**Contract Monthly Claim System (CMCS) – Full Documentation**

**1. Documentation**

**1.1 Design Choices**

* **Architecture**:
  + **.NET Core WPF** for modularity, scalability, and cross-platform compatibility.
  + Two-tier architecture:
    1. **Presentation Layer** (GUI/UI) – user-friendly forms for claims submission and approval.
    2. **Business Logic Layer** – validation, approval workflows, notifications.
* **UI Design Principles**:
  + Clean and minimalistic layout.
  + Consistent navigation (menu/sidebar).
  + Error-handling with clear validation messages.
  + Accessibility (keyboard shortcuts, readable fonts).

**1.2 Database Structure**

Main entities:

1. **Users** (Admin, Lecturer, )
2. **Claims** (submitted monthly by lecturers)
3. **SupportingDocuments** (uploads for claims)
4. **Approvals** (tracks approval workflow)

**1.3 GUI Layout**

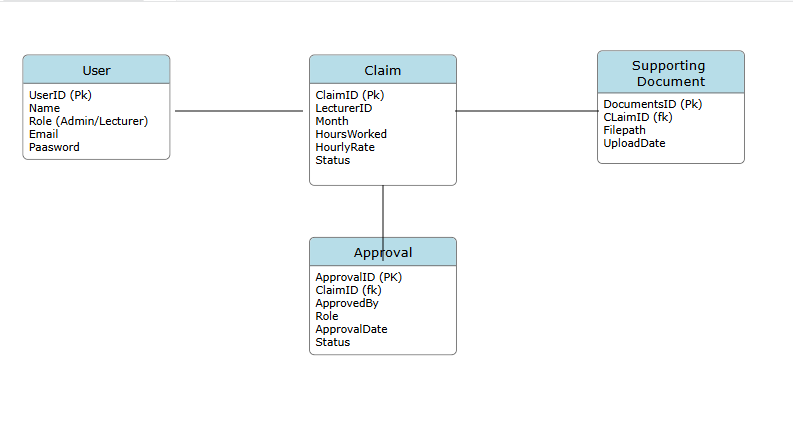
* **Login Screen** – authentication with role-based access.
* **Dashboard**
* **Submit Claim Form**
* **Verify & Approval Panel**
* **Upload documents**
* **Track status**

**1.4 Assumptions & Constraints**

* **Assumptions**:
  + Users have stable internet access.
  + Claims are submitted monthly (one claim per lecturer per month).
  + Each claim requires at least one supporting document.
* **Constraints**:
  + Limited budget for infrastructure → lightweight design.
  + Security requirements → all sensitive data encrypted.
  + Deadline → prototype must be non functional within the given semester.

