**Development Section**

**Prototype 1 - Main Page:**

***Overview:***The prototype for the **main page** of the traffic test application in C# focuses on a user-friendly navigation system between various forms, such as the Practice Page, Mock Test Page, Instructions Page, and Progress Page. Initially, button clicks were implemented to open new forms, with early iterations addressing navigation issues, such as incorrect form references. The logic evolved to introduce an Instructions Page before accessing the Mock Test. Further development included a new **Progress Page** that tracks user performance. This was achieved by incorporating a **dictionary to store test scores** and passing it as a parameter to the Progress Page constructor. Each iteration refined the flow of the main page, ensuring smoother navigation and enhanced functionality for users.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**  
A screenshot of a computer

Description automatically generated

**Problem:** How to Map buttons so that they open a new form?  
**Solution:** Applying the function with the Click Keyword with the name of the button.

private void Practice\_Click(object sender, EventArgs e)

{

PracticePage nextForm = new PracticePage();

nextForm.Show();

this.Hide();

}

Open new form:

PracticePage nextForm = new PracticePage();

nextForm.Show();

**Iteration 2:**

**Obstacle**: The Mock Test Page opening the Practice Page  
**Fix:** Reference Wasn’t Named Right Hence Causing the Practice Page to open Rather Than Mock test Page

private void Mock\_test\_Click(object sender, EventArgs e)

{

MockTest\_Page nextForm = new MockTest\_Page();

nextForm.Show();

this.Hide();

}

**Iteration 3:**

The Mock Test Page button now opens the Instruction form

private void Mock\_test\_Click(object sender, EventArgs e)

{

InstructionsForm nextForm = new InstructionsForm();

nextForm.Show();

this.Hide();

}

**Iteration 4:**

The New Progress\_Page constructor in my new logic which will require a Dictionary<int, int> argument that contains the scores for each test. This dictionary keeps track of which tests have been attempted and their respective scores. I need to pass this dictionary when creating an instance of Progress\_Page. In your MockTest\_Page, I already have a testScores dictionary that tracks the scores for each test. So, I can pass that dictionary when calling the constructor.

private void Progress\_Click(object sender, EventArgs e)

{

Progress\_Page nextForm = new Progress\_Page(GlobalData.TestScores);

nextForm.Show();

this.Hide();

}

**Iteration 5:**

Following the introduction of the Settings Page, the Hazard Perception Video Page, and the Update of Permanent Progress Storage, passing a dictionary parameter containing the test results is no longer required. In addition, I need to design new forms to map the Settings and Hazard Perception Video Page buttons.

private void Progress\_Click(object sender, EventArgs e)

{

Progress\_Page nextForm = new Progress\_Page();

nextForm.Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

Settings\_Page nextForm = new Settings\_Page();

nextForm.Show();

this.Hide();

}

private void hazardvidtester\_Click(object sender, EventArgs e)

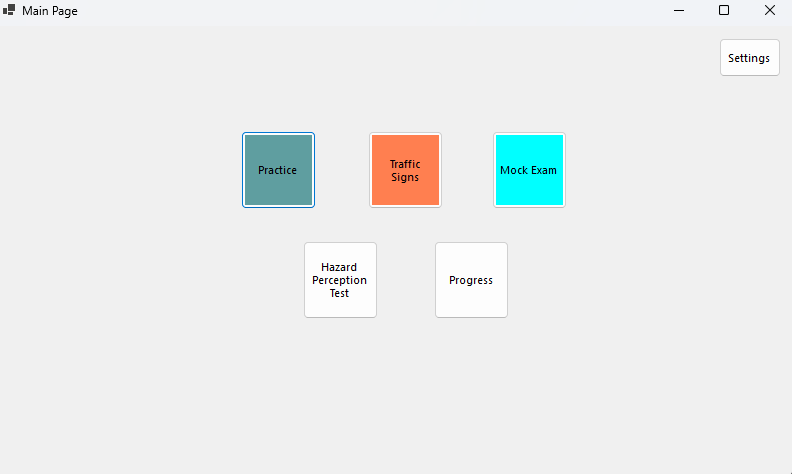
{

Hazard\_info nextForm = new Hazard\_info ();

nextForm.Show();

this.Hide();

}

******

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 2 – Practice Page:**

***Overview:***

The Practice Page is an interactive quiz-based learning platform designed to help users practice and test their knowledge on various topics. Users can select from multiple predefined tests, each containing a unique set of questions. The page presents one question at a time, offering multiple answer options for users to choose from. After submitting an answer, users receive immediate feedback on its correctness or view their results at the end of the test. The system tracks user progress, showing the number of questions answered out of the total, and provides a final score summary upon completion. The interface is simple, intuitive, and user-friendly, allowing for easy navigation between questions. By offering a self-paced learning experience, the Practice Page encourages users to improve their understanding of key concepts while tracking their performance. This approach makes it an effective tool for education, exam preparation, and skill development.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**



**Problem**: How to Map the start button to start the quiz?  
**Solution**: Use A Function class to call another class which clears the form from everything and loads in questions

Controls.Clear();

**Problem**: Tracker to track the question number you are on and iterating to the next question?  
**Solution**: Introduce a Tracker Label ***private Label trackerLabel;***   
The total number of questions in the test is determined by the `selectedTest.Questions.Count` property, which retrieves the size of the `Questions` collection in the `selectedTest` object. This count is used to ensure the test ends when the current question index (`currentQuestionIndex`) reaches the total number of questions

if (selectedTest == null || currentQuestionIndex >= selectedTest.Questions.Count)

**To update the tracker label:** that displays the current question number Each `Question` object in the `Questions` collection contains the question text and its possible options.

trackerLabel.Text = $"Question {currentQuestionIndex + 1} of {selectedTest.Questions.Count}";

**Iterate through the options**: A for loop is used for the collection selectedTest.Questions contains Question objects, where each question has its text (currentQuestion.Text) and a list of options (currentQuestion.Options).

for (int i = 0; i < currentQuestion.Options.Count; i++)

{

RadioButton optionButton = new RadioButton

{

Text = currentQuestion.Options[i],

AutoSize = true,

Tag = i // Store the option index

};

optionButton.Location = new Point((ClientSize.Width - optionButton.Width) / 2, yPosition);

Controls.Add(optionButton);

yPosition += 30;

}

**Iteration 2:**

**Problem**: I need buttons to navigate between questions but in a smoother way, ensuring I don't navigate to a non-existent question.  
**Solution**: I use the currentQuestionIndex variable to keep track of the current question I'm viewing. This variable updates when I navigate between questions, ensuring the correct question from selectedTest.Questions is loaded.

nextButton for Moving Forward by incrementing currentQuestionIndex which calls LoadQuestion() to load the next question

private void NextButton\_Click(object sender, EventArgs e)

{

currentQuestionIndex++;

LoadQuestion();

}

ensures the subsequent question is displayed as long as currentQuestionIndex is less than selectedTest.Questions.Count.

previousButton for Moving Backward by decrementing currentQuestionIndex which calls LoadQuestion() to load the previous question

private void PreviousButton\_Click(object sender, EventArgs e)

{

if (currentQuestionIndex > 0)

{

currentQuestionIndex--;

LoadQuestion();

}

}

Navigation backward is only allowed when currentQuestionIndex is greater than 0, ensuring the user doesn't navigate to a non-existent question.

**Problem**: I need to see how many questions I've gotten right and which option is correct. How will the scoring mechanism work?  
**Solution**: I will compare the selected option with the correct answer for each question. Each Question object has a property called CorrectOptionIndex, which stores the index of the correct option in the Options list.

private void PreviousButton\_Click(object sender, EventArgs e)

{

if (currentQuestionIndex > 0)

{

currentQuestionIndex--;

LoadQuestion();

}

}

When the user clicks the Next button, iterations through the Controls collection are done to check which RadioButton is selected. The Tag property of the selected RadioButton holds the index of the selected option by using foreach:

foreach (Control control in Controls)

{

if (control is RadioButton radioButton && radioButton.Checked)

{

int selectedOption = (int)radioButton.Tag;

...

}

}

The Scoring Mechanism works by comparing the user's selected option (selectedOption) with the correct option (CorrectOptionIndex) for the current question. If they match, the score is incremented:

foreach (Control control in Controls)

{

if (control is RadioButton radioButton && radioButton.Checked)

{

int selectedOption = (int)radioButton.Tag;

...

}

}

At the end of the quiz, the ShowScore() method is called. This method clears the form and displays the user's total score out of the total number of questions:

Label scoreLabel = new Label

{

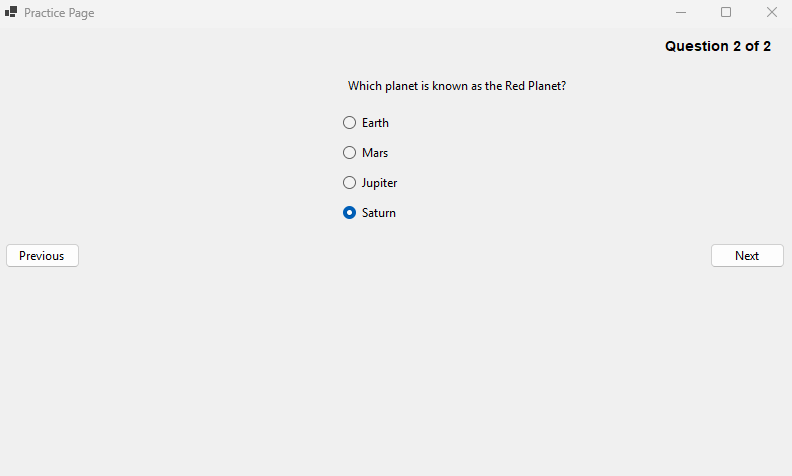
Text = $"Quiz Completed!\nYour score: {score} out of {questions.Count}",

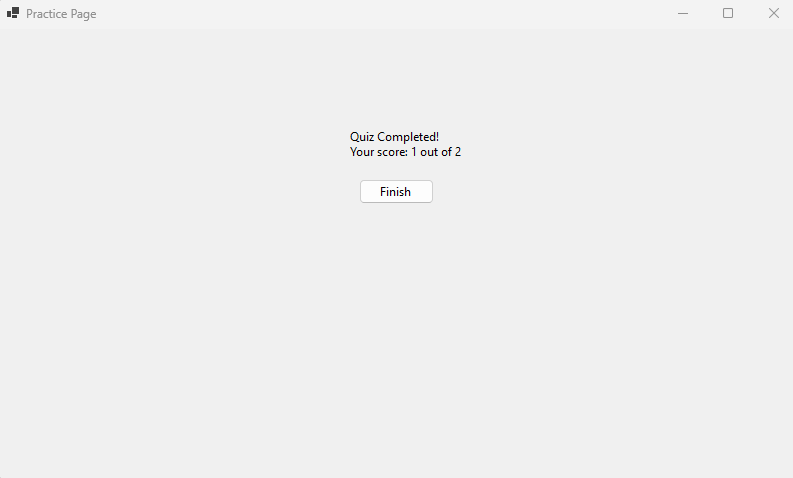
AutoSize = true

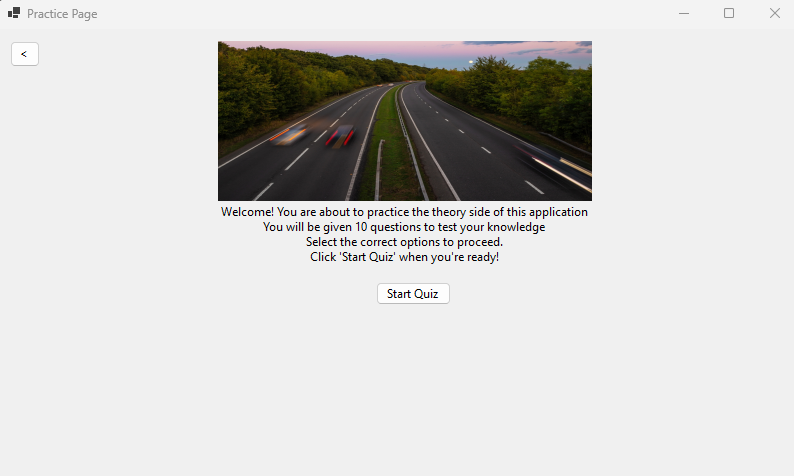
};

***A screenshot of a computer

Description automatically generated***

******

****

**Iteration 3:**   
  
**Problem:** How to Have Multiple Tests with different Questions? **Solution:** I am implementing multiple tests with different questions by organizing the questions into a List<Test>, where each Test object contains its own set of questions.

public *class* Test

{

public List<Question> Questions { *get*; *set*; } = new List<Question>();

}

public *class* Question

{

public string? Text { *get*; *set*; }

public List<string>? Options { *get*; *set*; }

public int CorrectOptionIndex { *get*; *set*; }

}

This structure allows each Test to encapsulate its own unique set of questions.

**Problem:** How to store these tests?  
**Solution:** I am creating multiple tests by using a List<Test> to store multiple tests, with each Test initialized with a unique set of questions.

tests = new List<Test>

{

new Test

{

Questions = new List<Question>

{

new Question

{

Text = "What is the capital of France?",

Options = new List<string> { "Berlin", "Madrid", "Paris", "Rome" },

CorrectOptionIndex = 2

},

new Question

{

Text = "Which planet is known as the Red Planet?",

Options = new List<string> { "Earth", "Mars", "Jupiter", "Saturn" },

CorrectOptionIndex = 1

},

}

},

new Test

{

Questions = new List<Question>

{

new Question

{

Text = "What is the largest ocean on Earth?",

Options = new List<string> { "Atlantic", "Indian", "Arctic", "Pacific" },

CorrectOptionIndex = 3

},

new Question

{

Text = "What is the square root of 64?",

Options = new List<string> { "6", "7", "8", "9" },

CorrectOptionIndex = 2

},

}

},

// More tests can be added here

};

**Problem:** How to make the tests Random and Dynamic at the same time? Why To Choose Tests at Random?   
**Solution:** I am making the test selection random by using the Random class in the StartQuizButton\_Click method.

private void StartQuizButton\_Click(object? sender, EventArgs e)

{

Random random = new Random(); // Create a new instance of the Random class

selectedTest = tests[random.Next(tests.Count)]; // Randomly select a test

currentQuestionIndex = 0;

score = 0;

LoadQuestion(); // Start loading the first question

}

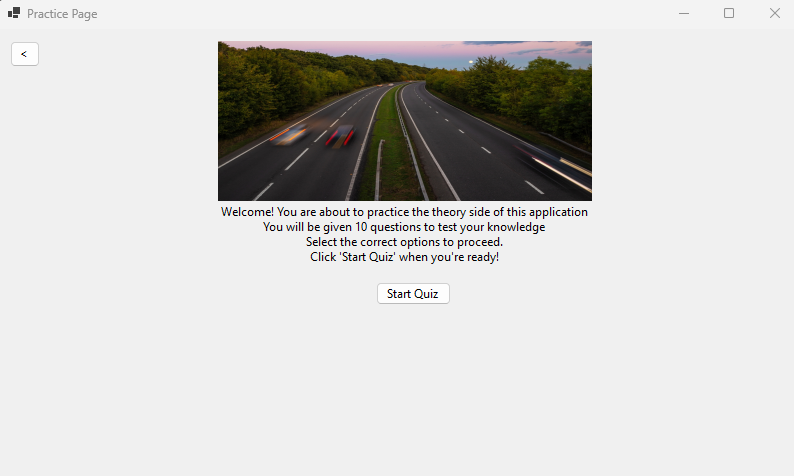
I use the Random class to generate pseudo-random numbers. The selection is done by random.Next(tests.Count), which generates a random integer between 0 (inclusive) and tests.Count (exclusive). This ensures that the index is valid within the bounds of the tests list, so a test is picked from the list.

