Table of Contents

Bulacan State University
City of Malolos, Bulacan

College of Information and Communications Technology
Information Technology Department

Citizen Complaint Management System

Project Title:

FINAL PROJECT

in

IT 207

OBJECT-ORIENTED PROGRAMMING 2

Submitted by:

DIMLA, Tyron B.

MENDOZA, Reymark M.

QUIÑONES JR, Romeo M.

STA MARIA, Prince Rusel B.

TAGANAS, Lorenz Romeo O.

BSIT 2G – G1

Submitted to:

ANDRES, Robethel D.

Introduction

The Citizen Complaint Management System is an easy-to-use software that helps organize and handle complaints in a community or organization. This system makes it simpler for people to submit their complaints and for administrators to keep track of, manage, and resolve issues efficiently.

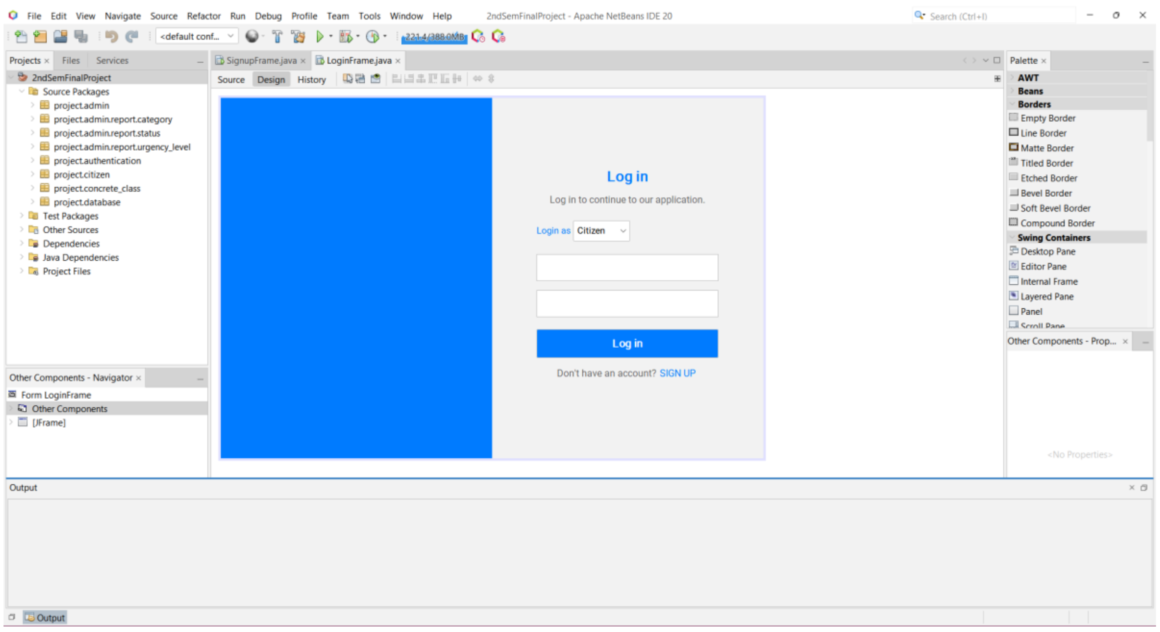
Business Problems

Complaints often aren't addressed quickly because they are handled manually and there's no streamlined process in place. This delay frustrates users and makes them lose trust in the organization's ability to resolve issues effectively. Additionally, complaints come in through various channels (email, phone, physical forms), making it hard to track their status and progress. This fragmentation can lead to lost complaints, duplicated efforts, and a lack of accountability.

Proposed Solutions

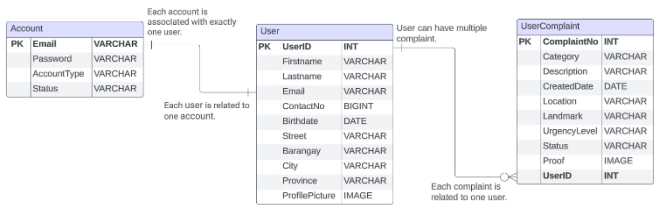
Our solution is to create a Complaint Management System (CMS). This system will be an online platform where people can submit their complaints, making sure all issues are recorded and tracked properly. The CMS will streamline the complaint-handling process, helping administrators resolve problems more quickly and efficiently. Users will be able to see the status of their complaints in real-time, which will make the process more transparent and accountable. By bringing all complaints into one system, we can avoid the problems caused by having multiple submission methods and ensure that no complaint is lost or ignored. This will result in faster response times, higher user satisfaction, and increased trust in the organization's ability to handle and resolve issues.

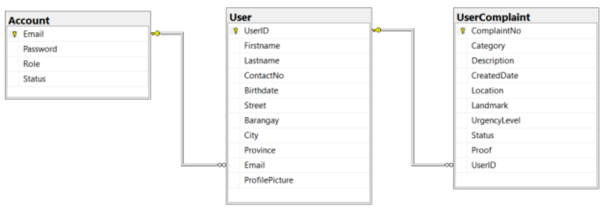
Project Window



Database Modeling

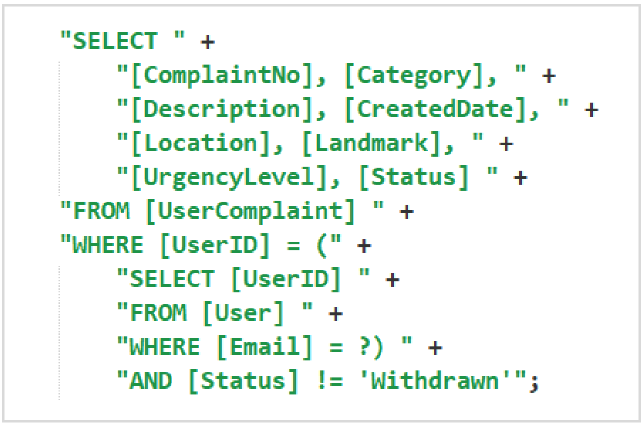
- **User Management:** The User table stores information about each user, facilitating easy identification and contact.
- **Account Security:** The Account table ensures secure authentication by storing encrypted passwords linked to user emails.
- **Complaint Tracking:** The User Complaint table allows users to submit and track complaints, complete with relevant details and proof images.





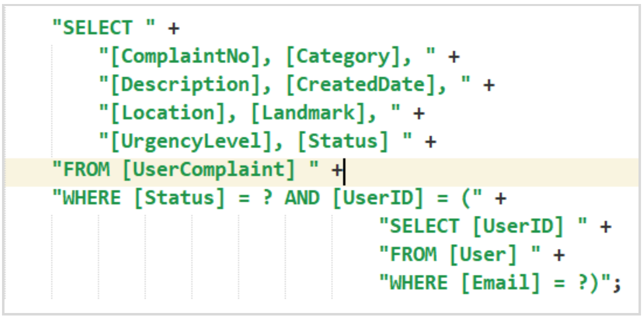
SQL Query Development

Citizen Interface Utilized Database Query



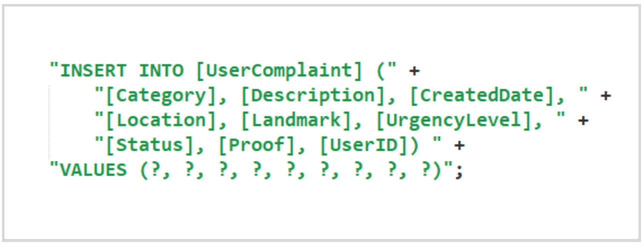
```
"SELECT " +
    "[ComplaintNo], [Category], " +
    "[Description], [CreatedDate], " +
    "[Location], [Landmark], " +
    "[UrgencyLevel], [Status] " +
"FROM [UserComplaint] " +
"WHERE [UserID] = (" +
    "SELECT [UserID] " +
    "FROM [User] " +
    "WHERE [Email] = ?) " +
    "AND [Status] != 'Withdrawn'";
```

This query retrieves complaint details for a user based on their email, excluding withdrawn complaints, utilizing a subquery to match the email to the corresponding UserID.



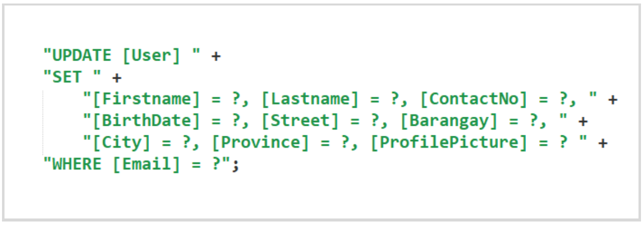
```
"SELECT " +
    "[ComplaintNo], [Category], " +
    "[Description], [CreatedDate], " +
    "[Location], [Landmark], " +
    "[UrgencyLevel], [Status] " +
"FROM [UserComplaint] " +
"WHERE [Status] = ? AND [UserID] = (" +
                                  "SELECT [UserID] " +
                                  "FROM [User] " +
                                  "WHERE [Email] = ?)";
```

This query retrieves complaint details for a user based on their email and a specified status, utilizing a subquery to match the email to the corresponding UserID.



```
"INSERT INTO [UserComplaint] (" +
    "[Category], [Description], [CreatedDate], " +
    "[Location], [Landmark], [UrgencyLevel], " +
    "[Status], [Proof], [UserID]) " +
"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

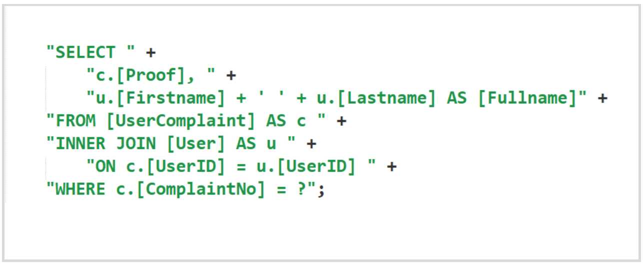
This query retrieves complaint details for a user based on their email, excluding withdrawn complaints, utilizing a subquery to match the email to the corresponding UserID.



```
"UPDATE [User] " +
"SET " +
    "[Firstname] = ?, [Lastname] = ?, [ContactNo] = ?, " +
    "[BirthDate] = ?, [Street] = ?, [Barangay] = ?, " +
    "[City] = ?, [Province] = ?, [ProfilePicture] = ? " +
"WHERE [Email] = ?";
```

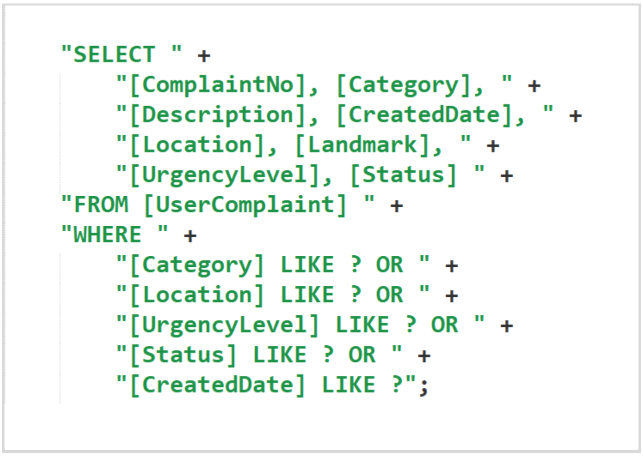
This query updates user information in the "User" table based on the provided email address.

Admin Interface Utilized Database Query



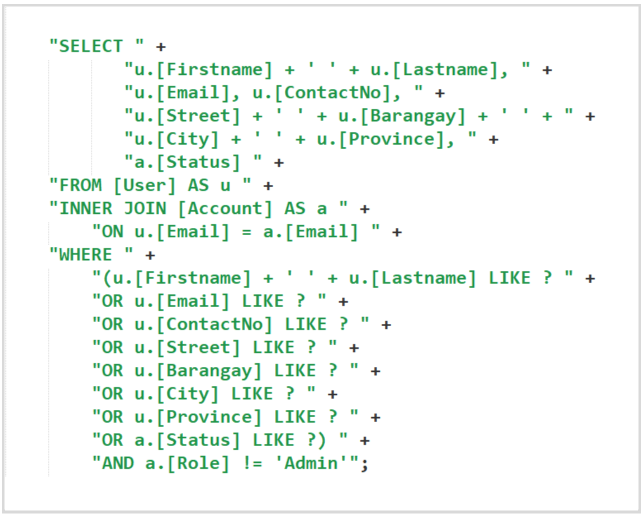
```
"SELECT " +
    "c.[Proof], " +
    "u.[Firstname] + ' ' + u.[Lastname] AS [Fullname]" +
"FROM [UserComplaint] AS c " +
"INNER JOIN [User] AS u " +
    "ON c.[UserID] = u.[UserID] " +
"WHERE c.[ComplaintNo] = ?";
```

This query retrieves the proof and the user's full name for a specific complaint number by joining the User Complaint and User tables.



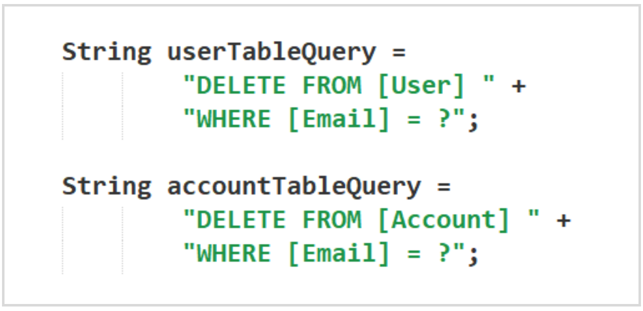
```
"SELECT " +
    "[ComplaintNo], [Category], " +
    "[Description], [CreatedDate], " +
    "[Location], [Landmark], " +
    "[UrgencyLevel], [Status] " +
"FROM [UserComplaint] " +
"WHERE " +
    "[Category] LIKE ? OR " +
    "[Location] LIKE ? OR " +
    "[UrgencyLevel] LIKE ? OR " +
    "[Status] LIKE ? OR " +
    "[CreatedDate] LIKE ?";
```

This query is used for the search feature to find complaints in the UserComplaint table based on category, location, urgency level, status, or created date.