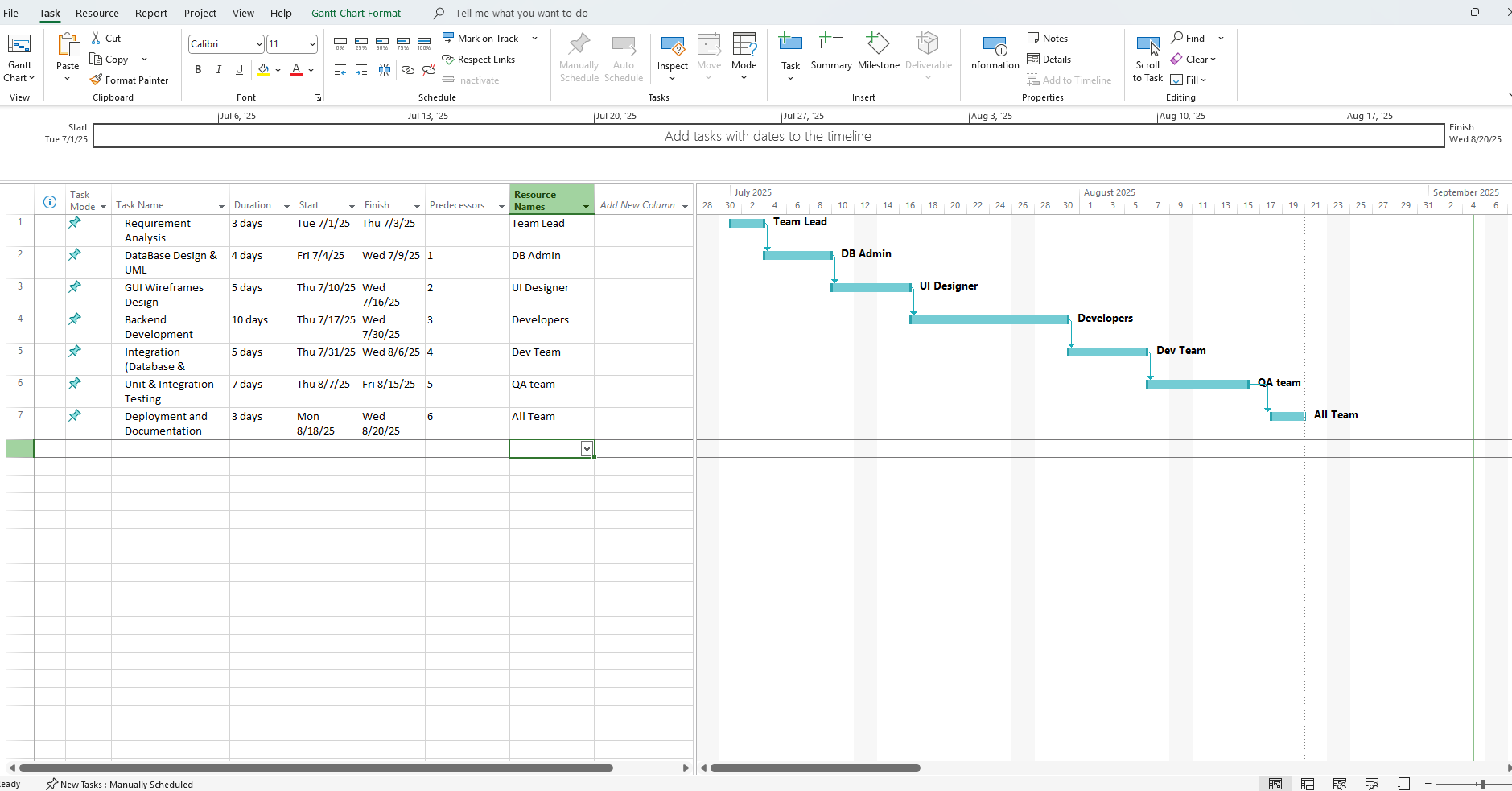
**2.Entities (summary):**

* User-(UserID, FullName, Email, Role, Password)
* Claim(ClaimID, LecturerID→Lecturer, Month, HoursWorked, HourlyRate, Status)
* SupportingDocument(DocumentID, ClaimID→Claim, FilePath, UploadDate)
* Approval(ApprovalID, ClaimID→Claim, ApprovedBy, Role, ApprovalDate, Status)



**3. Project plan**

| **Task ID** | **Task Description** | **Dependencies** | **Duration** | **Assigned To** |
| --- | --- | --- | --- | --- |
| 1 | Requirement Analysis | – | 3 days | Team Lead |
| 2 | Database Design & UML | Task 1 | 4 days | DB Admin |
| 3 | GUI Wireframes & UI Design | Task 1 | 5 days | UI Designer |
| 4 | Backend Development (C# .NET Core) | Task 2, Task 3 | 10 days | Developers |
| 5 | Integration (Database + UI) | Task 4 | 5 days | Dev Team |
| 6 | Testing (Unit & Integration) | Task 5 | 7 days | QA Team |
| 7 | Deployment & Documentation Finalize | Task 6 | 3 days | All Team |
|  |  |  |  |  |



**3) WPF Prototype (Non‑functional)**

**MainWindow.xaml.cs**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace ContractMonthlyClaimSystem2

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Login\_Click(object sender, RoutedEventArgs e)

{

try

{

// Get values from UI controls

string email = txtEmail.Text.Trim();

string password = txtPassword.Password;

string role = (cmbRole.SelectedItem as ComboBoxItem)?.Content.ToString();

// Validate input

if (string.IsNullOrEmpty(email))

