Hangman Game in VBA - Documentation

User Manual

Objective
The Hangman Game in VBA provides an interactive experience for guessing a hidden word within a limited number of attempts. The game utilizes Excel VBA forms, enabling users to guess letters and track progress via visual feedback and scores.

The game is integrated with a user login system. The game stores user data and tracks progress. The interface includes a multi-page design with the following components:

1. **Features**

- **User Role:**
  - Login to play the game.
  - Guess letters to uncover a hidden word.
  - View scores after each game.
  - Log out to return to the login screen.
- **Admin Role:**
  - Add or delete words in the word bank.
  - View and manage the leaderboard.
  - Log out to return to the login screen.

2. **How to Use the Game**

3. **Login:**
   - Enter your username and password.
   - Admins will access management features.
   - Users can play the game.
4. **For Users:**
   - Start a new game by clicking "New Game."
   - Guess letters using the input box and click "Submit."
   - The word is displayed as asterisks, updating as correct guesses are made.
   - Each incorrect guess deducts a life.
5. **For Admins:**
   - Use the “Add Word” button to input new words into the word bank.
   - Use the “Delete Word” button to remove specific words.
   - View the leaderboard of players by accessing the relevant section.
6. **Logout:**
   - Click “Logout” to exit the session.

How to Use:

1. **Login**: Enter your username and password in the provided fields. If no username is entered, an error message will prompt you to fill in the fields.
2. **Navigation**: Use the tabs to switch between the game, leaderboard, and other sections.

3. **Play**: Follow the game instructions displayed on the screen.
4. **Leaderboard**: View your score and compare it with other players.

How to Play

1. **Start a New Game**:
   - Click the "New Game" button to begin a new round.
   - A random word will be selected, masked with asterisks (*), and displayed along with its length (e.g., "Your word has 4 letters!!!").
   - The word remains hidden while the player guesses one letter at a time.
2. **Enter a Guess**:
   - Type a single letter into the text box provided.
   - Click the " =>" which classifies as a "Submit" button to validate the guess.
3. **Feedback**:
   - Correct guesses reveal the letter's position(s) in the word (e.g., *o** if the word is "goat" and the guess is o).
   - Incorrect guesses reduce the player's hearts (lives) by one. A message indicates how many hearts remain.
4. **Win or Lose**:
   - **Win**: If the player guesses all letters in the word, a congratulatory message is displayed.
   - **Lose**: If the player runs out of hearts, the game ends, and the correct word is revealed.

Interface Components

- **New Game Button**: Starts a new round.
- **Submit Button**: Submits a guess for validation.
- **Text Box**: For entering guesses.
- **Labels**: Display messages like "Your word has X letters!!!", the current word progress (*o**), and remaining hearts.

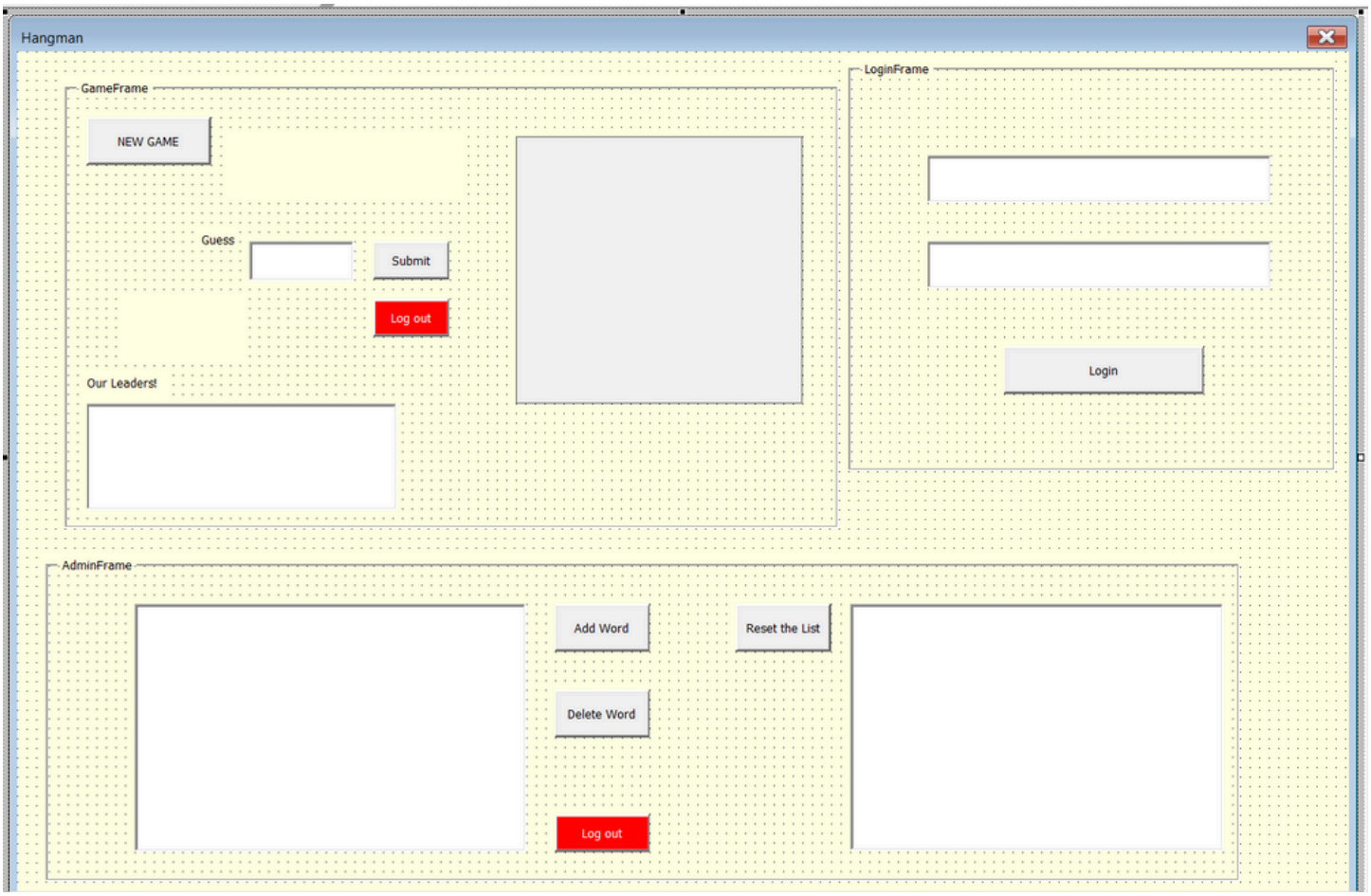
Troubleshooting Record / Known Bugs / Future Development

