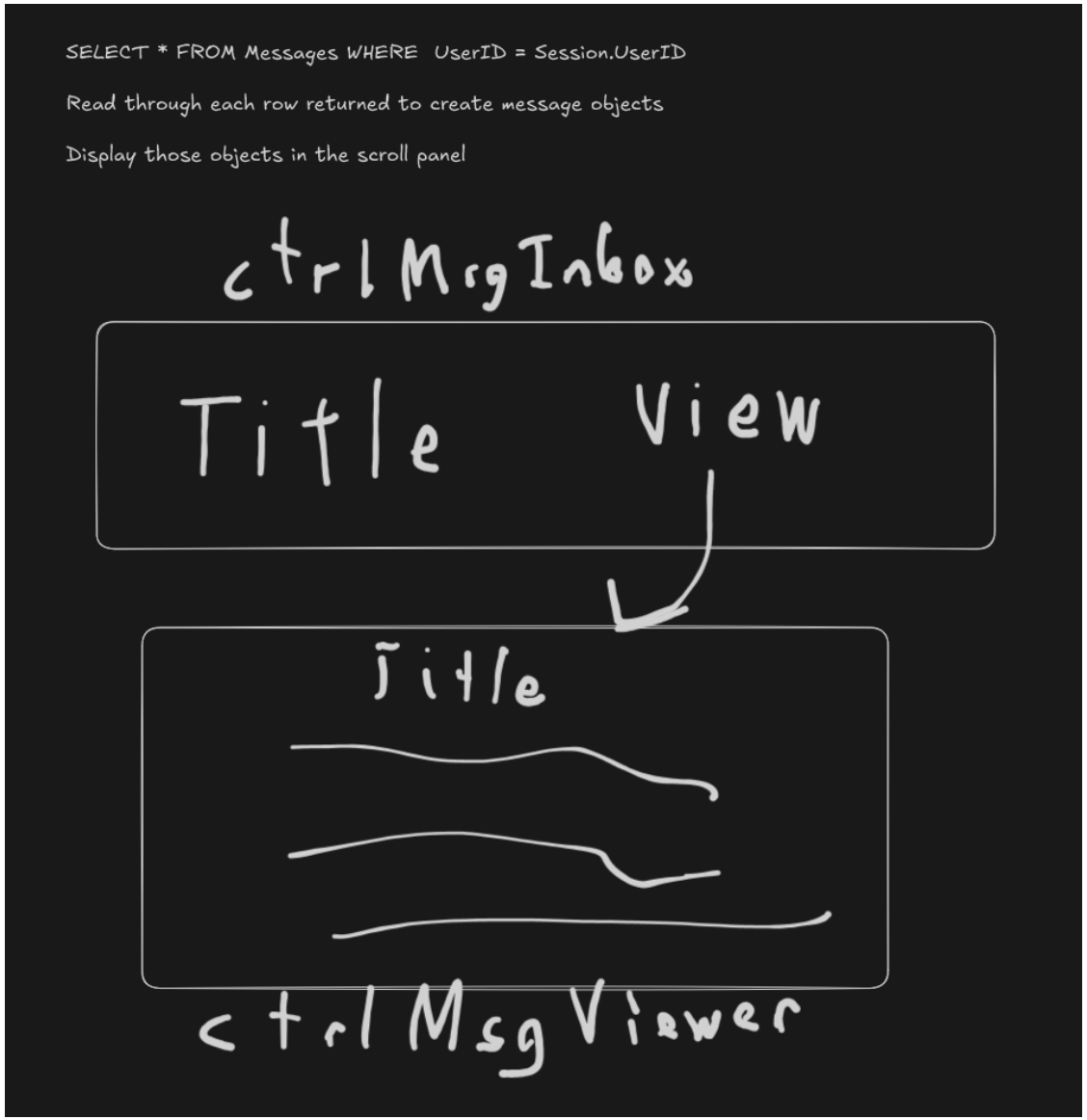
# SPRINT #4

Scrum Gang

# Goals

- Improve Functionality
  - Apply Claims
  - Download and Send Documents
  - Set up Messaging functionality
  - Claim and Finance Manager Requirements
- Continue to Add Quality of Life Updates
  - Branding Updates
  - Account updates