```
"SELECT " +
        "u.[Firstname] + ' ' + u.[Lastname], " +
        "u.[Email], u.[ContactNo], " +
        "u.[Street] + ' ' + u.[Barangay] + ' ' + " +
        "u.[City] + ' ' + u.[Province], " +
        "a.[Status] " +
"FROM [User] AS u " +
"INNER JOIN [Account] AS a " +
    "ON u.[Email] = a.[Email] " +
"WHERE " +
    "(u.[Firstname] + ' ' + u.[Lastname] LIKE ? " +
    "OR u.[Email] LIKE ? " +
    "OR u.[ContactNo] LIKE ? " +
    "OR u.[Street] LIKE ? " +
    "OR u.[Barangay] LIKE ? " +
    "OR u.[City] LIKE ? " +
    "OR u.[Province] LIKE ? " +
    "OR a.[Status] LIKE ?) " +
    "AND a.[Role] != 'Admin'";
```

This query is used for the search feature to find user details from the User and Account tables based on various attributes, excluding users with the 'Admin' role.



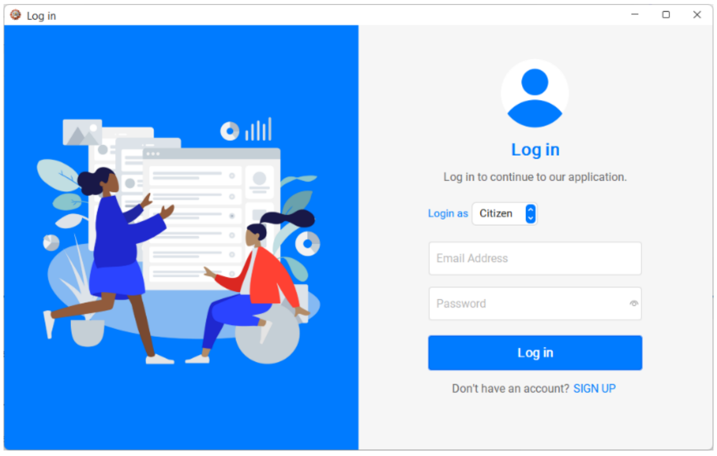
```
String userTableQuery =
        "DELETE FROM [User] " +
        "WHERE [Email] = ?";

