Assignment-3: Technical Documentation

CS346: Software Engineering Laboratory

Assignment-3 Report

Task:

To develop a **Community-Based Local Services Platform**

Author:

Group 1B

Instructor:

Prof. Pradip K. Das, Dept. of CSE, IITG

Preface

This project was assigned to **Group 1B** as the final project in the course CS346: Software Engineering Laboratory, Spring 2024 Semester, at IIT Guwahati.

Group 1B consists of the following students (listed roll number-wise):

1. Abhinav Kumar, 210101003
2. Anup Kumar, 210101019
3. Arani Rajesh Kumar, 210101020
4. Arvind Kumar, 210101022
5. Bhogi Sai Sathwik, 210101031
6. Bussa Sai Santhosh, 210101033
7. Devika Singh, 210101036
8. Divya Garg, 210101037
9. Gautam Sharma, 210101042
10. Gholap Sarvesh Sarjerao, 210101043
11. Gutthula Naga Satyam Preetam, 210101044
12. Kshitij Maurya, 210101059
13. Madala Sai Deekshitha, 210101063
14. Majji Aditya, 210101064
15. Mithilesh Gupta, 210101067
16. Parth Kasture, 210101074
17. Posa Mokshith, 210101077
18. Pratham, 210101079
19. Pratham Goyal, 210101080
20. Priyanshu Raj, 210101083
21. Rangu Rishvanja Simha, 210101084
22. Riya Mittal, 210101089
23. Sahil Jaiswal, 210101091
24. Shreya Saraf, 210101099
25. Sreehari C, 210101101
26. Swagat Bhupendra Sathwara, 210101101

27. Vanga Nikhitha, 210101107
28. Pratyush R, 210101116
29. Ketan Singh, 210101118
30. Shivam Agrawal, 210101119
31. Akshit Sharma, 210101124

The following is the team division:

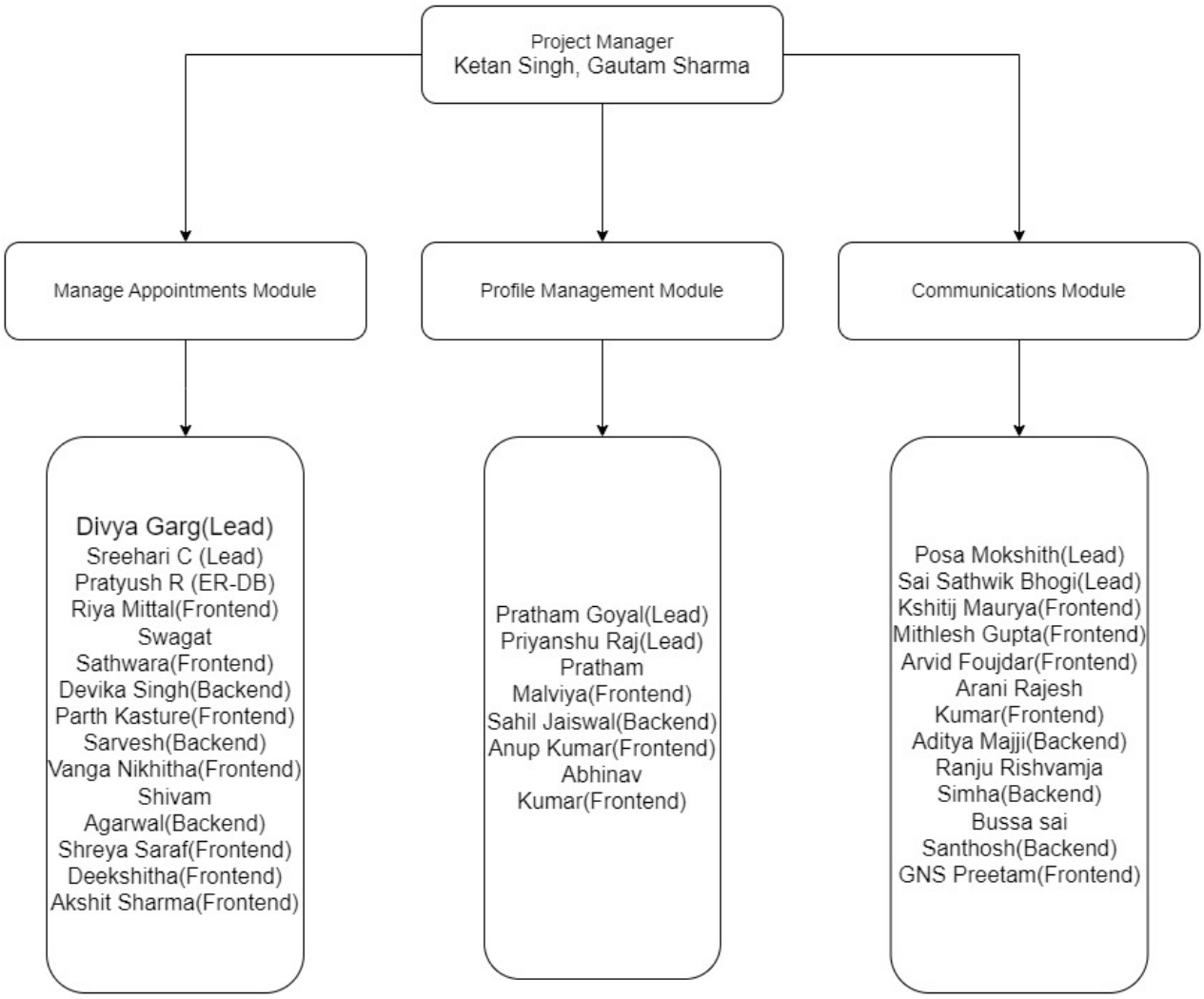

Figure 1: Team Division

Contents

1 Database

The description of tables is made available in INFORMATION_SCHEMA.TABLES
A total of 15 tables have been used, and they are as follows:

1. **admin**
2. **chat_room**
3. **customer**
4. **deals**
5. **feedback**
6. **last_update**
7. **location**
8. **messages**
9. **provider**
10. **refunded_details**
11. **review**
12. **schedule**
13. **subscription**
14. **support_msgs**
15. **support_room**

The description of constraints on these tables is made available in INFORMATION_SCHEMA.TABLE_C and are 25 in count and are as follows: Note that these can be represented more compactly in an ER.

1. **chat_room**
   (a) Primary Key: chat_room_id
   (b) Foreign key to customer: user_id
   (c) Foreign key to provider: provider_id
2. **customer**:
   (a) Primary key: user_id
3. **deals**:
   (a) Primary key: deal_id
   (b) Foreign key to customer: user_id
   (c) Foreign key to provider : provider_id
4. **feedback**:

   (a) Primary key: feedback_id
   (b) Check: user_type

5. **location**:
   (a) Foreign key: provider_id
6. **messages**:
   (a) Primary key: message_id
   (b) Check: sender_type
   (c) Foreign key to chat_room: chat_room_id
7. **provider**:
   (a) Primary key: provider_id
8. **refunded_deals**:
   (a) Primary key: deal_id
9. **review**:
   (a) Primary key: review_id
   (b) Foreign key to deals: deal_id
10. **schedule**:
   (a) Foreign key to customer: user_id
11. **subscription**:
   (a) Foreign key to provider: provider_id
12. **support_msgs**:
   (a) Primary key: message_id
   (b) Check: sender_type
   (c) Foreign key to support_room: support _room_id
13. **support_room**:
   (a) Primary key: support_room_id
   (b) Check : user_type

2 Admin related files

2.1 admin_template.vb

2.1.1 Method Descriptions

- **admin_template_Load**: Event handler method triggered when the admin_template form is loaded. It embeds an instance of the admin_dashboard form into the SplitContainer panel, ensuring proper properties are set for embedding.
- **switchPanel**: Method for switching the panel in the SplitContainer to display different forms. It clears the current panel, sets properties for the new panel, adds the specified form to the panel, and displays it.
- **Reset_Buttons**: Method to reset the background color of navigation buttons. It sets the background color of each button to the default color, ensuring only the selected button is highlighted.
- **Dashboard_btn_Click**: Event handler method triggered when the "Dashboard" button is clicked. It highlights the "Dashboard" button, resets other buttons, and switches the panel to display the admin_dashboard form.
- **Chats_btn_Click**: Event handler method triggered when the "Chats" button is clicked. It highlights the "Chats" button, resets other buttons, and switches the panel to display the admin_side_chat form.
- **Feedbacks_btn_Click**: Event handler method triggered when the "Feedbacks" button is clicked. It highlights the "Feedbacks" button, resets other buttons, and switches the panel to display the AdminFeedbackView form.
- **Logout_btn_Click**: Event handler method triggered when the "Logout" button is clicked. It closes the admin_template form and displays the Admin_Login form for logging out the administrator.