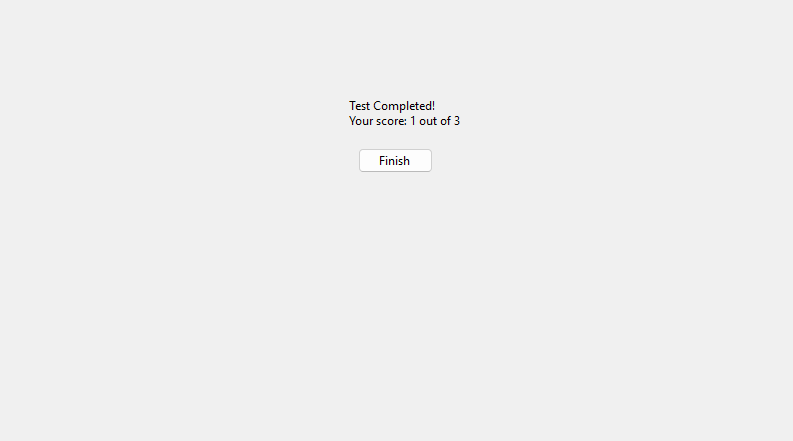
selectedTest = tests[random.Next(tests.Count)];

Dynamic Selection is ensured whenever the StartQuizButton\_Click method is executed (when the user starts the quiz), a new random test is selected.  
This ensures variety in the quiz experience every time it starts.

The Use of Choosing Random test is to ensure that the quiz feels fresh for the user by not always starting with the same test and to evenly distribute the selection across all available tests

**Obstacle**: When the test is End and the Quiz is started by the User it shows me the final result of the test rather than starting a new one



******

**Fix**: The issue arises because the currentQuestionIndex and score variables are not reset after completing the first test. When I click the "Start Test" button again, the previous values persist, causing the program to skip to the score display.

private void StartQuizButton\_Click(object? sender, EventArgs e)

{

// Reset test-specific variables

currentQuestionIndex = 0;

score = 0;

// Randomly select one test

Random random = new Random();

selectedTest = tests[random.Next(tests.Count)];

InitializeTimer();

LoadQuestion();

}

**Obstacle:** Facing the same test which the user has just attempted  
**Fix:** I will change the random selection logic to ensure I don't repeat the same test as the previous one. The previousTest variable is used to store the last selected test.

private void StartQuizButton\_Click(object? sender, EventArgs e)

{

// Randomly select a test that is not the same as the previous one

Random random = new Random();

Test? newTest;

do

{

newTest = tests[random.Next(tests.Count)];

} while (newTest == previousTest); // Repeat until a different test is selected

previousTest = newTest; // Update the previous test

//...(Below Logic Remains the same)

}

**Problem:** showing the user What test They are attempting at the moment.  
**Solution:** To display the name or index of the current test, adding a label that updates whenever a new test is selected

Modifying the StartQuizButton\_Click method to update the label when a test is selected

// Update the test name label

int testIndex = tests.IndexOf(selectedTest) + 1; // Test index starts from 1 testNameLabel.Text = $"Test {testIndex}";

Ensuring the Label is Visible During the Quiz by adding the testNameLabel to the form in the LoadQuestion method to ensure it remains visible when navigating through questions:

// Display the current question

Question currentQuestion = selectedTest.Questions[currentQuestionIndex]; Label questionLabel = new Label { Text = currentQuestion.Text, AutoSize = true }; questionLabel.Location = new Point((ClientSize.Width - questionLabel.Width) / 2, 50); Controls.Add(questionLabel);

**Iteration 4:**

I am changing the logic of random tests to user choice, as this is a practice mode and that logic is more suitable for the mock test page. I am also replacing the StartQuiz class with StartTest.

private void StartTest(int testIndex)

{

selectedTest = tests[testIndex];

currentQuestionIndex = 0;

score = 0;

// Update the test name label

testNameLabel.Text = $"Test {testIndex + 1}";

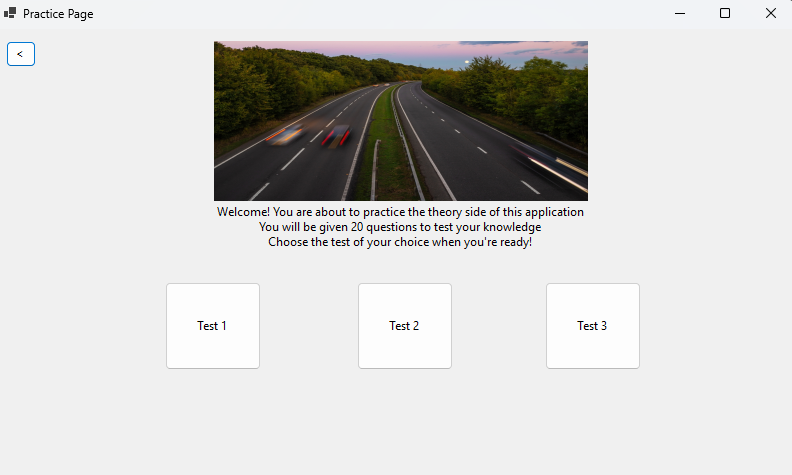
LoadQuestion();

}

// Update the test name label

testNameLabel.Text = $"Test {testIndex + 1}";

LoadQuestion();

******

***Problem:*** What if I want to quit the practice? A Quit button will help me quit at any given time.

***Solution:*** I will add a Quit button, which, when clicked, takes me back to the initial screen.

private void Quit\_Click(object sender, EventArgs e)

{

// Reset test-related variables

selectedTest = null;

previousTest = null;

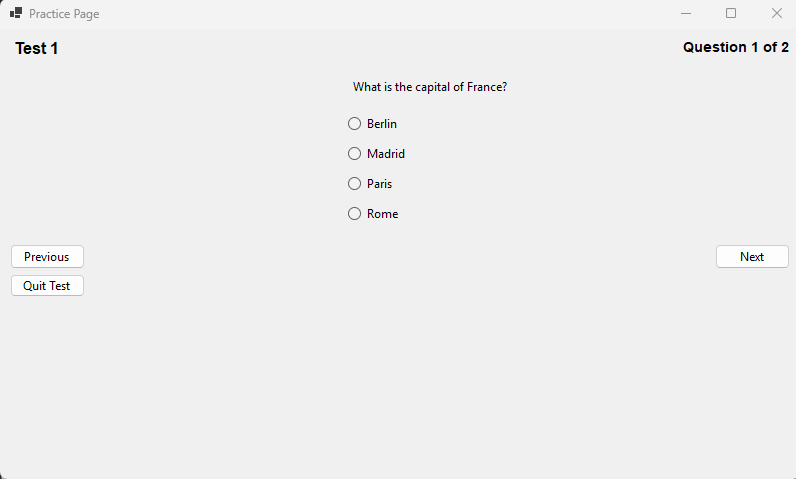
currentQuestionIndex = 0;

score = 0;

// Return to the introduction screen

ShowIntroduction();

}

****Problem:** What if the user wants to see the answer to the question right there?  
**Solution:** I will add a "Show Answer" button, which, when clicked, highlights the correct answer in green.

private void showAnswerButton\_Click(object sender, EventArgs e)

{

if (selectedTest == null || currentQuestionIndex >= selectedTest.Questions.Count)

{

MessageBox.Show("No question loaded to show the answer.");

return;

}

// Find the correct answer for the current question

Question currentQuestion = selectedTest.Questions[currentQuestionIndex];

int correctOptionIndex = currentQuestion.CorrectOptionIndex;

// Iterate through controls to find the RadioButton with the correct answer

foreach (Control control in Controls)

{

if (control is RadioButton radioButton && (int)radioButton.Tag == correctOptionIndex)

{

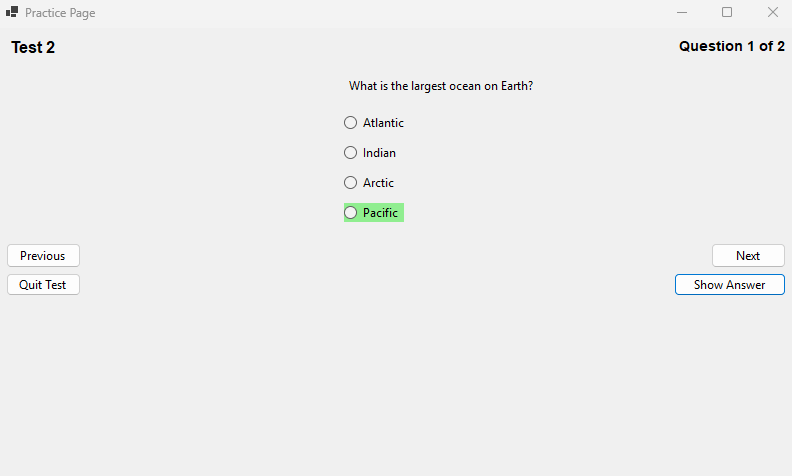
radioButton.BackColor = Color.LightGreen; // Highlight the correct answer

break;

}

}

}

******

**Iteration 5:**

Problem: How to add pictorial Questions?  
Solution: I will introduce a picture box to display the picture and set a proper location so that it doesn’t overlap with any button.

// Initialize PictureBox for question images

questionPictureBox = new PictureBox

{

SizeMode = PictureBoxSizeMode.Zoom,

Size = new Size(300, 200), // Default size

Visible = false, // Initially hidden; only shown if there's an image

Location = new Point((ClientSize.Width - 300) / 2, 200) // Centered

};

Edit load questions function to display and show the picture if it exists

// Display image if available

if (currentQuestion.Image != null)

{

questionPictureBox.Image = currentQuestion.Image;

questionPictureBox.Visible = true;

}

else

{

questionPictureBox.Visible = false;

}

Make the image property global and give proper path of the image as well

public Image? Image { get; set; } // New property for image

Image =Image.FromFile("E:\\CLIENT\\WindowsFormsApp1\\STOP\_SIGN\_PIC.jpg") // Add a valid image file path

******

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

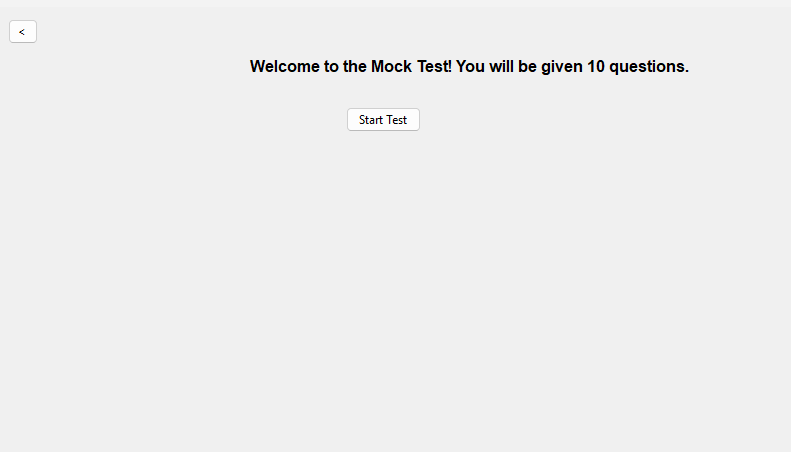
**Prototype 3 - Mock Test Page:**

***Overview:***

The Mock Test Page simulates a time-limited quiz, with a 57-minute countdown timer. Users can flag difficult questions for later review, and navigate through the test even without answering flagged questions. The test tracks answers, highlighting incorrect ones in red and correct ones in green in the results. A checkbox must be selected to confirm the user has read the instructions before starting the test. The page shows the time remaining and progress, and detailed feedback is provided on incorrect answers after the test ends.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**



Everything is the same as the **Practice Page** except there is a timer now.

**Problem**: How to Implement a CountDown?

**Solution:** I will implement the countdown timer by initializing a Timer in the StartQuizButton\_Click method.

private void InitializeTimer()

{

if (quizTimer == null)

{

quizTimer = new Timer();

quizTimer.Interval = 1000; // 1-second intervals

quizTimer.Tick += QuizTimer\_Tick; // Event handler when timer ticks

}

timeRemaining = 30; // Set the initial countdown time to 30 seconds

quizTimer.Start(); // Start the timer

timerLabel = new Label

{

Text = $"Time Left: {timeRemaining} seconds",

AutoSize = true,

Font = new Font("Arial", 10, FontStyle.Bold),

ForeColor = Color.Red,

Location = new Point(10, 10)

};

Controls.Add(timerLabel); // Add the timer label to the form

}

**Iteration 2:**

**Problem:** How will the timer would be Updated? This timer should Stop the Test when the timer runs out:

**Solution:** I will create the QuizTimer\_Tick method, which is triggered every second. It will update the timeRemaining value and display the remaining time on the timerLabel. If the timer reaches zero, the test will stop, and the score will be shown.

private void QuizTimer\_Tick(object? sender, EventArgs e)

{

timeRemaining--; // Decrease the time remaining by 1 second

if (timeRemaining <= 0)

{

quizTimer.Stop(); // Stop the timer when it reaches 0

ShowScore(); // Show the score when the time runs out

}

else

{

timerLabel.Text = $"Time Left: {timeRemaining} seconds"; // Update the label

}

}

The timer starts when the test begins with a countdown of 30 seconds. The quizTimer\_Tick event handler decreases the remaining time every second and updates the UI. Once the countdown reaches zero, the test is stopped by calling quizTimer.Stop(), and the ShowScore method is invoked to display the final score.

