Модификации ПО:

1. Внедрить систему уведомлений для клиентов об изменении статуса их заявки (через SMS или email).

2. Добавить возможность прикрепления фотографий или видео к заявкам для более точного описания проблемы.

3. Реализовать рейтинг механиков, основанный на оценках клиентов после завершения работы.

4. Ввести систему отчетов для анализа времени выполнения ремонтов и загрузки механиков.

5. Добавить возможность онлайн-оплаты услуг через интеграцию с платежными системами.

6. Внедрить автоматическое назначение заявок механикам в зависимости от их текущей загрузки.

7. Разработать мобильное приложение для клиентов, где можно отслеживать статус ремонта и записываться на сервис.

8. Добавить функционал для расчета примерной стоимости ремонта на основе описания проблемы.

9. Реализовать интеграцию с внешними системами учета запчастей для автоматического обновления информации о наличии.

10. Внедрить систему лояльности с накопительными скидками для постоянных клиентов.

11. Добавить возможность отправки клиентам напоминаний о необходимости прохождения технического обслуживания.

12. Ввести графики работ для механиков с возможностью их редактирования и мониторинга.

13. Реализовать функционал для отслеживания заявок в реальном времени с визуализацией этапов работы.

14. Добавить поддержку нескольких языков для интерфейса, чтобы расширить возможности использования для иностранных клиентов.

15. Создать модуль статистики для отслеживания доходов и расходов автосервиса.

16. Ввести возможность сохранения истории всех изменений, сделанных в заявке (например, комментарии, изменение статуса).

17. Разработать API для интеграции с другими системами (например, складские или бухгалтерские системы).

18. Внедрить систему предварительной записи с календарем доступных дат и времени.

19. Добавить функционал отправки автоматизированных счетов клиентам после завершения ремонта.

20. Реализовать возможность динамической фильтрации заявок по различным критериям, таким как статус, механик, клиент, дата.

**Качественные характеристики кода**

1. **Полнота обработки ошибочных данных**

В коде программы предусмотрена обработка возможных ошибок при взаимодействии с базой данных. Используются блоки try-catch, которые перехватывают исключения, возникающие при работе с SqlConnection и SqlCommand. В случае возникновения ошибки выводится сообщение с описанием ошибки:

try

{

connection.Open();

// Запрос для получения всех заявок с присоединением к другим таблицам для отображения имен клиентов и механиков

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке заявок: {ex.Message}");

}

finally

{

connection.Close();

}

}

Этот подход позволяет избежать некорректной работы программы при ошибках, связанных с подключением к базе данных или некорректным SQL-запросом.

1. **Наличие тестов для проверки допустимых значений входных данных.**

В коде реализована проверка допустимых значений при вводе данных в форме авторизации. Например, при вводе логина и пароля проверяется их наличие в бд через метод ValidateUser:

private bool ValidateUser(string login, string password)

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

SqlDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

string role = reader.GetString(0);

OpenUserForm(role);

return true;

}

else

{

return false;

}

}

}

Также, в форме авторизации при множественных неудачных попытках входа (3 раза) система блокирует доступ и сообщает об этом пользователю:

private int blockTime = 180000; // 3 минуты блокировки

if (attemptCount > 2)

{

BlockUser();

}

if (isBlocked)

{

MessageBox.Show("Вход заблокирован. Подождите 3 минуты.", "Блокировка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Наличие средств контроля корректности входных данных**

В коде присутствует множество проверок корректности входных данных, например, пустых полей при создании пользователем заявки и вывода сообщения об этом:

if (string.IsNullOrWhiteSpace(textBoxCarType.Text))

{

MessageBox.Show("Введите вид автомобиля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

1. **Наличие средств восстановления при сбоях оборудования**

Код не содержит явных механизмов восстановления после сбоев оборудования. Тем не менее, использование обработки исключений позволяет в определенной степени реагировать на проблемы с оборудованием (например, сбои при подключении к базе данных). В случае сбоя система выводит сообщение об ошибке и даёт пользователю информацию о том, что необходимо предпринять.

1. **Наличие комментариев**

В коде присутствуют комментарии для объяснения неоднозначных моментов, например некоторых методов и функций.

1. **Наличие проверки корректности передаваемых данных**

Код тщательно проверяет корректность вводимых пользователем данных, таких как логин, пароль и капча. Проверка происходит на уровне ввода данных с последующей валидацией с использованием запросов к базе данных:

if (ValidateUser(login, password))

{

attemptCount = 0;

LogLoginAttempt(login, true);

return;

}

else

{

attemptCount++;

LogLoginAttempt(login, false);

if (attemptCount == 1)

{

MessageBox.Show("Неверный логин или пароль. Попробуйте снова.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

ShowCaptcha();

}

1. **Наличие описания основных функций**

Каждый метод выполняет свою отдельную задачу, что соответствует принципам структурированного и читаемого кода.

* button1\_Click(object sender, EventArgs e) - Обработчик события для кнопки "Отправить", выполняет валидацию данных, добавляет информацию о автомобиле и запросе в базу данных, и отображает сообщения об успехе или ошибках.
* button1\_Click - Обрабатывает нажатие кнопки для входа в систему, проверяет, заблокирован ли пользователь, валидирует введенные данные, управляет попытками входа и отображает CAPTCHA при необходимости.
* button2\_Click - Обработчик события нажатия кнопки для удаления заявки. Проверяет, выбрана ли хотя бы одна заявка в DataGridView. Если выбрана, запрашивает подтверждение удаления. При положительном ответе вызывает метод удаления заявки и обновляет данные в таблице. Если ни одна заявка не выбрана, выводит предупреждающее сообщение.
* UpdateRequest - Метод для обновления данных заявки. Проверяет корректность введенных данных с помощью ValidateForm(). Если данные валидны, устанавливает соединение с базой данных и выполняет три обновляющих запроса: обновляет информацию о автомобиле, данные клиента и описание проблемы в заявке. После успешного выполнения запросов выводит сообщение об успешном обновлении, обновляет данные в родительской форме и закрывает текущую форму.

**Заключение:** код хорошо структурирован и включает все необходимые механизмы для обработки ошибок, контроля корректности входных данных и проверки передаваемых данных. Предусмотрены обработка ошибок при работе с базой данных, защита от злоупотребления входом через капчу и блокировку.