2.1.2 Exception Handling

The code includes exception handling to catch any errors that may occur during database operations. If an exception occurs, an error message dialog box is displayed with the details of the error.

2.2 admin_side_chat.vb

2.2.1 Variable Declarations

- **support_rooms**: A list containing tuples representing support rooms, each tuple consisting of user name, user type, support room, and user id.
- **support_msgs**: A list containing tuples representing support messages, each tuple consisting of room_id, sender type, message content, and send timestamp.
- **user_role**: A string variable indicating the role of the user, initialized as "admin".
- **roomId**: An integer variable initialized as -1 to store the current support room ID.
- **rooms_type**: A string variable indicating the type of rooms, initialized as "customer".

- **connectionString**: A string containing the connection information for the SQL Server database.

2.2.2 Method Descriptions

- **PopulateRooms**: Method to populate the room list with buttons representing available support rooms. It dynamically creates buttons for each room and sets their properties such as name, text, color, and event handlers.
- **LoadRoomsFromDatabase**: Method to load support rooms from the database into the support_rooms list. It executes a SQL query and populates the list with room details.
- **LoadMessagesFromDatabase**: Method to load support messages from the database into the support_msgs list. It executes a SQL query and populates the list with message details.
- **user_chats_Load**: Event handler method triggered when the admin_side_chat form is loaded. It initializes the messageTimer, loads rooms and messages from the database, sets initial UI state, and starts the messageTimer.
- **MainForm_FormClosed**: Event handler method triggered when the admin_side_chat form is closed. It stops the messageTimer.
- **MessageTimer_Tick**: Event handler method triggered when the messageTimer ticks. It reloads rooms and messages from the database and updates the UI if changes are detected.
- **userButton_Click**: Event handler method triggered when the "User" button is clicked. It sets the rooms_type to "customer", updates UI state, and populates rooms accordingly.
- **providerButton_Click**: Event handler method triggered when the "Provider" button is clicked. It sets the rooms_type to "provider", updates UI state, and populates rooms accordingly.
- **Button_Click**: Event handler method triggered when a room button is clicked. It sets the roomId, updates UI state, and prints messages for the selected room.
- **PrintMessages**: Method to print messages for a given room. It clears existing messages, filters and sorts messages for the room, and dynamically creates labels to display messages and their timestamps.
- **sendButton_Click**: Event handler method triggered when the "Send" button is clicked. It sends the typed message to the database, updates the support_msgs list, and prints messages.

2.3 admin_dashboard.vb

2.3.1 Library Imports

- **FastReport.DataVisualization.Charting**: Imports the Charting namespace from the FastReport.DataVisualization library, used for data visualization.
- **Microsoft.Data.SqlClient**: Imports the SqlConnection class from the Microsoft.Data.SqlClient namespace, used for database connectivity.

- **System.Data.SqlClient**: Imports the SqlConnection class from the System.Data.SqlClient namespace, used for database connectivity.
- **System.Windows.Forms.VisualStyles.VisualStyleElement**: Imports the VisualStyleElement class from the System.Windows.Forms.VisualStyles namespace, used for visual styles in Windows Forms applications.

2.3.2 Method Descriptions

- **admin_dashboard_Load**: Event handler method triggered when the admin_dashboard form is loaded. It executes SQL queries to retrieve data for various charts and statistics, populates the charts with data, and updates labels with statistics.

2.4 AdminFeedbackView.vb

2.4.1 Library Imports

- **System.Configuration**: Imports the Configuration namespace, used for accessing configuration files.
- **System.Data.SqlClient**: Imports the SqlConnection class from the System.Data.SqlClient namespace, used for database connectivity.
- **Microsoft.Data.SqlClient**: Imports the SqlConnection class from the Microsoft.Data.SqlClient namespace, used for database connectivity.

2.4.2 Class Description

The ‘AdminFeedbackView‘ class is responsible for displaying feedback from users (customers or providers) in an admin panel. It allows the admin to filter feedback based on user type (customer or provider) and rating.

2.4.3 Method Descriptions

- **AdminFeedbackView_Load**: Event handler method triggered when the form is loaded. It sets default button colors and loads feedback for customers by default.
- **userButton_Click**: Event handler method triggered when the user clicks the "Customer" button. It sets button colors and loads feedback for customers.
- **providerButton_Click**: Event handler method triggered when the user clicks the "Provider" button. It sets button colors and loads feedback for providers.
- **applyButton_Click**: Event handler method triggered when the user clicks the "Apply" button to apply filters. It retrieves the selected rating and loads feedback accordingly.
- **LoadFeedback**: Method to load feedback based on user type and rating filter. It constructs an SQL query, executes it, and dynamically creates panels to display feedback information.
- **GetSelectedRating**: Method to get the selected rating from the ComboBox or RadioButton controls.

2.5 Admin_Login.vb

2.5.1 Method Descriptions

- **Admin_Login_Load**: Event handler method triggered when the form is loaded. It initializes the error label to an empty string.
- **Showpassword_cb_CheckedChanged**: Event handler method triggered when the state of the "Show Password" checkbox changes. It toggles the visibility of the password characters in the password textbox.
- **Back_btn_Click**: Event handler method triggered when the "Back" button is clicked. It closes the current form and displays the Landing form.
- **Login_btn_Click**: Event handler method triggered when the "Login" button is clicked. It validates the administrator's password against the database and displays the admin_template form upon successful authentication.

2.5.2 Exception Handling

The code includes exception handling to catch any errors that may occur during database operations. If an exception occurs, an error message dialog box is displayed with the details of the error.

3 Customer Related Files

3.1 user_appointment_details.vb

3.1.1 Variable Declarations

- `dealID` (Integer): Represents the ID of the deal. Default value is 1.
- `startTime` (TimeSpan): Represents the start time of the appointment.
- `firstDate` (DateTime): Represents the date of the appointment.
- `bookDate` (DateTime): Represents the date when the appointment was booked.
- `provider` (Integer): Represents the ID of the provider. Default value is 0.

3.1.2 Methods

- `OnVisibleChanged(e As EventArgs)`: Overrides the `OnVisibleChanged` method to reload data and make chat visible when the form is visible.
- `MakeChatVisible()`: Creates and shows a chat form inside `SplitContainer1.Panel2`.
- `ReloadData()`: Reloads data related to the appointment such as deal details and appointment charges.
- `btn_cancel_Click(sender As Object, e As EventArgs)`: Handles the cancellation of the appointment and updates the deal status in the database.
- `btn_upcoming_Click(sender As Object, e As EventArgs)`: Handles the click event of the "Upcoming" button.
- `btn_completed_Click(sender As Object, e As EventArgs)`: Handles the click event of the "Completed" button.
- `btn_reschedule_Click(sender As Object, e As EventArgs)`: Handles the click event of the "Reschedule" button and displays reschedule options.

3.1.3 Other Notes

- The code uses SQL queries to retrieve and update data from the database.
- It handles cancellation fees based on the time difference between the current time and the booked time.
- The cancellation confirmation dialog prompts the user to confirm the cancellation.

3.2 user_appointments.vb

3.2.1 Variable Declarations

- `panelArray()` (Array of System.Windows.Forms.Panel): Array to hold panel controls.