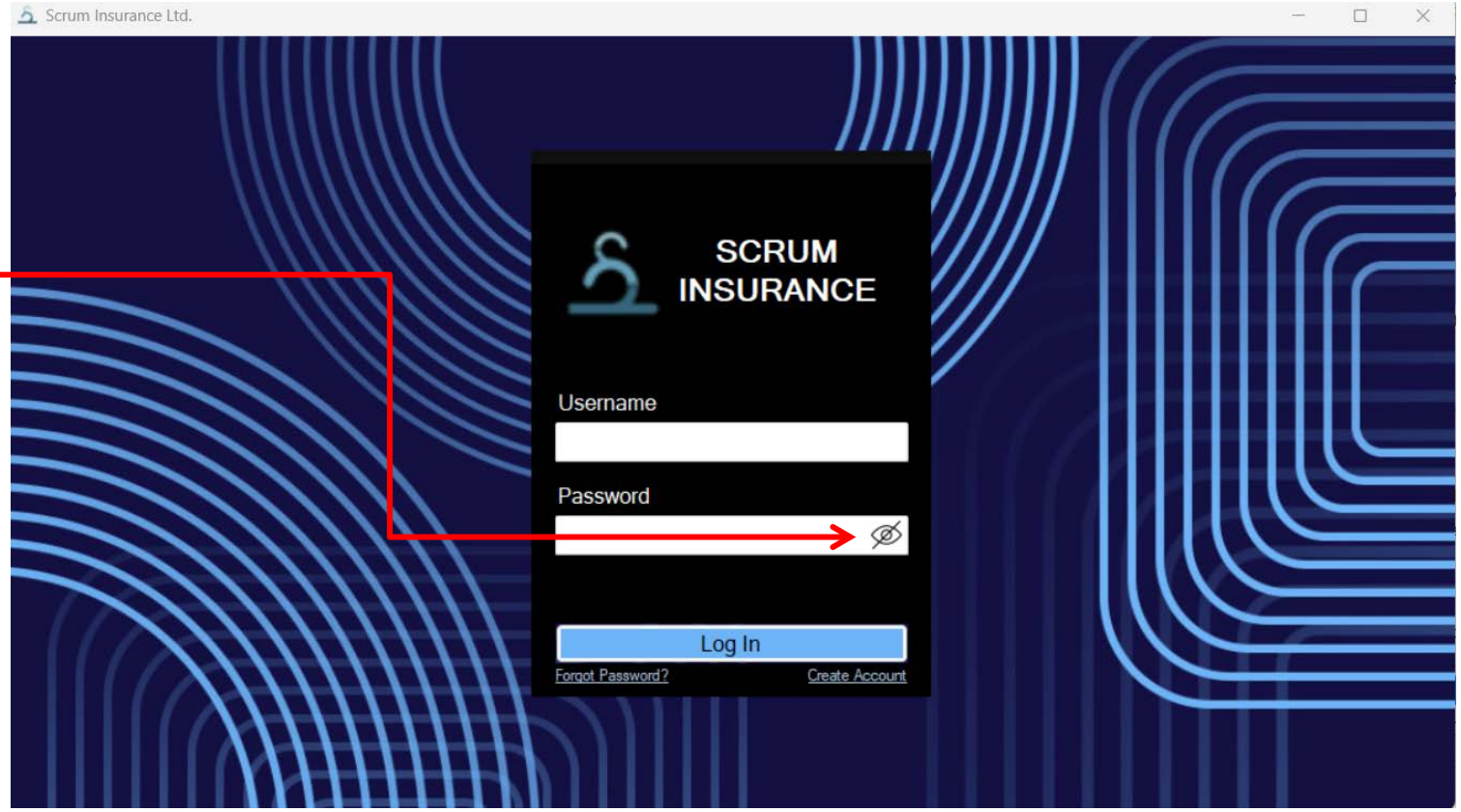
## Planning



# Front end

# Updated Gui Design

- Added Password Visibility
- Updated SCRUM INSURANCE © Background



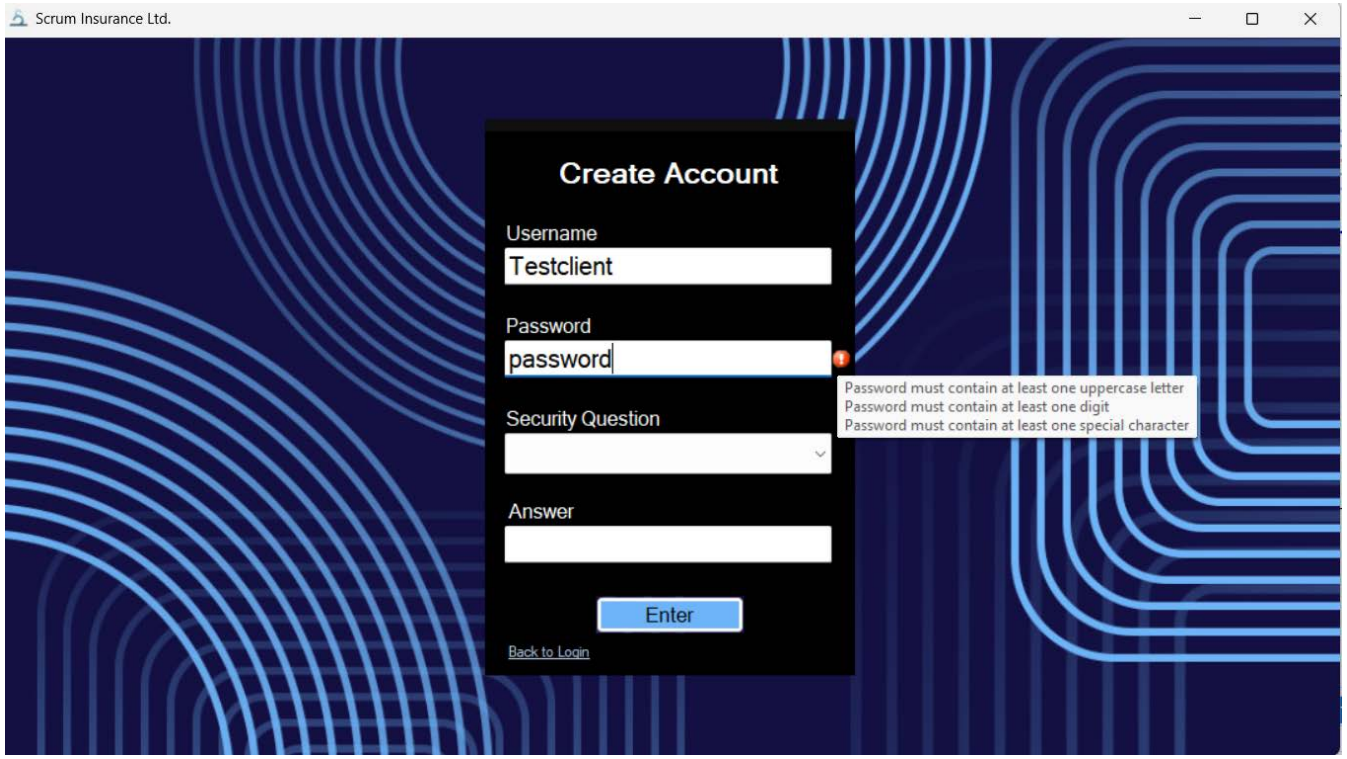

LOGIN ctrl

# Refined Password Requirements

```
2 references
public string ValidatePassword(string password, string username = "")
{
    StringBuilder errorMessages = new StringBuilder(string.Empty);

    if (password.Length < 8)
    {
        errorMessages.AppendLine("Password must be at least 8 characters long");
    }
    if (!Regex.IsMatch(password, @"^[a-zA-Z0-9!@#$%^&*]+$"))
    {
        errorMessages.AppendLine("Password must only contain letters, numbers, and special characters ! @ # $ % ^ & *");
    }
    if (!Regex.IsMatch(password, @"[A-Z]"))
    {
        errorMessages.AppendLine("Password must contain at least one uppercase letter");
    }
    if (!Regex.IsMatch(password, @"[a-z]"))
    {
        errorMessages.AppendLine("Password must contain at least one lowercase letter");
    }
    if (!Regex.IsMatch(password, @"[0-9]"))
    {
        errorMessages.AppendLine("Password must contain at least one digit");
    }
    if (!Regex.IsMatch(password, @"[\W_]"))
    {
        errorMessages.AppendLine("Password must contain at least one special character");
    }
    if (username != "" && password.Equals(username, StringComparison.OrdinalIgnoreCase))
    {
        errorMessages.AppendLine("Password must be different from username");
    }
    if (CheckDuplicatePassword(password))
    {
        errorMessages.AppendLine("Choose a different password");
    }

    return errorMessages.ToString();
}
```

- Allow multiple messages to be displayed at a time
- Validate Method that is called during account creation and new password creation (For forgot password)
- Refactoring