String accountTableQuery =
        "DELETE FROM [Account] " +
        "WHERE [Email] = ?";
```

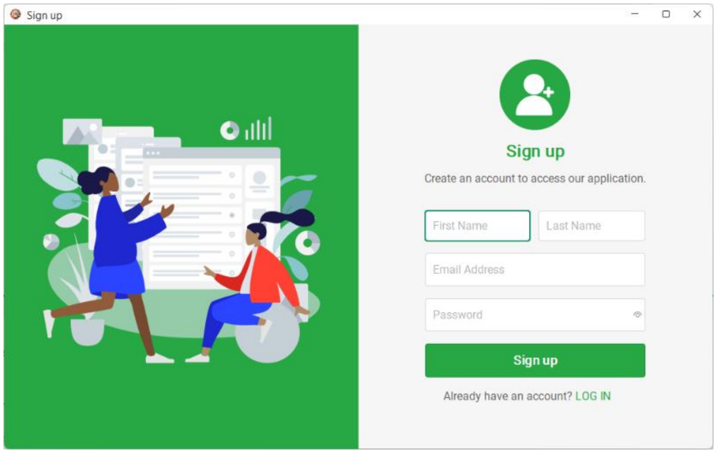
These queries are designed to delete a user's data from both the user and account tables based on their email address.

User Interface Design

User Authentication

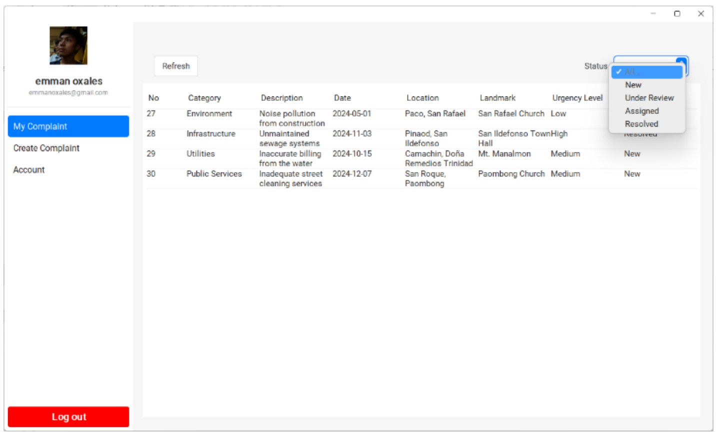

Log in
Log in
Log in to continue to our application.
Login as
Citizen
Email Address
Password
Log in
Don't have an account? SIGN UP

Login page allows users to securely access the platform by selecting their role (Admin or Citizen) from a dropdown menu and entering their email and password.


Sign up
Sign up
Create an account to access our application.
First Name
Last Name
Email Address
Password
Sign up
Already have an account? LOG IN

Signup page facilitates the creation of new user accounts by collecting essential information. Users are prompted to enter their first name, last name, email, and password to register for platform access.

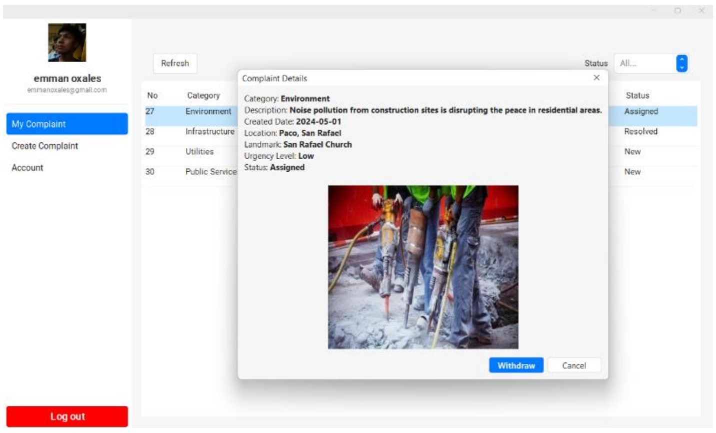
Citizen Interface



No	Category	Description	Date	Location	Landmark	Urgency Level	
27	Environment	Noise pollution from construction	2024-05-01	Paco, San Rafael	San Rafael Church	Low	[illegible]
28	Infrastructure	Unmaintained sewage systems	2024-11-03	Pinaod, San Ildefonso	San Ildefonso Town Hall	High	Resolved
29	Utilities	Inaccurate billing from the water	2024-10-15	Camachin, Doña Remedios Trinidad	Mt. Manalmon	Medium	New
30	Public Services	Inadequate street cleaning services	2024-12-07	San Roque, Paombong	Paombong Church	Medium	New

My Complaint interface features a table listing all user complaints with details like complaint number, category, description, date, location, landmark, urgency level, and status.

Sorting and Filtering: Users can sort and filter complaints using the table's sorting options and the status filter combo box.

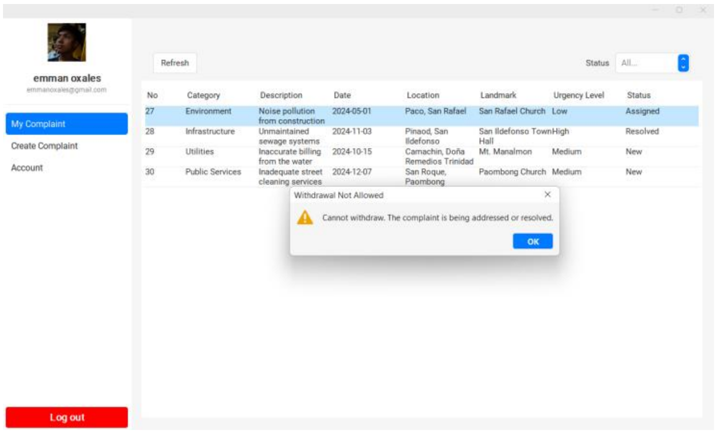


No	Category	Status
27	Environment	Assigned
28	Infrastructure	Resolved
29	Utilities	New
30	Public Service	New

Complaint Details

Category: **Environment**
Description: **Noise pollution from construction sites is disrupting the peace in residential areas.**
Created Date: **2024-05-01**
Location: **Paco, San Rafael**
Landmark: **San Rafael Church**
Urgency Level: **Low**
Status: **Assigned**

Popup Dialog: Double-clicking a complaint opens a detailed view in a popup, showing all relevant details and an associated image, if available.

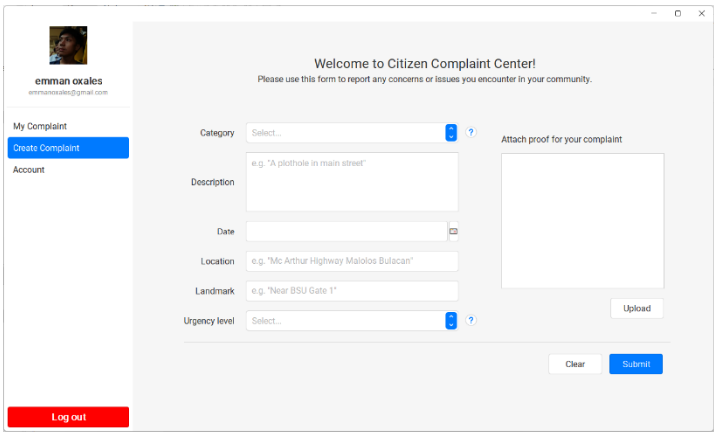


No	Category	Description	Date	Location	Landmark	Urgency Level	Status
27	Environment	Noise pollution from construction	2024-05-01	Paco, San Rafael	San Rafael Church	Low	Assigned
28	Infrastructure	Unmaintained sewage systems	2024-11-03	Pinaod, San Ildefonso	San Ildefonso Town Hall	High	Resolved
29	Utilities	Inaccurate billing from the water	2024-10-15	Camachin, Doña Remedios Trinidad	Mt. Manalmon	Medium	New
30	Public Services	Inadequate street cleaning services	2024-12-07	San Roque, Paombong	Paombong Church	Medium	New

Withdrawal Not Allowed

Cannot withdraw. The complaint is being addressed or resolved.

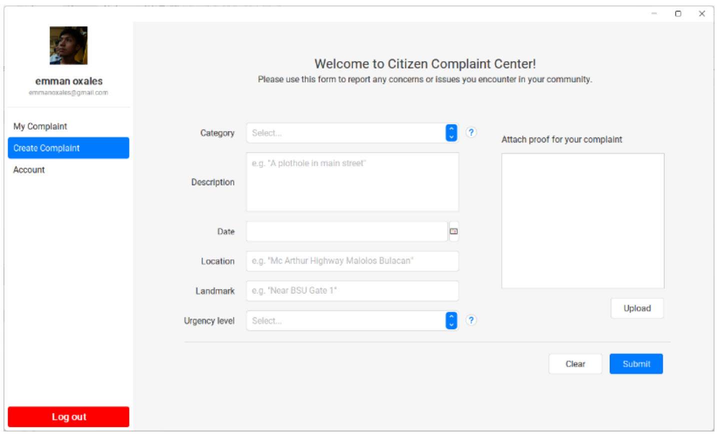
Withdrawal Option: Users can withdraw the complaint from the detailed view if its status is "New" or "Under Review". The system restricts withdrawal for complaints that are "Assigned" or "Resolved", displaying a warning message if a withdrawal is attempted on these.



Welcome to Citizen Complaint Center!
Please use this form to report any concerns or issues you encounter in your community.

Category: Select...
Description: e.g. "A plothole in main street"
Date:
Location: e.g. "Mc Arthur Highway Malolos Bulacan"
Landmark: e.g. "Near BSU Gate 1"
Urgency level: Select...
Attach proof for your complaint

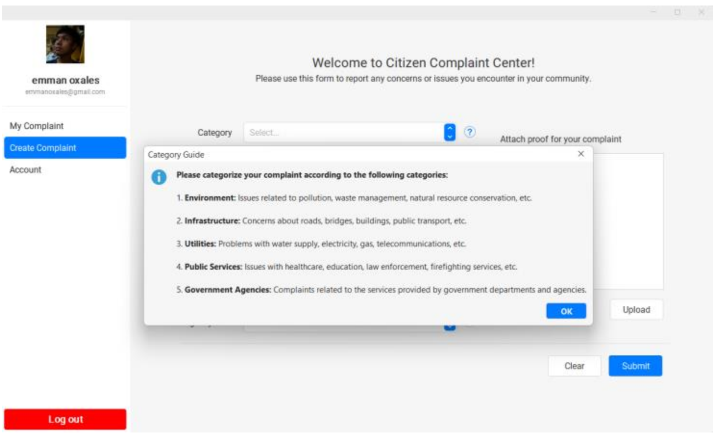
Popup Dialog: Double-clicking a complaint opens a detailed view in a popup, showing all relevant details and an associated image, if available.



Welcome to Citizen Complaint Center!
Please use this form to report any concerns or issues you encounter in your community.

Category: Select...
Description: e.g. "A plothole in main street"
Date:
Location: e.g. "Mc Arthur Highway Malolos Bulacan"
Landmark: e.g. "Near BSU Gate 1"
Urgency level: Select...
Attach proof for your complaint

Create Complaint interface have a complaint submission with clear prompts for essential details, including category, description, and location. It enables easy image proof uploads and provides real-time validation.



Welcome to Citizen Complaint Center!

Please use this form to report any concerns or issues you encounter in your community.

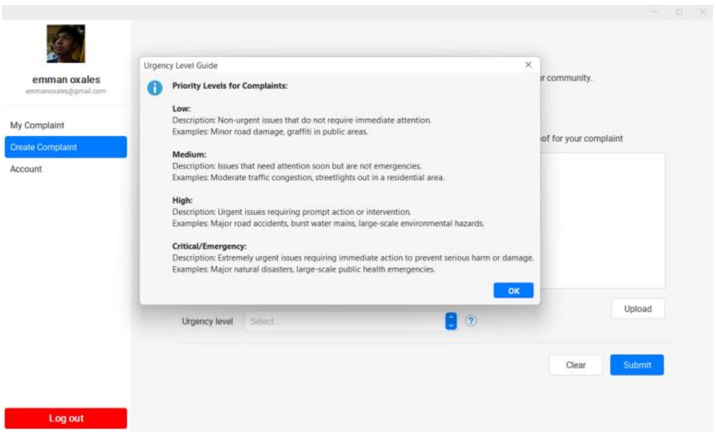
Category Guide

Please categorize your complaint according to the following categories:

1. **Environment:** Issues related to pollution, waste management, natural resource conservation, etc.
2. **Infrastructure:** Concerns about roads, bridges, buildings, public transport, etc.
3. **Utilities:** Problems with water supply, electricity, gas, telecommunications, etc.
4. **Public Services:** Issues with healthcare, education, law enforcement, firefighting services, etc.
5. **Government Agencies:** Complaints related to the services provided by government departments and agencies.

OK

Assistance for Category Selection: Users can click the help button while choosing a category to access detailed explanations. This aids users in selecting the most suitable category for their complaint.



Urgency Level Guide

Priority Levels for Complaints:

Low:
Description: Non-urgent issues that do not require immediate attention.
Examples: Minor road damage, graffiti in public areas.

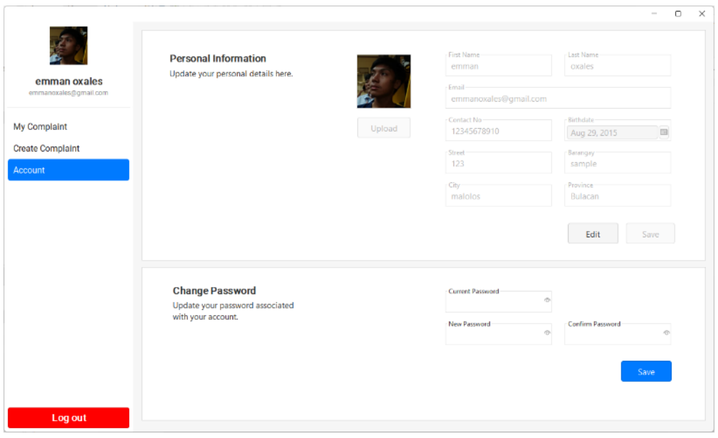
Medium:
Description: Issues that need attention soon but are not emergencies.
Examples: Moderate traffic congestion, streetlights out in a residential area.

High:
Description: Urgent issues requiring prompt action or intervention.
Examples: Major road accidents, burst water mains, large-scale environmental hazards.

Critical/Emergency:
Description: Extremely urgent issues requiring immediate action to prevent serious harm or damage.
Examples: Major natural disasters, large-scale public health emergencies.

OK

Assistance for Urgency Level Selection: The help button provides brief explanations for each urgency level option, aiding users in making informed choices for their complaints.



Personal Information
Update your personal details here.

First Name: emman	Last Name: oxales
Email: emmanoxales@gmail.com	
Contact No: 12345678910	Birthdate: Aug 29, 2015
Street: 123	Barangay: sample
City: malolos	Province: Bulacan

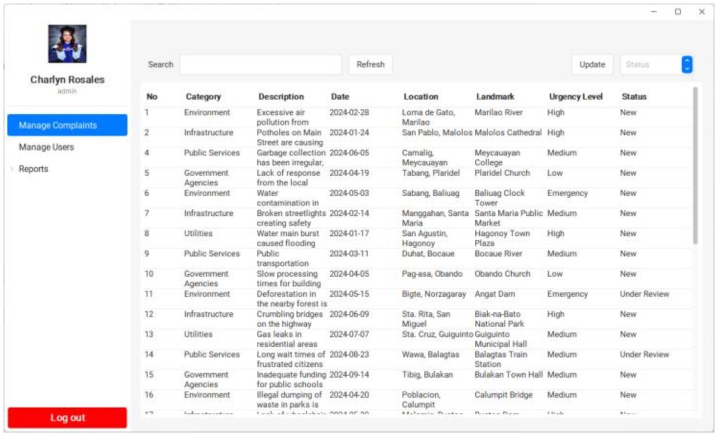
Change Password
Update your password associated with your account.

Current Password
New Password
Confirm Password

User Information: Users can easily edit personal details such as name, contact number, birthdate, and address through input fields. They can also personalize their account by uploading a profile picture.

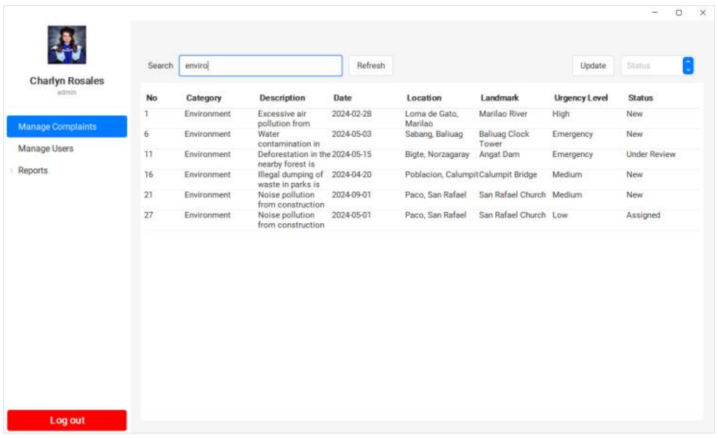
Change Password: Users can securely change their password by providing their current password, a new password, and confirming the new password. The interface ensures the current password is correct and that the new password matches the confirmation before saving changes.

Admin Interface



No	Category	Description	Date	Location	Landmark	Urgency Level	Status