3.2.2 Methods

- `Button2_Click(sender As Object, e As EventArgs)`: Handles the click event of Button2, changing button colors and calling the `payment()` method.
- `Button1_Click(sender As Object, e As EventArgs)`: Handles the click event of Button1, changing button colors and calling the `upcoming()` method.
- `Button3_Click(sender As Object, e As EventArgs)`: Handles the click event of Button3, changing button colors and calling the `completed()` method.
- `Panel_Click1(sender As Object, e As EventArgs)`: Handles the click event of panel controls for upcoming appointments, storing deal details and displaying appointment details.
- `Panel_Click2(sender As Object, e As EventArgs)`: Handles the click event of panel controls for pending payments, storing deal details and displaying payment details.
- `CompletedPanelClick(sender As Object, e As EventArgs)`: Handles the click event of panel controls for completed appointments, storing deal details and displaying feedback form.
- `spawnDivs(i As Integer, providerName As String, location As String, CostNum As Integer, Schedule As String, y As Integer, DealId As Integer)`: Dynamically creates panel controls for upcoming appointments.
- `spawnDivsFeedback(i As Integer, providerName As String, location As String, CostNum As Integer, Schedule As String, y As Integer, DealId As Integer)`: Dynamically creates panel controls for completed appointments with a feedback button.
- `spawnDivsWithButton(splitContainerArray As SplitContainer(), i As Integer, providerName As String, location As String, CostNum As Integer, Schedule As String, y As Integer, DealId As Integer)`: Dynamically creates panel controls for pending payments with a payment button.
- `Button_Click(sender As Object, e As EventArgs)`: Handles the click event of payment buttons in the lower panel (pending payments).
- `upcoming()`: Retrieves upcoming appointments from the database and dynamically creates panel controls for each appointment.
- `payment()`: Retrieves pending payment details from the database and dynamically creates panel controls for each payment with a payment button.
- `completed()`: Retrieves completed appointments from the database and dynamically creates panel controls for each appointment with a feedback button.
- `user_appointments_Load(sender As Object, e As EventArgs)`: Handles the load event of the form, initializing the display of upcoming appointments.

3.2.3 Other Notes

- The code uses SQL queries to retrieve data from the database asynchronously.
- It dynamically creates UI controls based on the retrieved data to display upcoming appointments, pending payments, and completed appointments.
- Each panel control is associated with a click event to handle user interactions.
- Feedback and payment buttons are added to panels for completed appointments and pending payments, respectively, enabling users to provide feedback or proceed with payment.

3.3 user_feedback.vb

3.3.1 Imports

```
Imports Microsoft.Data.SqlClient
Imports System.Configuration
```

3.3.2 Class Definition: user_feedback

```
Public Class user_feedback
```

3.3.3 Variables

- `dealID` (Integer) - Stores the deal ID retrieved from `Module_global.Appointment_Det_DealId`.
- `currentRating` (Double) - Stores the current rating given by the user.
- `clickCount1`, `clickCount2`, `clickCount3`, `clickCount4`, `clickCount5` (Integer) - Store the number of clicks on each star rating.
- `initialText` (String) - Stores the initial text shown in the feedback textbox.
- `connectionString` (String) - Stores the connection string retrieved from `ConfigurationManager.Co`

3.3.4 Event Handlers and Methods

- `PictureBox1_Click` to `PictureBox5_Click` - Event handlers for clicking on star ratings. Update the `currentRating` variable accordingly.
- `UpdateStars()` - Updates the star images based on the current rating.
- `feedback_Enter` - Event handler to clear the feedback textbox when it is clicked if it matches the initial text.
- `feedback_Leave` - Event handler to restore the initial text when the user leaves the textbox without entering anything.
- `InitializeFeedbackTextBox()` - Initializes the feedback textbox with the initial text and sets the text color to gray.
- `Label3_Click` - Event handler for submitting the feedback. It checks for existing reviews, updates or inserts the review into the database accordingly.
- `user_feedback_Load` - Event handler for loading the form. It sets the user's name in `Label4` and initializes the feedback textbox.

3.4 user_profile.vb

3.4.1 Class Definition: user_profile

```
Public Class user_profile
```

3.4.2 Variables

- `connectionString` (String) - Stores the connection string for connecting to the database.
- `edit_enable` (Boolean) - Indicates whether editing mode is enabled or not.

3.4.3 Event Handlers and Methods

- `User_profile_Load` - Event handler for loading the user profile form. Loads user data from the database and sets the profile picture, name, email, and phone number.
- `Changepic_pb_Click` - Event handler for changing the profile picture. Allows the user to select an image file and compresses it if needed.
- `CompressImageIfNeeded` - Method to compress an image if its size is greater than 1MB.
- `GetImageSize` - Method to get the size of an image in bytes.
- `GetEncoderInfo` - Method to get JPEG encoder information.
- `Edit_btn_Click` - Event handler for saving changes made to the user profile. Updates the user's information in the database.
- `greeting_label_Click` - Event handler for clicking on the greeting label.

3.5 user_provider_chats.vb

3.5.1 Class Definition: user_provider_chats

```
Public Class user_provider_chats
```

3.5.2 Variables

- `messages` (List of Tuples) - Stores the messages exchanged between users.
- `connectionString` (String) - Stores the connection string for connecting to the database.
- `user_role` (String) - Indicates the role of the user (provider or customer).
- `userId` (Integer) - Stores the user ID.
- `dealId` (Integer) - Stores the deal ID.
- `roomId` (Integer) - Stores the room ID.
- `header` (String) - Stores the header for the chat window.
- `timer` (Timer) - Timer object for periodic updates.

3.5.3 Event Handlers and Methods

- `GetSqlConnection` - Method to get SQL connection object.
- `LoadRoomsFromDatabase` - Method to load chat rooms from the database.
- `LoadMessagesFromDatabase` - Method to load messages from the database.
- `InsertMessageIntoDatabase` - Method to insert a message into the database.
- `PopulateRooms` - Method to populate chat rooms.
- `user_chats_Load` - Event handler for loading the user chats form.
- `timer_Tick` - Event handler for the timer tick event.
- `Button_Click` - Event handler for button click event.
- `sendTextBox_KeyDown` - Event handler for key down event in the send text box.
- `sendButton_Click` - Event handler for send button click event.
- `PrintMessagesBetweenUsers` - Method to print messages between users.
- `MainForm_FormClosed` - Event handler for form closed event.
- `Panel1_Paint` - Event handler for panel paint event.
- `chat_list_Paint` - Event handler for chat list paint event.
- `senderName_Click` - Event handler for sender name click event.

3.6 user_search.vb

3.6.1 Class: `user_search`

Variables

- `reviews`: Dictionary to store reviews, where the key is the provider ID, and the value is a tuple containing the sum of ratings and the count of reviews.
- `rating_prov`: Dictionary to store the average rating for each provider.
- `selected_service`: String representing the selected service.
- `selected_location`: String representing the selected location.
- `selected_sort`: String representing the selected sorting criteria.
- `binaryImageData`: Byte array representing binary image data.
- `is_null_image`: Integer representing whether the image is null (0 or 1).
- `providers`: List to store provider details as `Entry` structures.
- `temp_providers`: Temporary list to store provider details before adding them to `providers`.

3.6.2 Methods

- `user_search_Load`: Event handler for form load. Populates combo boxes, fetches data from the database, and populates provider details.
- `MakePictureBoxRound`: Makes the picture box round by adding an ellipse path.
- `TextBox1_GotFocus`: Event handler for when the search text box gains focus. Clears placeholder text.
- `TextBox1_LostFocus`: Event handler for when the search text box loses focus. Displays placeholder text if empty.
- `PopulateTable`: Populates the table with provider details.
- `LoadProviders`: Loads provider details from the database.
- `Button1_Click`: Event handler for the search button click. Filters providers based on search criteria and populates the table.
- `Button2_Click`: Event handler for the book slot button click. Navigates to the book slots page.
- `ComboBox1_SelectedIndexChanged`: Event handler for the service combo box selection change.
- `ComboBox2_SelectedIndexChanged`: Event handler for the location combo box selection change.
- `ComboBox3_SelectedIndexChanged`: Event handler for the sort by combo box selection change.