# Refined Account Creation



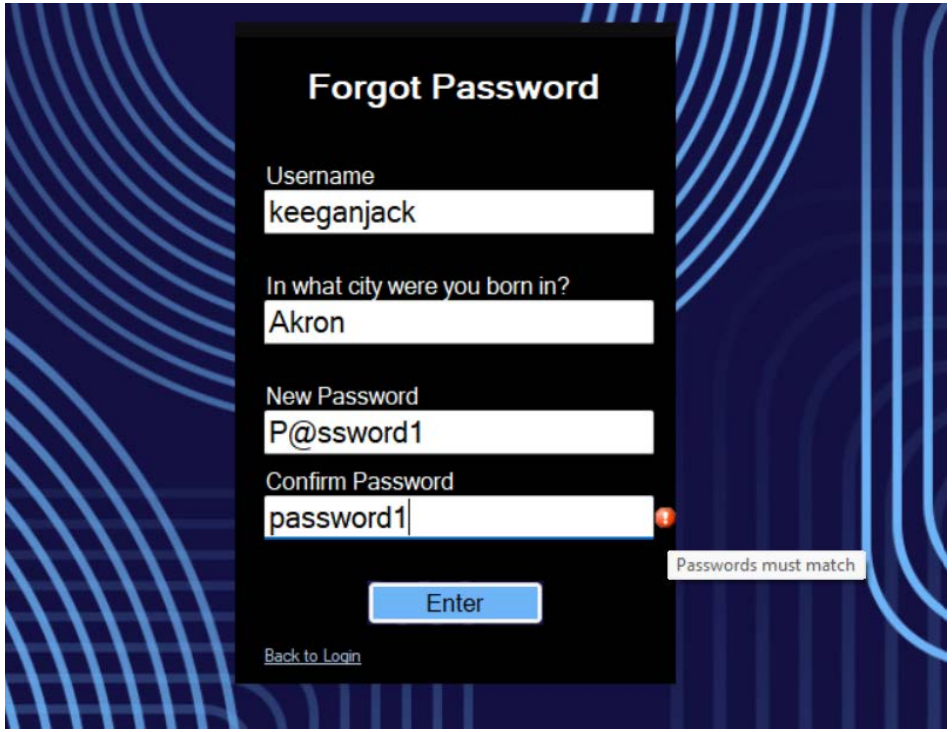


- Uses new Password Requirements
- Security Question Scroll Box
- Refactoring

# Refined Forgot Password

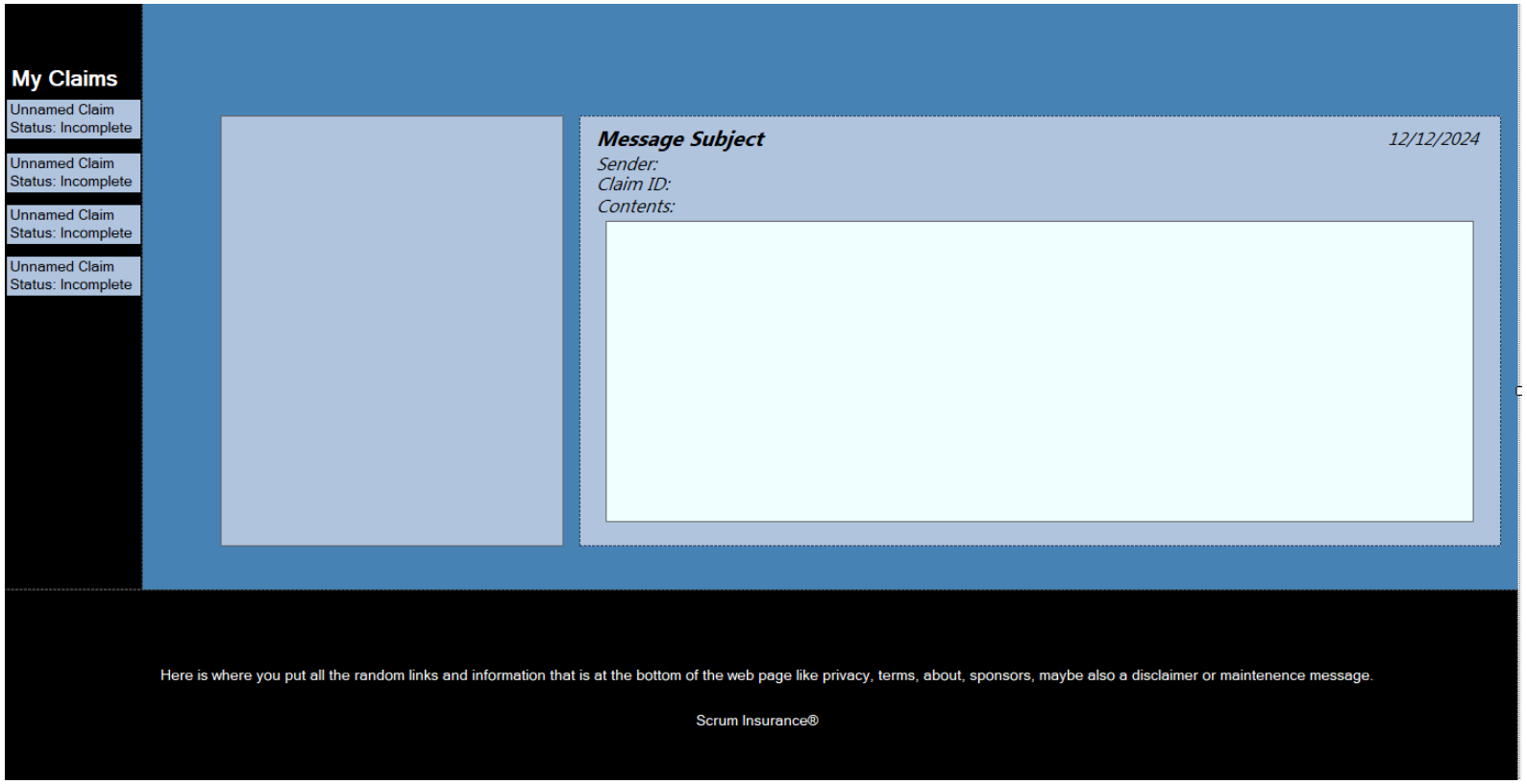
```
// Stores error count to kick user out of forgot password page
4 references
public int ForgotPassFailCount { get; set; }
```

```
// Shows security question step if email is associated with account
1 reference
private void btnConfirmUsername_Click(object sender, EventArgs e)
{
    // Get security info for input username and stores it in session account info
    string[] SecurityInfo = DBController.GetSecurityInfo(txtUsername.Text);

    // If the username was found in the database...
    if (SecurityInfo != null) {
        // Store user security info in session
        Session.UserAccount.Username = txtUsername.Text;
        Session.UserAccount.SecurityQuestion = SecurityInfo[0];
        Session.UserAccount.SecurityAnswer = SecurityInfo[1];

        // Show security question to user
        lblSecurityQuestion.Text = "Question: " + Session.UserAccount.SecurityQuestion;
        lblSecurityQuestion.Visible = true;
        txtQuestionAnswer.Visible = true;
        btnSubmitAnswer.Visible = true;
        lblUsernameError.Text = "";
    }
    else
    {
        lblUsernameError.Text = "Username not found";
    }
}
```



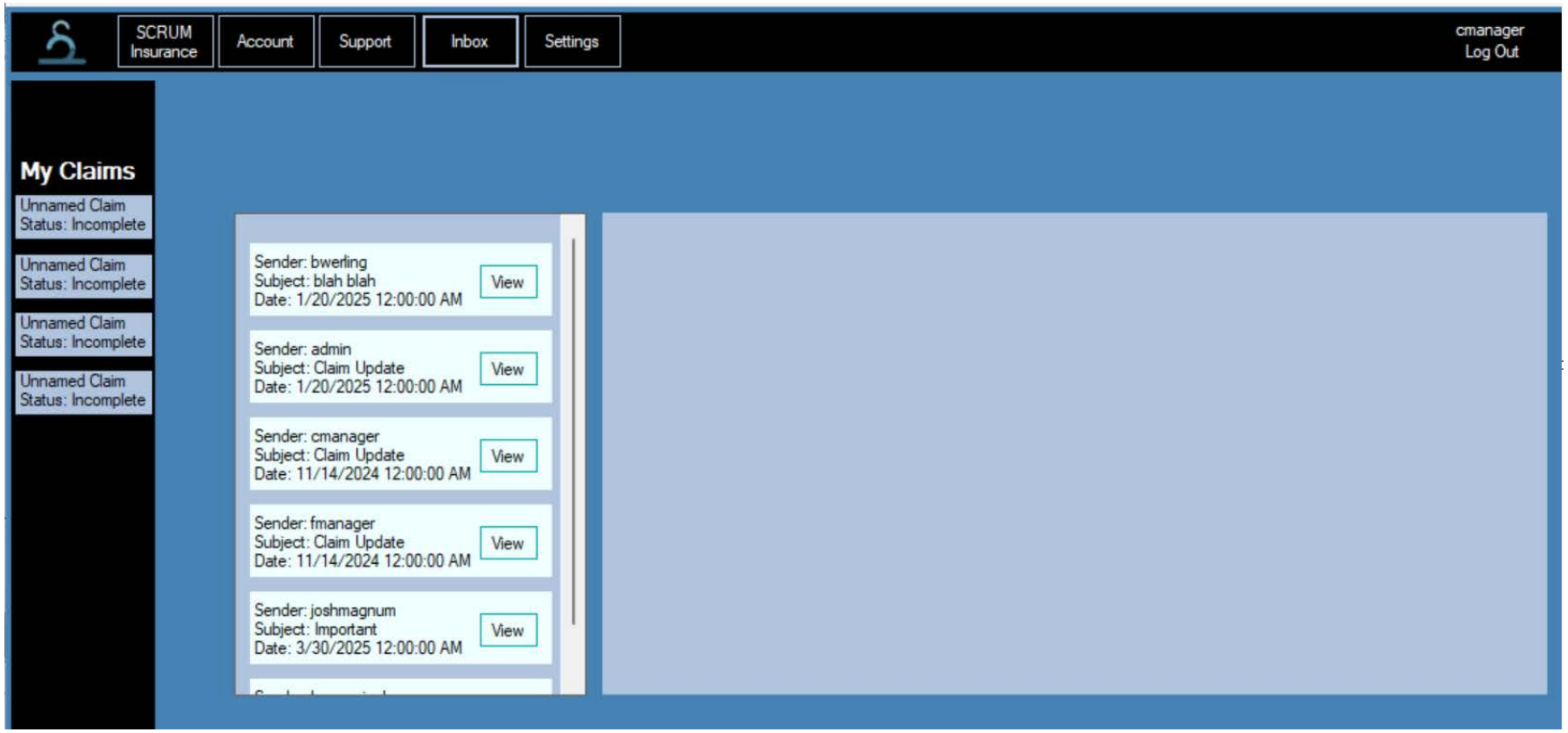

Security Question is used for password recovery along with more strict password constraints for the new password creation