{

MessageBox.Show("Please enter your email address.", "Validation Error",

MessageBoxButton.OK, MessageBoxImage.Warning);

txtEmail.Focus();

return;

}

if (string.IsNullOrEmpty(password))

{

MessageBox.Show("Please enter your password.", "Validation Error",

MessageBoxButton.OK, MessageBoxImage.Warning);

txtPassword.Focus();

return;

}

if (string.IsNullOrEmpty(role))

{

MessageBox.Show("Please select your role.", "Validation Error",

MessageBoxButton.OK, MessageBoxImage.Warning);

cmbRole.Focus();

return;

}

// Check credentials for Lecturer

if (email.ToLower() == "lecturer@test.com" && password == "123" && role == "Lecturer")

{

// Create and show the Dashboard window for Lecturer

DashboardWindow dashboardWindow = new DashboardWindow("Lecturer");

dashboardWindow.Show();

this.Close();

}

// Check credentials for Academic Manager/Admin

else if (email.ToLower() == "admin@test.com" && password == "123" && role == "Academic Manager")

{

// Create and show the Dashboard window for Admin

DashboardWindow dashboardWindow = new DashboardWindow("Admin");

dashboardWindow.Show();

this.Close();

}

// Check credentials for Programme Coordinator

else if (email.ToLower() == "coordinator@test.com" && password == "123" && role == "Programme Coordinator")

{

// Create and show the Dashboard window for Coordinator

DashboardWindow dashboardWindow = new DashboardWindow("Coordinator");

dashboardWindow.Show();

this.Close();

}

else

{

// Invalid credentials

MessageBox.Show("Invalid email, password, or role selection.\n\nValid credentials:\n" +

"Lecturer: lecturer@test.com / 123\n" +

"Admin: admin@test.com / 123\n" +

"Coordinator: coordinator@test.com / 123",

"Login Failed",

MessageBoxButton.OK,

MessageBoxImage.Error);

// Clear password field for security

txtPassword.Clear();

txtEmail.Focus();

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred during login: {ex.Message}",

"Error",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

}

private void TxtEmail\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

txtPassword.Focus();

}

}

private void TxtPassword\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

Login\_Click(sender, e);

}

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

txtEmail.Focus();

}

}

/// <summary>

/// Dashboard Window - front-end prototype

/// </summary>

public partial class DashboardWindow : Window

{

private string \_userRole;

public DashboardWindow(string userRole)

{

\_userRole = userRole;

InitializeComponent();

SetupDashboard();

}

private void SetupDashboard()

{

this.Title = $"CMCS Dashboard - {\_userRole}";

}

private void InitializeComponent()

{

// Window setup

this.Width = 1100;

this.Height = 700;

this.WindowStartupLocation = WindowStartupLocation.CenterScreen;

this.Background = new SolidColorBrush(Color.FromRgb(240, 242, 245));

// Root grid with 2 columns (sidebar + main content)

Grid rootGrid = new Grid();

this.Content = rootGrid;

rootGrid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(220) });

rootGrid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

// ---------------- Sidebar ----------------

StackPanel sidebar = new StackPanel

{

Background = new SolidColorBrush(Color.FromRgb(44, 62, 80)),

Orientation = Orientation.Vertical

};

Grid.SetColumn(sidebar, 0);

rootGrid.Children.Add(sidebar);

// Sidebar title

TextBlock sidebarTitle = new TextBlock

{

Text = "CMCS",

Foreground = Brushes.White,

FontSize = 28,

FontWeight = FontWeights.Bold,

Margin = new Thickness(20, 30, 0, 30)

};

sidebar.Children.Add(sidebarTitle);

// Sidebar buttons

sidebar.Children.Add(CreateSidebarButton("Lecture"));

sidebar.Children.Add(CreateSidebarButton("Academic Managers"));

sidebar.Children.Add(CreateSidebarButton("Claims"));

sidebar.Children.Add(CreateSidebarButton("Approvals"));

// Spacer

sidebar.Children.Add(new TextBlock { Height = 50 });

// Footer

TextBlock footer = new TextBlock

{

Text = "Consistent & Reliable",

Foreground = Brushes.LightGray,

FontSize = 14,

Margin = new Thickness(20, 40, 0, 0)