3.6.3 Structure: `Entry`

- `providerID`: Integer representing the provider ID.
- `providerName`: String representing the provider name.
- `service`: String representing the service offered by the provider.
- `location`: String representing the location of the provider.
- `cost`: Integer representing the cost per hour for the service.
- `rating`: String representing the rating of the provider.
- `RadioButton`: RadioButton control associated with the provider entry.

3.7 user_template.vb

3.7.1 Imports

- `System.Configuration`: Provides types that support reading configuration files.
- `Microsoft.Data.SqlClient`: Provides a data provider for Microsoft SQL Server.

3.7.2 Class: `user_template`

Methods

- `Form1_Load`: Event handler for form load. Loads the UserHome form into the split container's panel.
- `switchPanel`: Method to switch the panel in the split container.
- `Reset_Buttons`: Resets the color of all navigation buttons.
- `Home_btn_Click`: Event handler for the home button click. Sets the color of the home button and switches to the UserHome panel.
- `Search_btn_Click`: Event handler for the search button click. Sets the color of the search button and switches to the user_search panel.
- `Appointments_btn_Click`: Event handler for the appointments button click. Sets the color of the appointments button and switches to the user_appointments panel.
- `Profile_btn_Click`: Event handler for the profile button click. Sets the color of the profile button and switches to the user_profile panel.
- `Chats_btn_Click`: Event handler for the chats button click. Sets the color of the chats button and switches to the user_provider_chats panel.
- `Help_btn_Click`: Event handler for the help button click. Sets the color of the help button and switches to the support_chat panel.
- `Feedback_btn_Click`: Event handler for the feedback button click. Sets the color of the feedback button and switches to the FeedbackForm panel.
- `Logout_btn_Click`: Event handler for the logout button click. Closes the current form and shows the Login form.

3.8 UserHome.vb

3.8.1 Variables

- `map`: Dictionary to store provider information based on service type.
- `reviews`: Dictionary to store review information with provider IDs.
- `provider_locations`: Dictionary to store provider locations.
- `provider_keys`: Dictionary to store provider names.
- `provider_service`: Dictionary to store provider services.
- `buttonLoc`: Integer variable to store the location of buttons.
- `user_name`: String variable to store the username of the current user.
- `rating_prov`: Concurrent dictionary to store provider ratings.
- `labels`, `layout_panels`, `buttons`: Lists to store labels, panels, and buttons respectively.

3.8.2 Methods

- `InitializeForm`: Method to initialize the form by clearing existing components and dictionaries.
- `ReloadData`: Method to reload data by initializing the form and loading tasks asynchronously.
- `LoadTasks`: Asynchronous method to load tasks including retrieving reviews, user information, and provider information from the database.
- `UserHome_Load`: Event handler for the form's Load event. Calls `LoadTasks` method to load data.
- `btnViewMore_Click`: Event handler for the "View All" button click event. Navigates to another page to view all providers.
- `tileControl_Click`: Event handler for the provider tile click event. Clears existing controls and shows book slots for the selected provider.
- `FindParentProvTile`: Method to find the parent Prov_tile control recursively.

3.9 User_Signup.vb

3.9.1 Variables

- `code`: Integer variable to store the OTP (One-Time Password) generated for user verification.

3.9.2 Methods

- `User_Signup_Load`: Event handler for the form's Load event. Clears error label on form load.
- `Showpassword_cb_CheckedChanged`, `Showcnfpassword_cb_CheckedChanged`: Event handlers for checkbox state change to show/hide password characters.
- `Login_btn_Click`, `Back_btn_Click`: Event handlers for button clicks to navigate to login page or back to landing page.
- `SendEmail`: Method to send OTP via email using SMTP protocol.
- `SendOTP_btn_Click`: Event handler for sending OTP button click. Generates OTP and sends it via email.
- `Register_btn_Click`: Event handler for register button click. Validates user inputs, verifies OTP, and inserts user data into the database.

3.10 ViewAllUser.vb

This form is responsible for displaying a list of providers based on certain criteria, such as service type, rating, and location. It interacts with a database to retrieve provider data and presents this information in a scrollable panel of "viewMore" tiles, which allow users to click and view more details about each provider.

File Purpose:-

The ViewAllUser form serves as a user interface component that integrates with your application's navigation and data retrieval logic. It acts as a page to display all providers of a service type and also allows filtering or sorting the entries on the basis of rating, provider name and location.

Key Functions:-

InitializeForm() :- Clears the inner panel and resets global variables used for layout calculations.

ReloadData() :- Initialises the form, loads user data, and refreshes the view whenever we navigate from other page to this page. Calls InitializeForm() to prepare the form for data reloading.Dynamically adds the form to a panel and displays it. Invokes LoadTasks() to retrieve and display provider data.

LoadTasks() :- Retrieves provider data from a database and populates the inner panel with "viewMore" tiles. Queries a database to retrieve provider information based on the specified service type . Populates and sorts provider dictionary based on retrieved data.

viewMore_TileClicked :- Handles the event when a provider tile ("viewMore" tile) is clicked. Retrieves the provider ID from the clicked tile. Navigates to another page (Book_slots) for further interaction based on the selected provider.

ComboBox Event Handlers :- Handles selection changes in comboboxes (ComboBox1 for sorting criteria and ComboBox2 for location filtering). Dynamically updates the displayed provider tiles (innerPanel.Controls) based on selected criteria.

4 Provider Related Files

4.1 provider_appointment_details.vb

4.1.1 Fields

- `dealID`: Integer representing the deal ID.
- `startTime`: TimeSpan representing the start time of the appointment.
- `firstDate`: DateTime representing the date of the appointment.
- `bookDate`: DateTime representing the booking date.
- `advance`: Double representing the advance payment amount.
- `slots`: Integer representing the number of slots booked.
- `advancePercentage`: Integer representing the percentage of advance payment.
- `user`: String representing the user.
- `costPerHour`: Decimal representing the cost per hour.
- `status`: Integer representing the status of the appointment.
- `email`: String representing the email address.
- `COMPLETED`: Integer representing the completed status.
- `CANCELLED`: Integer representing the cancelled status.
- `user_id`: Integer representing the user ID.
- `provider_id`: Integer representing the provider ID.
- `connectionString`: String representing the connection string for SQL Server.
- `provider`: String representing the provider.
- `time`: String representing the time slots.

4.1.2 Methods

- `OnVisibleChanged(e As EventArgs)`: Overrides the `OnVisibleChanged` method to load data when the form is visible.
- `MakeChatVisible()`: Makes the chat window visible.
- `loadData()`: Loads data related to the appointment.
- `btn_Cancel_Click()`: Cancellation policy after provider cancels the service.
- `btn_Complete_Click()`: Completes the service from provider side.
- `WaitForVariableChangeOrTimeoutAsync(timeoutMilliseconds As Integer)`: Asynchronously waits for a variable change or timeout.

4.2 provider_appointments.vb

4.2.1 Fields

- `panelArray`: Array of Panel controls.

4.2.2 Methods

- `Button2_Click(sender As Object, e As EventArgs)`: Event handler for Button2 click event.
- `Button1_Click(sender As Object, e As EventArgs)`: Event handler for Button1 click event.
- `Button_ClickCompleted(sender As Object, e As EventArgs)`: Event handler for completing an appointment.
- `Button_Click(sender As Object, e As EventArgs)`: Event handler for button click.
- `Panel_Click(sender As Object, e As EventArgs)`: Event handler for panel click.
- `Panel_Click2(sender As Object, e As EventArgs)`: Event handler for panel click.
- `spawnDivs(i As Integer, providerName As String, location As String, CostNum As Integer, Schedule As String, y As Integer, DealId As Integer)`: Function to spawn appointment panels.
- `Payment()`: Function to handle payments.
- `spawnUpcomingDiv(splitContainerArray As SplitContainer(), i As Integer, UserName As String, location As String, CostNum As Integer, Schedule As String, y As Integer, DealId As Integer)`: Function to spawn upcoming appointment panels.
- `upcoming()`: Function to handle upcoming appointments.
- `provider_appointments_Load(sender As Object, e As EventArgs)`: Event handler for form load event.