1	Environment	Excessive air pollution from	2024-02-28	Loma de Gato, Marilao	Marilao River	High	New
2	Infrastructure	Potholes on Main Street are causing	2024-01-24	San Pablo, Malolos	Malolos Cathedral	High	New
4	Public Services	Garbage collection has been irregular,	2024-06-05	Camalig, Meycauayan	Meycauayan College	Medium	New
5	Government Agencies	Lack of response from the local	2024-04-19	Tabang, Plaridel	Plaridel Church	Low	New
6	Environment	Water contamination in	2024-05-03	Sabang, Baliuag	Baliuag Clock Tower	Emergency	New
7	Infrastructure	Broken streetlights creating safety	2024-02-14	Manggahan, Santa Maria	Santa Maria Public Market	Medium	New
8	Utilities	Water main burst caused flooding	2024-01-17	San Agustin, Hagonoy	Hagonoy Town Plaza	High	New
9	Public Services	Public transportation	2024-03-11	Duhat, Bocaue	Bocaue River	Medium	New
10	Government Agencies	Slow processing times for building	2024-04-05	Pag-asa, Obando	Obando Church	Low	New
11	Environment	Deforestation in the nearby forest is	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review
12	Infrastructure	Crumbling bridges on the highway	2024-06-09	Sta. Rita, San Miguel	Biak-na-Bato National Park	High	New
13	Utilities	Gas leaks in residential areas	2024-07-07	Sta. Cruz, Guiguinto	Guiguinto Municipal Hall	Medium	New
14	Public Services	Long wait times of frustrated citizens	2024-08-23	Wawa, Balagtas	Balagtas Train Station	Medium	Under Review
15	Government Agencies	Inadequate funding for public schools	2024-09-14	Tibig, Bulakan	Bulakan Town Hall	Medium	New
16	Environment	Illegal dumping of waste in parks is	2024-04-20	Poblacion, Calumpit	Calumpit Bridge	Medium	New

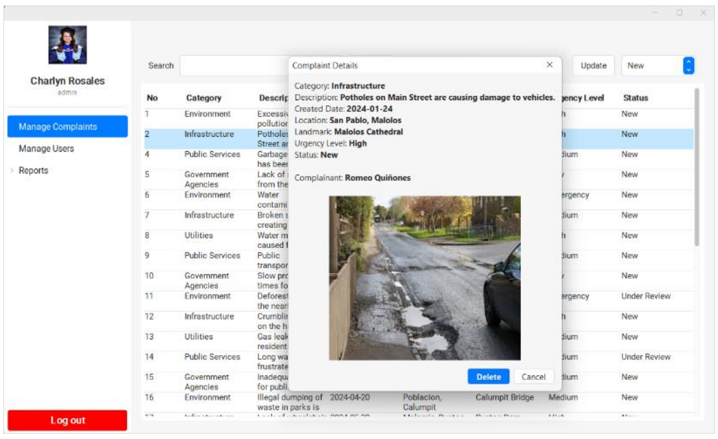
Displays a list of user complaints, populated from the database, showing details such as complaint number, category, description, date, location, landmark, urgency level, and status.



Search: enviro

No	Category	Description	Date	Location	Landmark	Urgency Level	Status
1	Environment	Excessive air pollution from	2024-02-28	Loma de Gato, Marilao	Marilao River	High	New
6	Environment	Water contamination in	2024-05-03	Sabang, Baliuag	Baliuag Clock Tower	Emergency	New
11	Environment	Deforestation in the nearby forest is	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review
16	Environment	Illegal dumping of waste in parks is	2024-04-20	Poblacion, Calumpit	Calumpit Bridge	Medium	New
21	Environment	Noise pollution from construction	2024-09-01	Paco, San Rafael	San Rafael Church	Medium	New
27	Environment	Noise pollution from construction	2024-05-01	Paco, San Rafael	San Rafael Church	Low	Assigned

Allows users to search for complaints by category, location, urgency level, status, or created date.



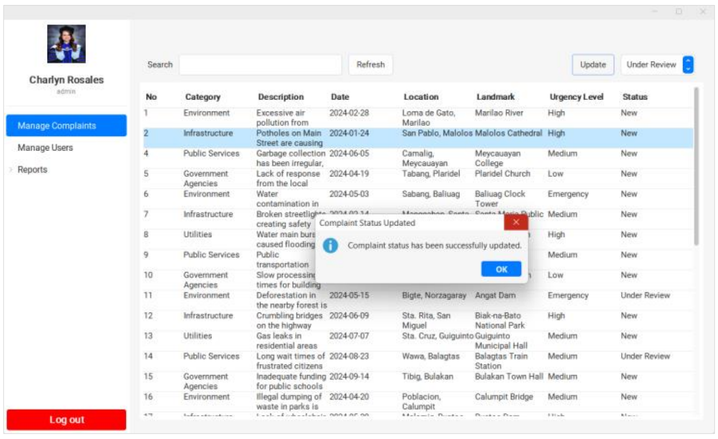
Complaint Details

Category: **Infrastructure**
Description: **Potholes on Main Street are causing damage to vehicles.**
Created Date: **2024-01-24**
Location: **San Pablo, Malolos**
Landmark: **Malolos Cathedral**
Urgency Level: **High**
Status: **New**

Complainant: **Romeo Quiñones**

No	Category	Descrip	ency Level	Status
1	Environment	Excessiv pollution	h	New
2	Infrastructure	Potholes Street ar	h	New
4	Public Services	Garbage has beer	dium	New
5	Government Agencies	Lack of from the	/	New
6	Environment	Water contami	ergency	New
7	Infrastructure	Broken s creating	dium	New
8	Utilities	Water m caused f	h	New
9	Public Services	Public transpor	dium	New
10	Government Agencies	Slow prc times fo	/	New
11	Environment	Deforest the near	ergency	Under Review
12	Infrastructure	Crumblir on the h	h	New
13	Utilities	Gas leak resident	dium	New
14	Public Services	Long wa frustrate	dium	Under Review
15	Government Agencies	Inadequa for publi	dium	New
16	Environment	Illegal dumping of waste in parks is 2024-04-20 Poblacion, Calumpit Calumpit Bridge	Medium	New

Clicking a complaint row once updates the status combo box; double-clicking displays detailed complaint information and an image and allows users to delete the complaint.

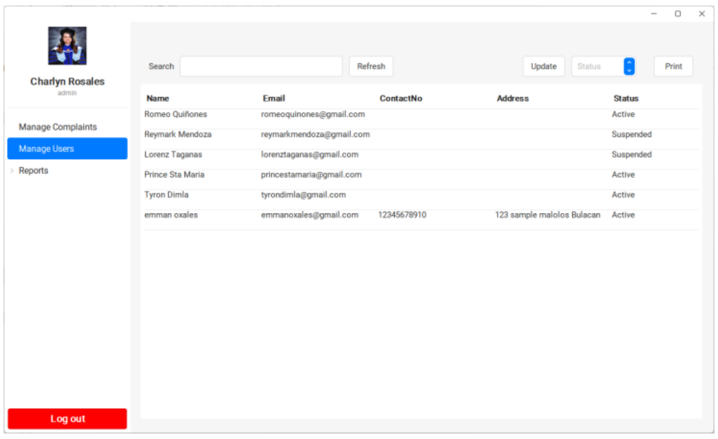


Complaint Status Updated

Complaint status has been successfully updated.

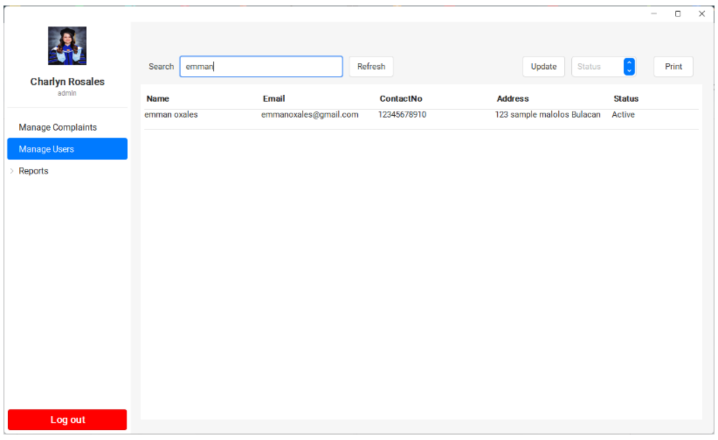
No	Category	Description	Date	Location	Landmark	Urgency Level	Status
1	Environment	Excessive air pollution from	2024-02-28	Loma de Gato, Marilao	Marilao River	High	New
2	Infrastructure	Potholes on Main Street are causing	2024-01-24	San Pablo, Malolos	Malolos Cathedral	High	New
4	Public Services	Garbage collection has been irregular,	2024-06-05	Camalig, Meycauayan	Meycauayan College	Medium	New
5	Government Agencies	Lack of response from the local	2024-04-19	Tabang, Plaridel	Plaridel Church	Low	New
6	Environment	Water contamination in	2024-05-03	Sabang, Baliuag	Baliuag Clock Tower	Emergency	New
7	Infrastructure	Broken streetlights creating safety	[illegible]	[illegible]	[illegible]	Medium	New
8	Utilities	Water main burs caused flooding	[illegible]	[illegible]	[illegible]	High	New
9	Public Services	Public transportation	[illegible]	[illegible]	[illegible]	Medium	New
10	Government Agencies	Slow processing times for building	[illegible]	[illegible]	[illegible]	Low	New
11	Environment	Deforestation in the nearby forest is	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review
12	Infrastructure	Crumbling bridges on the highway	2024-06-09	Sta. Rita, San Miguel	Biak-na-Bato National Park	High	New
13	Utilities	Gas leaks in residential areas	2024-07-07	Sta. Cruz, Guiguinto	Guiguinto Municipal Hall	Medium	New
14	Public Services	Long wait times of frustrated citizens	2024-08-23	Wawa, Balagtas	Balagtas Train Station	Medium	Under Review
15	Government Agencies	Inadequate funding for public schools	2024-09-14	Tibig, Bulakan	Bulakan Town Hall	Medium	New
16	Environment	Illegal dumping of waste in parks is	2024-04-20	Poblacion, Calumpit	Calumpit Bridge	Medium	New

Users can update the status of a selected complaint.



Name	Email	ContactNo	Address	Status
Romeo Quiñones	romeoquinones@gmail.com			Active
Reymark Mendoza	reymarkmendoza@gmail.com			Suspended
Lorenz Taganas	lorenztaganas@gmail.com			Suspended
Prince Sta Maria	princestamaria@gmail.com			Active
Tyron Dimla	tyrondimla@gmail.com			Active
emman oxales	emmanoxales@gmail.com	12345678910	123 sample malolos Bulacan	Active

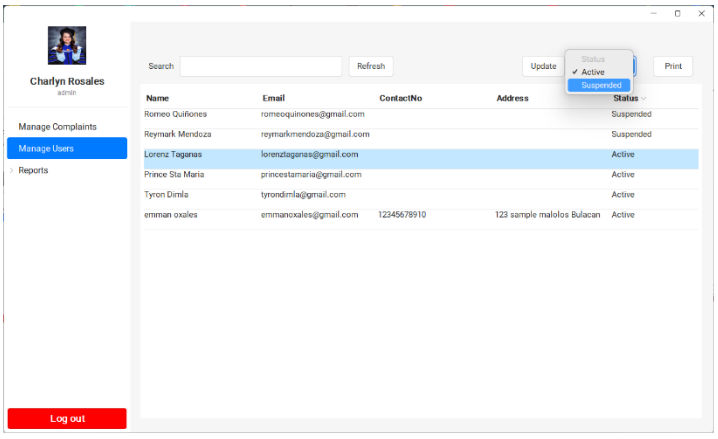
Displays user details such as full name, email, contact number, address, and account status, excluding admin accounts.



Search: emman

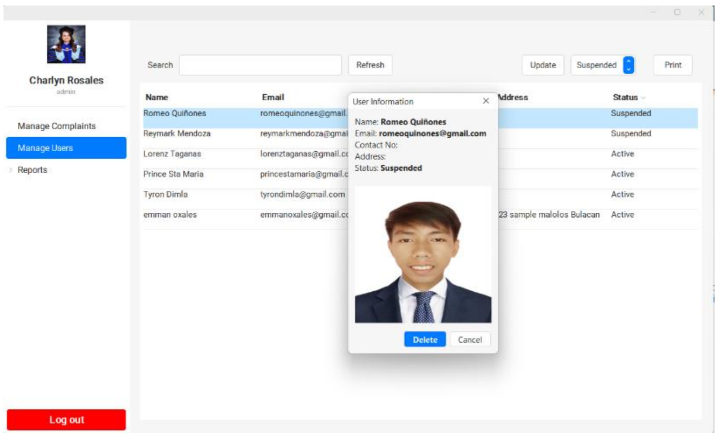
Name	Email	ContactNo	Address	Status
emman oxales	emmanoxales@gmail.com	12345678910	123 sample malolos Bulacan	Active

Search and Filter: Users can search for specific users by their name, email, contact number, or address, and filter the table based on these criteria.



Name	Email	ContactNo	Address	Status
Romeo Quiñones	romeoquinones@gmail.com			Suspended
Reymark Mendoza	reymarkmendoza@gmail.com			Suspended
Lorenz Taganas	lorenztaganas@gmail.com			Active
Prince Sta Maria	princestamaria@gmail.com			Active
Tyron Dimla	tyrondimla@gmail.com			Active
emman oxales	emmanoxales@gmail.com	12345678910	123 sample malolos Bulacan	Active

Update User Status: Administrators can change the status of a user (e.g., Active, Inactive) directly from the table.

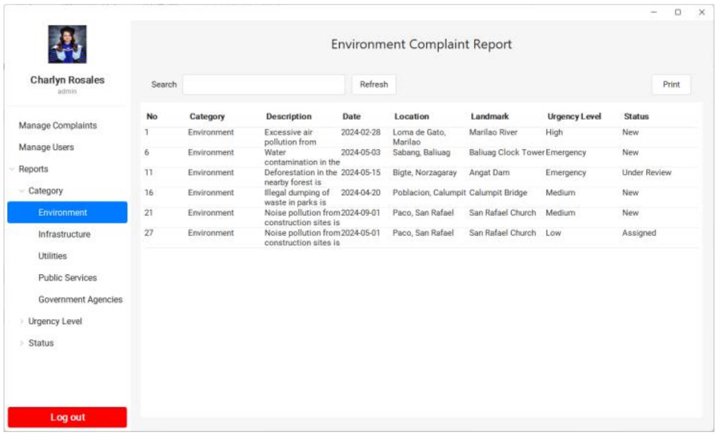


Name	Email	Address	Status
Romeo Quiñones	romeoquinones@gmail.		Suspended
Reymark Mendoza	reymarkmendoza@gmai		Suspended
Lorenz Taganas	lorenztaganas@gmail.c		Active
Prince Sta Maria	princestamaria@gmail.c		Active