# Message Inbox



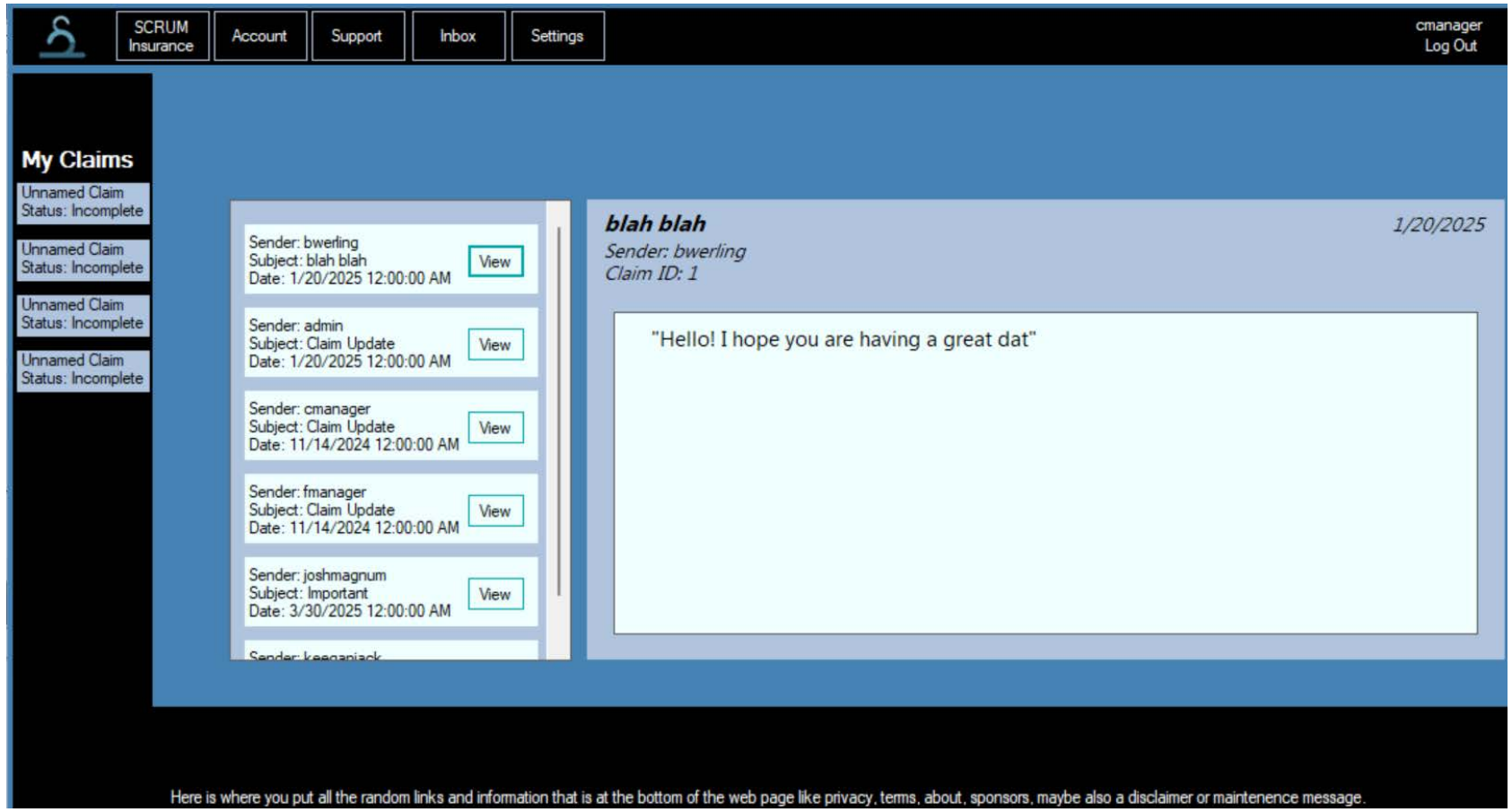

- Can send messages to finance Manager
- Doesn't return null for empty lists, so inbox can be empty and not crash the application

# Message Inbox




- Can send messages to finance Manager
- Doesn't return null for empty lists, so inbox can be empty and not crash the application

# Message Inbox



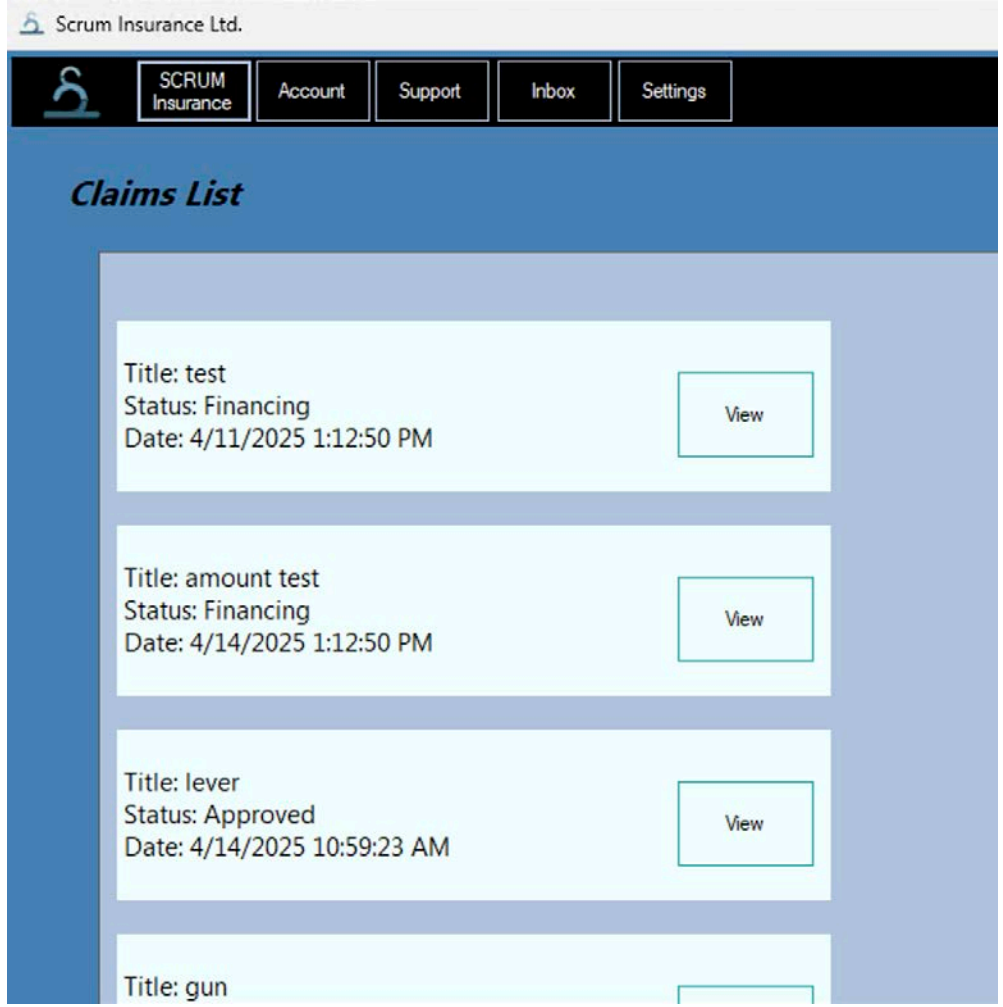

- Can send messages to finance Manager
- Doesn't return null for empty lists, so inbox can be empty and not crash the application

# Message Inbox

```
1 reference
private void btnMessageA_Click(object sender, EventArgs e)
{
    // Create a new button using the input sender object
    System.Windows.Forms.Button btn = sender as System.Windows.Forms.Button;

    // Get the message_id for this btn's message from its tag value
    int message_id = int.Parse(btn.Tag.ToString());

    // If this message is already loaded, return
    if (message_id == LoadedMessageID) return;

    // Store this message's id as the loaded message
    LoadedMessageID = message_id;

    // Unload previous message from contents panel
    if (pnlMessageContents.Controls.Count > 0)
    {
        pnlMessageContents.Controls.RemoveAt(0);
    }

    // Get the message for messageID
    Message message = DBController.GetMessage(message_id);
    message.initializeSender(DBController);

    //assign to lables and show
    lblSender.Text = "Sender: " + message.Sender;
    lblSender.Show();

    Console.WriteLine(message);

    lblDate.Text = message.Date.Split(' ')[0];
    lblDate.Show();

    lblHeader.Text = message.Subject;
    lblHeader.Show();

    String messageContent = message.Content;
    pnlMessageContents.Show();
```

```
private void AddMessage(Message msg)
{
    MessageCount++;

    // Create new button for this message and style it
    System.Windows.Forms.Button btn = new System.Windows.Forms.Button();
    btn.BackColor = Color.Azure;
    btn.Text = "View";
    btn.FlatAppearance.BorderColor = Color.Azure;
    btn.FlatStyle = FlatStyle.Popup;
    btn.TextAlign = ContentAlignment.MiddleCenter;
    btn.Width = 40;
    btn.Location = new Point(170, (MessageCount * 60) -25);

    // Store message id in button btn's 'Tag' property
    btn.Tag = msg.ID;

    // Add click handler to button btn
    btn.Click += new System.EventHandler(this.btnMessageA_Click);

    // Add button btn to the message panel
    pnlMessages.Controls.Add(btn);

    // Create new label for this message and style it
    Label lblMessage = new Label();
    lblMessage.BackColor = Color.Azure;
    lblMessage.TextAlign = ContentAlignment.MiddleLeft;
    lblMessage.Location = new Point(10, (MessageCount * 60) - 40);
    lblMessage.Width = 210;
    lblMessage.Height = 50;

    // Show message info as label text
    lblMessage.Text = "Sender: " + msg.Sender + "\nSubject: " + msg.Subject + "\nDate: " + msg.Date;

    // Add lblMessage to the message panel
    pnlMessages.Controls.Add(lblMessage);
}
```