**Obstacle:** Severity Code Description Project File Line Suppression State Details Error (active) CS0104 'Timer' is an ambiguous reference between 'System.Windows.Forms.Timer' and 'System.Threading.Timer' WinFormsApp1 **Fix:** The error occurs because both System.Windows.Forms.Timer and System.Threading.Timer are referenced, and the compiler doesn't know which one to use. To resolve this ambiguity, I need to explicitly qualify the Timer class with its namespace. For example, if I'm working with a Windows Forms application, I will use:

System.Windows.Forms.Timer quizTimer = new System.Windows.Forms.Timer();

This will ensure that the correct Timer class is used in the context of my Windows Forms application.

private System.Windows.Forms.Timer quizTimer; // Timer object

In the class of IntializeTimer we need to change the qualification

private void InitializeTimer()

{

if (quizTimer == null)

{

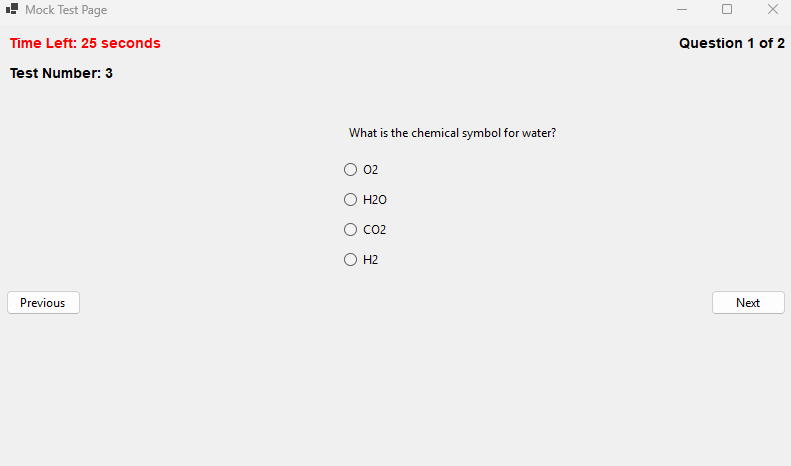
quizTimer = new System.Windows.Forms.Timer(); // Explicit namespace usage

quizTimer.Interval = 1000; // 1 second intervals

quizTimer.Tick += QuizTimer\_Tick;

}

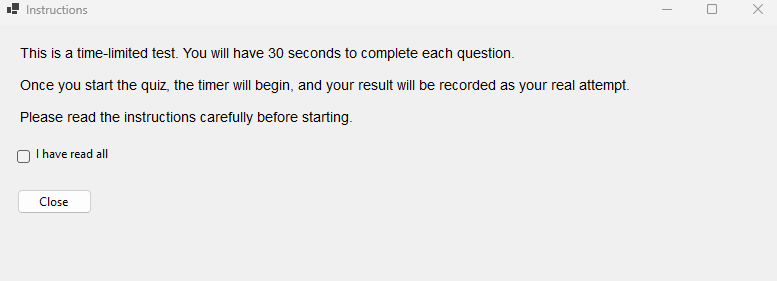
}



**Iteration 3:**

**Problem:** I need to make sure that I know this is a time-limited test and that I can't start until I have read the instructions carefully. To achieve this, I will clearly display a message stating that the test is time-limited, and ensure the "Start Test" button is only enabled once I check the "I have read all of the instructions" checkbox.

**Solution**: A new form is opened when the "Mock Test" button is clicked on the main page, containing the instructions that I must read before I can start the quiz. The Mock Test page form won't open until I have checked the "I have read all of the instructions" checkbox.



**Iteration 4:**  
Increasing the Timer from 30 sec to 57 Minutes by changing the Initialize Timer Code

private void InitializeTimer()

{

if (quizTimer == null)

{

quizTimer = new System.Windows.Forms.Timer();

quizTimer.Interval = 1000; // 1-second intervals

quizTimer.Tick += QuizTimer\_Tick;

}

timeRemaining = 57 \* 60; // 57 minutes in seconds

quizTimer.Start();

...//remains the same

}

Added new function to change the format

private string FormatTime(int seconds)

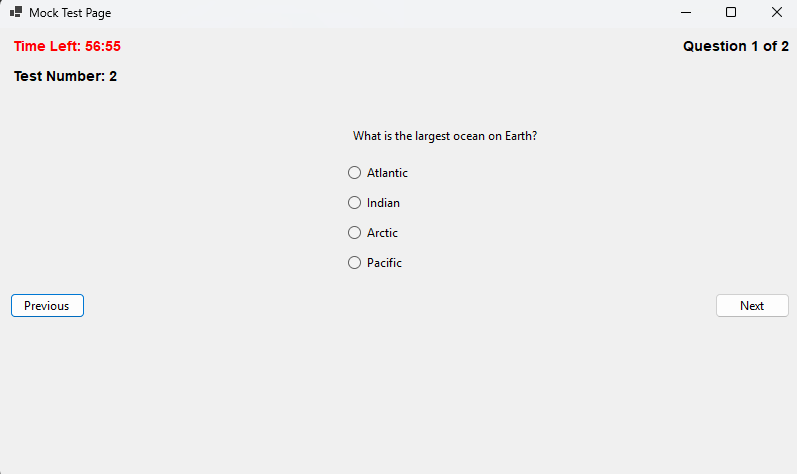
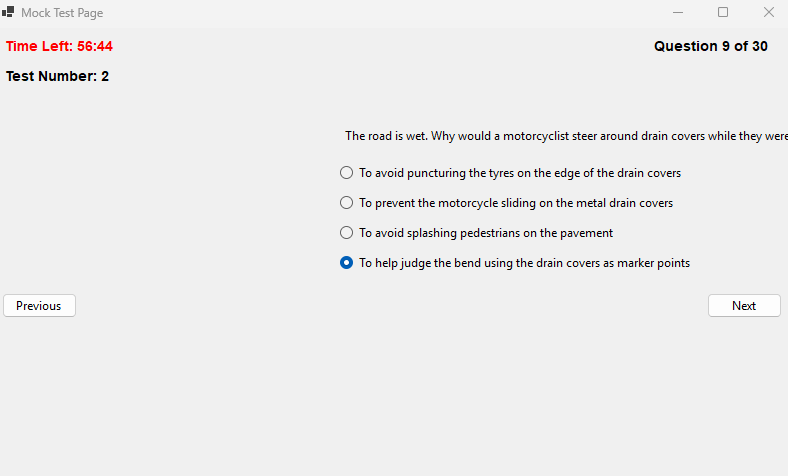
{

int minutes = seconds / 60;

int remainingSeconds = seconds % 60;

return $"{minutes:D2}:{remainingSeconds:D2}";

}

******Adding real traffic questions by putting 30 Questions in 3 mock test each  
  


**Iteration 5:**

**Problem:** If a user wants to mark a question they dont understand how would they?  
**Solution**: I will introduce a "Flag" button that allows me to flag questions I want to know the answer to. These flagged questions will then be displayed on the progress page. I can flag and unflag questions as needed. Additionally, I will introduce a list to save the selected flagged questions for easy tracking.

private List<int> flaggedQuestions = new List<int>();

Making a new Class for it.

private void Flag\_Click(object? sender, EventArgs e)

{

// Toggle flag for the current question

if (flaggedQuestions.Contains(currentQuestionIndex))

{

flaggedQuestions.Remove(currentQuestionIndex);

flagButton.Text = "Flag";

}

else

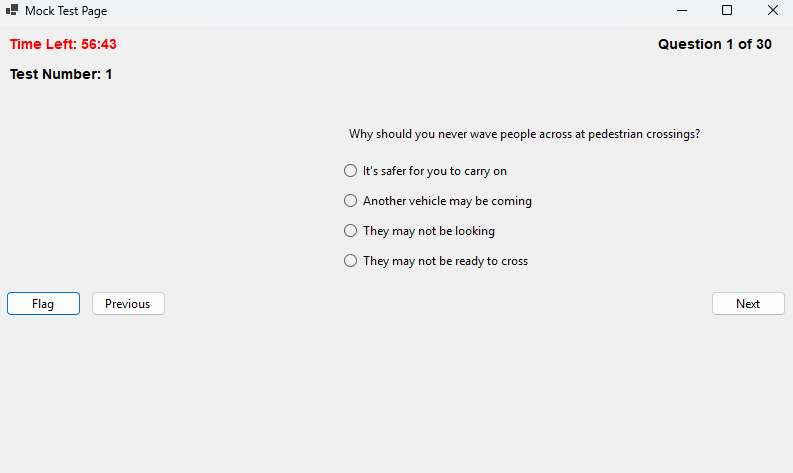
{

flaggedQuestions.Add(currentQuestionIndex);

flagButton.Text = "Unflag";

}

}

****

Made Questions Global as well so it could be accessed by the Progress page(Flagged questions Form)

public static List<List<Question>> AllTests { get; set; } = new List<List<Question>>

{

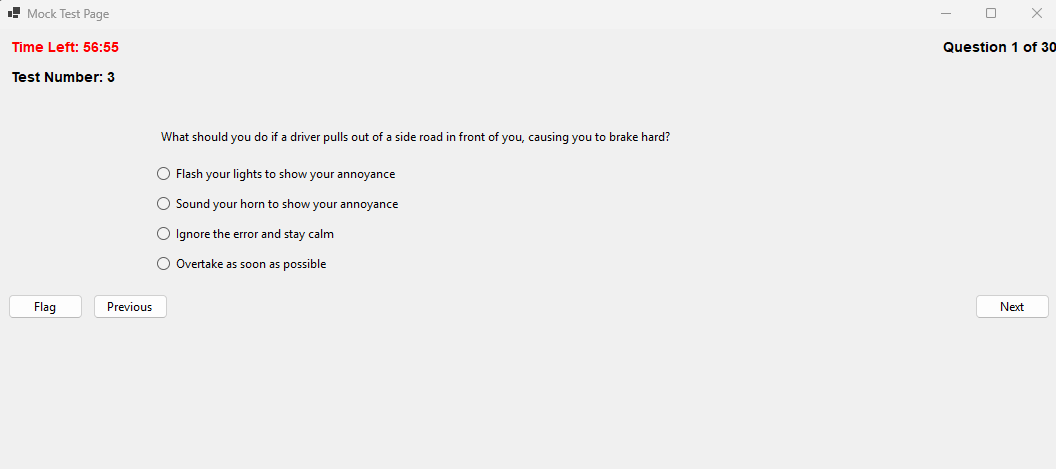
new List<Question>

{

new Question{Text = "Why should you never wave people across at pedestrian crossings?",Options = new List<string>{"It's safer for you to carry on","Another vehicle may be coming","They may not be looking","They may not be ready to cross"},CorrectOptionIndex = 1},

}

}

****

**Problem:** when the user is flagging the question they are still forced to select an option to move ahead  
**Solution:** Change the Next Click Class to include an if which checks if the question is flagged or not

private void Next\_Click(object? sender, EventArgs e)

{

// Check if the current question is flagged

if (flaggedQuestions.Contains(currentQuestionIndex))

{

// Proceed to the next question without requiring an answer

currentQuestionIndex++;

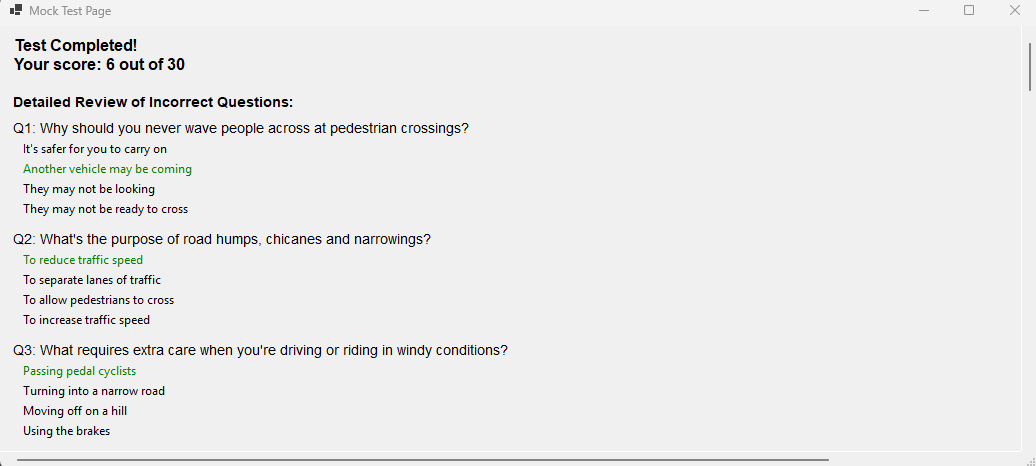
LoadQuestion();

return;

}

}

The Show screen also needs to show what questions the user has gotten right or wrong

 **Problem:** Showing all the questions without showing user choice

**Solution:** Make and implement a Dictionary which stores the users choice

private Dictionary<int, int> userAnswers = new Dictionary<int, int>();

implement it in next click class so the users choice can be stored

// Store the user's answer

userAnswers[currentQuestionIndex] = selectedOption;

I will change the ShowScore class to incorporate all of the new changes by creating a loop that goes through all of the incorrect questions answered by me. In this loop, red will represent the user’s choice that was wrong, while green will represent the correct answer. This will provide a clear visual representation of the mistakes made and the correct answers.

