding in clarifying requirements, gathering feedback, and ensuring that
- the final product aligns with user expectations before actual development begins.

- **Screen prototypes:** Screen prototypes are visual representations of software user interfaces, illustrating the layout, design, and interactions. These mockups serve as a tangible preview for stakeholders, aiding in clarifying requirements, gathering feedback, and ensuring that the final product aligns with user expectations before actual development begins.

- **Defect/Incident reports and summaries:** Defect/Incident reports and summaries for VMS (Vehicle Management System) software provide concise records of identified issues or anomalies in the system's functionality or performance. These reports highlight the defect's nature, severity, steps to reproduce, and relevant system information. Summaries offer an overview of all reported defects, enabling prioritization and efficient resolution by the development team, ensuring the VMS software meets quality standards and user expectations.

- **Test logs and turnover reports:** Test logs and turnover reports are essential documentation in software testing. Test logs record detailed information about the execution of test cases, including timestamps, test case IDs, outcomes (pass/fail), and any deviations from expected results. Turnover reports provide a comprehensive summary of the testing phase, highlighting key findings, outstanding issues, and the overall quality of the software. These documents ensure transparency, facilitate communication between testing and development teams, and guide decision-making for further development or deployment, contributing to the delivery of a robust and reliable software.

11. STAFFING AND TRAINING NEEDS

Staffing and training are essential for a proficient software quality testing team. Your team needs a diverse range of skills to ensure thorough testing. This includes testers who can meticulously design test cases, execute them, and report issues. Also, having test automation engineers who specialize in creating and maintaining automated test scripts is vital for efficiency. Quality assurance analysts should be on board to establish quality standards and monitor overall quality.

Consider the application's domain. If the software has various type of domain to perform such as finance, accounting, weather etc on that time it may require the expert people in the team. When it comes to training, your team should be well-versed in different testing methodologies, such as Agile or DevOps, to fit seamlessly into the development process. Testers need to be trained in test automation tools and best practices. Additionally, skills in performance testing (using tools like JMeter) and security testing should be imparted to identify vulnerabilities. It's not just technical skills that matter. Soft skills are equally important. Enhancing communication, collaboration, and problem-solving abilities is crucial as testers work closely with developers and must clearly communicate issues.

Lastly, encourage continuous learning. The field of software quality testing evolves rapidly, with new tools and methods emerging regularly. Encourage your team to attend workshops, webinars, and conferences to stay updated on the latest testing trends and technologies. This ensures your team's skills remain up-to-date and the testing process remains effective, contributing to the delivery of high-quality software products.

12. RESPONSIBILITIES

	TM	PM	Dev Team	Test Team	Client
Acceptance test Documentation and Execution	✗	✗	✓	✗	✗
System/ Integration test Documentation and Execution	✗	✓	✗	✗	✓
Unit test Documentation and Execution	✗	✓	✗	✗	✓
System Design Reviews	✗	✗	✗	✗	✗
Details Design Reviews	✗	✗	✗	✗	✓
Test Procedures and Rules	✗	✗	✗	✗	✓
Screen and Report Prototype Reviews	✓	✓	✗	✗	✗
Change Control and Regression Testing	✗	✗	✗	✗	✗

Figure 41: Responsibilities.

13. TESTING SCHEDULE

The project plan for the vehicle management system includes specific time allocations for testing activities. These activities are detailed in the project timeline and plan, including the dates and times for each task and the personnel required for each process. The project manager will coordinate the test team, development team, management, and customer to ensure smooth execution of each task. To facilitate this, a project management tool will be utilized to create and maintain the schedule. By allocating time and resources for testing, the project team can ensure that the system is thoroughly tested and validated before being delivered to the customer.

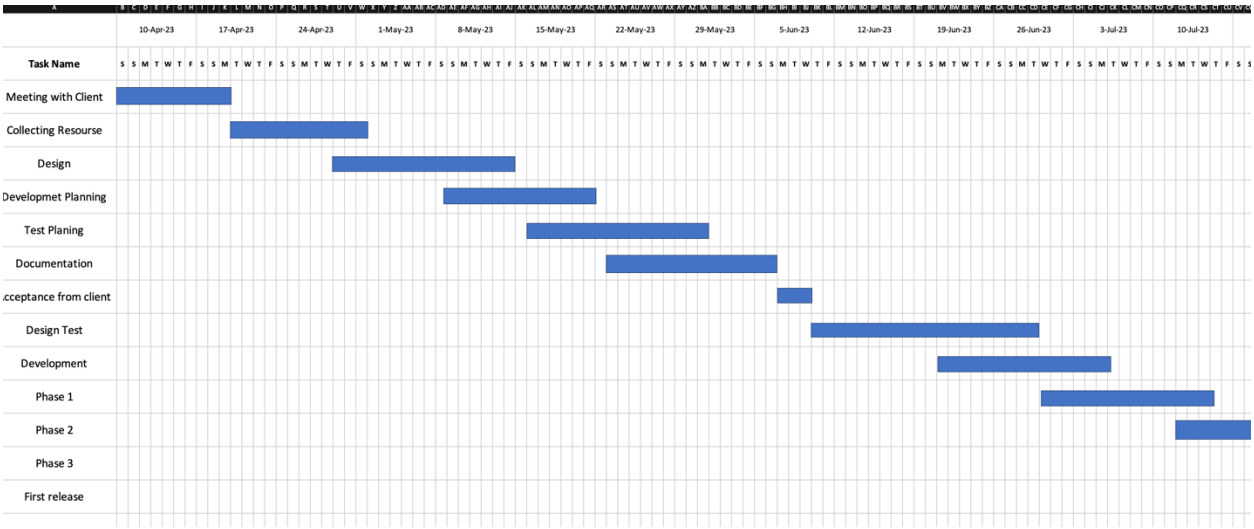


Figure 42: Time scheduling.

14. PLANNING RISKS AND CONTINGENCIES

Planning for risks and contingencies in software testing is essential to ensure a smooth testing process and mitigate potential issues. There are some approaches for handling them.

- Identify Risks: Begin by identifying potential risks that could impact the testing process. These risks could include technical challenges, resource constraints, scope changes, communication issues, or unexpected dependencies.
- Prioritize Risks: Prioritize the identified risks based on their potential impact and likelihood. Focus on addressing the most critical risks first, as they can have a significant impact on the testing timeline, quality, and project success.
- Monitor and Communicate: Continuously monitor the progress of the testing process and the status of identified risks. Regularly communicate the status of risks and contingencies to relevant stakeholders, including project managers, developers, and QA teams.
- Proper Documentation: After the testing phase, analyze the effectiveness of your risk management approach. Document lessons learned, noting what worked well and what could be improved for future projects.

By proactively planning for risks and having well-defined contingencies in place, your software testing process becomes more resilient. This helps ensure that potential issues are addressed promptly, minimizing disruptions and increasing the likelihood of a successful testing phase and project outcome.

15. APROVALS

Project Manager	APPROVED
Developer	APPROVED
Test Planner	APPROVED
Test Lead	APPROVED
Tester	APPROVED
End User	APPROVED

16. INDUSTRY VISIT PHOTO

Figure 43: Photo of Industry Visit.