# Claim List

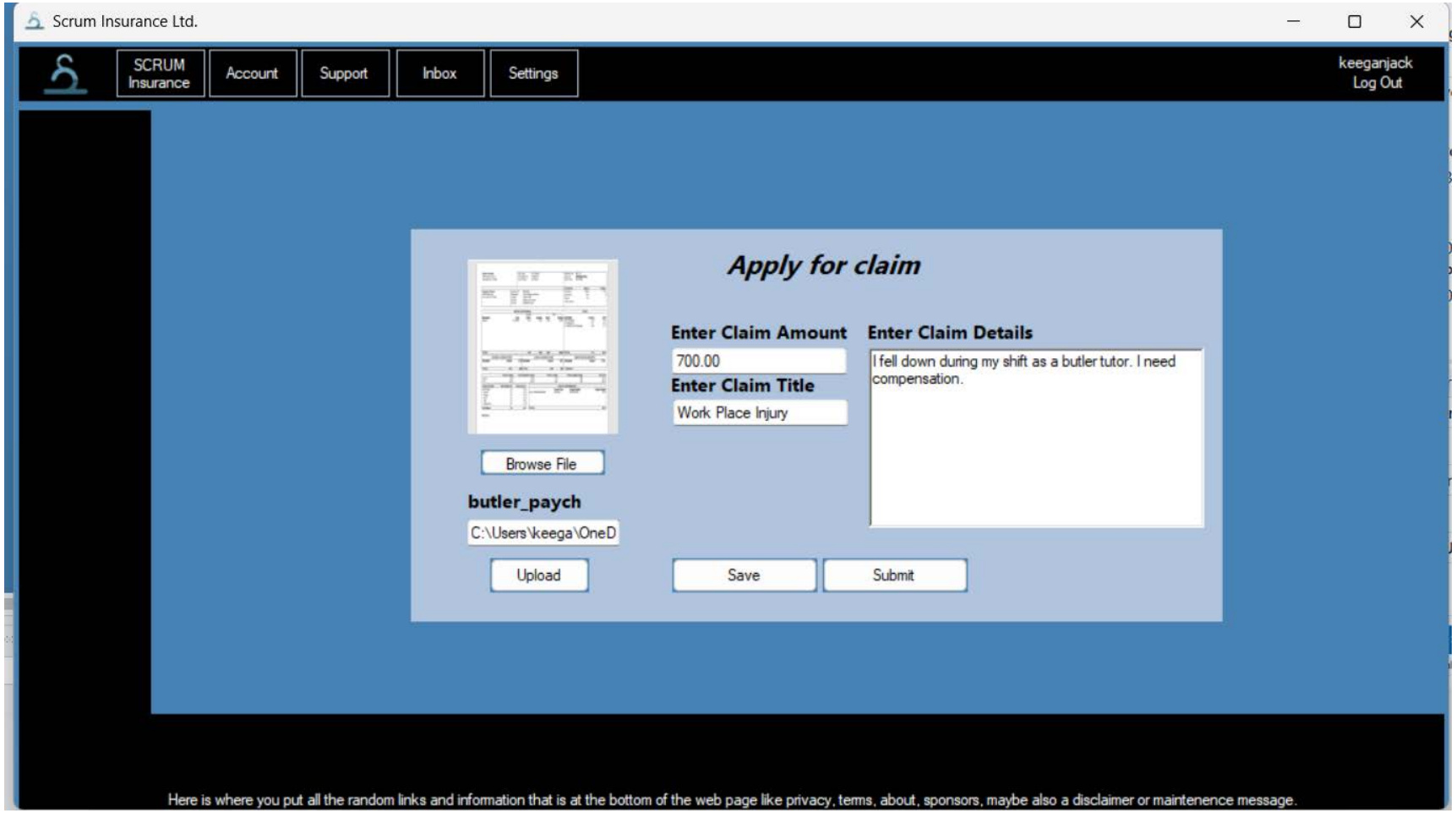



```
public ctrlClaimsList(Session session, DatabaseController DBController)
{
    InitializeComponent();
    session_ = session;
    //these are the columns we want to grab for the select query
    string[] columns = { "Claim_Title", "Claim_Date", "Claim_Status", "Claim_ID" };

    //these set the args.

    Dictionary<String, Object> args = new Dictionary<String, Object>();
    if (session.UserAccount.Role.Equals("claim_manager"))
    {
        args.Add("Claim_Status", "Validating");
    }
    else if (session.UserAccount.Role.Equals("finance_manager"))
    {
        args.Add("Claim_Status", "Financing");
    }


    List<Claim> claimList = DBController.GetClaimList(session.UserAccount);

    foreach (Claim claim in claimList)
    {
        addClaim(claim);
    }

}
```

# Claim Viewer

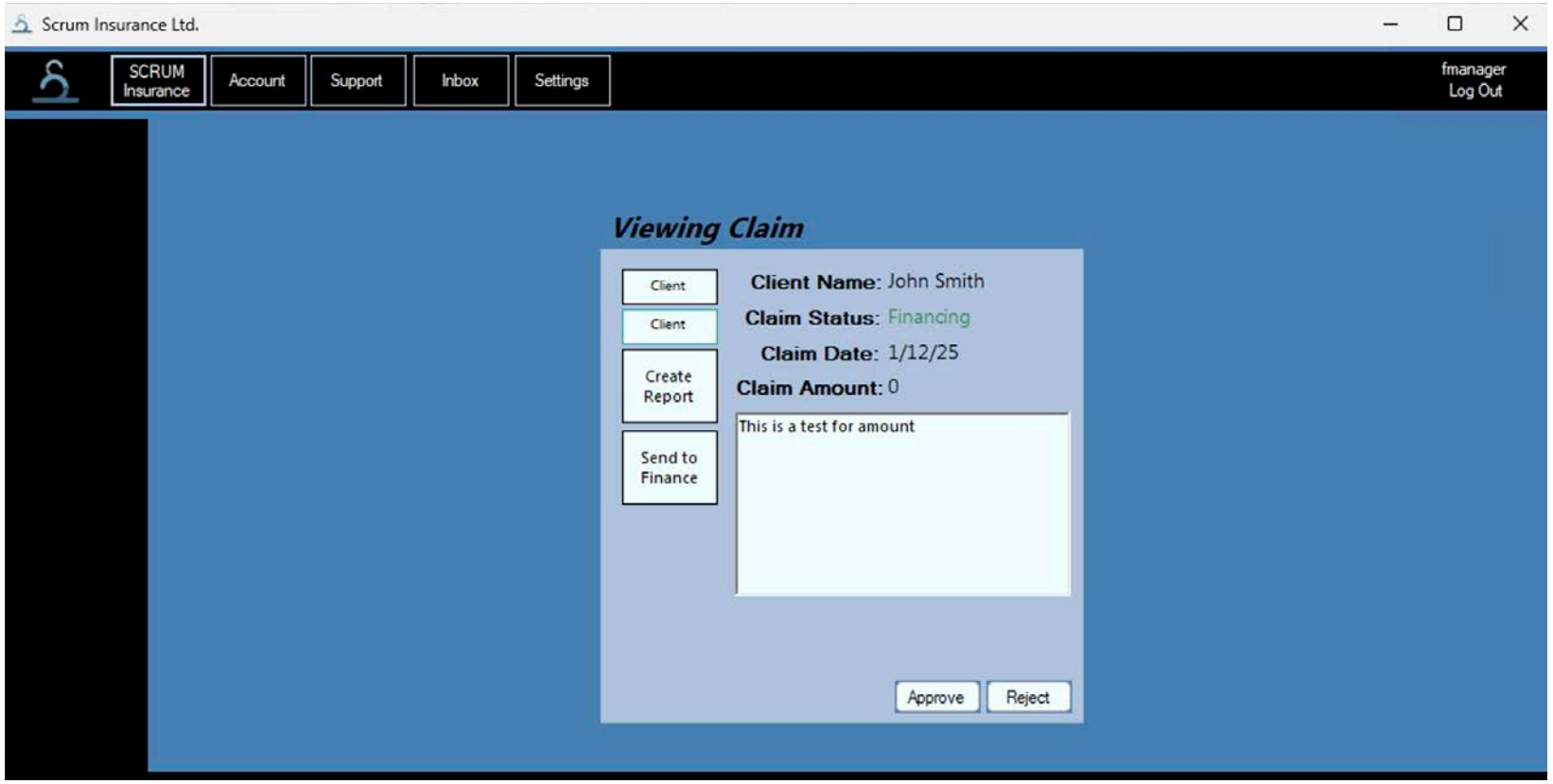
- Takes in a Claim object and pulls info