// Display detailed results for incorrect answers

Label reviewLabel = new Label

{

Text = "Review of Incorrect Questions:",

AutoSize = true,

Font = new Font("Arial", 10, FontStyle.Bold),

Location = new Point(10, yPosition)

};

scrollablePanel.Controls.Add(reviewLabel);

yPosition = reviewLabel.Bottom + 10;

// Loop through the questions and display only incorrect answers

for (int i = 0; i < selectedTest.Count; i++)

{

Question question = selectedTest[i];

int userAnswer = -1; // Default value for no answer

bool isCorrect = false;

// Determine user's selected answer

if (userAnswers.ContainsKey(i))

{

userAnswer = userAnswers[i];

isCorrect = userAnswer == question.CorrectOptionIndex;

}

// Skip correctly answered questions

if (isCorrect)

continue;

// Display the question

Label questionLabel = new Label

{

Text = $"Q{i + 1}: {question.Text}",

AutoSize = true,

Font = new Font("Arial", 10),

Location = new Point(10, yPosition)

};

scrollablePanel.Controls.Add(questionLabel);

yPosition = questionLabel.Bottom + 5;

// Display the options

for (int j = 0; j < question.Options.Count; j++)

{

Label optionLabel = new Label

{

Text = question.Options[j],

AutoSize = true,

Location = new Point(20, yPosition)

};

// Highlight the user's incorrect answer in red

if (j == userAnswer)

{

optionLabel.ForeColor = Color.Red;

}

// Highlight the correct answer in green

else if (j == question.CorrectOptionIndex)

{

optionLabel.ForeColor = Color.Green;

}

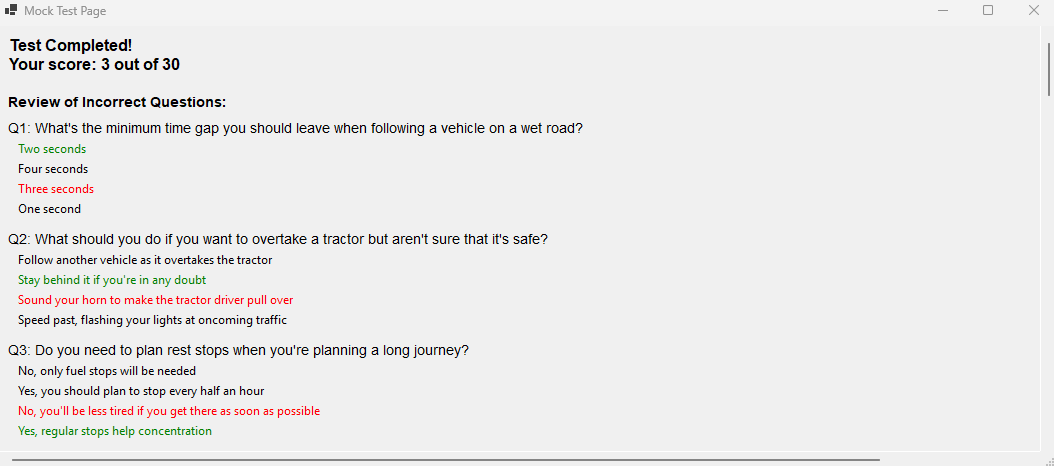
scrollablePanel.Controls.Add(optionLabel);

yPosition = optionLabel.Bottom + 5;

}

yPosition += 10; // Add spacing between questions

}

****

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 4 - Progress Page:**

***Overview:***

The project tracks and displays user progress in mock and practice tests using dynamic progress bars. It calculates progress based on the number of correct answers and the total questions per test, with scores stored globally. The interface also allows for flagging questions and displays a timer to track completed topics, updating the user's progress continuously. This system enhances user engagement by providing real-time feedback on test performance and completed content.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**

**Problem:** How can I track the progress of the user, including how many tests they have taken and their scores on those tests (graphically)? How will the score percentage be calculated? **Solution:** To implement a graphical progress bar for the progress page, I can use the ProgressBar control in Windows Forms and a Label to display the score percentage. I can create a dynamic progress bar for each test, where the value of the progress bar reflects the user's score as a percentage. I will calculate the score percentage based on the number of correct answers (e.g., 10 questions per test, or adjust this to the actual number of questions in the test).

Next to each progress bar, a Score Display label will show the score percentage or indicate if the test has not been attempted. When navigating to the Progress\_Page, I will pass a dictionary containing the scores for each test to ensure all progress is displayed correctly.

private void InitializeProgressBar()

{

Controls.Clear();

Label titleLabel = new Label

{

Text = "Progress Overview",

Font = new Font("Arial", 14, FontStyle.Bold),

AutoSize = true,

Location = new Point((ClientSize.Width - 200) / 2, 20)

};

Controls.Add(titleLabel);

int yPosition = titleLabel.Bottom + 30;

// Display progress bar

ProgressBar progressBar = new ProgressBar

{

Minimum = 0,

Maximum = 100,

Value = testScores.ContainsKey(testNumber)

? (testScores[testNumber] \* 100) / 10

: 0, // Assuming each test has 10 questions

Size = new Size(200, 20),

Location = new Point(100, yPosition)

};

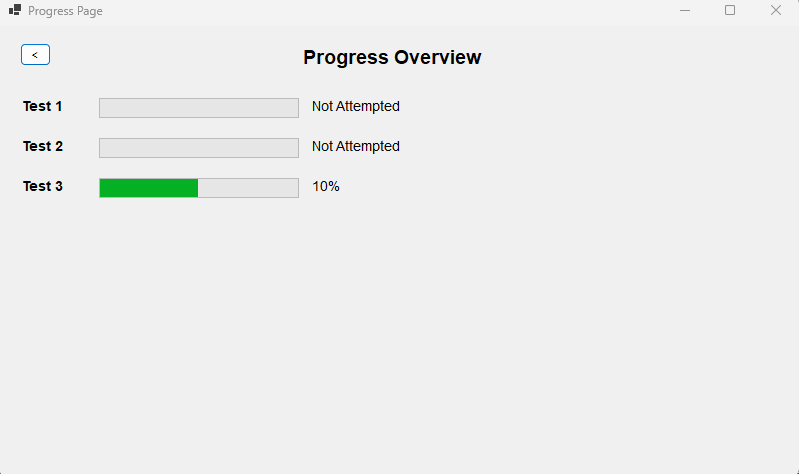
Controls.Add(progressBar);

}

**Problem:** How to Integrate it with the Mock test page?  
**Solution:** To integrate with my MockTest\_Page, updating the ShowScore method so it can save the score in a global object. Then, pass the scores to the Progress\_Page.

private Dictionary<int, int> testScores = new Dictionary<int, int>(); // Global dictionary to track scores

// Save the score for the current test if (!testScores.ContainsKey(testNumber)) testScores[testNumber] = score;



**Iteration 2:**

To Dynamically calculate the progress bar based on the number of questions in the test we do the following:

Dictionary<int, int> testQuestions = new Dictionary<int, int>

{

{ 1, 10 }, // Test 1 has 10 questions

{ 2, 8 }, // Test 2 has 8 questions

{ 3, 12 } // Test 3 has 12 questions

};

// Display progress bar

ProgressBar progressBar = new ProgressBar

{

Minimum = 0,

Maximum = 100,

Value = testScores.ContainsKey(testNumber) && testQuestions.ContainsKey(testNumber)

? (testScores[testNumber] \* 100) / testQuestions[testNumber]

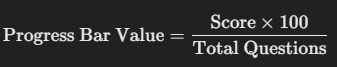
: 0, // Default to 0 if the test number is not found

Size = new Size(200, 20),

Location = new Point(100, yPosition)

};

If the test score for the current test (testNumber) exists in testScores and the total number of questions for that test exists in testQuestions, calculate the percentage:



Otherwise, default to 0. This logic ensures the progress bar adjusts according to the actual number of questions in the test rather than a fixed assumption of 10.  
  
**Problem:** The text next to the progress bar is 10%/20% it is being calculated by Multiplying the score by 10,Assuming that the number of question is 10  
the percentage should be calculated with the dynamic question logic now

Text = testScores.ContainsKey(testNumber)

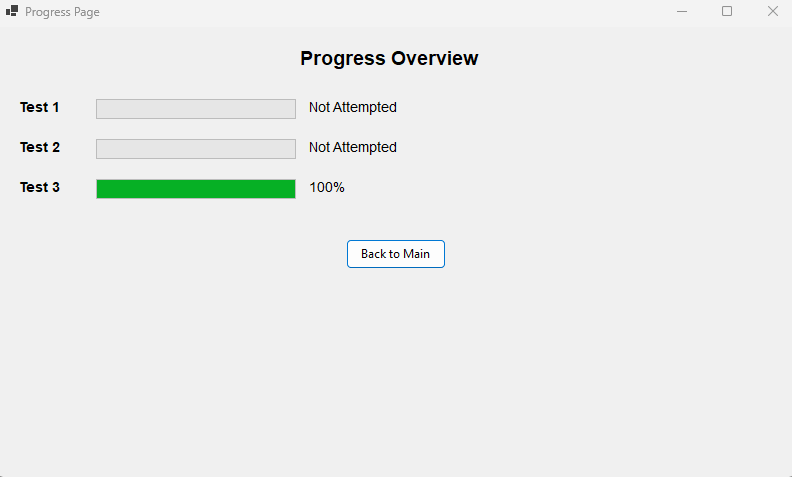
? $"{testScores[testNumber] \* 10}%"

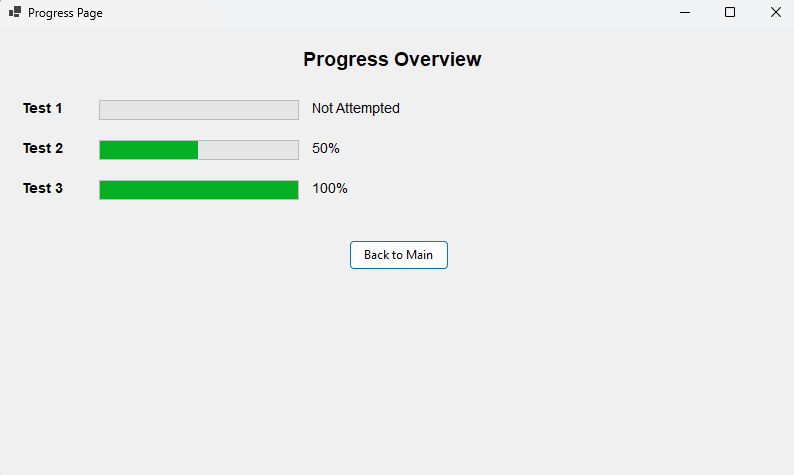
: "Not Attempted",

**Solution:** I can use a Label that dynamically calculates and displays the percentage based on the testScores and testQuestions.   
Calculate Percentage percentage is dynamically calculated now using the formula

****

If the score or total number of questions is missing, it defaults to 0%.  
The progress bar value is now set using the percentage.  
A Label is added to display the percentage next to the progress bar.  
The logic works for any number of questions and any score.





**Iteration 3:**

It only shows the progress of the mock test right now I want to change so that so it also shows the progress of practice page. I have to add A variable to score practice score globally

public static *class* GlobalData

{

public static Dictionary<int, int> TestScores { *get*; *set*; } = new Dictionary<int, int>();

public static Dictionary<int, int> PracticeScores { *get*; *set*; } = new Dictionary<int, int>(); // Practice test scores

}

I then have to store the scores as well

private void ShowScore()

{

// Clear the form

Controls.Clear();

// Save the score globally with the test index

if (selectedTest != null)

{

int testIndex = tests.IndexOf(selectedTest) + 1; // Test index starts from 1

GlobalData.PracticeScores[testIndex] = score; // Save the score for this test

}

}

I also changed the label to better identify the tests and show progress under

// Add mock test progress

Label mockProgressLabel = new Label

{

Text = "Mock Test Progress",

..//remains the same

};

foreach (*var* testNumber in testQuestions.Keys)

{

AddProgressRow("Test", testNumber, mockTestScores, ref yPosition);

}

// Add practice test progress

Label practiceProgressLabel = new Label

{

Text = "Practice Test Progress",

..//remains the same

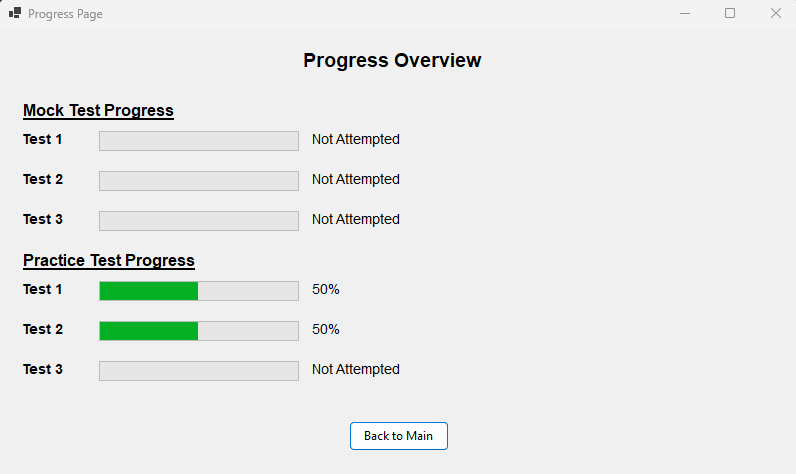
};

foreach (*var* testNumber in testQuestions.Keys)

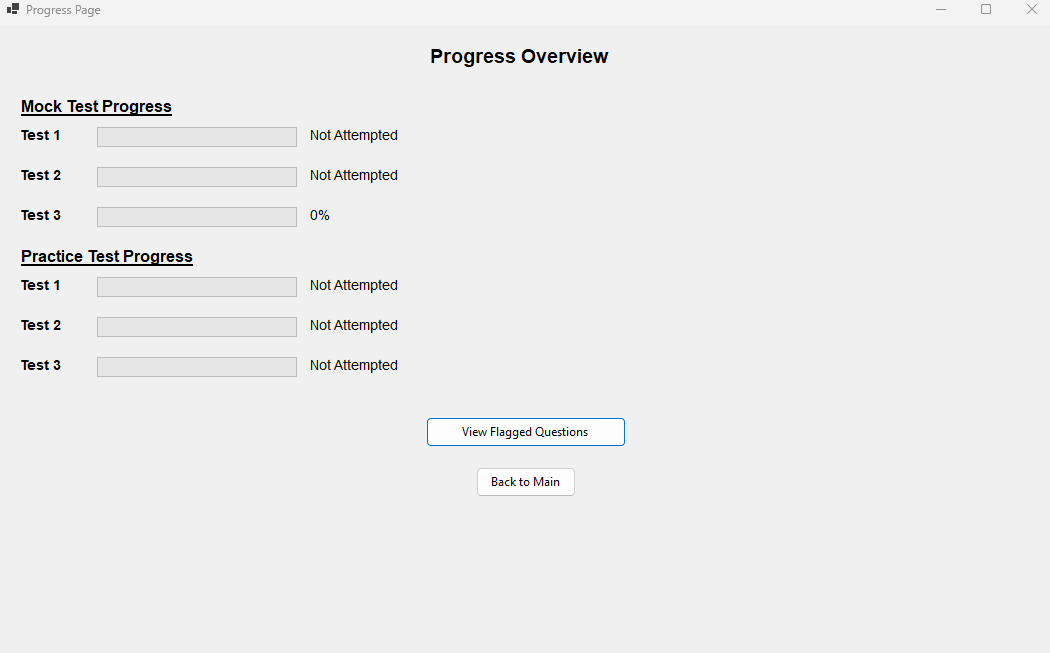
{

AddProgressRow("Test", testNumber, practiceTestScores, ref yPosition);

}

****Iteration 4:**

I will show the Flagged Questions here under a Flag Question Button, This Button would open a new form on which every flagged question alongside its answer would be there.