Tyron Dimla	tyrondimla@gmail.com		Active
emman oxales	emmanoxales@gmail.c	23 sample malolos Bulacan	Active

Name: **Romeo Quiñones**
Email: **romeoquinones@gmail.com**
Contact No:
Address:
Status: **Suspended**

View and Manage User Details: Double-clicking a row shows detailed information about the selected user, including a profile picture if available. This interface also provides an option to delete a user account, with a confirmation prompt to ensure intentional deletion.

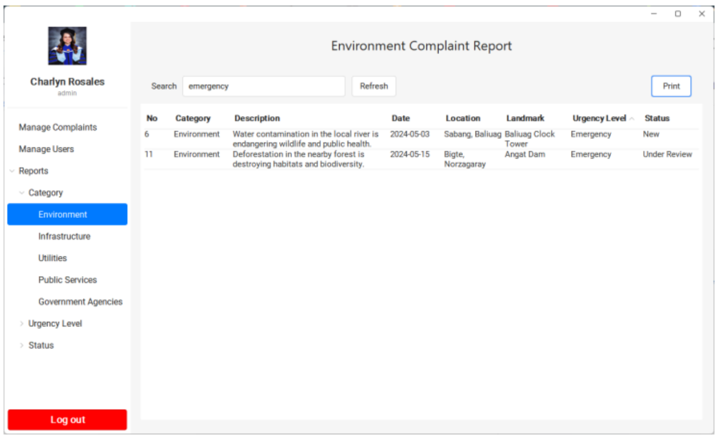
Reports



Environment Complaint Report

No	Category	Description	Date	Location	Landmark	Urgency Level	Status
1	Environment	Excessive air pollution from	2024-02-28	Loma de Gato, Marilao	Marilao River	High	New
6	Environment	Water contamination in the	2024-05-03	Sabang, Baliuag	Baliuag Clock Tower	Emergency	New
11	Environment	Deforestation in the nearby forest is	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review
16	Environment	Illegal dumping of waste in parks is	2024-04-20	Poblacion, Calumpit	Calumpit Bridge	Medium	New
21	Environment	Noise pollution from construction sites is	2024-09-01	Paco, San Rafael	San Rafael Church	Medium	New
27	Environment	Noise pollution from construction sites is	2024-05-01	Paco, San Rafael	San Rafael Church	Low	Assigned

Fetches and displays environment-related complaints from the database in a table format.

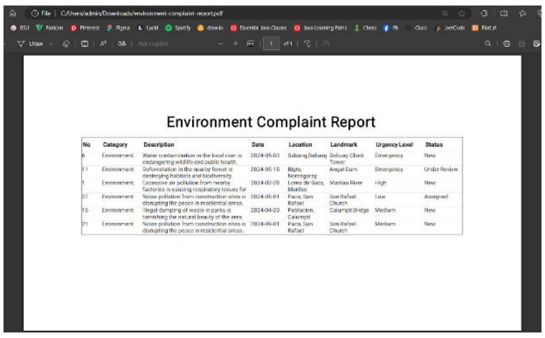


Environment Complaint Report

Search: emergency

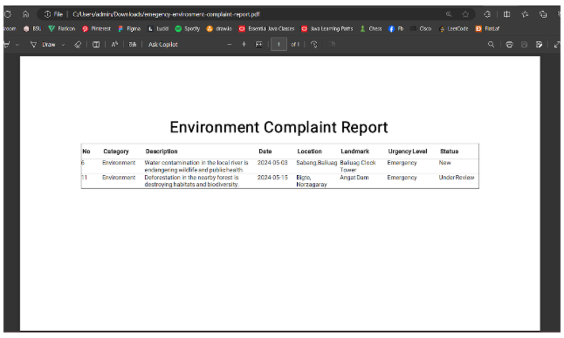
No	Category	Description	Date	Location	Landmark	Urgency Level	Status
6	Environment	Water contamination in the local river is endangering wildlife and public health.	2024-05-03	Sabang, Baliuag	Baliuag Clock Tower	Emergency	New
11	Environment	Deforestation in the nearby forest is destroying habitats and biodiversity.	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review

Users can search and filter complaints (category, location, etc.) with live updates. Print reports of filtered results.


C:/Users/admin/Downloads/environment-complaint-report.pdf

Environment Complaint Report

No	Category	Description	Date	Location	Landmark	Urgency Level	Status
6	Environment	Water contamination in the local river is endangering wildlife and public health.	2024-05-03	Sabang,Baliuag	Baliuag Clock Tower	Emergency	New
11	Environment	Deforestation in the nearby forest is destroying habitats and biodiversity.	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review
1	Environment	Excessive air pollution from nearby factories is causing respiratory issues for	2024-02-28	Loma de Gato, Marilao	Marilao River	High	New
27	Environment	Noise pollution from construction sites is disrupting the peace in residential areas.	2024-05-01	Paco, San Rafael	San Rafael Church	Low	Assigned
15	Environment	Illegal dumping of waste in parks is tarnishing the natural beauty of the area.	2024-04-20	Poblacion, Calumpit	Calumpit Bridge	Medium	New
21	Environment	Noise pollution from construction sites is disrupting the peace in residential areas.	2024-09-01	Paco, San Rafael	San Rafael Church	Medium	New


Environment Complaint Report

No	Category	Description	Date	Location	Landmark	Urgency Level	Status
6	Environment	Water contamination in the local river is endangering wildlife and public health.	2024-05-03	Sabang,Baliuag	Baliuag Clock Tower	Emergency	New
11	Environment	Deforestation in the nearby forest is destroying habitats and biodiversity.	2024-05-15	Bigte, Norzagaray	Angat Dam	Emergency	Under Review

Reflection

Building our project system as a student has been quite a ride! From brainstorming ideas to coding features, every part of the process taught me tons about technology and teamwork. Working with my classmates, I've not only improved my coding skills but also learned how to communicate and work together better. Seeing our system go from ideas to a real working application has been amazing, and I'm really proud of what we've accomplished together.

(Quiñones Jr., Romeo M.)

Through this project, I learned the significance of teamwork and communication, especially when creating wireframes and database models. Each member's contribution, from designing user interfaces to developing a database was crucial in building a system. This experience not only enhanced my technical skills but also taught me the value of coordination.

(Taganas, Lorenz Romeo O.)

From a practical point of view, it was pretty exciting to design and develop a program that connects to a SQL database using Java, applying the principles of OOP, and designing a GUI. Any further activity, such as setting up a connection with the database, writing SQL queries, or designing an interface easy for users to interact with, was challenging at each stage of the development process: spotting and fixing connection debugging issues, and making sure to get a seamless user interface. Such satisfactions are definitely aimed at consolidating the importance of planning and thoughtful design. This project helped me enhance not only my technical skills but learn to be persistent in the process of software development by looking into details.

(Sta Maria, Lorenz Romeo O.)

Working on the citizen complaint management system has been a valuable learning experience, Designing the system required a deep understanding of the complaint lifecycle, from submission to resolution. Moreover, the project highlighted the significance of collaboration and communication among team members. This process reinforced the importance of clear documentation and structured workflows to ensure that everyone remains aligned with the project goals. Overall, this experience has not only expanded my technical proficiency but also improved my project management and problem-solving skills.

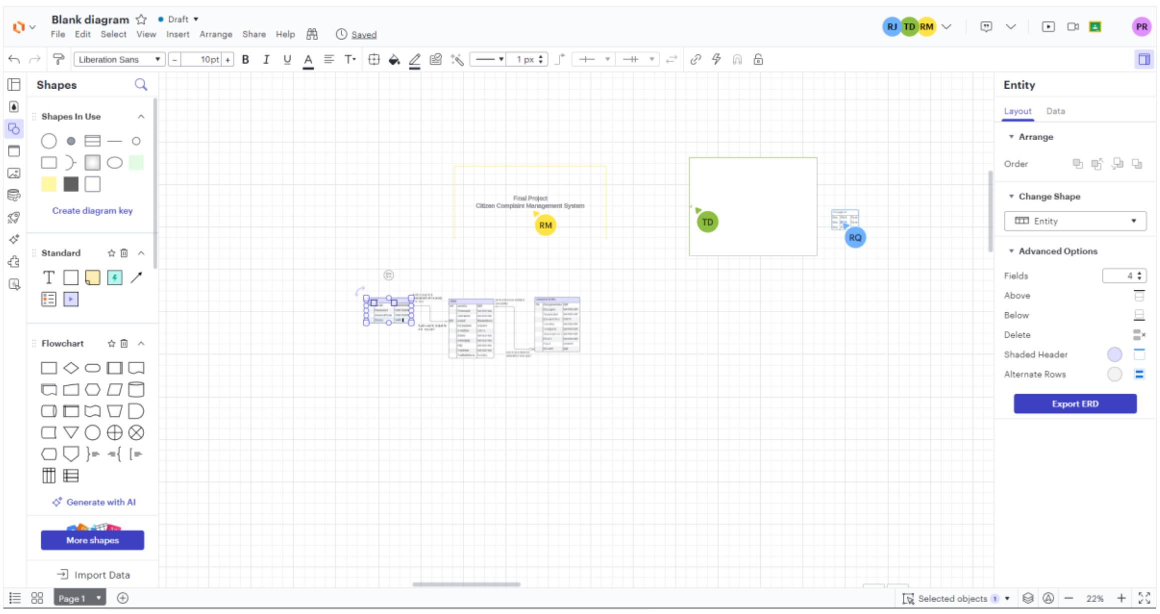
(Mendoza, Reymark M.)

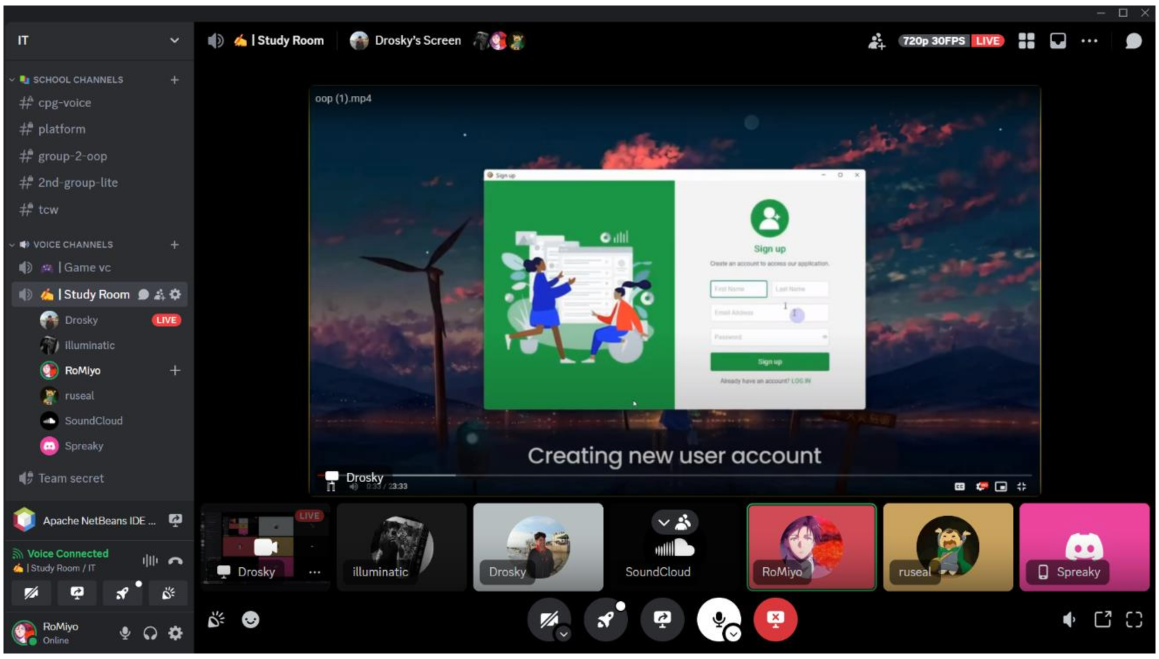
Citizen complaint management system (CCMS) has been an eye-opening experience. It's clear these systems hold great potential to reshape citizen-government interaction. On one hand, I see the power of empowering citizens with a clear platform to voice concerns and track progress. On the other hand, I recognize challenges like ensuring digital equity and safeguarding sensitive data. The key takeaway for me is that a successful CCMS requires a delicate balance between efficiency and human connection. While automation can make processes, citizen trust ultimately hinges on a system that builds open communication and delivers meaningful resolutions.

(Dimla, Tyron B.)

Collaboration

Anong nangyari sa kaniya pre?
Nahimatay sa actual coding kanina


Blank diagram
Draft
File Edit Select View Insert Arrange Share Help
Saved
Shapes
Shapes In Use
Create diagram key
Standard
Flowchart
Generate with AI
More shapes
Import Data
Final Project
Citizen Complaint Management System
RM
TD
RQ
Entity
Layout Data
Arrange
Order
Change Shape
Entity
Advanced Options
Fields
Above
Below
Delete
Shaded Header
Alternate Rows
Export ERD
Page 1
Selected objects
22%


IT
SCHOOL CHANNELS
cpg-voice
platform
group-2-oop
2nd-group-lite
tcw
VOICE CHANNELS
| Game vc
| Study Room
Drosky LIVE
illuminatic
RoMiyo
ruseal
SoundCloud
Spreaky
Team secret
Apache NetBeans IDE ...
Voice Connected
| Study Room / IT
RoMiyo
Online
| Study Room
Drosky's Screen
720p 30FPS LIVE
oop (1).mp4
Sign up
Creating new user account
Drosky
illuminatic
Drosky
SoundCloud
RoMiyo
ruseal
Spreaky

Source Code

Login