};

sidebar.Children.Add(footer);

// ---------------- Main Content ----------------

StackPanel mainPanel = new StackPanel

{

Margin = new Thickness(40),

HorizontalAlignment = HorizontalAlignment.Center

};

Grid.SetColumn(mainPanel, 1);

rootGrid.Children.Add(mainPanel);

// Title

TextBlock title = new TextBlock

{

Text = "Contract Monthly Claim System",

FontSize = 28,

FontWeight = FontWeights.Bold,

Foreground = new SolidColorBrush(Color.FromRgb(44, 62, 80)),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(0, 0, 0, 20)

};

mainPanel.Children.Add(title);

// Subtitle

TextBlock subtitle = new TextBlock

{

Text = "This is a front-end prototype only. Features will on Part 2.",

FontSize = 16,

Foreground = Brushes.Gray,

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(0, 0, 0, 40)

};

mainPanel.Children.Add(subtitle);

// Big colorful buttons in a WrapPanel

WrapPanel buttonPanel = new WrapPanel

{

HorizontalAlignment = HorizontalAlignment.Center,

ItemWidth = 200,

ItemHeight = 100,

Margin = new Thickness(0, 20, 0, 0)

};

mainPanel.Children.Add(buttonPanel);

buttonPanel.Children.Add(CreateMainButton("Submit Claims", Color.FromRgb(52, 152, 219)));

buttonPanel.Children.Add(CreateMainButton("Verify & Approve", Color.FromRgb(46, 204, 113)));

buttonPanel.Children.Add(CreateMainButton("Upload Documents", Color.FromRgb(243, 156, 18)));

buttonPanel.Children.Add(CreateMainButton("Track Status", Color.FromRgb(155, 89, 182)));

}

// Helper for sidebar buttons

private Button CreateSidebarButton(string text)

{

return new Button

{

Content = text,

Foreground = Brushes.White,

Background = Brushes.Transparent,

BorderBrush = Brushes.Transparent,

FontSize = 16,

Height = 40,

Margin = new Thickness(10, 5, 10, 5),

HorizontalAlignment = HorizontalAlignment.Stretch

};

}

// Helper for main colorful buttons

private Button CreateMainButton(string text, Color bgColor)

{

return new Button

{

Content = text,

Background = new SolidColorBrush(bgColor),

Foreground = Brushes.White,

FontSize = 16,

FontWeight = FontWeights.Bold,

Width = 200,

Height = 100,

Margin = new Thickness(10),

BorderThickness = new Thickness(0)

};

}

}

}

**MainWindow.xaml.**

<Window x:Class="ContractMonthlyClaimSystem2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="CMCS Login" Height="400" Width="500"

WindowStartupLocation="CenterScreen"

ResizeMode="NoResize">

<Grid Background="#F0F0F0">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<!-- Header -->

<TextBlock Grid.Row="0"

Text="Contract Monthly Claiming System"

FontSize="24"

FontWeight="Bold"

HorizontalAlignment="Center"

Margin="0,20,0,20"

Foreground="#2E3440"/>

<!-- Login Form -->

<StackPanel Grid.Row="1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Width="300">

<!-- Email -->

<TextBlock Text="Email:"

Margin="0,0,0,5"

FontWeight="SemiBold"/>

<TextBox x:Name="txtEmail"

Height="35"

Margin="0,0,0,15"

Padding="10,8"

BorderBrush="#CCCCCC"

BorderThickness="1"/>

<!-- Password -->

<TextBlock Text="Password:"

Margin="0,0,0,5"

FontWeight="SemiBold"/>

<PasswordBox x:Name="txtPassword"

Height="35"

Margin="0,0,0,15"

Padding="10,8"

BorderBrush="#CCCCCC"

BorderThickness="1"/>

<!-- Role Selection -->

<TextBlock Text="Role:"

Margin="0,0,0,5"

FontWeight="SemiBold"/>

<ComboBox x:Name="cmbRole"

Height="35"

Margin="0,0,0,20"

Padding="10,8"

BorderBrush="#CCCCCC"