******

**Iteration 5:**

**Problem:** After adding the checkbox for completed topics, how can I track what the user has completed? And if they go back and complete it later, how will that be updated?  
**Solution**: I will introduce a timer that checks what the user has selected to be completed. This timer will run every second. If the user hasn't completed any topics, it will display "None."  
add a timer and the topics completed

topicsCompletedLabel = new Label

{

Text = "Topics Completed: None",

Font = new Font("Arial", 12, FontStyle.Bold),

AutoSize = true,

Location = new Point(20, yPosition )

};

Controls.Add(topicsCompletedLabel);

System.Windows.Forms.Timer updateTimer = new System.Windows.Forms.Timer

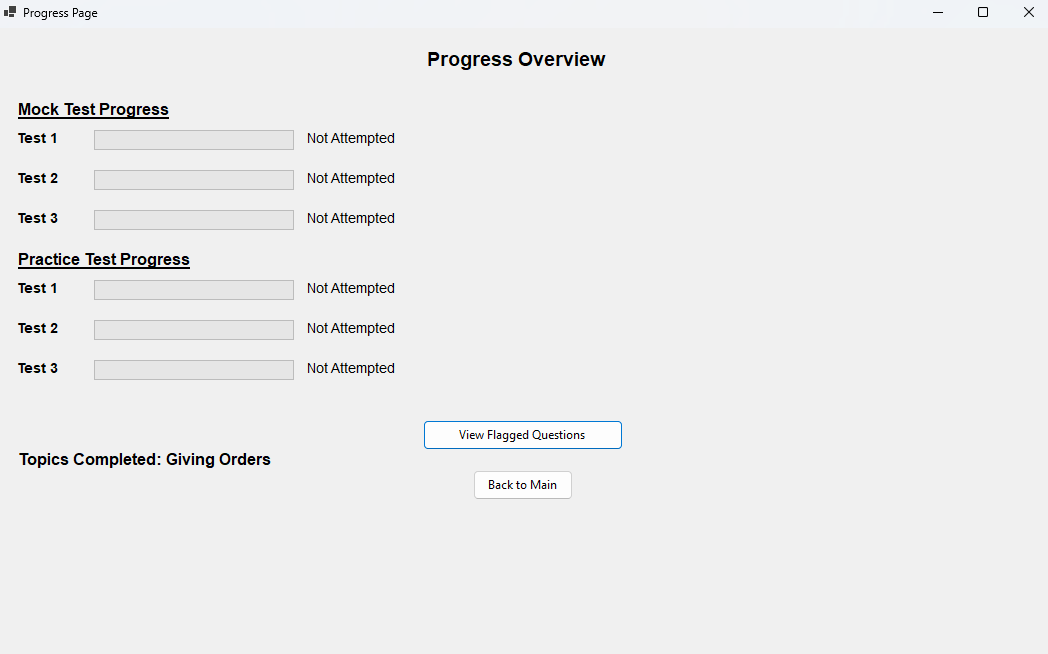
{

Interval = 1000 // Check for updates every second

};

updateTimer.Tick += (s, e) => UpdateTopicsCompleted();

updateTimer.Start();

***  
  
  
  
Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 5 - Flagged Questions Page:**

***Overview:***

In Iteration 1, the goal is to display all flagged questions from a test on a separate page. This involves iterating through a collection of flagged questions, retrieving the question text and its correct answer, and dynamically displaying them in a scrollable panel for easy viewing. Each flagged question is represented by a label showing its text and the correct answer, and the layout adjusts the position of each question as it is added to the panel.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**  
For getting all the flagged questions and showing them on one page

foreach (int questionIndex in test.Value)

{

Question question = GlobalData.AllTests[test.Key - 1][questionIndex]; // Get the flagged question

Label questionLabel = new Label

{

Text = $"Q: {question.Text}\nA: {question.Options[question.CorrectOptionIndex]}",

Font = new Font("Arial", 10, FontStyle.Regular),

AutoSize = true,

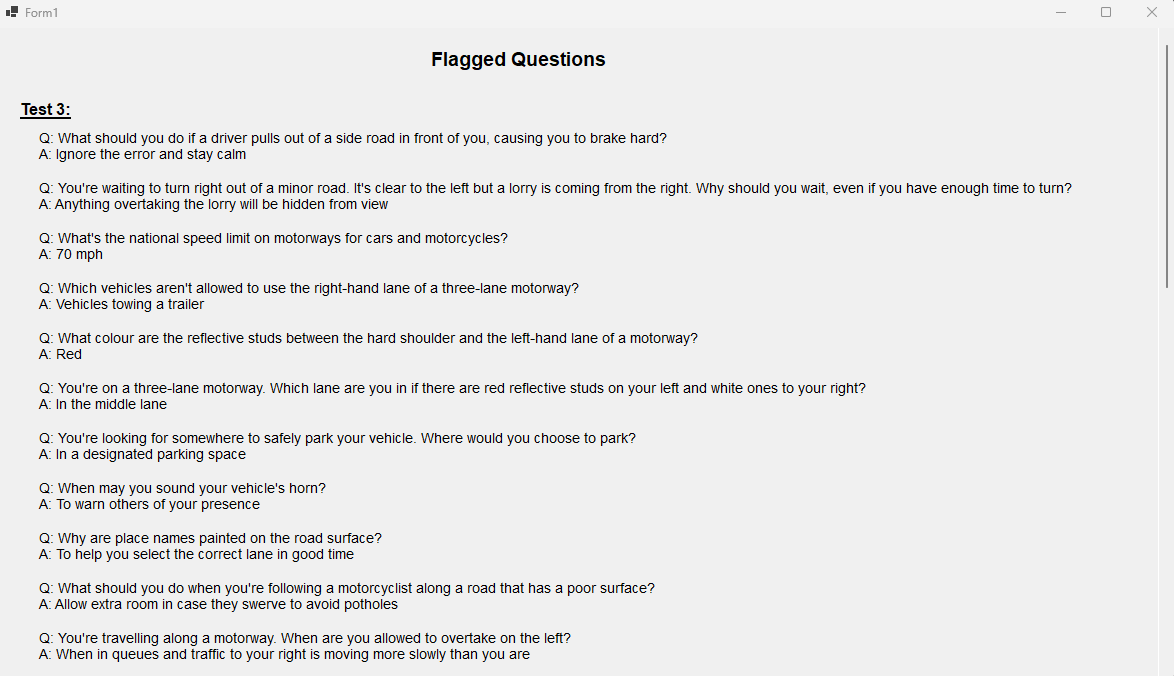
Location = new Point(40, yPosition)

};

scrollablePanel.Controls.Add(questionLabel);

yPosition += 50;

}



***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 6 - Traffic Signs Page:**

***Overview:***

This prototype aims to enhance a user's learning experience by tracking progress on traffic signs topics and displaying relevant sign images with descriptions. It includes two main features:

1. **Progress Tracking:**Users can mark topics as completed via checkboxes. Their progress is stored in a persistent dictionary, ensuring that their selections are remembered even when they navigate away from the page. The system dynamically updates the checkbox states to reflect the user's progress.
2. **Sign Study Interface:**For each traffic signs topic, a dedicated page displays related signs with images and descriptions. A DataGridView is utilized to show these signs in a scrollable format, with an image column and a text column for additional information. This interface allows users to visually study and learn the traffic signs.

The prototype efficiently tracks user progress and provides an interactive study interface for traffic signs, helping users easily learn and monitor their advancement.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:  
Problem:** How to know what topics are completed by the user?  
**Solution:** I will add a checkbox under every topic so that when the user completes a topic, they can check it. This allows the user to track their progress and mark topics as completed.

public static *List*<string> CompletedTopics { get; private set; } = new *List*<*string*>();

public Traffic\_Signs\_page()

{

InitializeComponent();

}

private *void* Giving\_Order\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Giving Orders", Giving\_Order\_Complete.Checked);

}

private *void* Warning\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Warning Signs", Warning\_Signs\_Complete.Checked);

}

private *void* Direction\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Direction Signs", Direction\_Signs\_Complete.Checked);

}

private *void* Information\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Information Signs", Information\_Signs\_Complete.Checked);

}

private *void* Road\_Work\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Road Work Signs", Road\_Work\_Complete.Checked);

}

private *void* UpdateCompletedTopics(string topic, bool isCompleted)

{

if (isCompleted)

{

if (!CompletedTopics.Contains(topic))

{

CompletedTopics.Add(topic);

}

}

else

{

CompletedTopics.Remove(topic);

}

}

**Obstacle:** when the user goes back to the menu and open traffic page again the checkbox becomes empty  
**Solution:** The issue arises because the state of the checkboxes (whether they are checked or not) is not being preserved when navigating away from the Traffic\_Signs\_page and reopening it. This happens because each time the form is opened, a new instance of Traffic\_Signs\_page is created, and the checkboxes are reset to their default state (unchecked).

To fix this, I will store the state of the checkboxes (whether they are checked or unchecked) in a persistent location, such as a global variable, a file, or a database. When the Traffic\_Signs\_page is reopened, I can retrieve the saved state and update the checkboxes accordingly.

public static *List*<string> CompletedTopics { get; private set; } = new *List*<*string*>();

public Traffic\_Signs\_page()

{

InitializeComponent();

}

private *void* Giving\_Order\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Giving Orders", Giving\_Order\_Complete.Checked);

}

private *void* Warning\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Warning Signs", Warning\_Signs\_Complete.Checked);

}

private *void* Direction\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Direction Signs", Direction\_Signs\_Complete.Checked);

}

private *void* Information\_Signs\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Information Signs", Information\_Signs\_Complete.Checked);

}

private *void* Road\_Work\_Complete\_CheckedChanged(object sender, *EventArgs* e)

{

UpdateCompletedTopics("Road Work Signs", Road\_Work\_Complete.Checked);

}

private *void* UpdateCompletedTopics(string topic, bool isCompleted)

{

if (isCompleted)

{

if (!CompletedTopics.Contains(topic))

{

CompletedTopics.Add(topic);

}

}

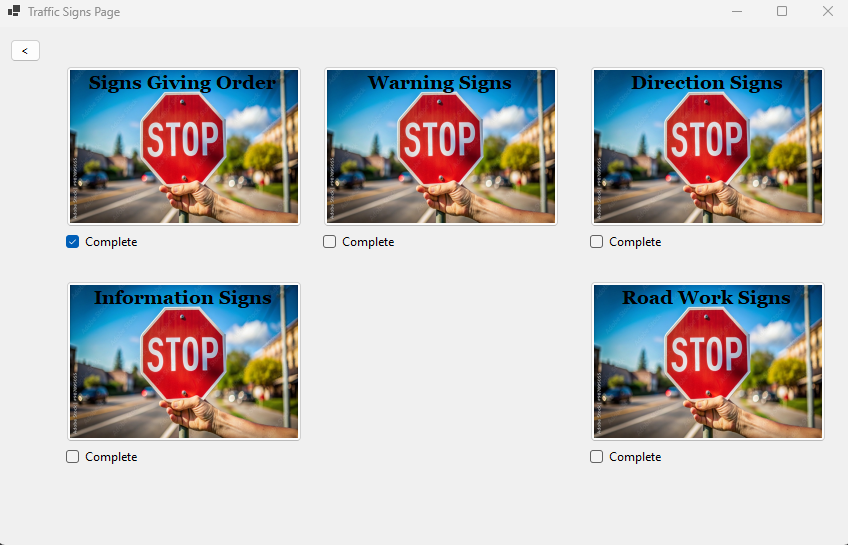
else

{

CompletedTopics.Remove(topic);

}

}

******

**Iteration 2:**

**Problem:** How to put pictures with text so the user can study signs from?  
**Solution:** I will add a new page under each topic so that the user can see what signs belong to that particular topic. When the user clicks on a topic, they will be directed to a page that displays all the relevant signs under that topic.

private void Orders\_Signs\_Click(object sender, EventArgs e)

{

Orders\_Signs nextForm = new Orders\_Signs();

nextForm.Show();

this.Hide();

}

**Problem:** How to put pictures with text?  
**Solution:** Implementation of a datagrid with scroll wheel

private void InitializeGrid()

{

// Create and configure DataGridView

DataGridView signsGridView = new DataGridView

{

Dock = DockStyle.Fill,

AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,

RowTemplate = { Height = 100 }, // Adjust row height to fit images

AllowUserToAddRows = false,

ReadOnly = true

};

// Add columns

DataGridViewImageColumn imageColumn = new DataGridViewImageColumn

{

HeaderText = "Sign Image",

Name = "ImageColumn",

ImageLayout = DataGridViewImageCellLayout.Zoom // Adjust image display

};

signsGridView.Columns.Add(imageColumn);

DataGridViewTextBoxColumn infoColumn = new DataGridViewTextBoxColumn

{

HeaderText = "Information",

Name = "InfoColumn"

};

signsGridView.Columns.Add(infoColumn);

// Add data

AddSignRow(signsGridView, "Signs with red circles are mostly prohibitive.\r\nPlates below signs qualify their message.", "Blank.png");

AddSignRow(signsGridView, "Entry to 20 mph zone", "Entry\_to\_20\_mph\_zone.png");

AddSignRow(signsGridView, "End of 20 mph zone", "End\_of\_20\_mph\_zone.png");

if (signsGridView.Rows.Count > 1) // Ensure the row exists

{

signsGridView.Rows[0].DefaultCellStyle.Font = new Font("Arial", 10, FontStyle.Bold);

}

// Add DataGridView to form

Controls.Add(signsGridView);

}

private void AddSignRow(DataGridView grid, string info, string imagePath)

{

// Load image

string appDirectory = AppDomain.CurrentDomain.BaseDirectory;

string imageFullPath = System.IO.Path.Combine(appDirectory, "Signs\_Giving\_Order", imagePath);

if (!System.IO.File.Exists(imageFullPath))

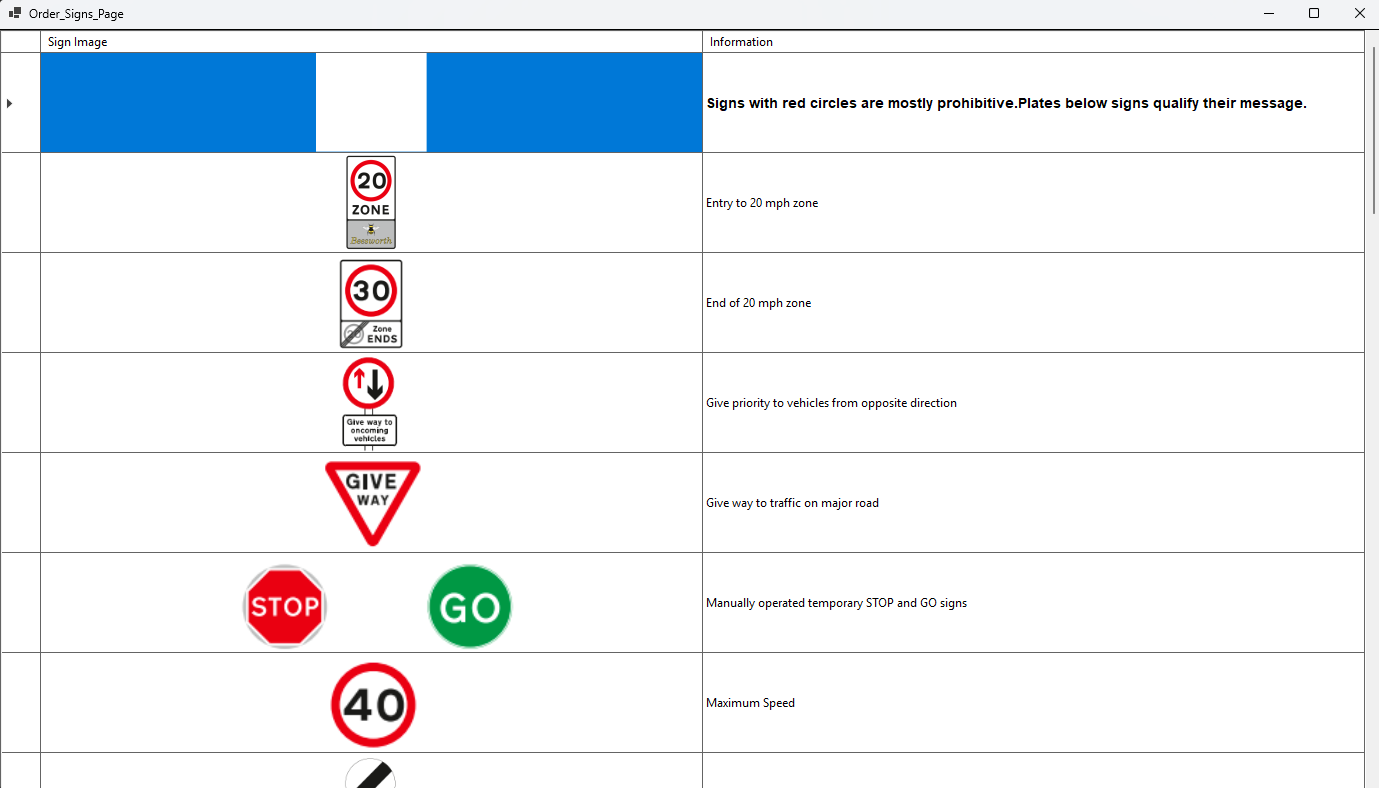
throw new System.IO.FileNotFoundException($"Image file not found: {imageFullPath}");

Image signImage = Image.FromFile(imageFullPath);

// Add row

grid.Rows.Add(signImage, info);

}

******

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**New Update – Permanent Progress Storage:**  
This update implements a persistent state-saving mechanism that allows users to track their progress, scores, and flagged questions. The progress is stored in local text files, and this information is loaded when required, such as when the Progress Page is opened. This approach enables continuity in user experience, even after closing the application.