Troubleshooting Record

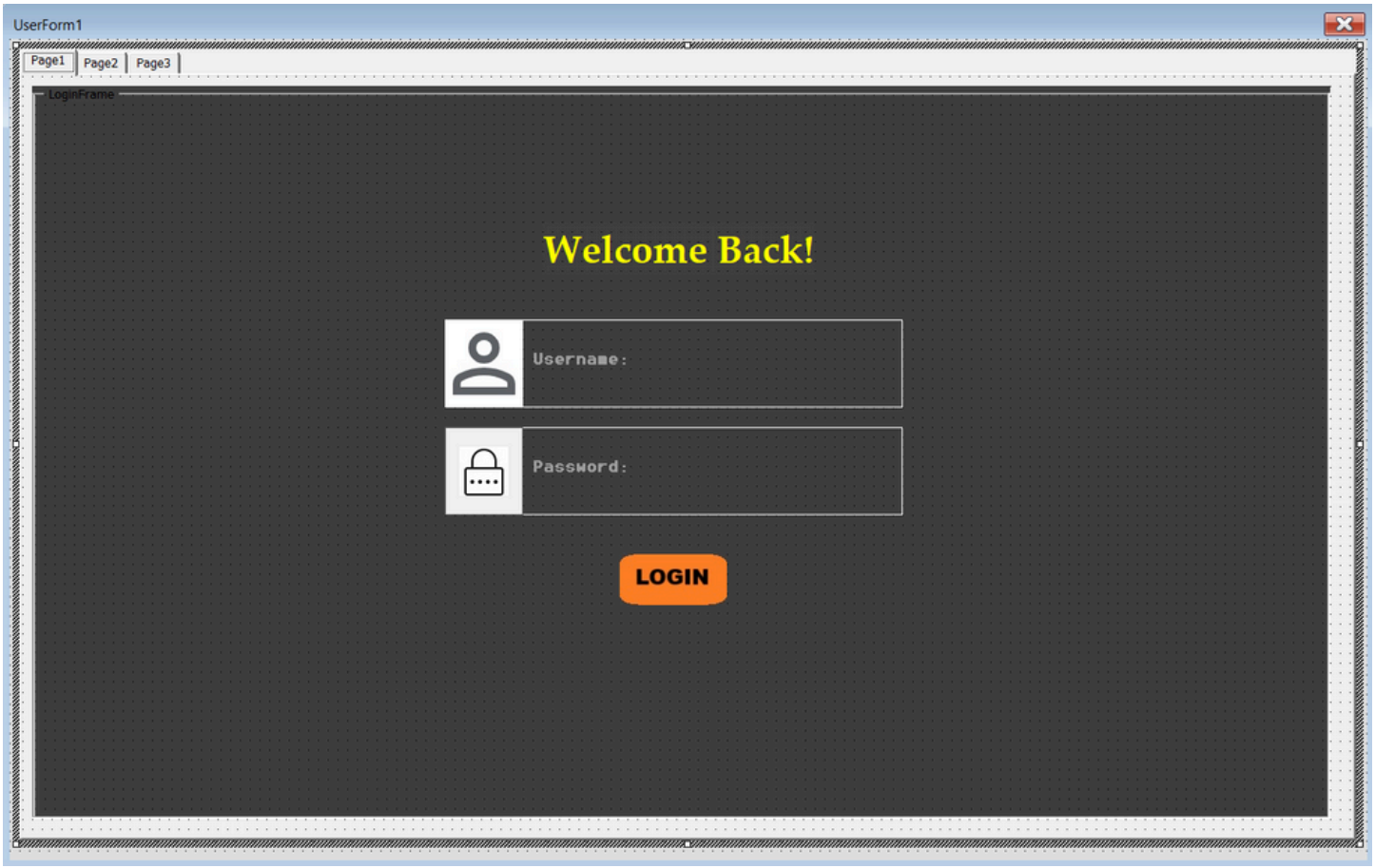
- **Error**: "Method or Data Member Not Found".
  - **Cause**: Attempted to reference non-existent labels like CurrentWord.
  - **Solution**: Added required labels to the UserForm.
- **Display Not Updating**:
  - **Cause**: stDisplay was not updating incrementally.
  - **Solution**: Introduced a temporary string (tempDisplay) to build the updated display step-by-step.
- **Error with overlapping frames**
  - **Cause:** Added another frame in one user form, which caused problems, and it was too discomfort playing.
  - **Solution:** The Professor has suggested using the multipage function to resolve the problem.
- **Compile Error**:

