**FULL CODE WITH SCREENSHOTS**

Backend :

Form controller :

import { Request, Response } from 'express';

import { readDB, writeDB } from '../db';

import { IForm } from '../models/Form';

const isError = (error: unknown): error is Error => {

  return error instanceof Error;

};

// Create Form

export const createForm = async (req: Request, res: Response) => {

    try {

      console.log('Received request:', req.body); // Log the request body

      const { name, email, phone, githubLink, stopwatchTime } = req.body;

      // Validate input

      if (!name || !email || !phone || !githubLink || !stopwatchTime) {

        return res.status(400).json({ error: 'All fields are required' });

      }

      const newForm: IForm = { name, email, phone, githubLink, stopwatchTime };

      const forms = await readDB();

      forms.push(newForm);

      await writeDB(forms);

      res.status(201).json(newForm);

    } catch (err) {

      if (isError(err)) {

        console.error(`Error creating form: ${err.message}`);

        res.status(500).json({ error: err.message });

      } else {

        console.error('An unknown error occurred');

        res.status(500).json({ error: 'An unknown error occurred' });

      }

    }

  };

// Get Forms

export const getForms = async (req: Request, res: Response) => {

  try {

    const forms = await readDB();

    res.status(200).json(forms);

  } catch (err) {

    if (isError(err)) {

      console.error(`Error getting forms: ${err.message}`);

      res.status(500).json({ error: err.message });

    } else {

      console.error('An unknown error occurred');

      res.status(500).json({ error: 'An unknown error occurred' });

    }

  }

};

// Get Form by Index

export const getFormById = async (req: Request, res: Response) => {

  try {

    const index = parseInt(req.query.index as string, 10);

    const forms = await readDB();

    if (index < 0 || index >= forms.length) {

      return res.status(404).json({ msg: 'Form not found' });

    }

    res.status(200).json(forms[index]);

  } catch (err) {

    if (isError(err)) {

      console.error(`Error getting form by ID: ${err.message}`);

      res.status(500).json({ error: err.message });

    } else {

      console.error('An unknown error occurred');

      res.status(500).json({ error: 'An unknown error occurred' });

    }

  }

};

// Get Form by Email

export const getFormByEmail = async (req: Request, res: Response) => {

  try {

    const email = req.params.email;

    const forms = await readDB();

    const form = forms.find((form: IForm) => form.email === email);

    if (!form) {

      return res.status(404).json({ msg: 'Form not found' });

    }

    res.status(200).json(form);

  } catch (err) {

    if (isError(err)) {

      console.error(`Error getting form by email: ${err.message}`);

      res.status(500).json({ error: err.message });

    } else {

      console.error('An unknown error occurred');

      res.status(500).json({ error: 'An unknown error occurred' });

    }

  }

};

// Delete Form by Index

export const deleteFormById = async (req: Request, res: Response) => {

  try {

    const index = parseInt(req.params.id, 10);

    const forms = await readDB();

    if (index < 0 || index >= forms.length) {

      return res.status(404).json({ msg: 'Form not found' });

    }

    forms.splice(index, 1);

    await writeDB(forms);

    res.status(200).json({ msg: 'Form deleted' });

  } catch (err) {

    if (isError(err)) {

      console.error(`Error deleting form by ID: ${err.message}`);

      res.status(500).json({ error: err.message });

    } else {

      console.error('An unknown error occurred');

      res.status(500).json({ error: 'An unknown error occurred' });

    }

  }

};

// Update Form by Email

export const updateFormByEmail = async (req: Request, res: Response) => {

    try {

      const email = req.params.email;

      const { name, phone, githubLink, stopwatchTime } = req.body;

      const forms = await readDB();

      const formIndex = forms.findIndex((form: IForm) => form.email === email);

      if (formIndex === -1) {

        return res.status(404).json({ msg: 'Form not found' });

      }

      // Ensure only updating the specific fields and not creating a new entry

      forms[formIndex] = { ...forms[formIndex], name, phone, githubLink, stopwatchTime };

      await writeDB(forms);

      res.status(200).json(forms[formIndex]);