To save the history of user for previous test scores, completed topics and flagged questions to a .txt file which is then read by the progress page.

***For Practice Page:***

private *void* *ShowScore*()

{

if (selectedTest != null)

{

*int* testIndex = tests.*IndexOf*(selectedTest) + 1;

*GlobalData*.*PracticeScores*[testIndex] = score;

}

string filePath = "PracticeScores.txt";

*List*<string> fileLines = *File*.*Exists*(filePath) ? new *List*<string>(*File*.*ReadAllLines*(filePath)) : new *List*<string>();

string testIdentifier = $"Test: {testNameLabel.Text}";

string updatedScoreText = $"{testIdentifier}, Score: {score}/{selectedTest?.Questions.Count}";

*int* existingIndex = fileLines.*FindIndex*(line => line.*StartsWith*(testIdentifier));

if (existingIndex >= 0) fileLines[existingIndex] = updatedScoreText;

else fileLines.*Add*(updatedScoreText);

*File*.*WriteAllLines*(filePath, fileLines);

}

A screenshot of a computer

Description automatically generated

***For Traffic Signs Page:***

private void SaveData()

{

try

{

*var* uniqueTopics = new HashSet<string>(CompletedTopics.Select(topic => topic.Trim()));

using (StreamWriter writer = new StreamWriter("data.txt"))

{

writer.WriteLine(string.Join(",", uniqueTopics));

foreach (*var* entry in CheckboxStates) writer.WriteLine($"{entry.Key}:{entry.Value}");

}

}

catch (Exception ex) { Console.WriteLine("Error saving data: " + ex.Message); }

}

else File.WriteAllText(filePath, scoreLine);

A screenshot of a computer

Description automatically generated

**For Mock Test Page (*flagged questions):***

private void Flag\_Click(object? sender, EventArgs e)

{

string filePath = "Flagged\_Questions.txt";

Question currentQuestion = selectedTest[currentQuestionIndex];

string flaggedQuestionEntry = $"Test {testNumber}, Question {currentQuestionIndex + 1}:{Environment.NewLine}" +

$"Question: {currentQuestion.Text}{Environment.NewLine}" +

$"Answer: {currentQuestion.Options[currentQuestion.CorrectOptionIndex]}{Environment.NewLine}";

if (flaggedQuestions.Contains(currentQuestionIndex))

{

flaggedQuestions.Remove(currentQuestionIndex);

if (File.Exists(filePath))

{

*var* lines = File.ReadAllLines(filePath).ToList();

int startIndex = lines.FindIndex(line => line.StartsWith($"Test {testNumber}, Question {currentQuestionIndex + 1}:"));

if (startIndex != -1) lines.RemoveRange(startIndex, 3);

File.WriteAllLines(filePath, lines);

}

}

else

{

flaggedQuestions.Add(currentQuestionIndex);

if (!File.Exists(filePath)) File.WriteAllText(filePath, flaggedQuestionEntry + Environment.NewLine);

else if (!File.ReadAllLines(filePath).Contains($"Test {testNumber}, Question {currentQuestionIndex + 1}:"))

File.AppendAllText(filePath, flaggedQuestionEntry + Environment.NewLine);

}

}

**A screenshot of a computer

Description automatically generated**

**For Mock Test Page Scores:**

private void SaveScoreToFile(int testNumber, int score)

{

string filePath = "Mock\_Score.txt";

string scoreLine = $"Test: Test {testNumber}, Score: {score}/{selectedTest.Count}";

if (File.Exists(filePath))

{

*var* lines = File.ReadAllLines(filePath).ToList();

bool testFound = false;

for (int i = 0; i < lines.Count; i++)

{

if (lines[i].StartsWith($"Test: Test {testNumber}"))

{

lines[i] = scoreLine;

testFound = true;

break;

}

}

if (!testFound) lines.Add(scoreLine);

File.WriteAllLines(filePath, lines);

}

}

A screenshot of a computer

Description automatically generated

**For Progress Page (saving completed topics progress):**

private void UpdateTopicsCompleted()

{

string filePath = "data.txt";

if (!File.Exists(filePath)) return;

*var* completedTopics = new List<string>();

try

{

*var* lines = File.ReadAllLines(filePath);

foreach (*var* line in lines)

{

*var* parts = line.Split(':');

if (parts.Length == 2 && bool.TryParse(parts[1], out bool isCompleted) && isCompleted)

completedTopics.Add(parts[0].Trim());

}

}

catch (Exception ex) { Debug.WriteLine($"Error reading or parsing file {filePath}: {ex.Message}"); }

}

// Progress Page: Load Mock Scores

private Dictionary<int, int> LoadMockScores(string filePath)

{

*var* scores = new Dictionary<int, int>();

if (!File.Exists(filePath)) return scores;

*var* lines = File.ReadAllLines(filePath);

foreach (*var* line in lines)

{

try

{

*var* parts = line.Split(new[] { "Test: ", ", Score: " }, StringSplitOptions.RemoveEmptyEntries);

if (parts.Length == 2

&& int.TryParse(parts[0].Replace("Test ", string.Empty), out int testNumber)

&& int.TryParse(parts[1].Split('/')[0], out int score))

{

scores[testNumber] = score;

}

}

catch { }

}

return scores;

}

// Progress Page: Load Practice Scores

private Dictionary<int, int> LoadPracticeScores(string filePath)

{

*var* scores = new Dictionary<int, int>();

if (!File.Exists(filePath)) return scores;

*var* lines = File.ReadAllLines(filePath);

foreach (*var* line in lines)

{

try

{

*var* parts = line.Split(new[] { "Test: ", ", Score: " }, StringSplitOptions.RemoveEmptyEntries);

if (parts.Length == 2

&& int.TryParse(parts[0].Replace("Test ", string.Empty), out int testNumber)

&& int.TryParse(parts[1].Split('/')[0], out int score))

{

scores[testNumber] = score;

}

}

catch { }

}

return scores;

}

// Flagged Questions Page: Display Flagged Questions

private void DisplayFlaggedQuestions()

{

string filePath = "Flagged\_Questions.txt";

if (!File.Exists(filePath)) return;

try

{

*var* lines = File.ReadAllLines(filePath);

string currentTest = string.Empty;

for (int i = 0; i < lines.Length; i++)

{

string line = lines[i].Trim();

if (line.StartsWith("Test") && line.Contains(", Question"))

{

currentTest = line; // Test identifier

}

else if (line.StartsWith("Question:"))

{

string questionText = line.Substring("Question:".Length).Trim();

string answerText = lines[++i].Substring("Answer:".Length).Trim();

}

}

}

catch (Exception ex) { Debug.WriteLine($"Error reading flagged questions: {ex.Message}"); }

}

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 7 – Settings Page:**  
***Overview:***

The Settings Page is a centralized form designed to provide users with control over customization options for the application. It allows for changes to visual elements (background color, font size, and style) and allows users to reset stored progress.

***Development and Debugging (broken down into chronological iterations/updates):***

**Iteration 1:**

Setting button to change the background colour of each form for user visibility  
Creating a new Form Page of Settings:

private void Change\_Color\_Click(object sender, EventArgs e)

{

using (ColorDialog colorDialog = new ColorDialog())

{

if (colorDialog.ShowDialog() == DialogResult.OK)

{

GlobalBackgroundColor = colorDialog.Color; // Set the global background color

MessageBox.Show("Background color updated. It will apply to all forms when they are reopened.",

"Settings",

MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

}

}

**A yellow square with a white text

Description automatically generatedA white background with red text

Description automatically generatedA screenshot of a computer screen

Description automatically generated**

**Problem:** How can the user change the size of the font? **Solution:** Introduce a slider on the Settings Page So That The User can change the size of the font as well as choose if they want to make it bold/italic

private void FontSizeSlider\_Scroll(object sender, EventArgs e)

{

// Update the global font size based on the slider value

GlobalFontSize = fontSizes[fontSizeSlider.Value];

// Update the font size of the preview label

fontPreviewLabel.Font = new Font("Arial", GlobalFontSize, GlobalFontStyle);

}

private void FontStyleCheckBox\_CheckedChanged(object sender, EventArgs e)

{

FontStyle fontStyle = FontStyle.Regular;

if (boldCheckBox.Checked)

fontStyle |= FontStyle.Bold;

if (italicCheckBox.Checked)

fontStyle |= FontStyle.Italic;

GlobalFontStyle = fontStyle;

// Update the font style of the preview label

fontPreviewLabel.Font = new Font("Arial", GlobalFontSize, GlobalFontStyle);

MessageBox.Show($"Font style updated to {GlobalFontStyle}.", "Settings",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

**Problem:** Forms Opening anywhere on the screen  
**Solution**: Make The Forms on center when they are being loaded in runtime.

this.StartPosition = FormStartPosition.CenterScreen;

A screenshot of a computer

Description automatically generated

*A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated*

**Iteration 2:**

Adding a Reset Button which Deletes the mock test practice test scores as well as topics completed too.

private void reset\_Click(object sender, EventArgs e)

{

// File paths (update these with the correct file paths as needed)

string[] filesToDelete = {

"PracticeScores.txt",

"Mock\_Score.txt",

"Flagged\_Questions.txt",

"data.txt"

};

bool filesDeleted = false; // Track if any files were deleted

bool filesExist = false; // Track if any files existed

foreach (string file in filesToDelete)

{

try

{

if (System.IO.File.Exists(file))

{

filesExist = true;

System.IO.File.Delete(file);

filesDeleted = true; // Mark that at least one file was deleted

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while deleting '{file}': {ex.Message}", "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return; // Exit the method on error

}

}

// Show appropriate message after processing all files

if (filesDeleted)

{

MessageBox.Show("Data has been deleted.", "Reset",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else if (!filesExist)

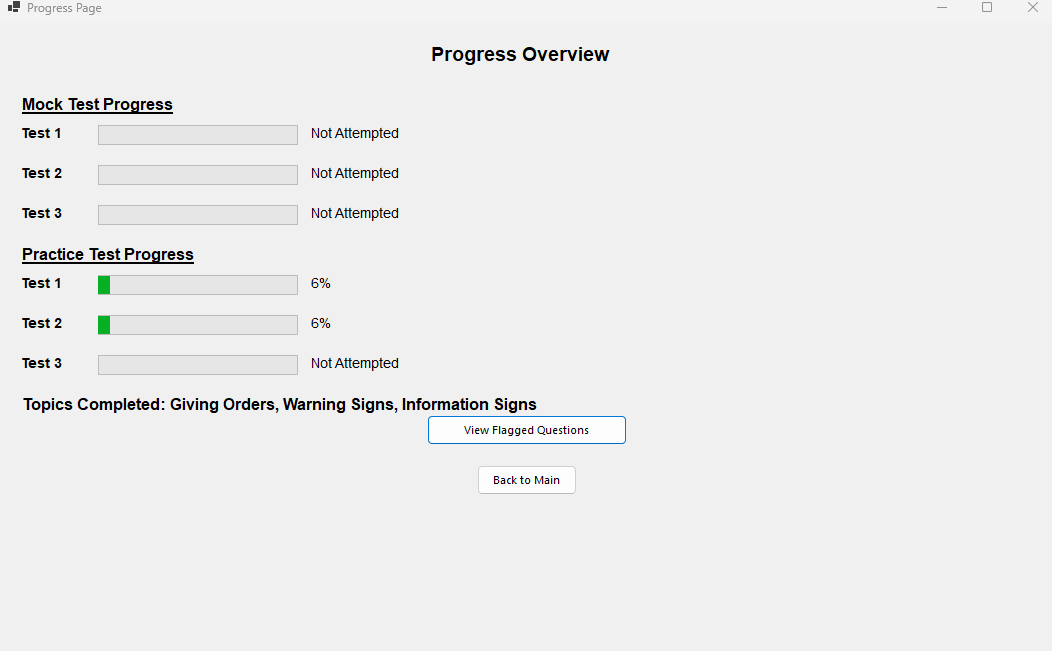
{

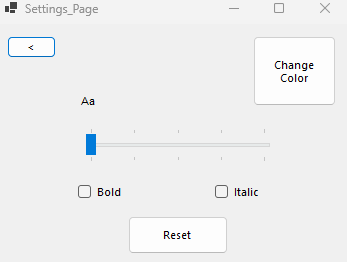
MessageBox.Show("No data to reset.", "Reset",

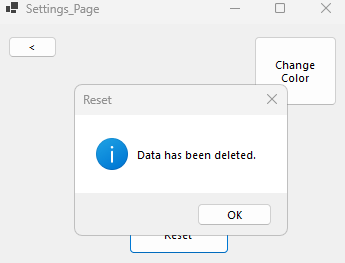
MessageBoxButtons.OK, MessageBoxIcon.Warning);