```java
package project.authentication;

import com.formdev.flatlaf.FlatClientProperties;
import com.formdev.flatlaf.themes.FlatMacLightLaf;
import java.awt.Color;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import project.admin.AdminFrame;
import project.citizen.CitizenFrame;
import project.database.Database;

/**
 * Represents the login frame of the application where users can log in.
 * This frame allows users to enter their credentials and log in to the
system.
 * It provides a user interface for authentication purposes.
 * @since 4/25/2024
 */
public class LoginFrame extends javax.swing.JFrame {

    // Variables for database connection and user credentials
    private Connection connection;
    private String role;
    public static String email;
    private String password;
    private int attemptLeft = 3;

    public LoginFrame() {
        initComponents();
        try {
            connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(LoginFrame.class.getName()).log(Level.SEVERE,
ex.getMessage(), ex);
        }

        loadFrameIcon();

        loadPanelIcon();

        customizeComponentProperties();
    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        leftPanel = new javax.swing.JPanel();
        loginIcon = new javax.swing.JLabel();
        rightPanel = new javax.swing.JPanel();
        loginPanel = new javax.swing.JPanel();
        userIcon = new javax.swing.JLabel();
        headerTitle = new javax.swing.JLabel();
        description = new javax.swing.JLabel();
        emailTextField = new javax.swing.JTextField();
        passwordField = new javax.swing.JPasswordField();
        loginButton = new javax.swing.JButton();
        question = new javax.swing.JLabel();
        signupLink = new javax.swing.JButton();
        loginAs = new javax.swing.JLabel();
        roleComboBox = new javax.swing.JComboBox<>();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Log in");
        getContentPane().setLayout(new java.awt.GridLayout(1, 0));

        leftPanel.setBackground(new java.awt.Color(0, 123, 255));
        leftPanel.setPreferredSize(new java.awt.Dimension(500, 600));

        javax.swing.GroupLayout leftPanelLayout = new
javax.swing.GroupLayout(leftPanel);
        leftPanel.setLayout(leftPanelLayout);
        leftPanelLayout.setHorizontalGroup(

leftPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
leftPanelLayout.createSequentialGroup()
                .addGap(46, 46, 46)
                .addComponent(loginIcon,
javax.swing.GroupLayout.PREFERRED_SIZE, 400,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        leftPanelLayout.setVerticalGroup(

leftPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
leftPanelLayout.createSequentialGroup()
                .addGap(129, 129, 129)
                .addComponent(loginIcon,
javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(121, 121, 121))
        );

        getContentPane().add(leftPanel);

        rightPanel.setBackground(new java.awt.Color(255, 255, 255));
        rightPanel.setPreferredSize(new java.awt.Dimension(500, 600));
        rightPanel.setLayout(new java.awt.CardLayout());

        java.awt.GridBagLayout loginPanelLayout = new
java.awt.GridBagLayout();
```

```
        loginPanelLayout.columnWidths = new int[] {0, 50, 0, 50, 0, 50, 0, 50,
0};
        loginPanelLayout.rowHeights = new int[] {0, 15, 0, 15, 0, 15, 0, 15,
0, 15, 0, 15, 0, 15, 0, 15, 0, 15, 0};
        loginPanel.setLayout(loginPanelLayout);

        userIcon.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = 9;
        loginPanel.add(userIcon, gridBagConstraints);

        headerTitle.setFont(new java.awt.Font("Roboto", 1, 24)); // NOI18N
        headerTitle.setForeground(new java.awt.Color(0, 123, 255));
        headerTitle.setText("Log in");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = 9;
        loginPanel.add(headerTitle, gridBagConstraints);

        description.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        description.setForeground(new java.awt.Color(102, 102, 102));
        description.setText("Log in to continue to our application.");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = 9;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 10, 0);
        loginPanel.add(description, gridBagConstraints);

        emailTextField.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        emailTextField.setForeground(new java.awt.Color(51, 51, 51));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.gridwidth = 9;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        loginPanel.add(emailTextField, gridBagConstraints);

        passwordField.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        passwordField.setForeground(new java.awt.Color(51, 51, 51));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.gridwidth = 9;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 5, 0);
        loginPanel.add(passwordField, gridBagConstraints);

        loginButton.setBackground(new java.awt.Color(0, 123, 255));
        loginButton.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        loginButton.setForeground(new java.awt.Color(255, 255, 255));
        loginButton.setText("Log in");
        loginButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
```

```
        loginButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 12;
        gridBagConstraints.gridwidth = 9;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        loginPanel.add(loginButton, gridBagConstraints);

        question.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        question.setForeground(new java.awt.Color(102, 102, 102));
        question.setText("Don't have an account?");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 14;
        gridBagConstraints.gridwidth = 5;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.LINE_START;
        gridBagConstraints.insets = new java.awt.Insets(0, 34, 0, 3);
        loginPanel.add(question, gridBagConstraints);

        signupLink.setBackground(new java.awt.Color(242, 242, 242));
        signupLink.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        signupLink.setForeground(new java.awt.Color(0, 123, 255));
        signupLink.setText("SIGN UP");
        signupLink.setToolTipText("Go to sign up");
        signupLink.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1,
1, 1));
        signupLink.setBorderPainted(false);
        signupLink.setContentAreaFilled(false);
        signupLink.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        signupLink.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                showSignupFrame(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 14;
        gridBagConstraints.gridwidth = 5;
        gridBagConstraints.ipadx = 5;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.LINE_END;
        gridBagConstraints.insets = new java.awt.Insets(0, 9, 0, 34);
        loginPanel.add(signupLink, gridBagConstraints);

        loginAs.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        loginAs.setForeground(new java.awt.Color(0, 123, 255));
        loginAs.setText("Login as");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 6, 0);
        loginPanel.add(loginAs, gridBagConstraints);

        roleComboBox.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        roleComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Citizen", "Admin" }));
        roleComboBox.setPreferredSize(new java.awt.Dimension(92, 35));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 5;
        gridBagConstraints.ipadx = 3;
        gridBagConstraints.insets = new java.awt.Insets(0, 14, 6, 0);
        loginPanel.add(roleComboBox, gridBagConstraints);

        rightPanel.add(loginPanel, "card3");

        getContentPane().add(rightPanel);

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>


    /**
     * Loads the icon for the frame title.
     */
    private void loadFrameIcon() {
        final ImageIcon titlePaneIcon = new
ImageIcon(getClass().getResource("/icon/CICT-logo-icon.png"));
        this.setIconImage(titlePaneIcon.getImage());
    }

    /**
     * Loads icons for various panels in the login frame.
     */
    private void loadPanelIcon() {
        loginIcon.setIcon(new ImageIcon(getClass().getResource("/icon/left-
panel-icon.png")));
        userIcon.setIcon(new
ImageIcon(getClass().getResource("/icon/user.png")));
    }

    /**
     * Customizes the properties of GUI components using FlatLaf styles.
     */
    private void customizeComponentProperties() {

//emailTextField.putClientProperty(FlatClientProperties.TEXT_FIELD_LEADING_ICO
N,
        //        new ImageIcon(getClass().getResource("/mail.png")));


emailTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT, "Email
Address");
        emailTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;" +
                "showClearButton: true;");


//passwordField.putClientProperty(FlatClientProperties.TEXT_FIELD_LEADING_ICON
,
        //        new ImageIcon(getClass().getResource("/pass.png")));
        passwordField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
"Password");
        passwordField.putClientProperty(FlatClientProperties.STYLE,
                "showRevealButton: true;" +
                "showCapsLock: true;" +
                "margin: 3, 10, 3, 10;");

        signupLink.putClientProperty(FlatClientProperties.STYLE,
                "hoverForeground: #4d97ff;" +
                "pressedForeground: #0058cc;");

        roleComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 0, 10, 0, 10;");
    }

    /**
    * Opens the sign-up frame when the corresponding button is clicked.
    * This method creates a new instance of the SignupFrame class and makes it
visible.
    * It also disposes of the current login frame.
    */
    private void showSignupFrame(java.awt.event.ActionEvent evt) {

        new SignupFrame().setVisible(true);
        dispose();

    }


    /**
     * Handles login button action.
     * Performs login operation when the login button is clicked.
     * Retrieves user input, verifies credentials, and opens the Citizen
application frame if successful.
     * Displays error message if login fails.
     */
    private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {

        role = roleComboBox.getSelectedItem().toString();
        email = emailTextField.getText();
        password = String.valueOf(passwordField.getPassword());

        if (isInputFieldsEmpty()) {
            JOptionPane.showMessageDialog(
                    rightPanel,
                    "Fill out needed information.",
                    "Reminder",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }


       String selectQuery = "SELECT * FROM Account WHERE Email = ?";
       try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{
           pst.setString(1, email);
           ResultSet rs = pst.executeQuery();
```

```
            while (rs.next()) {
                 String dbEmail = rs.getString("Email");
                 String dbPassword = rs.getString("Password");
                 String dbRole = rs.getString("Role");
                 String dbStatus = rs.getString("Status");

                 if (isCredentialMatch(dbEmail, dbPassword, dbRole) &&
!dbStatus.equals("Suspended")) {
                     JOptionPane.showMessageDialog(
                             rightPanel,
                             "Log in succesful.",
                             "Login",
                             JOptionPane.INFORMATION_MESSAGE);

                     currentLoginAccount(role);

                     dispose();

                     return;
                 }

                 if (isCredentialMatch(dbEmail, dbPassword, dbRole) &&
dbStatus.equals("Suspended")) {
                     JOptionPane.showMessageDialog(
                             rightPanel,
                             "Your account has been suspended.",
                             "Login Failed",
                             JOptionPane.ERROR_MESSAGE);

                     return;
                 }

             }



             attemptLeft--;

             JOptionPane.showMessageDialog(
                     rightPanel,
                     "This credential doesn't match our record. \n" +
                     "You have " + attemptLeft + " attempts remaining.",
                     "Log in failed",
                     JOptionPane.ERROR_MESSAGE);


             String updateQuery =
                     "UPDATE [ACCOUNT] " +
                     "SET [Status] = 'Suspended' " +
                     "WHERE [Email] = ?";

             if (attemptLeft == 0) {

                 try (PreparedStatement updateAccountStatus =
connection.prepareStatement(updateQuery)) {

                     updateAccountStatus.setString(1, this.email);
                     updateAccountStatus.executeUpdate();
```

```
                    JOptionPane.showMessageDialog(
                            rightPanel,
                            "Your account has been suspended.",
                            "Login Failed",
                            JOptionPane.ERROR_MESSAGE);

                    clearUserInput();


                } catch(SQLException ex) {
                      Logger.getLogger(LoginFrame.class.getName())
                            .log(Level.SEVERE, ex.getMessage(), ex);
                }
            }


        } catch(SQLException ex) {
            Logger.getLogger(LoginFrame.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

    }

    private void clearUserInput() {

        emailTextField.setText("");
        passwordField.setText("");

    }


    private void currentLoginAccount (String role) {

        switch (role) {
            case "Citizen": new CitizenFrame().setVisible(true); break;
            case "Admin": new AdminFrame().setVisible(true); break;
        }
    }


    /**
    * Checks if the email and password input fields are empty.
    *
    * @return true if either the email or password input field is empty, false
otherwise.
    */
    private boolean isInputFieldsEmpty() {

        return email.isBlank() || password.isBlank();

    }


    /**
     *
     * Check if the credentials match.
     * Checks if the provided email, password, and role match the
corresponding values in the database.
     *
     * @param dbEmail The email retrieved from the database.
     * @param dbPassword The password retrieved from the database.
     * @param dbRole The role retrieved from the database.
     * @return true if the provided email, password, and role match the
database values, false otherwise.
     */
    private boolean isCredentialMatch(String dbEmail, String dbPassword,
String dbRole) {

        return email.equals(dbEmail) &&
               password.equals(dbPassword) &&
               role.equals(dbRole);

    }

    public static void main(String args[]) {

        FlatMacLightLaf.setup();

        // FlatLaf setup and UI theme customization...
        UIManager.put("TitlePane.unifiedBackground", false);
        UIManager.put("TitlePane.background", Color.decode("#ffffff"));

        UIManager.put( "Button.arc", 8);
        UIManager.put("TextComponent.arc", 8);
        UIManager.put( "Component.focusWidth", 1);

        // Menu Customization
        UIManager.put("List.selectionBackground", Color.decode("#007AFF"));
        UIManager.put("List.selectionInactiveBackground",
Color.decode("#007AFF"));
        UIManager.put("List.selectionInactiveForeground",
Color.decode("#FFFFFF"));

        // Table UI Customization
        UIManager.put("TableHeader.separatorColor", Color.decode("#FFFFFF"));
        UIManager.put("TableHeader.hoverBackground", Color.decode("#F2F2F2"));
        UIManager.put("TableHeader.height", 40);
        UIManager.put("Table.selectionBackground", Color.decode("#C2E7FF"));
        UIManager.put("Table.selectionForeground", Color.decode("#333333"));
        UIManager.put("Table.showCellFocusIndicator", false);
        UIManager.put("Table.selectionInactiveBackground",
Color.decode("#C2E7FF"));
        //UIManager.put("Table.alternateRowColor", Color.decode("#F2F2F2"));
        UIManager.put("Table.cellFocusColor", Color.decode("#B0E2FF"));

        // TextField UI Customization
        UIManager.put("TextComponent.arc", 8);
        UIManager.put("TextField.disabledBackground",
Color.decode("#FFFFFF"));
        UIManager.put("TextField.disabledForeground",
Color.decode("#333333"));


        java.awt.EventQueue.invokeLater(() -> {
            new LoginFrame().setVisible(true);
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JLabel description;
    private javax.swing.JTextField emailTextField;
    private javax.swing.JLabel headerTitle;
    private javax.swing.JPanel leftPanel;
    private javax.swing.JLabel loginAs;
    private javax.swing.JButton loginButton;
    private javax.swing.JLabel loginIcon;
    private javax.swing.JPanel loginPanel;
    private javax.swing.JPasswordField passwordField;
    private javax.swing.JLabel question;
    private javax.swing.JPanel rightPanel;
    private javax.swing.JComboBox<String> roleComboBox;
    private javax.swing.JButton signupLink;
    private javax.swing.JLabel userIcon;
    // End of variables declaration

}
```

Signup

```
package project.authentication;

import com.formdev.flatlaf.FlatClientProperties;
import com.formdev.flatlaf.themes.FlatMacLightLaf;
import java.awt.Color;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import project.database.Database;

/**
 * Represents the signup frame of the application.
 * Allows users to create a new account.
 *
 * @since 4/27/2024
 */
public class SignupFrame extends javax.swing.JFrame {