    } catch (err) {

      if (isError(err)) {

        console.error(`Error updating form by email: ${err.message}`);

        res.status(500).json({ error: err.message });

      } else {

        console.error('An unknown error occurred');

        res.status(500).json({ error: 'An unknown error occurred' });

      }

    }

  };

export const ping = (req: Request, res: Response) => {

  res.json({ success: true });

};

Screenshot :

A computer screen with a white screen

Description automatically generated

Front End :

Creation Form :

Code :

Imports System.Net.Http

Imports System.Text

Imports Newtonsoft.Json

Imports System.Windows.Forms

Imports System.Diagnostics

Public Class CreateSubmissionForm

Private stopwatch As New Stopwatch()

Private httpClient As New HttpClient()

Public Sub New()

InitializeComponent()

Me.KeyPreview = True ' Enable KeyPreview for the form to capture key events

Timer1.Interval = 1 ' Set Timer interval to 1 millisecond

End Sub

Private Sub btnStartStop\_Click(sender As Object, e As EventArgs) Handles btnStartStop.Click

If stopwatch.IsRunning Then

stopwatch.Stop()

btnStartStop.Text = "&Start"

Else

stopwatch.Start()

btnStartStop.Text = "&Stop"

Timer1.Start() ' Start the Timer1 component

End If

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

lblStopwatch.Text = stopwatch.Elapsed.ToString("hh\:mm\:ss\.fff")

End Sub

Private Async Sub btnSubmit\_Click(sender As Object, e As EventArgs) Handles btnSubmit.Click

Dim formData = New With {

.name = txtName.Text,

.email = txtEmail.Text,

.phone = txtPhone.Text,

.githubLink = txtGithubLink.Text,

.stopwatchTime = stopwatch.Elapsed.ToString("hh\:mm\:ss\.fff")

}

' Check if all fields are filled

If String.IsNullOrWhiteSpace(formData.name) OrElse

String.IsNullOrWhiteSpace(formData.email) OrElse

String.IsNullOrWhiteSpace(formData.phone) OrElse

String.IsNullOrWhiteSpace(formData.githubLink) Then

MessageBox.Show("All fields must be filled", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Return

End If

Dim json = JsonConvert.SerializeObject(formData)

Dim content = New StringContent(json, Encoding.UTF8, "application/json")

Try

Dim response = Await httpClient.PostAsync("http://localhost:5000/submit", content)

If response.IsSuccessStatusCode Then

MessageBox.Show("Submission successful!")

Else

MessageBox.Show("Error submitting the form. " & response.ReasonPhrase)

End If

Catch ex As Exception

MessageBox.Show("Error submitting the form: " & ex.Message)

End Try

' Reset form fields and stopwatch after submission

ResetForm()

End Sub

Private Sub ResetForm()

txtName.Text = ""

txtEmail.Text = ""

txtPhone.Text = ""

txtGithubLink.Text = ""

lblStopwatch.Text = "00:00:00.000"

stopwatch.Reset()

btnStartStop.Text = "&Start"

Timer1.Stop()

End Sub

Private Sub CreateSubmissionForm\_KeyDown(sender As Object, e As KeyEventArgs) Handles MyBase.KeyDown

If e.Control AndAlso e.KeyCode = Keys.S Then

' Ctrl + S: Start/Stop Stopwatch

btnStartStop.PerformClick()

e.SuppressKeyPress = True ' Prevent the key event from being processed further

ElseIf e.Control AndAlso e.KeyCode = Keys.Enter Then

' Ctrl + Enter: Submit Form

btnSubmit.PerformClick()

e.SuppressKeyPress = True ' Prevent the key event from being processed further

End If

End Sub

Private Sub CreateSubmissionForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

Private Sub Label1\_Click(sender As Object, e As EventArgs) Handles Label1.Click

End Sub

End Class

Screenshot :

A computer screen shot of a computer

Description automatically generated

Instruction form :

Code :