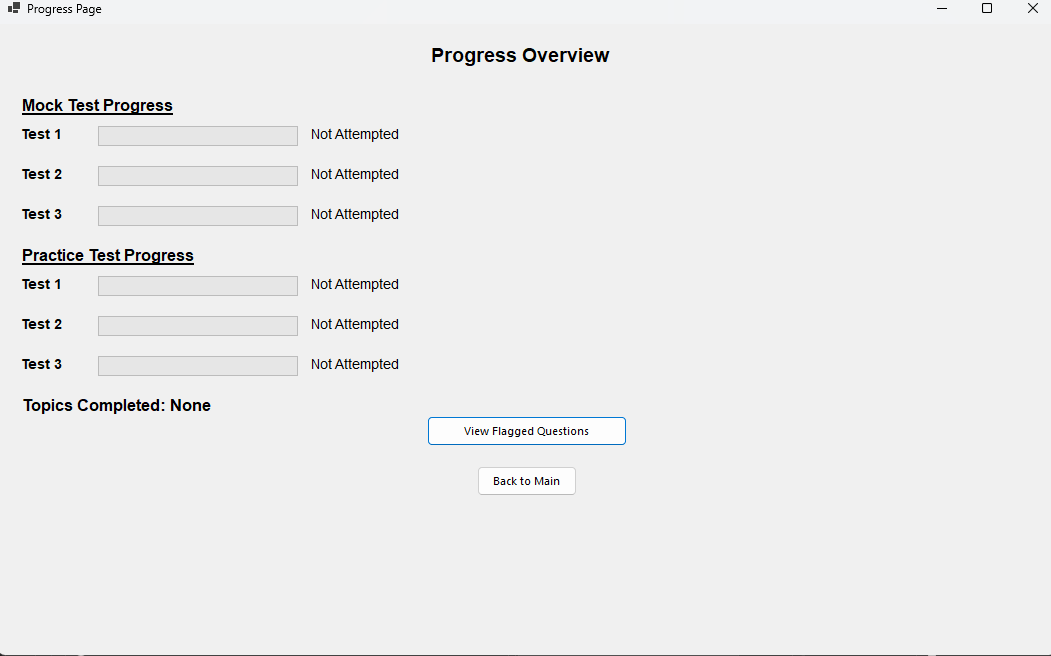
}

}

******

******

******

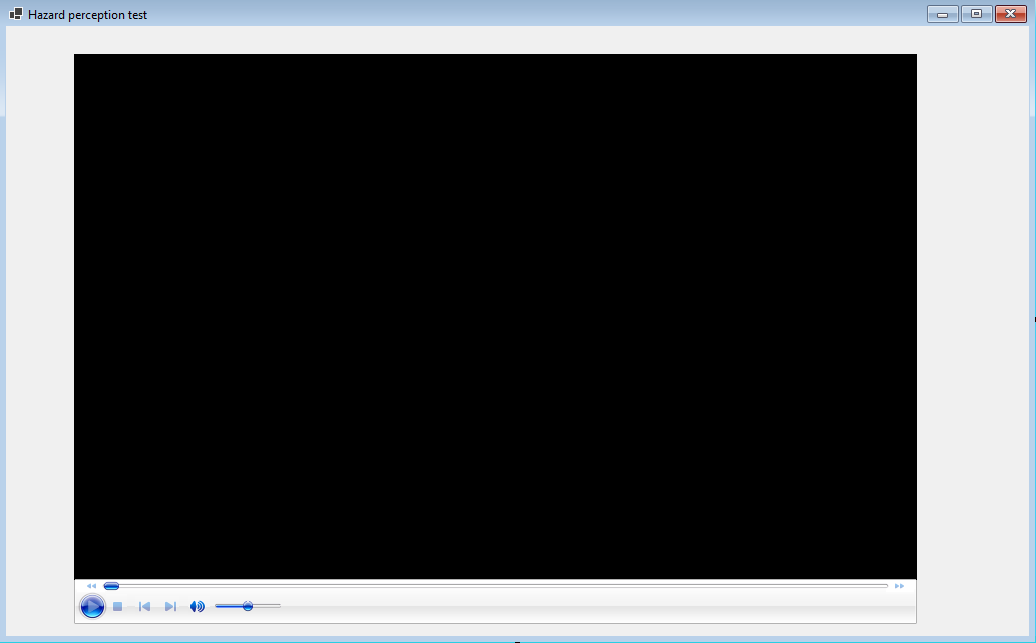
******

***Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |

**Prototype 8 - Hazard Perception Test Page:**  
***Overview:***

The Hazard Perception Test Page is an interactive application designed to test users' ability to detect and respond to hazards in a video-based environment. The user interacts with a video by flagging potential hazards at specific timestamps, earning points based on the timing and accuracy of their responses. The page has undergone several development iterations to improve functionality, user experience, and scoring logic.

***Development and Debugging (broken down into chronological iterations/updates):***  
**Iteration 1:**A Hazard Perception Video which plays a video where the user have to mark/flag where the upcoming obstacle is coming from  
**Problem:** How do add a video on the user form?  
**Solution:** Add Windows Media player from the tool box.  
  
  
  
Adding a Label for the Timestamp & adding a timer:

private void Hazard\_perception\_test\_Load(object sender, EventArgs e)

{

axWindowsMediaPlayer1.URL = "C:\\path\\to\\your\\video.mp4";

axWindowsMediaPlayer1.Ctlcontrols.play();

timer1.Interval = 500; timer1.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

Update the label with the current timestamp if (axWindowsMediaPlayer1.currentMedia != null) { double currentTime = axWindowsMediaPlayer1.Ctlcontrols.currentPosition; label1.Text = TimeSpan.FromSeconds(currentTime).ToString(@"hh\:mm\:ss"); }

}

**Iteration 2:Problem:** Remove the ability for the user to interact with video ui and media control  
**Solution:** I can configure the axWindowsMediaPlayer control to hide the UI elements for playback control.

using System;

using System.Windows.Forms;

*namespace* WinFormsApp1

{

public partial *class* Hazard\_perception\_test : Form

{

public Hazard\_perception\_test()

{

InitializeComponent();

// Configure axWindowsMediaPlayer on form load

axWindowsMediaPlayer1.uiMode = "none"; // Hide user controls

}

private System.Windows.Forms.Timer timer1;

private System.Windows.Forms.Label label1;

private void axWindowsMediaPlayer1\_Enter(object sender, EventArgs e)

{

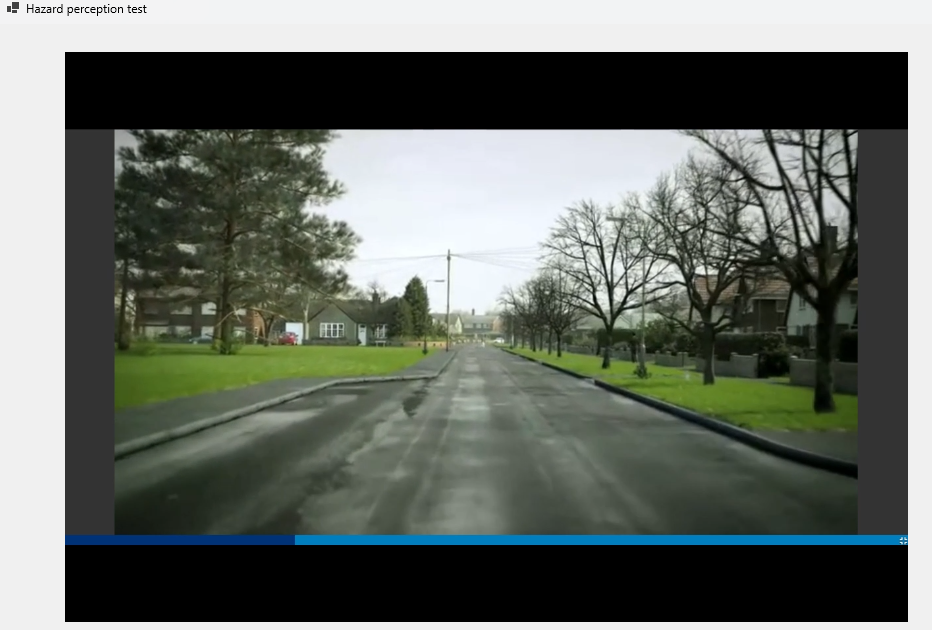
axWindowsMediaPlayer1.URL = "MY EYES \_ Miles Morales Edit (4K).mp4"; // Set the path to your video

axWindowsMediaPlayer1.Ctlcontrols.play(); // Play video automatically

}

}

}

******

**Iteration 3:Problem:** How to mark flags on the time stamp?   
**Solution:** Whenever a user clicks anywhere on the form make it so it flags that specific time stamp of the video and shows it to the user

using System;

using System.Windows.Forms;

*namespace* WinFormsApp1

{

public partial *class* Hazard\_perception\_test : Form

{

public Hazard\_perception\_test()

{

InitializeComponent();

// Configure axWindowsMediaPlayer on form load

axWindowsMediaPlayer1.uiMode = "none"; // Hide user controls

// Configure form's click event to flag timestamps

this.Click += Form\_Click;

}

private void axWindowsMediaPlayer1\_Enter(object sender, EventArgs e)

{

axWindowsMediaPlayer1.URL = "MY EYES \_ Miles Morales Edit (4K).mp4"; // Set the path to your video

axWindowsMediaPlayer1.Ctlcontrols.play(); // Play video automatically

}

private void Form\_Click(object sender, EventArgs e)

{

// Get the current position of the video

double currentTime = axWindowsMediaPlayer1.Ctlcontrols.currentPosition;

// Format the timestamp to minutes and seconds

TimeSpan timestamp = TimeSpan.FromSeconds(currentTime);

string formattedTime = timestamp.ToString(@"mm\:ss");

// Display the flagged time

MessageBox.Show($"Flagged Timestamp: {formattedTime}", "Flagged Time", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

**Problem:** How to award points?  
**Solution:** User earns points based on their clicks within the first 5 seconds of the video for now , and display a prompt with the final points when the video ends

using System;

using System.Windows.Forms;

*namespace* WinFormsApp1

{

public partial *class* Hazard\_perception\_test : Form

{

private int correctClicks;

private System.Windows.Forms.Timer videoTimer;

private Label timestampDisplay;

private System.Collections.Generic.List<string> timestamps;

public Hazard\_perception\_test()

{

InitializeComponent();

// Configure axWindowsMediaPlayer on form load

axWindowsMediaPlayer1.uiMode = "none"; // Hide user controls

// Initialize the timer

videoTimer = new System.Windows.Forms.Timer();

videoTimer.Interval = 1000; // Check every second

videoTimer.Tick += VideoTimer\_Tick;

// Configure form's click event to flag timestamps

this.Click += Form\_Click;

// Initialize the counter for correct clicks

correctClicks = 0;

// Initialize the label to display timestamps

timestampDisplay = new Label();

timestampDisplay.Location = new System.Drawing.Point(943, 28); // Adjust location as needed

timestampDisplay.Size = new System.Drawing.Size(200, 300); // Size of the display area

timestampDisplay.BorderStyle = BorderStyle.FixedSingle;

this.Controls.Add(timestampDisplay);

// List to store timestamps

timestamps = new System.Collections.Generic.List<string>();

}

private void axWindowsMediaPlayer1\_Enter(object sender, EventArgs e)

{

axWindowsMediaPlayer1.URL = "MY EYES \_ Miles Morales Edit (4K).mp4"; // Set the path to your video

axWindowsMediaPlayer1.Ctlcontrols.play(); // Play video automatically

// Start the timer to monitor video end

videoTimer.Start();

}

private void Form\_Click(object sender, EventArgs e)

{

// Get the current position of the video

double currentTime = axWindowsMediaPlayer1.Ctlcontrols.currentPosition;

// Format the timestamp and add to the list

TimeSpan timestamp = TimeSpan.FromSeconds(currentTime);

string formattedTime = timestamp.ToString(@"mm\:ss");

timestamps.Add(formattedTime);

// Update the timestamp display

timestampDisplay.Text = string.Join("\n", timestamps);

}

private void VideoTimer\_Tick(object sender, EventArgs e)

{

// Check if the video has reached its end

if (axWindowsMediaPlayer1.Ctlcontrols.currentPosition >= axWindowsMediaPlayer1.currentMedia.duration)

{

// Video has ended, stop the timer

videoTimer.Stop();

// Gather all timestamps and categorize them as correct or incorrect

string resultText = "Flagged Timestamps:\n";

foreach (*var* timestamp in timestamps)

{

// Determine if the timestamp was flagged within the first 5 seconds of the video

double timestampInSeconds = TimeSpan.Parse(timestamp).TotalSeconds;

string correctness = timestampInSeconds <= 5 ? "Correct" : "Incorrect";

if (correctness == "Correct")

{

correctClicks++;

}

resultText += $"{correctness}: {timestamp}\n";

}

// Show the result as a message box

MessageBox.Show($"{resultText}Your total points: {correctClicks}", "Final Points", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

**Obstacle:** Comparison of time format is wrong   
**Solution:** Convert the hardcoded 5 seconds into a formatted timestamp string "00:05" and then compare it to the timestamps recorded during the video.

private void VideoTimer\_Tick(object sender, EventArgs e)

{

// Check if the video has reached its end

if (axWindowsMediaPlayer1.Ctlcontrols.currentPosition >= axWindowsMediaPlayer1.currentMedia.duration)

{

// Video has ended, stop the timer

videoTimer.Stop();

// Prepare the hardcoded timestamp for comparison (5 seconds)

string fiveSecondsTimestamp = "00:05";

double fiveSeconds = TimeSpan.Parse(fiveSecondsTimestamp).TotalSeconds;

// Gather all timestamps and categorize them as correct or incorrect

string resultText = "Flagged Timestamps:\n";

foreach (*var* timestamp in timestamps)

{

// Convert the timestamp into seconds

TimeSpan ts = TimeSpan.Parse(timestamp);

double timestampInSeconds = ts.TotalSeconds;

// Compare the timestamp to the "00:05" mark

string correctness = timestampInSeconds <= fiveSeconds ? "Correct" : "Incorrect";

if (correctness == "Correct")

{

correctClicks++;

}

resultText += $"{correctness}: {timestamp}\n";

}

// Show the result as a message box

MessageBox.Show($"{resultText}Your total points: {correctClicks}", "Final Points", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

**Iteration 4:Problem:** Want to show flags in front of time stamps   
**Solution:** To show a flag in front of the timestamps text to indicate flagged times, we can modify the Video\_Click method to add some visual indicator next to each timestamp in the timestampDisplay.

using AxWMPLib;

using System;

using System.Windows.Forms;

*namespace* WinFormsApp1

{

public partial *class* Hazard\_perception\_test : Form

{

private int correctClicks;

private System.Windows.Forms.Timer videoTimer;

private Label timestampDisplay;

private System.Collections.Generic.List<string> timestamps;

public Hazard\_perception\_test()

{

InitializeComponent();