4.3 provider_dashboard.vb

4.3.1 Class: provider_dashboard

This class manages the provider dashboard.

4.3.2 Methods

- `provider_dashboard_Load(sender As Object, e As EventArgs)`: Event handler for form load event.
- `ChangeCellBackgroundColor(rowIndex As Integer, colIndex As Integer, backColor As Color)`: Changes the background color of a cell in a table layout panel.
- `Chart_Init()`: Initializes the chart with data.
- `GetTimeTable()`: Retrieves the provider's timetable from the database.
- `SetWorkingDayColors(dates As String, workingDays As String)`: Sets the background color of cells based on the provider's working hours.

4.4 provider_feedback_view.vb

4.4.1 Class Declaration

[language=[Sharp]C] Public Class provider$_f eedback_v iew$

4.4.2 Variables

- `connectionString`: A string variable to store the connection string retrieved from the application's configuration file.

4.4.3 Methods

- `provider_feedback_view_Load`: Event handler for the form's Load event. Calls the `LoadReviews` method to load reviews when the form loads.
- `LoadReviews`: Method to load reviews from the database and display them in a ListView.

4.4.4 ListView Configuration

- Configures the ListView control (`listViewReviews`) to display reviews and ratings.

4.4.5 SQL Query

- Constructs an SQL query to retrieve review text and ratings from the `review` table.

4.4.6 Database Interaction

- Opens a connection to the database using the connection string.
- Executes the SQL query to retrieve review data.
- Reads data from the SqlDataReader and adds it to the ListView control.

4.4.7 Error Handling

- Displays a message box with an error message if an exception occurs during database interaction.

4.5 provider_info.vb

4.5.1 Variables

- `user`, `provider`: Integer variables to store user and provider IDs.
- `ProviderName`, `user_name`: String variables to store provider name and user name.
- `binaryImageData`: Byte array to store profile image data.
- `availability`: 2D array to store availability schedule.
- `BookedList`: List to store booked slots.
- `Avaiability_String`: String variable to store availability schedule as string.
- `cost_per_hour`, `is_null_image`: Integer variables for cost per hour and null image flag.

4.5.2 Methods

- `Book_slots_load`: Event handler for the form's Load event. Runs the `LoadData` function asynchronously.
- `LoadData`: Asynchronous method to load data from the database including provider and user details.
- `SplitContainer1_Panel1_Paint`, `providerProfilePicture_Click`: Event handlers for UI events (paint and click).

4.6 provider_notifications.vb

This page displays the provider all the advance payments,complete payments,cancelled payments and the feedback and rating given by the users to that provider. Provider can also set filters to see the feedbacks of particular ratings only. There are 4 buttons in this page to view advance payments,complete payments,cancelled bookings,reviews respectively and flowlayout panel in which the data is displayed. These are the following functions used in the page : provider_notifications_Load: On opening the page in the provider_template it displays according to the load function which displays the reviews by default to the provider.The function is same as the button4_click.
Button1_Click(sender As Object, e As EventArgs): This is the method to fetch and display the details of all the advanced payments(50%) of the customers to that provider. SQl query is executed and the data is displayed in the panel using dynamically created elements.The name,adress,time of transaction,deal amount were displayed.
button2_click(sender As Object, e As EventArgs): This is the method to fetch and display the details of all the complete payments of the customers to that provider. SQl query is executed and the data is displayed in the panel using dynamically created elements.The name,adress,time of transaction,deal amount were displayed.
button3_click(sender As Object, e As EventArgs): This is the method to fetch and display the details of all the cancelled appointments of that provider. Dynamically created elements shows the name,adress,a button for the provider to pay refund. On clicking on pay refund button its makes the transaction to customer to pay the refund and mail will be sent to the customer regarding the payment details.
button4_click(sender As Object, e As EventArgs): This is the method to fetch and display the reviews and ratings using SQL query by calling the showAllReviews() method and loads it in the flow layout panel.
ShowAllReviews(): Fetches and displays all the reviews and rating along with username to that particular provider.
RatingComboBox_SelectedIndexChanged(): Updates the data in panel according to the selected value in the combo box(filter) on applying filter.
ApplyComboBoxStyling(): This method used to styles and format the combobox used to apply filter on reviews based on rating.

4.7 provider_template.vb

4.7.1 Class Declaration

```
Public Class provider_template
```

4.7.2 Methods

4.7.3 `provider_template_Load` Method

```
Private Sub provider_template_Load(sender As Object, e As EventArgs) _
    Handles MyBase.Load
```

- **Description:** This method is an event handler for the load event of the `provider_template` form. It loads the provider dashboard form (`provider_dashboard`) when the application starts.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.7.4 `ShowForm` Method

```
Public Sub ShowForm(form As Form)
```

- **Description:** This method clears the panel and adds a new form to it. It sets the properties of the form such as `TopLevel`, `FormBorderStyle`, `Dock`, and then displays it.
- **Parameters:** `form` (Form) - The form to be displayed.
- **Access:** Public

4.7.5 Navigation Button Click Event Handlers

There are several event handlers for the click events of navigation buttons (`Profile_Navi_btn_Click`, `Dashboard_Navi_btn_Click`, etc.). Each event handler changes the background color of the clicked button and shows the corresponding form using the `ShowForm` method.

4.7.6 `Logout_btn_Click` Method

```
Private Sub Logout_btn_Click(sender As Object, e As EventArgs) _
    Handles logout_btn.Click
```

- **Description:** This method is an event handler for the click event of the logout button. It closes the current form and shows the login form (`Login`).
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.7.7 Provider_Signup.vb

4.7.8 Class Declaration

```
Public Class Provider_Signup
```

4.7.9 Fields

```
Dim code As Integer
```

4.7.10 Methods

4.7.11 `Provider_Signup_Load` Method

```
Private Sub Provider_Signup_Load(sender As Object, e As EventArgs) _
    Handles MyBase.Load
```

- **Description:** This method is an event handler for the load event of the `Provider_Signup` form. It initializes the error label.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.7.12 `Showpassword_cb_CheckedChanged` and `Showcnfpassword_cb_CheckedChanged` Methods

These methods are event handlers for the checked changed event of the show password checkboxes. They toggle the visibility of the password characters.

4.7.13 `Login_btn_Click` and `Back_btn_Click` Methods

These methods are event handlers for the click events of the login and back buttons respectively. They close the current form and show the login form (`Login`) or the landing page form (`Landing`).

4.7.14 `SendEmail` Method

```
Private Sub SendEmail(randomNumber As Integer)
```

- **Description:** This method sends an email with a generated OTP to the provided email address.
- **Parameters:** `randomNumber` (Integer) - The OTP to be sent.
- **Access:** Private

4.7.15 `SendOTP_btn_Click` Method

```
Private Sub SendOTP_btn_Click(sender As Object, e As EventArgs) _
    Handles sendOTP_btn.Click
```

- **Description:** This method generates an OTP and calls the `SendEmail` method to send it via email.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.7.16 `Register_btn_Click` Method

```
Private Sub Register_btn_Click(sender As Object, e As EventArgs) _
    Handles register_btn.Click
```

- **Description:** This method validates user input, generates an OTP, sends it via email, and registers a new provider in the database.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.7.17 `Panel2_Paint` Method

```
Private Sub Panel2_Paint(sender As Object, e As PaintEventArgs) _
    Handles Panel2.Paint
```

- **Description:** This method is an event handler for the paint event of Panel2. It is currently empty.
- **Parameters:** `sender` (Object), `e` (PaintEventArgs)
- **Access:** Private