Public Class InstructionsForm

Private Sub InstructionsForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim instructions As String = "This is the Slidely Project made by Nandana Pradeep. " &

vbCrLf & vbCrLf &

"This is a small set of instructions to use the application:" & vbCrLf &

" " & vbCrLf &

"1. You have 3 options: view submissions, create submission, and edit responses." & vbCrLf &

" " & vbCrLf &

"2. Before you can access, you need to login as an admin or user. " &

" If you are the admin, you can only view different submissions and delete them if you want. " &

" If you are a user, you can create your submission and edit them but cannot view responses or delete them." & vbCrLf &

" " & vbCrLf &

"3. Shortcut keys are available: " & vbCrLf &

" - Submit:'Ctrl + Enter" & vbCrLf &

" - Start timer:'Ctrl + S'" & vbCrLf &

" - Next:'Ctrl + N'" & vbCrLf &

" - Previous:'Ctrl + P'" & vbCrLf &

vbCrLf & "Thank you for the opportunity, hope you like it :)"

txtInstructions.Text = instructions

txtInstructions.AutoSize = True

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Me.Close()

End Sub

Private Sub txtInstructions\_TextChanged(sender As Object, e As EventArgs) Handles txtInstructions.TextChanged

End Sub

Private Sub PictureBox1\_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click

End Sub

End Class

Screenshot :

A computer screen shot of a computer

Description automatically generated

login form :

Code :

Public Class LoginForm

Private Sub rbAdmin\_CheckedChanged(sender As Object, e As EventArgs) Handles rbAdmin.CheckedChanged

If rbAdmin.Checked Then

txtUsername.Text = "admin"

txtPassword.Text = "admin"

End If

End Sub

Private Sub rbUser\_CheckedChanged(sender As Object, e As EventArgs) Handles rbUser.CheckedChanged

If rbUser.Checked Then

txtUsername.Text = "user"

txtPassword.Text = "user"

End If

End Sub

Private Sub btnLogin\_Click(sender As Object, e As EventArgs) Handles btnLogin.Click

Dim username = txtUsername.Text

Dim password = txtPassword.Text

If rbAdmin.Checked AndAlso username = "admin" AndAlso password = "admin" Then

Dim mainForm As New MainForm(True) ' Passing False for isAdmin

mainForm.Show()

Hide()

ElseIf rbUser.Checked AndAlso username = "user" AndAlso password = "user" Then

Dim mainForm As New MainForm(False) ' Passing True for isAdmin

mainForm.Show()

Hide()

Else

MessageBox.Show("Invalid username or password", "Login Failed", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End Sub

Private Sub LoginForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Show InstructionForm as popup

Dim instructionForm As New InstructionsForm()

instructionForm.StartPosition = FormStartPosition.CenterParent

instructionForm.ShowDialog(Me)

End Sub

Private Sub PictureBox1\_Click(sender As Object, e As EventArgs)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

End

End Sub

End Class

Screenshot :

A computer screen shot of a computer

Description automatically generated

Main form :

Code :

Public Class MainForm

Inherits System.Windows.Forms.Form

Private isAdmin As Boolean

' Constructor with parameters

Public Sub New(isAdmin As Boolean)

' Initialize components from the designer-generated code

InitializeComponent()

' Set isAdmin field based on parameter

Me.isAdmin = isAdmin

' Configure buttons based on isAdmin

ConfigureButtons()

End Sub

' Method to configure button visibility and enable/disable based on isAdmin

Private Sub ConfigureButtons()

If isAdmin Then

' Admin-specific settings

btnViewSubmissions.Enabled = True

btnCreateSubmission.Enabled = False

btnEditSubmission.Enabled = False

Else

' User-specific settings

btnViewSubmissions.Enabled = False

btnCreateSubmission.Enabled = True

btnEditSubmission.Enabled = True

End If

End Sub

' Event handler for btnViewSubmissions click event

Private Sub btnViewSubmissions\_Click(sender As Object, e As EventArgs) Handles btnViewSubmissions.Click

Dim viewForm As New ViewSubmissionsForm()

viewForm.Show()

End Sub

' Event handler for btnCreateSubmission click event

Private Sub btnCreateSubmission\_Click(sender As Object, e As EventArgs) Handles btnCreateSubmission.Click