    Connection connection;
    String firstName;
    String lastName;
    String email;
    String password;

    public SignupFrame() {
        initComponents();
        try {
            connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(SignupFrame.class.getName()).log(Level.SEVERE,
ex.getMessage(), ex);
        }

        loadPanelIcon();

        loadFrameIcon();

        customizeComponent();
    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        leftPanel = new javax.swing.JPanel();
        loginIcon = new javax.swing.JLabel();
        rightPanel = new javax.swing.JPanel();
        sigupPanel = new javax.swing.JPanel();
        addUserIcon = new javax.swing.JLabel();
        headerTitle = new javax.swing.JLabel();
        description = new javax.swing.JLabel();
        firstNameTextField = new javax.swing.JTextField();
        lastNameTextField = new javax.swing.JTextField();
        emailTextField = new javax.swing.JTextField();
        passwordField = new javax.swing.JPasswordField();
        signupButton = new javax.swing.JButton();
        questionText = new javax.swing.JLabel();
        loginLink = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Sign up");
        getContentPane().setLayout(new java.awt.GridLayout(1, 2));

        leftPanel.setBackground(new java.awt.Color(40, 167, 69));
        leftPanel.setPreferredSize(new java.awt.Dimension(500, 600));

        javax.swing.GroupLayout leftPanelLayout = new
javax.swing.GroupLayout(leftPanel);
        leftPanel.setLayout(leftPanelLayout);
        leftPanelLayout.setHorizontalGroup(

leftPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(leftPanelLayout.createSequentialGroup()
                .addGap(46, 46, 46)
                .addComponent(loginIcon,
javax.swing.GroupLayout.PREFERRED_SIZE, 400,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        leftPanelLayout.setVerticalGroup(

leftPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
leftPanelLayout.createSequentialGroup()
                .addContainerGap(129, Short.MAX_VALUE)
                .addComponent(loginIcon,
javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(121, 121, 121))
        );

        getContentPane().add(leftPanel);

        rightPanel.setBackground(new java.awt.Color(255, 255, 255));
        rightPanel.setPreferredSize(new java.awt.Dimension(500, 600));
        rightPanel.setLayout(new java.awt.CardLayout());

        java.awt.GridBagLayout sigupPanelLayout = new
java.awt.GridBagLayout();
        sigupPanelLayout.columnWidths = new int[] {0, 5, 0, 5, 0, 5, 0};
        sigupPanelLayout.rowHeights = new int[] {0, 15, 0, 15, 0, 15, 0, 15,
0, 15, 0, 15, 0, 15, 0, 15, 0};
        sigupPanel.setLayout(sigupPanelLayout);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = 7;
        sigupPanel.add(addUserIcon, gridBagConstraints);

        headerTitle.setFont(new java.awt.Font("Roboto", 1, 24)); // NOI18N
        headerTitle.setForeground(new java.awt.Color(40, 167, 69));
        headerTitle.setText("Sign up");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = 7;
        sigupPanel.add(headerTitle, gridBagConstraints);

        description.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        description.setForeground(new java.awt.Color(102, 102, 102));
        description.setText("Create an account to access our application.");
        description.setCursor(new
java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = 7;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 20, 0);
        sigupPanel.add(description, gridBagConstraints);

        firstNameTextField.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        firstNameTextField.setForeground(new java.awt.Color(51, 51, 51));
        firstNameTextField.setPreferredSize(new java.awt.Dimension(150, 25));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        sigupPanel.add(firstNameTextField, gridBagConstraints);

        lastNameTextField.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        lastNameTextField.setForeground(new java.awt.Color(51, 51, 51));
        lastNameTextField.setPreferredSize(new java.awt.Dimension(150, 25));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        sigupPanel.add(lastNameTextField, gridBagConstraints);

        emailTextField.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        emailTextField.setForeground(new java.awt.Color(51, 51, 51));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.gridwidth = 7;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        sigupPanel.add(emailTextField, gridBagConstraints);

        passwordField.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        passwordField.setForeground(new java.awt.Color(51, 51, 51));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.gridwidth = 7;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        sigupPanel.add(passwordField, gridBagConstraints);

        signupButton.setBackground(new java.awt.Color(40, 167, 69));
        signupButton.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        signupButton.setForeground(new java.awt.Color(255, 255, 255));
        signupButton.setText("Sign up");
        signupButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        signupButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                signupButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 12;
        gridBagConstraints.gridwidth = 7;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.ipady = 20;
        sigupPanel.add(signupButton, gridBagConstraints);

        questionText.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        questionText.setForeground(new java.awt.Color(102, 102, 102));
        questionText.setText("Already have an account?");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 14;
        gridBagConstraints.gridwidth = 7;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.LINE_START;
        gridBagConstraints.insets = new java.awt.Insets(0, 27, 0, 0);
        sigupPanel.add(questionText, gridBagConstraints);
        loginLink.setBackground(new java.awt.Color(242, 242, 242));
        loginLink.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        loginLink.setForeground(new java.awt.Color(40, 167, 69));
        loginLink.setText("LOG IN");
        loginLink.setToolTipText("Go to log in");
        loginLink.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1,
1, 1));
        loginLink.setBorderPainted(false);
        loginLink.setContentAreaFilled(false);
        loginLink.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        loginLink.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                showLoginFrame(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 14;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 2);
        sigupPanel.add(loginLink, gridBagConstraints);

        rightPanel.add(sigupPanel, "card3");

        getContentPane().add(rightPanel);

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>


    /**
     * Loads the icon for the frame title.
     */
    private void loadFrameIcon() {

        final ImageIcon titlePaneIcon = new
ImageIcon(getClass().getResource("/icon/CICT-logo-icon.png"));
        this.setIconImage(titlePaneIcon.getImage());

    }


    /**
     * Loads icons for panels.
     * This method loads the icons for the left and right panels of the signup
frame.
     */
    private void loadPanelIcon() {

        loginIcon.setIcon(new ImageIcon(getClass().getResource("/icon/left-
panel-icon.png")));
        addUserIcon.setIcon(new ImageIcon(getClass().getResource("/icon/add-
user.png")));

    }


    /**
```

```
     * Customizes UI components.
     * This method customizes the appearance and behavior of various UI
components
     * such as text fields, buttons, and labels.
     */
    private void  customizeComponent() {


firstNameTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
"First Name");
        firstNameTextField.putClientProperty(FlatClientProperties.STYLE,
                "focusColor: #28a745;" +
                "margin: 3, 10, 3, 10;");


lastNameTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
"Last Name");
        lastNameTextField.putClientProperty(FlatClientProperties.STYLE,
                "focusColor: #28a745;" +
                "margin: 3, 10, 3, 10;");


emailTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT, "Email
Address");
        emailTextField.putClientProperty(FlatClientProperties.STYLE,
                "focusColor: #28a745;" +
                "margin: 3, 10, 3, 10;" +
                "showClearButton: true;");

        passwordField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
"Password");
        passwordField.putClientProperty(FlatClientProperties.STYLE,
                "focusColor: #28a745;" +
                "showRevealButton: true;" +
                "showCapsLock: true;" +
                "margin: 3, 10, 3, 10;");

        signupButton.putClientProperty(FlatClientProperties.STYLE,
"focusColor: #28a745;");

        loginLink.putClientProperty(FlatClientProperties.STYLE,
                "hoverForeground: #5cb85c;" +
                "pressedForeground: #1e7e34;");

    }


    /**
     * Displays the login frame.
     * This method is invoked when the user clicks on the login link.
     * It closes the current signup frame and opens the login frame.
     */
    private void showLoginFrame(java.awt.event.ActionEvent evt) {

        new LoginFrame().setVisible(true);
        dispose();

    }
    /**
     * Handles sign-up button action.
     * This method is invoked when the user clicks on the sign-up button.
     * It retrieves the user input from the text fields, validates the input,
     * and then attempts to create a new user account in the database.
     */
    private void signupButtonActionPerformed(java.awt.event.ActionEvent evt) {

        firstName = firstNameTextField.getText();
        lastName = lastNameTextField.getText();
        email = emailTextField.getText();
        password = String.valueOf(passwordField.getPassword());

        if (isInputFieldEmpty()) {
            JOptionPane.showMessageDialog(
                    rightPanel,
                    "Fill out needed information!",
                    "Reminder",
                    JOptionPane.ERROR_MESSAGE);

            return;
        }


        if (isPasswordLengthValid(passwordField.getPassword())) {

            if (!isDigitCountValid(passwordField.getPassword())) {

                JOptionPane.showMessageDialog(
                        rightPanel,
                        "Must contain atleast 2 digits!",
                        "Password requirement",
                        JOptionPane.WARNING_MESSAGE);

                return;
            }

        }
        else {
            JOptionPane.showMessageDialog(
                    rightPanel,
                    "Password must be 8 characters long!",
                    "Password requirement",
                    JOptionPane.WARNING_MESSAGE);

                return;
        }

        String insertToUserQuery =
                "INSERT INTO [User] ([Firstname], [Lastname], [Email]) " +
                "VALUES (?, ?, ?)";

        String insertToAccountQuery =
                "INSERT INTO [Account] ([Email], [Password], [Role], [Status])
" +
                "VALUES (?, ?, ?, ?)";
        try (PreparedStatement insertToAccount =
connection.prepareStatement(insertToAccountQuery);
             PreparedStatement insertToUser =
connection.prepareStatement(insertToUserQuery);) {

            connection.setAutoCommit(false);

            insertToAccount.setString(1, email);
            insertToAccount.setString(2, password);
            insertToAccount.setString(3, "Citizen");
            insertToAccount.setString(4, "Active");
            insertToAccount.executeUpdate();

            insertToUser.setString(1, firstName);
            insertToUser.setString(2, lastName);
            insertToUser.setString(3, email);
            insertToUser.executeUpdate();

            connection.commit();

            JOptionPane.showMessageDialog(
                    rightPanel,
                    "Sign up Succesful",
                    "Notice",
                    JOptionPane.INFORMATION_MESSAGE);

            new LoginFrame().setVisible(true);
            dispose();

        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(
                    rightPanel,
                    "An account with this email already exists. Please use a
different email or log in.",
                    "Sign Up Error",
                    JOptionPane.ERROR_MESSAGE);

            clearUserInput();

            Logger.getLogger(SignupFrame.class.getName())
                  .log(Level.INFO, "Sign up failed", ex);
        }

    }


    private boolean isPasswordLengthValid(char[] password) {

        return password.length <= 8;

    }


    private boolean isDigitCountValid(char[] password) {

        int digitCount = 0;
        for (char character : password) {

            if (Character.isDigit(character)) {
                digitCount++;
            }

        }

        return digitCount >= 2;

    }

    private void clearUserInput() {
        firstNameTextField.setText("");
        lastNameTextField.setText("");
        emailTextField.setText("");
        passwordField.setText("");
    }

    /**
     * Checks if any input field is empty.
     * This method checks whether any of the input fields (first name, last
name, email, password) are empty.
     *
     * @return true if any input field is empty, false otherwise.
     */
    boolean isInputFieldEmpty() {

        return firstName.isBlank() || lastName.isBlank() ||
               email.isBlank() || password.isBlank();

    }


    public static void main(String args[]) {

        FlatMacLightLaf.setup();

        // FlatLaf setup and UI theme customization...
        UIManager.put("TitlePane.unifiedBackground", false);
        UIManager.put("TitlePane.background", Color.decode("#ffffff"));
        UIManager.put("TitlePane.showIcon", true);

        UIManager.put( "Button.arc", 8);
        UIManager.put("TextComponent.arc", 8);
        UIManager.put( "Component.focusWidth", 1);


        java.awt.EventQueue.invokeLater(() -> {
            new SignupFrame().setVisible(true);
        });

    }

    // Variables declaration - do not modify
    private javax.swing.JLabel addUserIcon;
    private javax.swing.JLabel description;
    private javax.swing.JTextField emailTextField;
    private javax.swing.JTextField firstNameTextField;
    private javax.swing.JLabel headerTitle;
    private javax.swing.JTextField lastNameTextField;
    private javax.swing.JPanel leftPanel;
    private javax.swing.JLabel loginIcon;
    private javax.swing.JButton loginLink;
    private javax.swing.JPasswordField passwordField;
    private javax.swing.JLabel questionText;
    private javax.swing.JPanel rightPanel;
    private javax.swing.JButton signupButton;
    private javax.swing.JPanel sigupPanel;
    // End of variables declaration
}
```

Citizen

Citizen Frame

```
package project.citizen;

import com.formdev.flatlaf.FlatClientProperties;
import com.formdev.flatlaf.FlatLaf;
import com.formdev.flatlaf.fonts.roboto.FlatRobotoFont;
import com.formdev.flatlaf.themes.FlatMacLightLaf;
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Image;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import project.authentication.LoginFrame;
import project.database.Database;


public class CitizenFrame extends javax.swing.JFrame {

    Connection connection;

    CardLayout cl;

    private String email;
    private String fullName;
    private byte[] profilePictureData;

    public CitizenFrame() {
        initComponents();

        try {
            connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(CitizenFrame.class.getName()).log(Level.SEVERE,
ex.getMessage(), ex);
        }

        cl = (CardLayout) contentPanel.getLayout();
        customizeComponent();

        initProfileInformation();

    }

    private void customizeComponent() {
        menuList.putClientProperty(FlatClientProperties.STYLE,
                "cellMargins: 0, 10, 0, 0;" +
                "selectionArc: 10;");
    }


    private void initProfileInformation() {

        this.email = LoginFrame.email;