BorderThickness="1">

<ComboBoxItem Content="Lecturer"/>

<ComboBoxItem Content="Academic Manager"/>

</ComboBox>

<!-- Login Button -->

<Button Content="Login"

Height="40"

Background="#4CAF50"

Foreground="White"

BorderThickness="0"

FontSize="16"

FontWeight="SemiBold"

Click="Login\_Click"

Cursor="Hand">

<Button.Style>

<Style TargetType="Button">

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="#45a049"/>

</Trigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

</StackPanel>

<!-- Footer -->

<TextBlock Grid.Row="2"

Text="Test Credentials: lecturer@test.com / admin@test.com | Password: 123"

HorizontalAlignment="Center"

Margin="0,0,0,10"

FontSize="10"

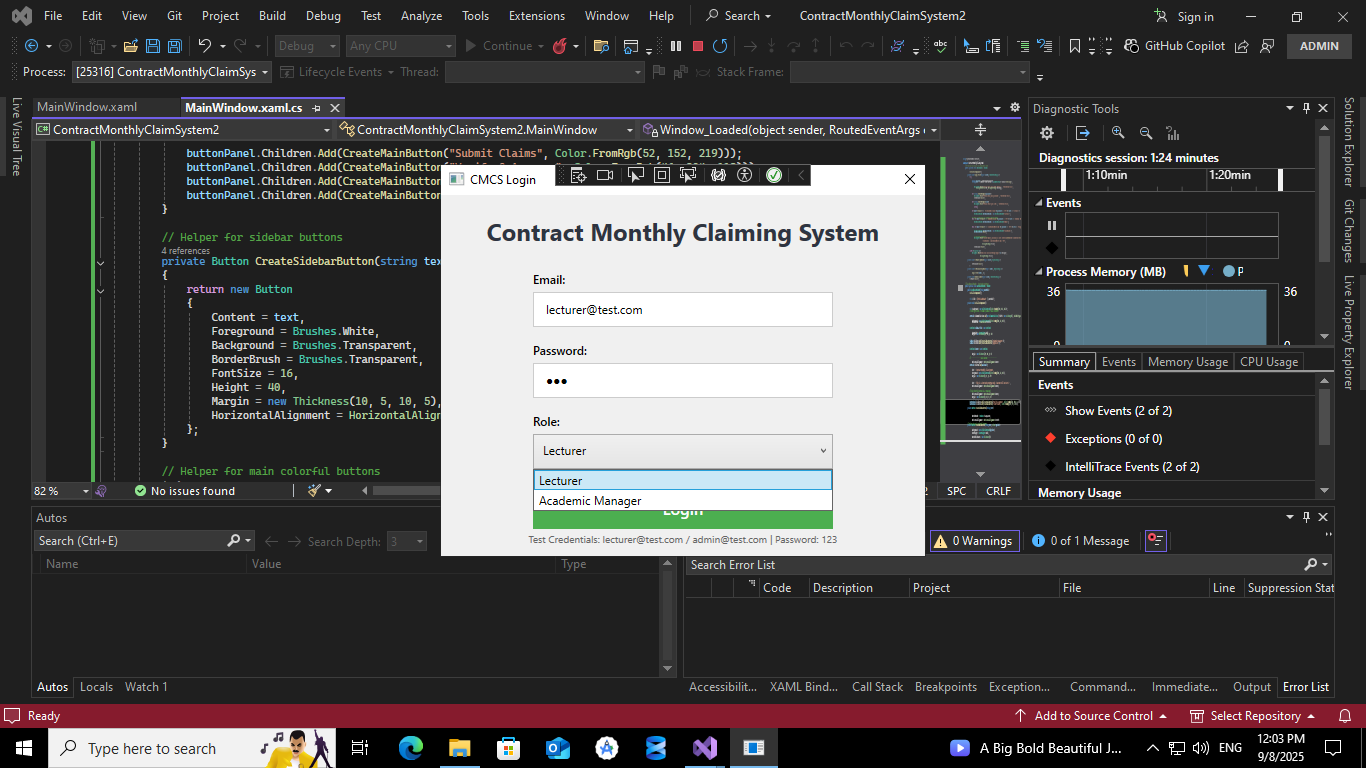
Foreground="#666666"/>

</Grid>

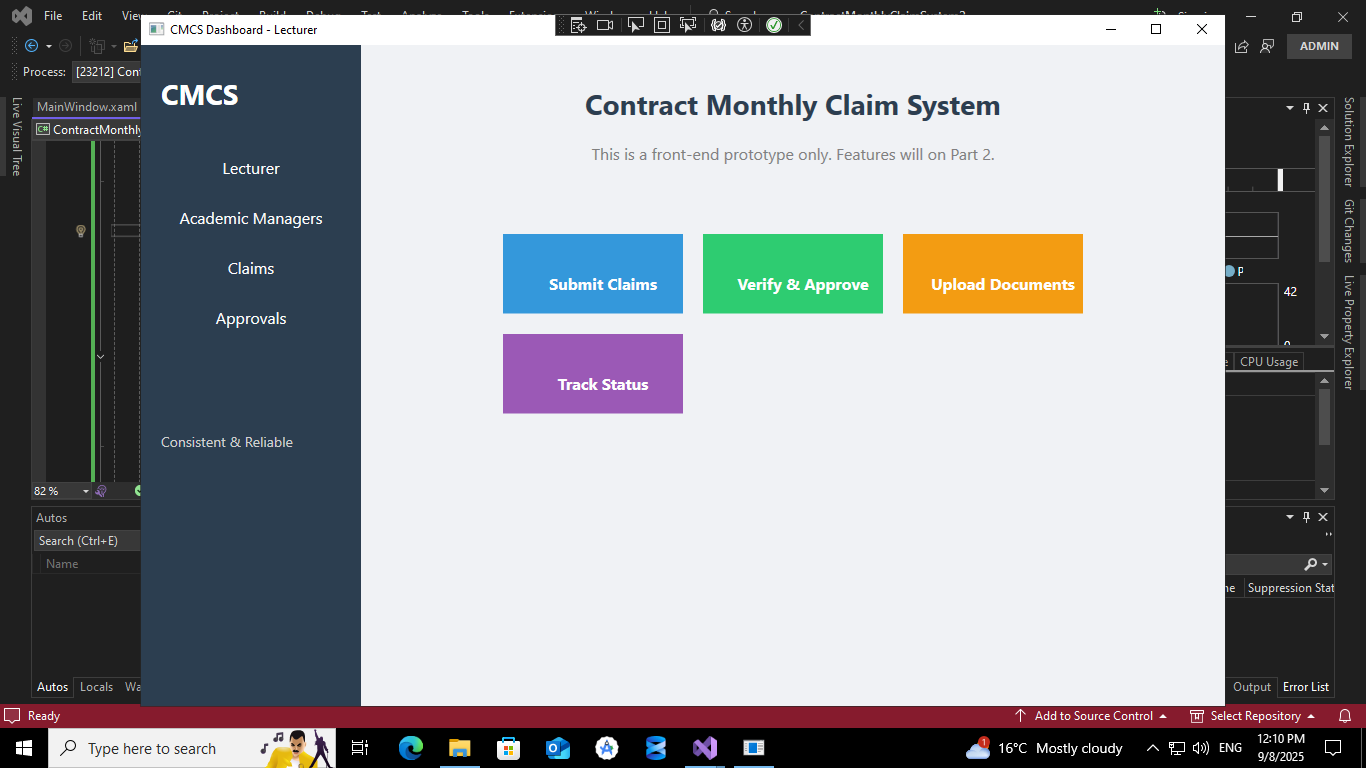
</Window>

**VISUAL PROTOTYPE**

**LOGIN CREDENTIALS**



**Contract Monthly Claim System**



**HOW TO RUN**

Click “**Start**”

Enter login credentials for;

**Lecturer-**

[Email-lecturer@test.com](mailto:Email-lecturer@test.com)

Password-123

**Admin-**

[Email-admin@test.com](mailto:Email-admin@test.com)

password-123

**NB- USE THE CREDENTIALS ABOVE TO LOGIN**

After entering the login credentials, Dashboard will appear

Visual presentation of the prototype that is **NON FUNTIONAL**