Dim createForm As New CreateSubmissionForm()

createForm.Show()

End Sub

' Event handler for btnEditSubmission click event

Private Sub btnEditSubmission\_Click(sender As Object, e As EventArgs) Handles btnEditSubmission.Click

Dim searchForm As New SearchEmailForm()

searchForm.Show()

End Sub

Private Sub MainForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' You can add additional initialization logic here if needed

End Sub

Private Sub btnBack\_Click(sender As Object, e As EventArgs) Handles btnBack.Click

Dim loginForm As New LoginForm

loginForm.Show

Close

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Dim instructionForm As New InstructionsForm()

instructionForm.StartPosition = FormStartPosition.CenterParent

instructionForm.ShowDialog(Me)

End Sub

Private Sub Label1\_Click(sender As Object, e As EventArgs) Handles Label1.Click

End Sub

Private Sub ToolTip1\_Popup(sender As Object, e As PopupEventArgs)

End Sub

Private Sub PictureBox3\_Click(sender As Object, e As EventArgs)

End Sub

End Class

Screenshot :

A screenshot of a computer

Description automatically generated

Instruction form :

Code :

Imports System.Net.Http

Imports Newtonsoft.Json

Public Class SearchEmailForm

Private httpClient As New HttpClient()

Private Async Sub btnSearch\_Click(sender As Object, e As EventArgs) Handles btnSearch.Click

Dim email As String = txtEmail.Text

If String.IsNullOrWhiteSpace(email) Then

MessageBox.Show("Please enter a valid email address.")

Return

End If

Try

Dim response = Await httpClient.GetAsync($"http://localhost:5000/form/email/{email}")

If response.IsSuccessStatusCode Then

Dim jsonResponse = Await response.Content.ReadAsStringAsync()

Dim form = JsonConvert.DeserializeObject(Of FormModel)(jsonResponse)

If form IsNot Nothing Then

' Pass the form data and index (assuming it's retrieved from the response) to the UpdateForm

Dim updateForm As New UpdateForm(form.id, form)

updateForm.Show()

Me.Close()

Else

MessageBox.Show("No form entry found for the provided email address.")

End If

Else

Dim errorMessage = Await response.Content.ReadAsStringAsync()

MessageBox.Show($"Error fetching form entry: {errorMessage}")

End If

Catch ex As Exception

MessageBox.Show($"An error occurred: {ex.Message}")

End Try

End Sub

Private Sub SearchEmailForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

Private Sub txtEmail\_TextChanged(sender As Object, e As EventArgs) Handles txtEmail.TextChanged

End Sub

End Class

Screenshot :

A computer screen shot of a computer

Description automatically generated

Instruction form :

Code :

Imports System.Net.Http

Imports System.Text

Imports Newtonsoft.Json

Public Class UpdateForm

Private httpClient As New HttpClient()

Private formIndex As Integer

Private currentForm As FormModel

Public Sub New(index As Integer, form As FormModel)

InitializeComponent()

formIndex = index

currentForm = form

End Sub

Private Sub UpdateForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

txtName.Text = currentForm.name

txtEmail.Text = currentForm.email

txtPhone.Text = currentForm.phone

txtGithubLink.Text = currentForm.githubLink

txtStopwatchTime.Text = currentForm.stopwatchTime

End Sub

Private Async Sub btnSave\_Click(sender As Object, e As EventArgs) Handles btnSave.Click

currentForm.name = txtName.Text

currentForm.email = txtEmail.Text

currentForm.phone = txtPhone.Text

currentForm.githubLink = txtGithubLink.Text

currentForm.stopwatchTime = txtStopwatchTime.Text

Dim json = JsonConvert.SerializeObject(currentForm)

Dim content = New StringContent(json, Encoding.UTF8, "application/json")

' Ensure the URL uses the email to target the correct form entry

Dim response = Await httpClient.PutAsync($"http://localhost:5000/form/email/{currentForm.email}", content)

If response.IsSuccessStatusCode Then

MessageBox.Show(" updated successfully.")