// Configure axWindowsMediaPlayer on form load

axWindowsMediaPlayer1.uiMode = "none"; // Hide user controls

axWindowsMediaPlayer1.fullScreen = false;

axWindowsMediaPlayer1.stretchToFit = true;

// Initialize the timer

videoTimer = new System.Windows.Forms.Timer();

videoTimer.Interval = 1000; // Check every second

videoTimer.Tick += VideoTimer\_Tick;

// Initialize the counter for correct clicks

correctClicks = 0;

// Initialize the label to display timestamps

timestampDisplay = new Label();

timestampDisplay.Location = new System.Drawing.Point(943, 28); // Adjust location as needed

timestampDisplay.Size = new System.Drawing.Size(200, 300); // Size of the display area

timestampDisplay.BorderStyle = BorderStyle.FixedSingle;

this.Controls.Add(timestampDisplay);

// List to store timestamps

timestamps = new System.Collections.Generic.List<string>();

// Event handlers for clicks on the video player

axWindowsMediaPlayer1.ClickEvent += Video\_Click;

}

private void axWindowsMediaPlayer1\_Enter(object sender, EventArgs e)

{

axWindowsMediaPlayer1.URL = "MY EYES \_ Miles Morales Edit (4K).mp4"; // Set the path to your video

axWindowsMediaPlayer1.Ctlcontrols.play(); // Play video automatically

// Start the timer to monitor video end

videoTimer.Start();

}

private void Video\_Click(object sender, \_WMPOCXEvents\_ClickEvent e)

{

// Get the current position of the video

double currentTime = axWindowsMediaPlayer1.Ctlcontrols.currentPosition;

// Format the timestamp and add to the list with marker

TimeSpan timestamp = TimeSpan.FromSeconds(currentTime);

string formattedTime = timestamp.ToString(@"mm\:ss");

string timestampWithMarker = $"⚑ {formattedTime}";

timestamps.Add(timestampWithMarker);

// Update the timestamp display

timestampDisplay.Text = string.Join("\n", timestamps);

}

private void VideoTimer\_Tick(object sender, EventArgs e)

{

// Check if the video has reached its end

if (axWindowsMediaPlayer1.Ctlcontrols.currentPosition >= axWindowsMediaPlayer1.currentMedia.duration)

{

// Video has ended, stop the timer

videoTimer.Stop();

// Prepare the hardcoded timestamp for comparison (5 seconds)

string fiveSecondsTimestamp = "00:05";

double fiveSeconds = TimeSpan.Parse(fiveSecondsTimestamp).TotalSeconds;

// Gather all timestamps and categorize them as correct or incorrect

string resultText = "Flagged Timestamps:\n";

foreach (*var* timestamp in timestamps)

{

// Convert the timestamp into seconds

TimeSpan ts = TimeSpan.Parse(timestamp.Substring(2)); // Remove the ⚑ marker

double timestampInSeconds = ts.TotalSeconds;

// Compare the timestamp to the "00:05" mark

string correctness = timestampInSeconds <= fiveSeconds ? "Correct" : "Incorrect";

if (correctness == "Correct")

{

correctClicks++;

}

resultText += $"{correctness}: {timestamp}\n";

}

// Show the result as a message box

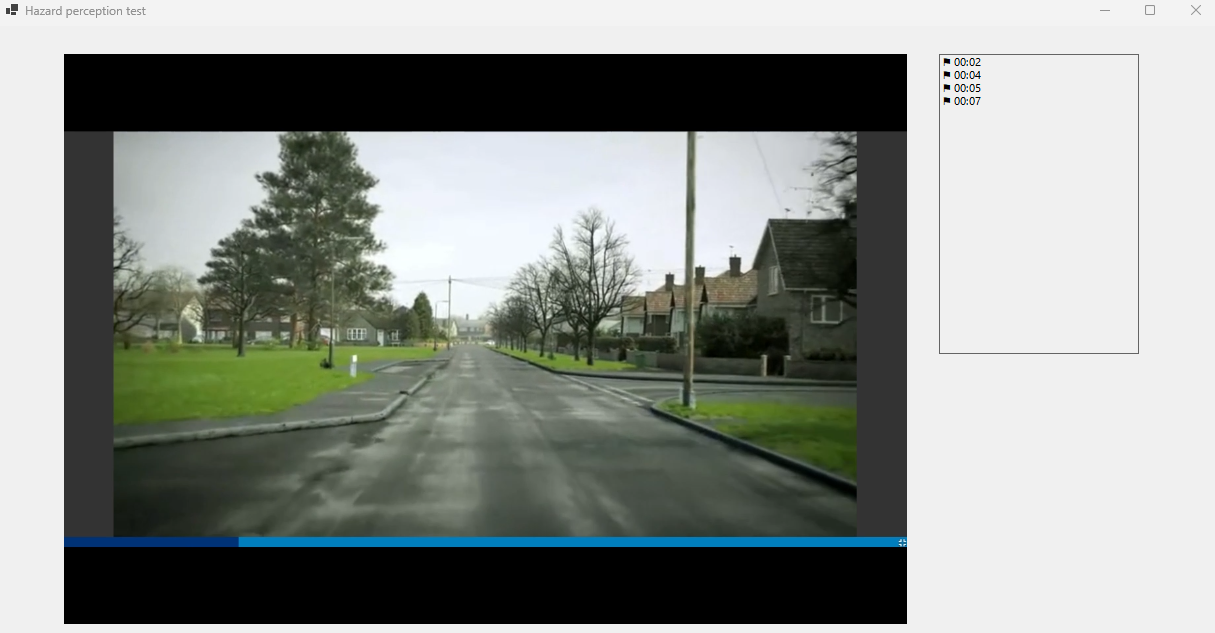
MessageBox.Show($"{resultText}Your total points: {correctClicks}", "Final Points", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

****Problem:** Want to change the time stamps  
**Solution:** Use if else and range to see if the click/flag lies within it

private *void* VideoTimer\_Tick(*object* *sender*, *EventArgs* *e*)

{

// Check if the video has reached its end

if (*axWindowsMediaPlayer1.Ctlcontrols*.currentPosition >= axWindowsMediaPlayer1.currentMedia.duration)

{

// Video has ended, stop the timer

*videoTimer.Stop*();

// Prepare the hardcoded timestamp for comparison (5 seconds)

*double* fiveSeconds = *TimeSpan.Parse*("00:05").*TotalSeconds*;

// Gather all timestamps and categorize them as correct or incorrect with points

string resultText = "Flagged Timestamps:\n";

*int* totalPoints = 0;

bool pointsAdded = false; // Track if any points have been added

foreach (var timestamp in timestamps)

{

// Extract time part from the formatted timestamp

string timePart = *timestamp.Substring*(2); // Remove the ⚑ marker

*TimeSpan* ts = *TimeSpan.Parse*(timePart);

*double* timestampInSeconds = *ts.TotalSeconds*;

// Compare timestamp against each specific time point for points

*int* points = 0;

if (timestampInSeconds <= *TimeSpan.Parse*("00:01").*TotalSeconds*)

{

points = 5;

}

else if (timestampInSeconds <= *TimeSpan.Parse*("00:02").*TotalSeconds*)

{

points = 4;

}

else if (timestampInSeconds <= *TimeSpan.Parse*("00:03").*TotalSeconds*)

{

points = 3;

}

else if (timestampInSeconds <= *TimeSpan.Parse*("00:04").*TotalSeconds*)

{

points = 2;

}

else if (timestampInSeconds <= *TimeSpan.Parse*("00:05").*TotalSeconds*)

{

points = 1;

}

// Only add the first set of points within each interval

if (points > 0 && !pointsAdded)

{

correctClicks += points;

pointsAdded = true; // Mark points as added

}

// Append the result to the display text

resultText += $"{(points > 0 ? "*Correct*" : "*Incorrect*")}: {timestamp} (+{points} points)\n";

}

// Show the result as a message box

*MessageBox.Show*($"{resultText}Your total points: {correctClicks}", "Final Points", *MessageBoxButtons*.OK, *MessageBoxIcon.Information*);

}

}

**Obstacle:** Time stamps Range not being implemented correctly  
**Solution:**

// Show the results in a new form with a "Next" button

Form resultsForm = new Form

{

Text = "Final Results",

Size = new System.Drawing.Size(342, 151),

StartPosition = FormStartPosition.CenterParent

};

Label resultsLabel = new Label

{

Text = $"Your total points: {correctClicks}",

Dock = DockStyle.Fill,

AutoSize = false,

TextAlign = ContentAlignment.TopLeft,

Padding = new Padding(10)

};

Button nextButton = new Button

{

Text = "Next",

Dock = DockStyle.Bottom,

Height = 40

};

nextButton.Click += (s, e) =>

{

resultsForm.Close();

// Open the next form with the new video

Hazard\_per\_test3 nextForm = new Hazard\_per\_test3(); // Replace with your next form logic

nextForm.Show();

this.Close();

};

resultsForm.Controls.Add(resultsLabel);

resultsForm.Controls.Add(nextButton);

resultsForm.ShowDialog();

**Obstacle:** Want to Add a form when the video ends so it can open the next video  
**Solution:**

// Show the results in a new form with a "Next" button

Form resultsForm = new Form

{

Text = "Final Results",

Size = new System.Drawing.Size(342, 151),

StartPosition = FormStartPosition.CenterParent

};

Label resultsLabel = new Label

{

Text = $"Your total points: {correctClicks}",

Dock = DockStyle.Fill,

AutoSize = false,

TextAlign = ContentAlignment.TopLeft,

Padding = new Padding(10)

};

Button nextButton = new Button

{

Text = "Next",

Dock = DockStyle.Bottom,

Height = 40

};

nextButton.Click += (s, e) =>

{

resultsForm.Close();

// Open the next form with the new video

Hazard\_per\_test3 nextForm = new Hazard\_per\_test3(); // Replace with your next form logic

nextForm.Show();

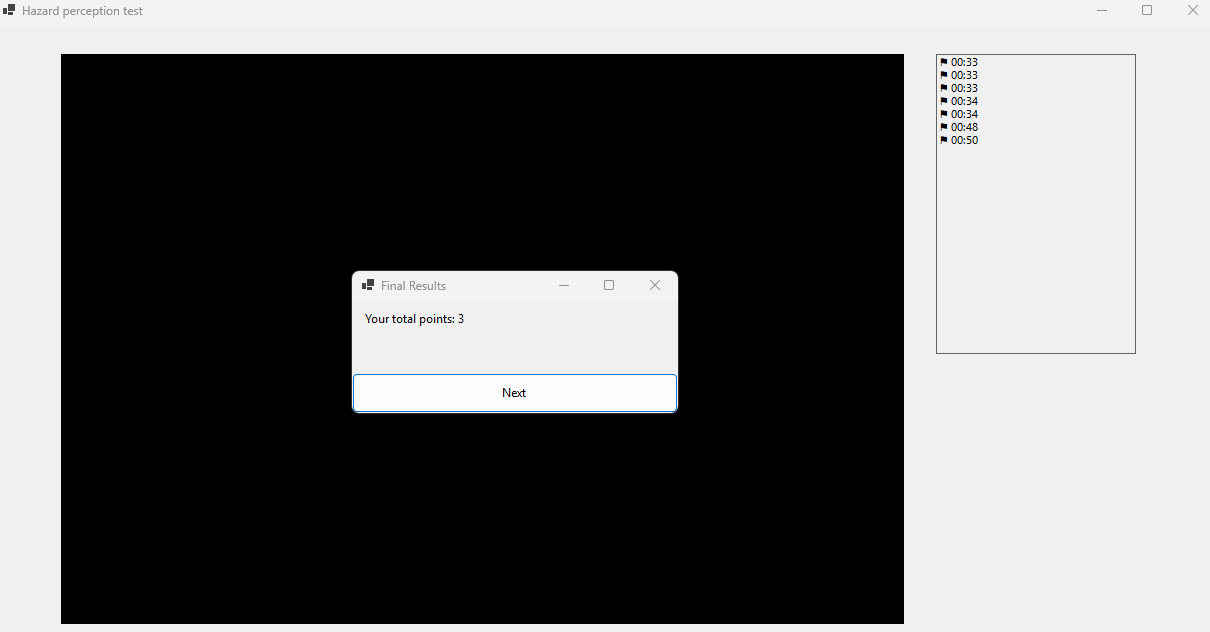
this.Close();

};

resultsForm.Controls.Add(resultsLabel);

resultsForm.Controls.Add(nextButton);

resultsForm.ShowDialog();

***  
  
Review:***

|  |  |
| --- | --- |
| **Success Criteria** | **Result** |
| Does the system allow users to mark topics as completed? |  |
| Does the system persist the checkbox state even when the user navigates away from the page? |  |
| Can users view all signs related to a specific topic when clicking on it? |  |
| Are all traffic signs displayed with clear images and descriptions? |  |
| Is the progress of each topic updated immediately when the user marks it as completed or incomplete? |  |
| Can users navigate seamlessly between different pages (topics, progress, settings, signs)? |  |
| Does the progress page show a complete list of all completed topics with accurate indicators? |  |
| Is the user interface intuitive for selecting topics, viewing signs, and interacting with checkboxes? |  |
| Can users undo or reset their progress on specific topics or all topics if needed? |  |
| Is there a way for users to track and view their overall progress on a single page? |  |
| Are checkboxes and progress indicators (checkmarks, completion percentages) clearly visible and easy to interact with? |  |
| Can users dynamically change the background color using a ColorDialog in the settings page? |  |
| Can users adjust font size using a slider, and toggle font styles (bold/italic) using checkboxes? |  |
| Are all user settings (background color, font size, font styles) applied globally and persisted between sessions? |  |
| Does the settings page provide real-time updates with previews for font and style changes? |  |
| Are images loaded efficiently without causing performance issues, even with a large number of topics or signs? |  |
| Can users interact with images, such as zooming in for better visibility of signs? |  |
| Does the system handle missing or corrupted image files gracefully by showing placeholders? |  |
| Does the layout remain clean, with proper spacing, alignment, and legibility for readability? |  |
| Is there clear error handling for issues like missing data, image errors, or faulty inputs? |  |
| Does the system load quickly without significant delays or lag when switching between pages or topics? |  |
| Does the system visually distinguish between completed and incomplete topics? |  |
| Is the reset button functional, deleting all relevant data files (scores, flagged questions) with success confirmation? |  |
| Does the system prevent crashes or unexpected behavior from faulty inputs or user actions? |  |
| Can flagged questions be viewed separately for easy access? |  |
| Are all forms and pages centered on the screen when opened? |  |
| Does the system handle high data volumes (e.g., many topics or signs) without performance issues? |  |
| Can additional content, such as multimedia (videos, animations), be added without major redesigns? |  |
| Is the system flexible enough to integrate new topics, signs, or complex features (e.g., quizzes, assessments)? |  |
| Does the overall user interface and experience align with personalization, usability, and consistency goals? |  |