    - **Issue**: *"Procedure declaration does not match description of event or procedure having the same name."*
    - **Solution**: Updated the event handler for Exit to include the Cancel parameter.
- **Placeholder Visibility Issue**:
    - **Issue**: The placeholder (lblUsername) wasn't disappearing upon typing.
    - **Solution**: Implemented Enter and Exit event handlers to manage label visibility dynamically

- **FROM THIS**



- TO THIS



Known Bugs

1. **Case Sensitivity**: The game treats uppercase and lowercase letters as different (e.g., a ≠ A).
2. **Input Validation**: There is no restriction on invalid inputs (e.g., numbers or multiple characters).

Future Development

1. **Multi-Player Support**: Add functionality for two players (one selects the word and the other guesses).
2. **Word Bank Expansion**: Load words dynamically from an Excel sheet or external file for greater variety.
3. **Difficulty Levels**: Add levels based on word length or limited guesses.
4. **Feature Enhancements:**
   - Implement a "Hint" button to provide the first letter of the word.
   - Allow for customizable word lists.

Detailed Code Analysis

Word Selection

The game starts with a predefined list of 15 words. A random word is selected when the **"New Game"** button is clicked.

```
Dim stAnimalList(1 To 15) As String
Dim stWord As String

Private Sub btnNewGame_Click()
    stAnimalList(1) = "cat"
    stAnimalList(2) = "dog"
    ' ... other words ...
    stAnimalList(15) = "monkey"

    Dim iSelection As Integer
    iSelection = Int(Rnd * 15) + 1
    stWord = stAnimalList(iSelection)

    Me.CurrentStatusMessage.Caption = "Your word has got " & Len(stWord) & " letters!!
    stDisplay = String(Len(stWord), "*")
    Me.CurrentWord.Caption = stDisplay
End Sub
```

Letter Guessing Logic

The **"Submit"** button validates the player's guess and updates the game state accordingly.

1. **Input Validation**:
   - Ensures the player enters only one character.
2. **Correct Guess**:
   - Reveals the guessed letter at all correct positions in the word.
3. **Incorrect Guess**:
   - Deducts one life.
4. **Win/Loss Conditions**:
   - Displays a message if the player wins or loses.

```
Private Sub btnSubmit_Click()
    Dim stGuess As String
    Dim iLetter As Integer
    Dim tempDisplay As String
    Dim bFound As Boolean

    stGuess = Me.txtGuess.Text
    bFound = False
    tempDisplay = ""

    For iLetter = 1 To Len(stWord)
        If Mid(stWord, iLetter, 1) = stGuess Then
            tempDisplay = tempDisplay & stGuess
            bFound = True
        Else
            tempDisplay = tempDisplay & Mid(stDisplay, iLetter, 1)
        End If
    Next

    stDisplay = tempDisplay
    Me.CurrentWord.Caption = stDisplay

    If bFound Then
        MsgBox "Correct guess!"
    Else
        hearts = hearts - 1
        MsgBox "Wrong guess! Remaining hearts: " & hearts
    End If

    If hearts = 0 Then
        MsgBox "Game Over! The word was: " & stWord
    ElseIf stDisplay = stWord Then
        MsgBox "Congratulations! You guessed the word: " & stWord
    End If
End Sub
```

Score Calculation

The score depends on the word length and the player's performance (lives left and wrong guesses).
Formula:
$Score = (WordLength \times 10) + (LivesLeft \times 5) - (WrongGuesses \times 2)$

Key snippet for score:

```
score = (Len(stWord) * 10) + (iLives * 5) - (iWrongGuesses * 2)
```

References:

[VBA Documentation - MSDN]

[Random Number Generation in VBA]

[Hangman Game Logic Inspiration]

[Solutions and community discussions for debugging specific compile errors.]

This documentation provides a comprehensive guide to the Hangman game in VBA, covering its usage, code, references, troubleshooting, and potential enhancements.