Me.Close()

Else

Dim errorMessage = Await response.Content.ReadAsStringAsync()

MessageBox.Show($"Error updating form entry: {errorMessage}")

End If

End Sub

Private Sub PictureBox1\_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click

End Sub

End Class

Screenshot :

A computer screen shot of a computer

Description automatically generated

View form :

Code :

Imports System.Net.Http

Imports System.Text

Imports Newtonsoft.Json

Public Class ViewSubmissionsForm

Private currentIndex As Integer = 0

Private forms As List(Of FormModel)

Private httpClient As New HttpClient()

Public Sub New()

InitializeComponent()

Me.KeyPreview = True ' Enable KeyPreview for the form to capture key events

End Sub

Private Async Sub ViewSubmissionsForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Await LoadForms()

DisplayForm()

End Sub

Private Async Function LoadForms() As Task

Dim response = Await httpClient.GetStringAsync("http://localhost:5000/forms")

forms = JsonConvert.DeserializeObject(Of List(Of FormModel))(response)

End Function

Private Sub DisplayForm()

If forms IsNot Nothing AndAlso forms.Count > 0 Then

Dim form = forms(currentIndex)

lblName.Text = form.name

lblEmail.Text = form.email

lblPhone.Text = form.phone

lblGithubLink.Text = form.githubLink

lblStopwatchTime.Text = form.stopwatchTime

End If

End Sub

Private Async Sub btnDelete\_Click(sender As Object, e As EventArgs) Handles btnDelete.Click

If forms IsNot Nothing AndAlso forms.Count > 0 Then

Dim formToDelete = forms(currentIndex)

Dim confirmation = MessageBox.Show($"Are you sure you want to delete the form entry for {formToDelete.name}?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

If confirmation = DialogResult.Yes Then

Await DeleteFormFromServer(currentIndex)

forms.RemoveAt(currentIndex)

If forms.Count > 0 Then

If currentIndex >= forms.Count Then

currentIndex = forms.Count - 1

End If

DisplayForm()

Else

ClearForm()

End If

End If

End If

End Sub

Private Async Function DeleteFormFromServer(index As Integer) As Task

Dim response = Await httpClient.DeleteAsync("http://localhost:5000/form/" & index)

If response.IsSuccessStatusCode Then

MessageBox.Show("Data Deleted Successfully.")

Else

MessageBox.Show("Error deleting form entry from the server.")

End If

End Function

Private Sub ClearForm()

lblName.Text = ""

lblEmail.Text = ""

lblPhone.Text = ""

lblGithubLink.Text = ""

lblStopwatchTime.Text = ""

End Sub

Private Sub btnNext\_Click(sender As Object, e As EventArgs) Handles btnNext.Click

NavigateNext()

End Sub

Private Sub btnPrevious\_Click(sender As Object, e As EventArgs) Handles btnPrevious.Click

NavigatePrevious()

End Sub

Private Sub NavigateNext()

If currentIndex < forms.Count - 1 Then

currentIndex += 1

DisplayForm()

End If

End Sub

Private Sub NavigatePrevious()

If currentIndex > 0 Then

currentIndex -= 1

DisplayForm()

End If

End Sub

Private Sub ViewSubmissionsForm\_KeyDown(sender As Object, e As KeyEventArgs) Handles MyBase.KeyDown

If e.Control AndAlso e.KeyCode = Keys.N Then

' Ctrl + N: Next form

NavigateNext()

e.SuppressKeyPress = True ' Prevent the key event from being processed further

ElseIf e.Control AndAlso e.KeyCode = Keys.P Then

' Ctrl + P: Previous form

NavigatePrevious()

e.SuppressKeyPress = True ' Prevent the key event from being processed further

End If

End Sub

Public Sub btnUpdateForm\_Click(sender As Object, e As EventArgs)

End Sub

End Class

Public Class FormModel

Public Property id As Integer ' Assuming each form entry has an ID for identification

Public Property name As String

Public Property email As String

Public Property phone As String

Public Property githubLink As String

Public Property stopwatchTime As String

End Class

Screenshot :

A computer screen shot of a black screen

Description automatically generated