4.8 Provider_Profile_page.vb

4.8.1 Class Declaration

```
Public Class Provider_Profile_page
```

4.8.2 Fields

```
Public editprovprof As Boolean = False
```

4.8.3 Methods

4.8.4 `Provider_Profile_page_Load` Method

```
Private Sub Provider_Profile_page_Load(sender As Object, e As EventArgs) _
    Handles MyBase.Load
```

- **Description:** This method is an event handler for the load event of the `Provider_Profile_page` form. It loads the provider's information from the database and displays it on the form.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.8.5 `Edit_profile_btn_Click` Method

```
Private Sub Edit_profile_btn_Click(sender As Object, e As EventArgs) _
    Handles Edit_profile_btn.Click
```

- **Description:** This method is an event handler for the click event of the "Edit Profile" button. It raises the `EditProfileClicked` event and closes the current form.
- **Parameters:** `sender` (Object), `e` (EventArgs)
- **Access:** Private

4.8.6 `SetStarRating` Method

```
Private Sub SetStarRating(rating As Double)
```

- **Description:** This method sets the star rating based on the provided rating value.
- **Parameters:** `rating` (Double) - The rating value.
- **Access:** Private

4.8.7 `SetStarImage` Method

```
Private Sub SetStarImage(index As Integer, image As Image)
```

- **Description:** This method sets the image of a star based on the provided index and image.
- **Parameters:** `index` (Integer) - The index of the star. `image` (Image) - The image of the star.
- **Access:** Private

4.8.8 `LoadWorkingHours` Method

```
Private Sub LoadWorkingHours(workingHour As String)
```

- **Description:** This method loads the working hours of the provider and marks the checkboxes in the slot matrix table layout accordingly.
- **Parameters:** `workingHour` (String) - The working hours of the provider.
- **Access:** Private

4.9 Provider_Signup.vb

4.9.1 Explanation

1. **Form Load Event Handler:** Initializes the form by clearing the error label.
2. **Showpassword_cb CheckedChanged Event Handler:** Toggles the visibility of the password characters based on the checkbox state.
3. **Showcnfpassword_cb CheckedChanged Event Handler:** Toggles the visibility of the confirm password characters based on the checkbox state.
4. **Login_btn Click Event Handler:** Closes the current form and opens the login form.
5. **Back_btn Click Event Handler:** Closes the current form and returns to the landing page.
6. **SendEmail Method:** Sends an email with an OTP to the provided email address using SMTP.
7. **SendOTP_btn Click Event Handler:** Generates and sends an OTP to the provided email address.
8. **Register_btn Click Event Handler:** Validates the user input, checks the OTP, and inserts the user data into the database if all conditions are met.

5 Payment related files

5.1 pending_payment.vb

5.1.1 Explanation

1. **Imports:** The code imports necessary libraries for database operations, email sending, and threading.
2. **Class Declaration:** The class `pending_payment` is declared.
3. **Variables:** Various variables are declared, including `dealID`, `provider`, `slots`, `costPerHour`, etc.
4. **OnVisibleChanged Method:** Overrides the `OnVisibleChanged` method to reload data when the form is made visible.
5. **ReloadData Method:** Retrieves data from the database related to the deal and updates UI elements.
6. **WaitForVariableChangeOrTimeoutAsync Method:** Asynchronously waits for a variable change or timeout.
7. **sendEmail Method:** Sends an email using SMTP.
8. **btn_pay_Click Event Handler:** Handles the click event of the pay button. Initiates payment process, updates deal status, sends email, etc.
9. **btn_upcoming_Click and btn_completed_Click Event Handlers:** Handle click events of the upcoming and completed buttons, respectively.

5.1.2 Conclusion

This document provided an overview and explanation of the provided VB.NET code.

5.2 payments.vb

5.2.1 Explanation

1. **Imports:** The code imports necessary libraries for various operations such as email sending, database connectivity, PDF generation, etc.
2. **Global Variables:** Global variables are defined to hold user identifiers, database connection strings, and payment-related information.
3. **Payment_load Method:** Handles form load event and initializes global variables.
4. **payButton_Click Event Handler:** Handles the click event of the pay button. Initiates payment process, sends OTP via email, updates user balances, generates receipts, etc.
5. **sendEmail Method:** Sends an email with OTP for payment authentication.
6. **ApplyCashback Method:** Applies cashback to the user's account based on a random probability.
7. **payments_Load Method:** Handles form load event and initializes form controls with payment information.

8. **addButton_Click Event Handler:** Handles the click event of the add button to add money to the user's wallet. Sends OTP via email, updates user balance, generates receipts, etc.

5.3 otp-auth.vb

5.3.1 Explanation

1. **Properties:** The class defines a property named `InputValue` which returns the value entered in a text box.

2. **button1_Click Event Handler:** Handles the click event of the OK button. It sets the `DialogResult` to OK, stores the input value from the text box, clears the text box, and closes the form.

5.3.2 Conclusion

This document provided an overview and explanation of the provided VB.NET code.

5.4 captcha.vb

5.4.1 Explanation

1. **Form Load Event Handler:** Initializes the form by calling the `DrawCaptcha` method to generate a CAPTCHA image.

2. **DrawCaptcha Method:** Generates a CAPTCHA image with random text, noise, and lines. Displays the image in a picture box.

3. **GenerateRandomText Method:** Generates a random string of characters for the CAPTCHA text.

4. **Button1_Click Event Handler:** Validates the entered text against the CAPTCHA text. If matched, displays a success message and closes the form. If not matched, displays a failure message, clears the text box, generates a new CAPTCHA text, and redraws the CAPTCHA image.

5.4.2 Conclusion

This document provided an overview and explanation of the provided VB.NET code.

6 Miscellaneous

6.1 Prov_tile.vb

It creates a custom UI element which is used for displaying all providers in the User Home Page. Public Class Prov_tile - It initializes the class for the provider tile.
Public Sub New(provider As Int32, providerName As String, loc As String, rating As Double, itemImage As Image) - It is the constructor for the class Prov _tile.
Input parameters - provider_id, provider name, provider location, provider rating, provider image.
Private Sub UserControl_MouseEnter(sender As Object, e As EventArgs) - Changes the color of the tile on hovering mouse over it.
Private Sub UserControl_MouseLeave(sender As Object, e As EventArgs) - Restores the color of the tile.

6.2 Login.vb

Login_Load: Initializes the form and clears the error label.
Showpassword_cb_CheckedChanged: Toggles password visibility in the password textbox.
Provider_btn_Click: Handles provider login. Validates user input, checks credentials from the database, and navigates to the provider template if login is successful.
User_btn_Click: Handles user login. Validates user input, checks credentials from the database, and navigates to the user template if login is successful.
Register_btn_Click: Closes the current form and opens the Landing form for user registration.
Admin_ll_LinkClicked: Navigates to the Admin_Login form.
LoadRoomsFromDatabase: Loads chat room information from the database based on the user's role (provider or customer).
LinkLabel1_LinkClicked: Navigates to the Forgot_password form.
Properties: login_status: Boolean field to track the login status.

6.3 Landing.vb

6.4 Forgot_password.vb

send_Click: It first validates the email address ,if the email address is valid and either the "customer" or "provider" checkbox is selected, it generates a random OTP and sends it to the entered email address using the SendEmail function.
proceed_Click: It validates whether the entered OTP matches the randomly generated OTP. If it matches, it displays the panel for resetting the password; otherwise, it displays an error message.
save_Click: It checks if the password and confirm password fields are not empty and match. Then, it checks the strength of the password and asks the user if they are satisfied with the strength. If yes, it updates the password in the database for the corresponding user (either customer or provider).
IsValidEmail: This function validates whether the entered email address exists in the database for either a customer or provider.
SendEmail: This function is responsible for sending an email containing the OTP to the provided email address. It configures the SMTP client with Gmail's SMTP server settings and sends an email with the OTP as the message body. If the email is sent successfully, it returns True; otherwise, it returns False.

6.5 FeedbackForm.vb

The FeedbackForm class facilitates the submission and updating of feedback within a software application. It allows users, either customers or providers, to rate their experience and provide additional comments if desired. The class includes methods for submitting and updating feedback, event handlers for user interactions, and properties for managing form elements.