# Apply Claim




- Claims are inserted into database
- Added Save button
- Upload Document functionality almost completed
- Can select multiple file type, preview, and see file name
- Finance Managers can be assigned to claims

# Claim Status Progression




1. Client
2. Claim Manager
3. Finance Manager
4. Client

# Claim Status Progression Cont.

```
1 reference
public ctrlClaimViewer(DatabaseController dbController, int claim_id, Session session)
{
    InitializeComponent();
    DBController = dbController;
    role = session.UserAccount.Role;

    // Queries the database for input claim
    Claim claim = DBController.GetClaim(claim_id);
    status = claim.Status.ToString();
    lblStatusType.Text = status;
    if (status == "Pending")
    {
        lblStatusType.ForeColor = Color.Olive;
    }
    else if (status == "Financing")
    {
        lblStatusType.ForeColor = Color.SeaGreen;
    }
    else if (status == "Approved")
    {
        lblStatusType.ForeColor = Color.Green;
    }
    else if (status == "Rejected")
    {
        lblStatusType.ForeColor = Color.Red;
    }

    lblAmount.Text = claim.Amount.ToString();

    rtxDetails.Text = claim.Content.ToString();
    id = claim_id;

}
```

- Claim Manager and Finance Manager both can change the claim status

```
private void btnApprove_Click(object sender, EventArgs e)
{
    if (role.Equals("claim_manager"))
    {
        if(DBController.UpdateClaim(id, "Status", "Financing"))
        {
            lblStatusType.Text = "Financing";
            lblStatusType.ForeColor = Color.SeaGreen;
        }

    }
    else if (role.Equals("finance_manager"))
    {
        DBController.UpdateClaim(id, "Status", "Approved");
        lblStatusType.Text = "Approved";
        lblStatusType.ForeColor = Color.Green;
    }
}

1 reference
private void btnReject_Click(object sender, EventArgs e)
{
    DBController.UpdateClaim(id, "Status", "Rejected");
    lblStatusType.Text = "Rejected";
    lblStatusType.ForeColor = Color.Red;
}

1 reference
private void btnReport_Click(object sender, EventArgs e)
{

}

1 reference
private void btnTransfer_Click(object sender, EventArgs e)
{
    int financerId = Convert.ToInt32(DBController.getFinanceManager().ID);
    DBController.UpdateClaim(id, "finance_manager_id", financerId.ToString());
}
```

# Upload Documents

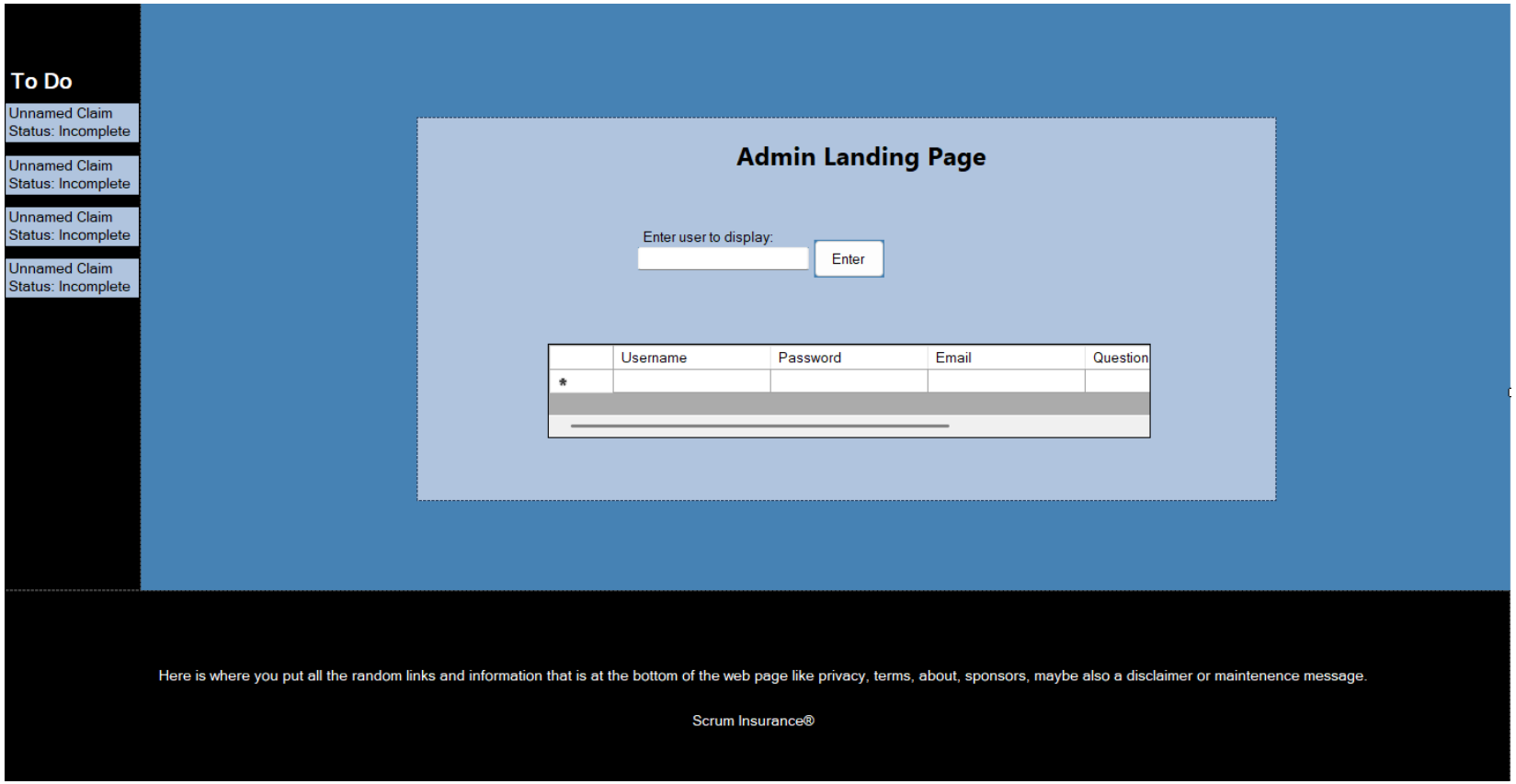
```
1 reference
private void btnBrowseDocument_Click(object sender, EventArgs e)
{
    DialogResult dr = this.ofdClaimDocument.ShowDialog();

    // If file selection was successful...
    if (dr == DialogResult.OK)
    {
        // Loop through each file selected in the dialog
        foreach (string file_path in ofdClaimDocument.FileNames)
        {
            // Add this file's file path to the list of document paths
            DocumentPaths.Add(file_path);

            // Displays file names, seperated by ';'
            lblFileName.Text = Path.GetFileName(file_path) + ";";
        }
    }
}
```

```
1 reference
private void btnTestUpload_Click(object sender, EventArgs e)
{
    // Loop through each document in the List
    foreach (string document_path in DocumentPaths)
    {
        // Stores file name from path
        string file_name = Path.GetFileName(document_path);

        // Stores file data as byte array
        byte[] file_data = File.ReadAllBytes(document_path);

        // Upload document info to database
        DBController.UploadDocument(7, file_name, file_data);
    }
}
```

## Calling the Method

```
1 reference
public void UploadDocument(long claim_id, string file_name, byte[] file_data)
{
    // Create parameter dictionary for document upload
    Dictionary<string, object> document_info = new Dictionary<string, object>
    {
        { "claim_id", claim_id },
        { "file_name", file_name },
        { "file_data", file_data },
        { "upload_date", DateTime.Now }
    };

    // Create insertion query and store it in Connection.Query property
    Connection.Query = new InsertQuery(document_info).Into("documents");

    // Upload document
    if (!Connection.ExecuteNonQuery()) throw new InvalidOperationException("File upload failed.");
}

}
```

# Admin Page Updates



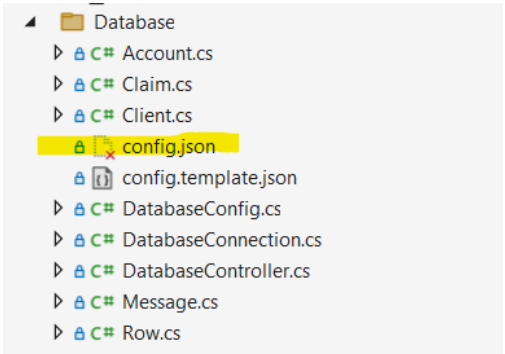

- Scroll Fixed

# Back End

# Database Security

- config.json FILE - Database Privacy

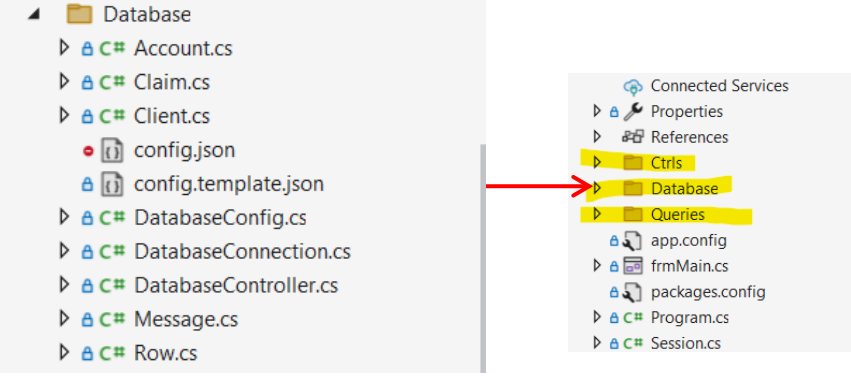
```
/*
* Copy this file's contents into a file titled 'config.json' in this Database directory
* Input database connection values into each line
* In the properties for the file set 'Copy to output directory' to 'Always True'
*
* This exists so we don't have our database info publicly available on GitHub
*/
{
  "DatabaseConfig": {
    "ServerName": "",
    "Name": "",
    "Username": "",
    "Password": ""
  }
}
```

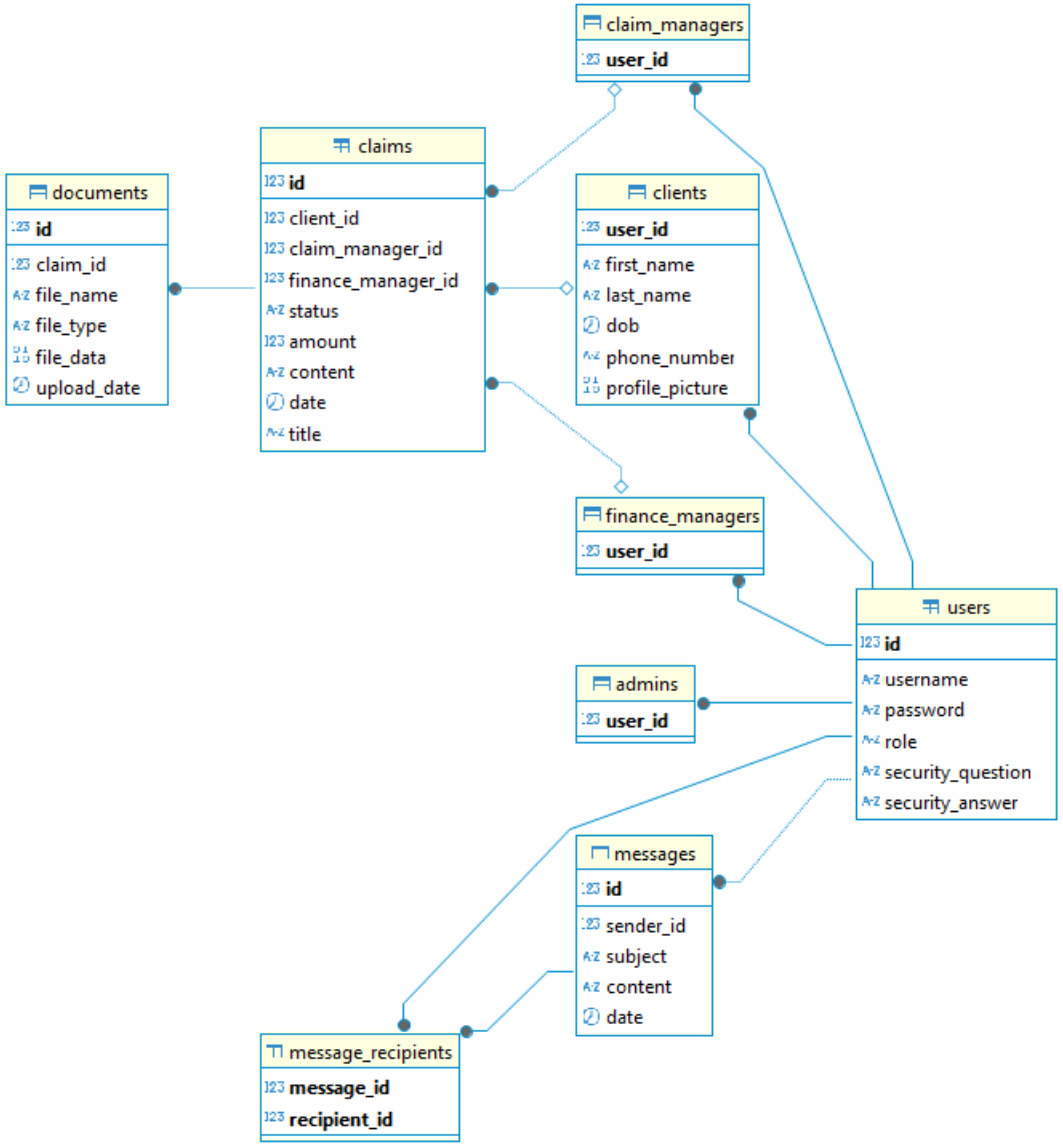



```
5 references
public class DatabaseConfig
{
    [JsonPropertyName("DatabaseConfig")]
    4 references
    public DatabaseInfo Database { get; set; }
}

1 reference
public class DatabaseInfo
{
    [JsonPropertyName("ServerName")]
    1 reference
    public string ServerName { get; set; }

    [JsonPropertyName("Name")]
    1 reference
    public string Name { get; set; }

    [JsonPropertyName("Username")]
    1 reference
    public string Username { get; set; }

    [JsonPropertyName("Password")]
    1 reference
    public string Password { get; set; }
}
}
```

# Database + VS Rework

- Current Organization
- Database files moved to their own folder



# Example: Row Class

- New Row Object
- Can act as a constructor for Claim Class

```
namespace ScrumInsurance
{
    35 references
    public class Row
    {
        27 references
        public Dictionary<string, object> Columns { get; set; }

        3 references
        public Row()
        {
            Columns = new Dictionary<string, object>();
        }

        2 references
        public void AddColumn(string name, object value)
        {
            Columns.Add(name, value);
        }
```

```
public Message(Row row)
{
    if (row.Columns.TryGetValue("id", out var id)) ID = id.ToString();
    if (row.Columns.TryGetValue("sender_id", out var sender)) SenderID = sender.ToString();
    if (row.Columns.TryGetValue("recipient_id", out var recipient)) RecipientID = recipient.ToString();
    if (row.Columns.TryGetValue("subject", out var subject)) Subject = subject.ToString();
    if (row.Columns.TryGetValue("content", out var content)) Content = content.ToString();
    if (row.Columns.TryGetValue("date", out var date)) Date = date.ToString();
}
```

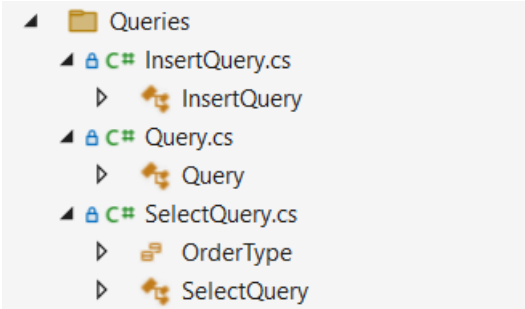
# Query changes

- Query Class
  - Abstract
  - Used to build queries with methods
  - Added Query object property to Database Connection

```
namespace ScrumInsurance.Queries
{

    public abstract class Query
    {

        public string TableName { get; set; }


        public Query (string tableName)
        {
            TableName = tableName;
        }


        public override abstract string ToString();


        public abstract void InsertParameters(MySqlCommand cmd);
    }
}
```