Methods:

FeedbackForm_Load: Initialises the feedback form, checking if feedback has been previously submitted and populating the form accordingly. Star_Click: Event handler for setting the rating when a star is clicked.
txtFeedback_Enter and txtFeedback_Leave: Event handlers for managing default text in the feedback textbox.
btnSend_Click: Event handler for submitting or updating feedback, including error handling and database insertion or update.
GetNextFeedbackId: Retrieves the next available feedback ID from the database.
InsertFeedback: Inserts a new feedback entry into the database.
UpdateFeedback: Updates existing feedback entry in the database.

Properties:

stars: List of PictureBox controls representing the star rating.
connectionString: Connection string to the database.
senderId: ID of the user submitting feedback.
userType: Indicates whether the user is a customer or provider.
Rating: Current rating provided by the user.
sendername: Name of the user submitting feedback.

Unique Feedback Submission:

To ensure fairness and accuracy in feedback submission, the class implements a mechanism to prevent multiple submissions of feedback from the same user. This prevents scenarios where a user may excessively submit feedback with similar ratings, skewing the overall feedback data.

Feedback Update Feature:

In addition to unique feedback submission, the class allows users to update their feedback if their opinions change over time or if newer versions of the application alter their experience. Users have the option to modify their ratings and comments, providing an up-to-date reflection of their satisfaction with the application.

6.6 EditProfilePage.vb

Methods:

Edit_profile_btn_Click: Handles the click event of the "Edit_profil_btn" button. Raises the EditProfileClicked event.
SetStarRating:Sets the star rating based on the provided rating value. Determines the star rating based on the integer and fractional parts of the rating value, and updates the star images accordingly.
SetStarImage: Sets the image of a star PictureBox based on the provided index. Sets the image of the PictureBox corresponding to the specified index to the provided image.
LoadWorkingHours: Loads the working hour's data into the slot_matrix_tablelayout. Parses the binary workingHour string, updates the checkboxes in the slot_matrix_tablelayout based on the parsed data.

Properties: Edit_profile_btn: Button for editing the provider's profile.
editprovprof: Boolean flag indicating whether the provider profile is being edited.
Name_label: Label for displaying the provider's name.
email_label: Label for displaying the provider's email.

Service_label: Label for displaying the provider's service.
rate_label: Label for displaying the provider's rate per hour.
contact_label: Label for displaying the provider's contact number.
profilepic_pb: PictureBox for displaying the provider's profile picture.
slot_matrix_tablelayout: TableLayoutPanel for displaying the provider's working hours.

Events:
EditProfileClicked: Event raised when the edit profile button is clicked.

6.7 viewMore.vb

The viewMore UserControl is a custom user interface component used to display provider information as a tile. It encapsulates details about a provider, such as their username, locations, and rating, and provides event handling for interaction when the tile is clicked.
Constructor :-Initialises the viewMore control with specified parameters: index: Unique identifier for the tile. startX, startY: Initial position coordinates of the tile. locations: List of locations associated with the provider. rating: Average rating of the provider.
Properties :- Index -¿ Retrieves the index of the viewMore tile.
Username -¿Gets or sets the username displayed on the tile.
Locations -¿ Gets or sets the list of locations associated with the provider.
Rating -¿Gets or sets the average rating of the provider.
Events -¿ TileClicked Event
Triggered when the viewMore tile is clicked.
On clicking on tile it navigates to the page to book slots of that provider

6.7.1 Fields

- `_startX`: Integer representing the starting X-coordinate of the control.
- `_startY`: Integer representing the starting Y-coordinate of the control.
- `_index`: Integer representing the index of the location.
- `_locations`: List of String representing the locations associated with the view.
- `_rating`: Double representing the rating of the location.

6.7.2 Events

- `TileClicked`: EventHandler(Of Integer) - Raised when the user clicks on any part of the control.

6.7.3 Properties

- `Index`: ReadOnly Property - Gets the index of the location.
- `Username`: Property - Gets or sets the username associated with the view.
- `Locations`: Property - Gets or sets the list of locations associated with the view.
- `Rating`: Property - Gets or sets the rating of the location.

6.7.4 Methods

- None

6.7.5 Events

- `viewMore_Click`: Handles the click event for the entire control.
- `usernamelabel_Click`: Handles the click event for the username label.
- `PictureBox1_Click`: Handles the click event for the picture box.
- `label2_Click`: Handles the click event for label2.
- `label3_Click`: Handles the click event for label3.

6.7.6 Custom Drawing

- `OnPaint`: Overrides the OnPaint method to customize the appearance of the control by drawing a rounded rectangle with a border.

6.8 appointment_history.vb

6.8.1 Class Declaration

6.8.2 Fields

- `data`: List of tuples containing appointment information.

6.8.3 Events

- None

6.8.4 Methods

- `Appointment_history_Load`: Handles the form load event and retrieves appointment data from the database.
- `Panel_Click`: Handles the click event for appointment panels and displays detailed information.
- `Label_MouseEnter`: Handles the mouse enter event for appointment panels and changes their color.
- `Label_MouseLeave`: Handles the mouse leave event for appointment panels and changes their color back to default.
- `PictureBox1_Click`: Handles the click event for the picture box to hide detailed information and display the appointment history.

Custom Drawing

- None

6.8.5 Dependencies

- `Microsoft.Data.SqlClient`: Used for database connectivity.
- `Org.BouncyCastle.Utilities`: Used for utility functions.

7 Appointments

7.1 Reschedule_Slots.vb

7.2 Book_slots.vb

7.2.1 Class Members

- `user`: Integer representing the user ID.
- `provider`: Integer representing the provider ID.
- `ProviderName`: String representing the name of the service provider.
- `user_name`: String representing the username of the user.
- `binaryImageData`: Byte array representing the profile image data of the user.
- `availability`: 2D array representing the availability of slots (7 days, 12 slots per day).
- `BookedList`: List of booked slots.
- `Avaiability_String`: String representing the availability of slots in a compact format.
- `cost_per_hour`: Integer representing the cost per hour of service.
- `is_null_image`: Integer flag indicating whether the user has a profile image.

7.2.2 Methods

- `Book_slots_load`: Asynchronously loads data required for booking slots.
- `LoadData`: Asynchronously retrieves data from the database.
- `Make_Schedule_Table`: Asynchronously generates the schedule table based on availability data.
- `PopulateScheduleTableAsync`: Asynchronously populates the schedule table with time slots.
- `AddControlToScheduleTableAsync`: Asynchronously adds control (button) to the schedule table.
- `TimeSlot_Click`: Handles click events for time slots.
- `MakePictureBoxRound`: Asynchronously makes the profile picture box round.
- `ImageFromBinary`: Converts binary image data to an Image object.
- `WaitForVariableChangeOrTimeoutAsync`: Asynchronously waits for a variable change or timeout.
- `Book_Btn_Click`: Handles click events for the book button.
- `Back_Btn_Click`: Handles click events for the back button.

7.2.3 Conclusion

The `Book_slots` class provides functionality for users to book slots with a service provider. It handles loading data, generating the schedule table, managing slot bookings, and handling user interactions.

8 Communication Module

8.1 user_provider_chats.vb

The user_provider_chats class is responsible for managing user-provider chats within the application. It handles loading chat rooms and messages from the database, populating the UI with chat room buttons, sending and displaying messages, and managing user interactions with the chat interface. It helps to interact before making deal.