# Query changes

- Insert Query Class
  - Stores insert query string to build



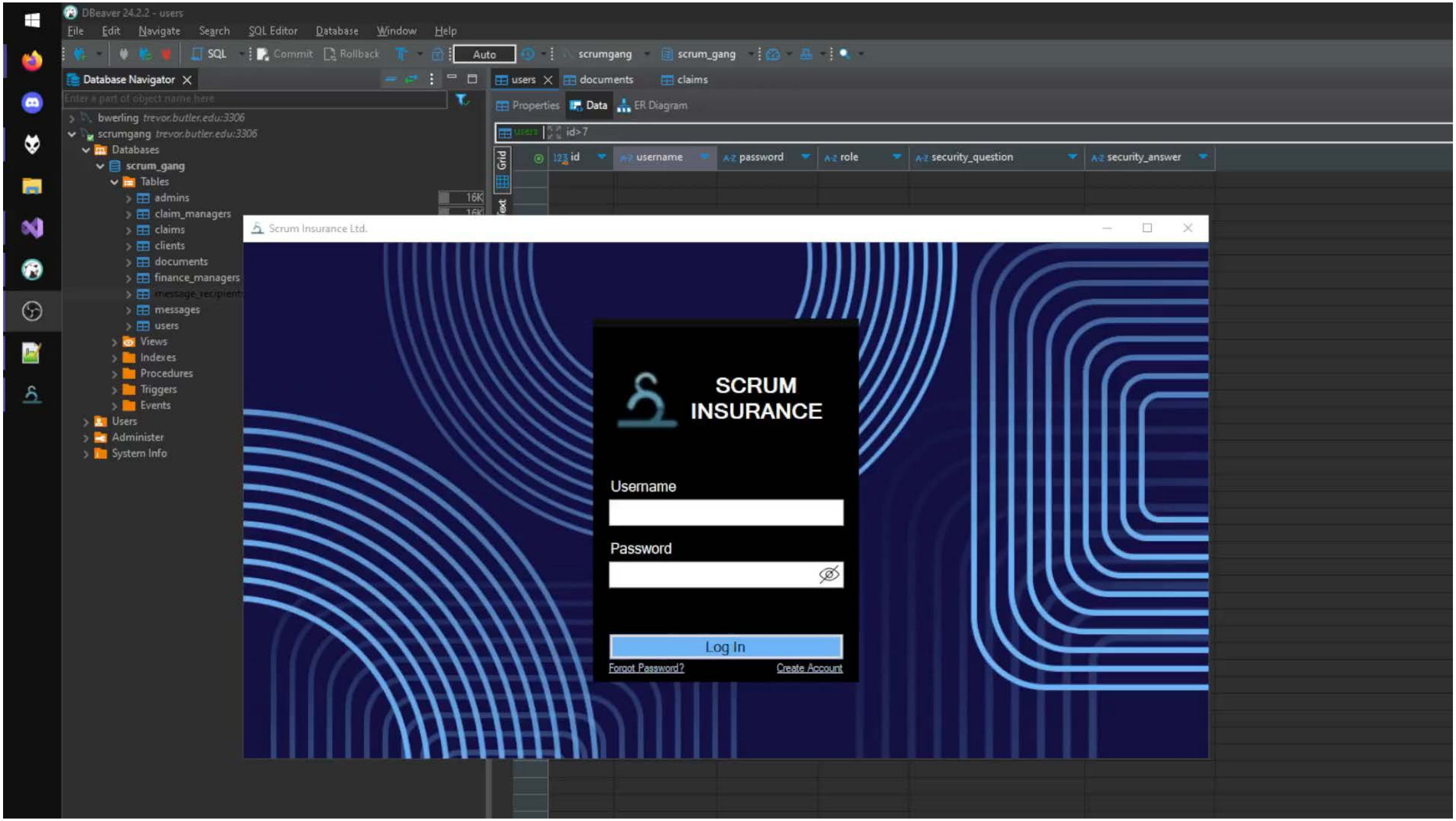

```
namespace ScrumInsurance.Queries
{
    5 references
    public class InsertQuery : Query
    {
        4 references
        public Dictionary<string, object> Data { get; set; }

        3 references
        public InsertQuery(Dictionary<string, object> data) : base(string.Empty)
        {
            Data = data;
        }

        // Set the table to insert into
        3 references
        public InsertQuery Into(string tableName)
        {
            TableName = tableName;
            return this;
        }

        // Convert this query to string ready for execution
        3 references
        public override string ToString()
        {
            // Stores insert query string to build
            StringBuilder query = new StringBuilder();

            // Join keys with commas in a string
            var columns = string.Join(", ", Data.Keys);

            // Join @keys as parameters with commas for later insertion
            var parameters = string.Join(", ", Data.Keys.Select(k => "@" + k));

            // Creates string of requested columns
            query.Append($"INSET INTO {TableName} ({columns})\nVALUES ({parameters})");

            // Print query to console
            Console.WriteLine(query.ToString());

            return query.ToString();
        }

        // Insert actual values into parameterized select query stored in input 'cmd'
        3 references
        public override void InsertParameters(MySqlCommand cmd)
        {
            // Loop through each column in Data Dictionary
            foreach (var pair in Data)
            {
                // Insert actual value into the @ parameters
                cmd.Parameters.AddWithValue("@" + pair.Key, pair.Value);
            }
        }
    }
}
```

# Query changes

- Select Query
- Order Type – allows methods to change the order, important for the message inbox
- Adaptable

```
// Add one where condition
11 references
public SelectQuery Where(string arg1, string operatr, string arg2) {
    WhereConditions.Add((arg1, operatr, arg2));
    return this;
}

// Add a list of where conditions
0 references
public SelectQuery Where(List<(string, string, string)> conditions) {
    WhereConditions = conditions;
    return this;
}

// Add a column to order by and set the order to ASC or DESC
0 references
public SelectQuery OrderBy(string column, OrderType order = OrderType.ASC) {
    OrderColumn = column;
    OrderType = order;
    return this;
}

// Add a limit of rows returned
0 references
public SelectQuery Limit(int limit) {
    RowLimit = limit;
    return this;
}
```

```
namespace ScrumInsurance.Queries
{
    5 references
    public enum OrderType
    {
        ASC,
        DESC
    }

    21 references
    public class SelectQuery : Query
    {
        4 references
        public List<string> RequestColumns { get; set; }
        9 references
        public List<(string, string, string)> WhereConditions { get; set; }
        6 references
        public string JoinTable {  get; set; }
        5 references
        public string JoinColumn { get; set; }
        5 references
        public string OrderColumn { get; set; }
        4 references
        public OrderType OrderType { get; set; }
        5 references
        public int RowLimit { get; set; }

        4 references
        public SelectQuery(string column = null) : base(string.Empty)
        {
            RequestColumns = new List<string> { column == null ? "*" : column };
            WhereConditions = new List<(string, string, string)>();
            JoinTable = string.Empty;
            JoinColumn = string.Empty;
            OrderColumn = string.Empty;
            OrderType = OrderType.ASC;
            RowLimit = 0;
        }

        4 references
        public SelectQuery(List<string> requestColumns) : base(string.Empty) {
            RequestColumns = requestColumns;
            WhereConditions = new List<(string, string, string)>();
            JoinTable = string.Empty;
            JoinColumn = string.Empty;
            OrderColumn = string.Empty;
            OrderType = OrderType.ASC;
            RowLimit = 0;
        }
```


DBeaver 24.2.2 - users
File Edit Navigate Search SQL Editor Database Window Help
SQL Commit Rollback Auto scrumgang scrum_gang
Database Navigator
Enter a part of object name here
bwerling trevor.butler.edu:3306
scrumgang trevor.butler.edu:3306
Databases
scrum_gang
Tables
admins
claim_managers
claims
clients
documents
finance_managers
messages
users
Views
Indexes
Procedures
Triggers
Events
Users
Administer
System Info
users
documents
claims
Properties
Data
ER Diagram
id>7
id
username
password
role
security_question
security_answer
Scrum Insurance Ltd.
SCRUM INSURANCE
Username
Password
Log In
Forgot Password?
Create Account

# Obstacles

- Figuring out how to work with Blobs  (Downloading images to the database)
- “Value does not fall within the expected range” error – difficulty troubleshooting
- End of Semester busyness

# Future Work

- Streamline communication between different user roles
- Compose message control
- Document download functionality
- View profile functionality

# Questions?

Scrum Gang