8.1.1 Variables

1. userId: The ID of the user or provider
2. user_role: The role of the user (either "customer" or "provider")
3. room: The ID of the chat room where the message is sent
4. dealId: The ID of the deal associated with the message (if any)
5. header: Represents the header text for the chat room interface
6. connection_string: Represents the connection string used to connect to the database
7. messages: The content of the message

8.1.2 Functions

1. GetSqlConnection(): Private function that returns a new SqlConnection object using the connection string retrieved from the application's configuration settings
2. LoadRoomsFromDatabase(): Private subprocedure that loads chat room information from the database based on the user's role (provider or customer). It populates the Module_global.roomchat list with tuples containing room details
3. LoadMessagesFromDatabase(): Private subprocedure that loads messages from the database based on the user's role (provider or customer). It populates the messages list with tuples containing message details
4. InsertMessageIntoDatabase(): Private function that inserts a new message into the database. It takes parameters for the room ID, deal ID, user role, and message content. Returns True if the message is successfully inserted, False otherwise
5. PopulateRooms(): Private subprocedure that populates the chat interface with buttons representing chat rooms. Each button corresponds to a chat room retrieved from the database
6. PrintMessagesBetweenUsers(): Private subprocedure that prints messages between users in the chat interface. It retrieves messages for the specified room ID from the messages list and displays them in the UI
7. PrintMessagesBetweenUsers(): Private subprocedure that prints messages between users in the chat interface. It retrieves messages for the specified room ID from the messages list and displays them in the UI

8.1.3 Event Handlers

1. user_chats_Load(): Event handler for the form's Load event. It initializes the chat interface, loads chat rooms and messages from the database, and populates the UI with chat room buttons
2. timer_Tick(): Event handler for the timer's Tick event. It reloads chat rooms and messages periodically to keep the UI updated
3. Button_Click(): Event handler for button clicks within the chat interface. It handles switching between chat rooms and displaying messages accordingly
4. sendTextBox_KeyDown(): Event handler for key presses in the message input textbox. It allows sending messages by pressing the Enter key
5. sendButton_Click(): Event handler for the send button click event. It sends the message typed by the user, inserts it into the database, and updates the UI with the new message
6. MainForm_FormClosed(): Event handler for the form's Closed event. It stops the timer when the form is closed to prevent unnecessary processing

8.2 admin_side_chat.vb

The admin_side_chat class manages the administration side of a chat application. It facilitates communication between administrators and users by providing functionality to load and display chat rooms and messages and send messages.

8.2.1 Variables

1. support_rooms: A list of tuples containing information about chat rooms. Each tuple consists of the user name, user type, support room ID, and user ID
2. support_msgs: A list of tuples containing information about messages in the chat. Each tuple consists of the room ID, sender type, message content, and sent timestamp
3. user_role: A string indicating the role of the user. In this context, it's set to "admin"
4. roomId: An integer representing the ID of the current chat room. Initialized to -1
5. rooms_type: A string indicating the type of rooms being displayed. Initialized to "customer"
6. connectionString: A string containing the connection information for the database

8.2.2 Methods

1. PopulateRooms(): Populates the UI with buttons representing chat rooms based on the data in support_rooms
2. LoadRoomsFromDatabase(): Loads chat room data from the database into the support_rooms list
3. LoadMessagesFromDatabase(): Loads message data from the database into the support_msgs list
4. PrintMessages(roomId As Integer): Displays messages for the specified room in the UI

8.2.3 Event Handlers

1. user_chats_Load(): Event handler for the form load event. Initializes the form and starts the message timer.
2. MainForm_FormClosed(): Event handler for the form closed event. Stops the message timer.
3. MessageTimer_Tick(): Event handler for the tick event of the message timer. Checks for new messages and updates the UI.
4. userButton_Click(): Event handler for the click event of the user button. Filters rooms by customer type and updates the UI.
5. sendTextBox_KeyDown(): Event handler for the key down event of the send text box. Handles sending messages when the enter key is pressed.
6. providerButton_Click(): Event handler for the click event of the provider button. Filters rooms by provider type and updates the UI.
7. Button_Click(): Event handler for the click event of chat room buttons. Handles displaying messages for the selected room.
8. sendButton_Click(): Event handler for the click event of the send button. Sends a message to the current chat room and updates the UI.

8.3 support_chat.vb

The support_chat class provides functionality for managing a support chat interface within the application. It allows users to exchange messages with a support team or system administrator in real-time.

Properties:

1. user_role: Represents the role of the current user (e.g., "customer" or "provider").
2. userId: Represents the ID of the current user.
3. roomId: Represents the ID of the support room.
4. messageTimer: A Timer object used to periodically check for new messages.
5. user_type: Represents the type of user (e.g., "admin" or "user").
6. supportmessages: A list to store tuples representing support messages. Each tuple contains message details such as message ID, sender type, message content, and sent timestamp.

Methods

1. LoadMessagesFromDatabase(): Loads messages from the database for the current support room. True if new messages were loaded, False otherwise.
2. PrintMessages(): Prints messages in the support chat panel. Messages are sorted by timestamp and displayed with appropriate formatting.

Event Handlers:

1. user_chats_Load(): Event handler for the form load event. Initializes the form, loads initial messages, and starts the message timer.
2. MainForm_FormClosed(): Event handler for the form closed event. Stops the message timer when the form is closed.
3. MessageTimer_Tick(): Event handler for the tick event of the message timer. Checks for new messages periodically and prints them if found.
4. sendTextBox_KeyDown(): Event handler for key down event in the input text box. Handles the Enter key press to send messages.
5. sendButton_Click(): Event handler for the send button click event. Sends a new message, saves it to the database, and prints it.

8.4 appointmentChat.vb

The appointmentChat class is designed to facilitate communication between users involved in an appointment, such as a customer and a service provider. It manages message sending, receiving, and display within a chat interface. **Properties**:

1. roomId: Represents the ID of the chat room.
2. dealId: Represents the ID of the appointment or deal.
3. user_role: Represents the role of the user (customer or provider).
4. userId: Represents the ID of the user.
5. providerId: Represents the ID of the service provider.
6. providerName: Stores the name of the service provider.
7. customerName: Stores the name of the customer.
8. messages: Stores the messages exchanged in the chat.
9. connectionString: Stores the connection string to the database.

Methods:

1. GetSqlConnection(): Establishes a connection to the database using the connection string.
2. GetSenderName(): Retrieves and stores the name of the sender (either customer or provider).
3. GetRoom(): Checks for an existing chat room and creates a new one if not found.
4. InsertMessageIntoDatabase: Inserts a new message into the database.
5. LoadMessagesFromDatabase: Loads messages from the database for display.
6. PrintMessages(): Displays the messages in the chat interface.

Event Handlers:

1. sendTextBox_KeyDown(): Event handler for sending messages when Enter key is pressed.

2. sendButton_Click(): Event handler for sending messages when send button is clicked.
3. MessageTimer_Tick(): Event handler for the tick event of the message timer.
4. appointmentChat_Load(): Event handler for the form load event.
5. MainForm_FormClosed(): Event handler for the form closed event.

8.5 Chat.vb

8.5.1 Fields

- `messageList`: List of `Message` objects representing chat messages.

8.5.2 Events

- None

8.5.3 Methods

- `Button1_Click`: Handles the click event of the send button and adds a new message to the chat.
- `DisplayMessages`: Displays the chat messages in the message panel.

8.5.4 Custom Drawing

- None

8.5.5 Dependencies

- `System.Drawing.Printing`: Provides printing functionality for the application.
- `iText.Kernel.Pdf.Navigation`: Provides navigation functionality for PDF documents.

8.5.6 Nested Class: Message

- `ChatRoom_id`: Integer representing the ID of the chat room.
- `Deal_id`: Integer representing the ID of the deal related to the message.
- `Sender_type`: String representing the type of sender (e.g., user name).
- `Message_content`: String representing the content of the message.
- `SendTimeStamp`: DateTime representing the timestamp when the message was sent.

9 Errors Encountered and Resolved

1. On editing the phone number in user profile, you get an error. Although it gets updated even after getting the error.
2. Time not correct while chatting
3. Two people able to schedule a slot simultaneously
4. The user appointment and user appointment detail
5. integration no done, button of cancel no work
6. Edit profile not working in user
7. The user_feedback form is not integrated with the project
8. Concurrency
9. Edit profile not working in user
10. Payment async await resolved
11. Optimization using multithreading
12. Page not reloading after going back
13. Clicking on a tile, then coming back and clicking on another tile does not change the provider id because the form does not get loaded again.
14. Multiple accounts could be made using the same email address