        String selectQuery =
                "SELECT [Firstname] + ' ' + [Lastname] AS Fullname,
[ProfilePicture] " +
                "FROM [User]" +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{

            pst.setString(1, this.email);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.fullName = rs.getString("Fullname");
                this.profilePictureData = rs.getBytes("ProfilePicture");

            }

        } catch (SQLException ex) {
            Logger.getLogger(CitizenFrame.class.getName())
                  .log(Level.SEVERE, "initProfileInformation method", ex);
        }

        nameLabel.setText(this.fullName);
        emailLabel.setText(this.email);

        try {

this.profilePictureLabel.setIcon(scaledImageIcon(profilePictureData));
        } catch (NullPointerException ex) {
            Logger.getLogger(CitizenFrame.class.getName())
                  .log(Level.SEVERE, "scaledImageIcon method", ex);
        }

    }

    private ImageIcon scaledImageIcon(byte[] pictureData) throws
NullPointerException {
        Image imageIcon = new ImageIcon(profilePictureData).getImage()

.getScaledInstance(profilePictureLabel.getWidth(),

profilePictureLabel.getHeight(),

Image.SCALE_SMOOTH);

        return new ImageIcon(imageIcon);
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        mainPanel = new javax.swing.JPanel();
        menuPanel = new javax.swing.JPanel();
        profilePanel = new javax.swing.JPanel();
        profilePictureLabel = new javax.swing.JLabel();
        nameLabel = new javax.swing.JLabel();
        emailLabel = new javax.swing.JLabel();
        menuScrollPane = new javax.swing.JScrollPane();
        menuList = new javax.swing.JList<>();
        jSeparator1 = new javax.swing.JSeparator();
        logoutButton = new javax.swing.JButton();
        contentPanel = new javax.swing.JPanel();
        myComplaint1 = new project.citizen.MyComplaint();
        createComplaint1 = new project.citizen.CreateComplaint();
        accountSetting1 = new project.citizen.AccountSetting();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        mainPanel.setPreferredSize(new java.awt.Dimension(1300, 800));

        menuPanel.setBackground(new java.awt.Color(255, 255, 255));

        profilePanel.setBackground(new java.awt.Color(255, 255, 255));
        profilePanel.setPreferredSize(new java.awt.Dimension(240, 170));
        profilePanel.setLayout(new java.awt.GridBagLayout());

        profilePictureLabel.setPreferredSize(new java.awt.Dimension(70, 70));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        profilePanel.add(profilePictureLabel, gridBagConstraints);

        nameLabel.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        nameLabel.setForeground(new java.awt.Color(51, 51, 51));
        nameLabel.setText("Full Name");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(20, 0, 0, 0);
        profilePanel.add(nameLabel, gridBagConstraints);

        emailLabel.setFont(new java.awt.Font("Roboto", 0, 12)); // NOI18N
        emailLabel.setForeground(new java.awt.Color(128, 128, 128));
        emailLabel.setText("Email");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
        profilePanel.add(emailLabel, gridBagConstraints);

        menuScrollPane.setBorder(null);

menuScrollPane.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HO
RIZONTAL_SCROLLBAR_NEVER);

menuScrollPane.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERT
ICAL_SCROLLBAR_NEVER);
        menuScrollPane.setWheelScrollingEnabled(false);

        menuList.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1,
1, 1));
        menuList.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        menuList.setModel(new javax.swing.AbstractListModel<String>() {
            String[] strings = { "My Complaint", "Create Complaint", "Account"
};
            public int getSize() { return strings.length; }
            public String getElementAt(int i) { return strings[i]; }
        });

menuList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        menuList.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        menuList.setFixedCellHeight(40);
        menuList.setPreferredSize(new java.awt.Dimension(220, 400));
        menuList.setSelectedIndex(0);
        menuList.setVisibleRowCount(4);
        menuList.addListSelectionListener(new
javax.swing.event.ListSelectionListener() {
            public void valueChanged(javax.swing.event.ListSelectionEvent evt)
{
                menuListValueChanged(evt);
            }
        });
        menuScrollPane.setViewportView(menuList);

        jSeparator1.setForeground(new java.awt.Color(235, 235, 235));
        jSeparator1.setPreferredSize(new java.awt.Dimension(230, 10));

        logoutButton.setBackground(java.awt.Color.red);
        logoutButton.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        logoutButton.setForeground(new java.awt.Color(255, 255, 255));
        logoutButton.setText("Log out");
        logoutButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        logoutButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                logoutButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout menuPanelLayout = new
javax.swing.GroupLayout(menuPanel);
        menuPanel.setLayout(menuPanelLayout);
        menuPanelLayout.setHorizontalGroup(

menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
menuPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)
                    .addComponent(logoutButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jSeparator1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addComponent(menuScrollPane,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 226, Short.MAX_VALUE)
                    .addComponent(profilePanel,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
                .addContainerGap())
        );
        menuPanelLayout.setVerticalGroup(

menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(menuPanelLayout.createSequentialGroup()
                .addComponent(profilePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(menuScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 126,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(logoutButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        contentPanel.setBackground(new java.awt.Color(204, 204, 255));
        contentPanel.setPreferredSize(new java.awt.Dimension(800, 600));
        contentPanel.setLayout(new java.awt.CardLayout());
        contentPanel.add(myComplaint1, "myComplaintCard");
        contentPanel.add(createComplaint1, "createComplaintCard");
        contentPanel.add(accountSetting1, "accountSettingCard");

        javax.swing.GroupLayout mainPanelLayout = new
javax.swing.GroupLayout(mainPanel);
        mainPanel.setLayout(mainPanelLayout);
        mainPanelLayout.setHorizontalGroup(

mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addComponent(menuPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(contentPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1064, Short.MAX_VALUE))
        );
        mainPanelLayout.setVerticalGroup(

mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addComponent(contentPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 750,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addComponent(menuPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
1308, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
756, Short.MAX_VALUE)
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void menuListValueChanged(javax.swing.event.ListSelectionEvent
evt) {

        final int MENU1 = 0;
        final int MENU2 = 1;
        final int MENU3 = 2;
        int selectedMenu = menuList.getSelectedIndex();

        switch(selectedMenu) {
            case MENU1: cl.show(contentPanel, "myComplaintCard"); break;
            case MENU2: cl.show(contentPanel, "createComplaintCard"); break;
            case MENU3: cl.show(contentPanel, "accountSettingCard"); break;
        }
    }

    private void logoutButtonActionPerformed(java.awt.event.ActionEvent evt) {

        int response = JOptionPane.showConfirmDialog(this,
                "Are you sure you want to log out?",
                "Log out Confirmation",
                JOptionPane.YES_NO_OPTION);

        final int YES = 0;
        if (response == YES) {
            new LoginFrame().setVisible(true);
            dispose();
        }

    }


    public static void main(String args[]) {

        FlatLaf.registerCustomDefaultsSource("/theme");

        FlatMacLightLaf.setup();

        // Font used
        FlatRobotoFont.install();

        // TitlePane Customization
        UIManager.put("TitlePane.unifiedBackground", false);

        // Menu Customization
        UIManager.put("List.selectionBackground", Color.decode("#007AFF"));
        UIManager.put("List.selectionInactiveBackground",
Color.decode("#007AFF"));
        UIManager.put("List.selectionInactiveForeground",
Color.decode("#FFFFFF"));

        // TextField UI Customization
        UIManager.put("TextComponent.arc", 8);
        UIManager.put("TextField.disabledBackground",
Color.decode("#FFFFFF"));
        UIManager.put("TextField.disabledForeground",
Color.decode("#333333"));

        // Table UI Customization
        UIManager.put("TableHeader.separatorColor", Color.decode("#FFFFFF"));
        UIManager.put("TableHeader.hoverBackground", Color.decode("#F2F2F2"));
        UIManager.put("TableHeader.height", 40);
        UIManager.put("Table.selectionBackground", Color.decode("#C2E7FF"));
        UIManager.put("Table.selectionForeground", Color.decode("#333333"));
        //nUIManager.put("Table.showCellFocusIndicator", false);
        UIManager.put("Table.selectionInactiveBackground",
Color.decode("#C2E7FF"));
        //UIManager.put("Table.alternateRowColor", Color.decode("#F2F2F2"));
        UIManager.put("Table.cellFocusColor", Color.decode("#B0E2FF"));

        java.awt.EventQueue.invokeLater(() -> {
            new CitizenFrame().setVisible(true);
        });
    }

    // Variables declaration - do not modify
    private project.citizen.AccountSetting accountSetting1;
    private javax.swing.JPanel contentPanel;
    private project.citizen.CreateComplaint createComplaint1;
    private javax.swing.JLabel emailLabel;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JButton logoutButton;
    private javax.swing.JPanel mainPanel;
    private javax.swing.JList<String> menuList;
    private javax.swing.JPanel menuPanel;
    private javax.swing.JScrollPane menuScrollPane;
    private project.citizen.MyComplaint myComplaint1;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JPanel profilePanel;
    private javax.swing.JLabel profilePictureLabel;
    // End of variables declaration
}
```

My Complaint

```
package project.citizen;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


/**
 * The MyComplaint class represents a panel for displaying and managing user
complaints.
 * It provides functionalities to populate, filter, and interact with the
complaints table.
 * This panel includes options to view complaint details, withdraw complaints,
and refresh the table.
 *
 * The complaints are fetched from the database based on the currently logged-
in user's email.
 * Users can filter complaints by their status (e.g., New, Assigned,
Resolved).
```

```
 * Double-clicking on a complaint row displays detailed information about the
complaint,
 * including its category, description, date, location, landmark, urgency
level, and status.
 * Additionally, users can withdraw complaints by selecting the respective
option from the pop-up window.
 */
public class MyComplaint extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    private int complaintNo;
    private String category;
    private String description;
    private String date;
    private String location;
    private String landmark;
    private String urgencyLevel;
    private String status;

    /**
     * Constructs a new MyComplaint panel.
     * Initializes GUI components, sets up table header, customizes
components,
     * establishes database connection, and populates the complaints table.
     */
    public MyComplaint() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeComponents();
        customizeCellRender();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(MyComplaint.class.getName()).log(Level.SEVERE,
ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();
        this.email = LoginFrame.email;

        populateTable();

    }

    /**
     * Populates the complaints table with user-specific complaints retrieved
from the database.
     */
    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [UserID] = (" +
                    "SELECT [UserID] " +
                    "FROM [User] " +
                    "WHERE [Email] = ?) " +
                    "AND [Status] != 'Withdrawn'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            complaint.setString(1, this.email);

            ResultSet rs = complaint.executeQuery();
            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    rowData[i-1] = rs.getObject(i);
                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(MyComplaint.class.getName()).log(Level.SEVERE,
"populateTable method", ex);
        }

    }


    /**
     * Initializes the table header with a specific font and alignment.
     */
    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.PLAIN, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
```

```
        complaintNoColumn.setPreferredWidth(5);

    }


    /**
     * Customizes components, such as setting padding for the status combo
box.
     */
    private void customizeComponents() {

        statusComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 3, 10, 3, 10;");

    }


    /**
     * Customizes cell rendering for the complaints table to support multiline
text.
     */
    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {
            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        filterBy = new javax.swing.JLabel();
        statusComboBox = new javax.swing.JComboBox<>();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        filterBy.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        filterBy.setForeground(new java.awt.Color(51, 51, 51));
        filterBy.setText("Status");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 10, 0, 0);
        topPanel.add(filterBy, gridBagConstraints);

        statusComboBox.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        statusComboBox.setForeground(new java.awt.Color(51, 51, 51));
        statusComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "<html><font color='gray'>All...</font></html>", "New", "Under
Review", "Assigned", "Resolved" }));
        statusComboBox.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        statusComboBox.setPreferredSize(new java.awt.Dimension(140, 40));
        statusComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                statusComboBoxActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(statusComboBox, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setForeground(new java.awt.Color(51, 51, 51));
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
```

```
            }
        ) {
            boolean[] canEdit = new boolean [] {
                true, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
```

```
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(11, Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    /**
     * Refreshes the complaints table by clearing existing data and
repopulating it.
     */
    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        tableModel.setRowCount(0);
        statusComboBox.setSelectedIndex(0);
        tableModel.setRowCount(0);
        populateTable();

    }


    /**
     * Handles the action event when the status combo box selection changes.
     * Filters complaints based on the selected status.
     */
    private void statusComboBoxActionPerformed(java.awt.event.ActionEvent evt)
{

        String selectedStatus = statusComboBox.getSelectedItem().toString();
        tableModel.setRowCount(0);

        String selectQuery =

                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Status] = ? AND [UserID] = (" +
                                                "SELECT [UserID] " +
                                                "FROM [User] " +
                                                "WHERE [Email] = ?)";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{

            pst.setString(1, selectedStatus);
            pst.setString(2, this.email);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(MyComplaint.class.getName())
                  .log(Level.SEVERE, "statusComboBoxActionPerformed method",
ex);
        }


        if (selectedStatus.equals("<html><font
color='gray'>All...</font></html>") ) {
           populateTable();
        }

    }



    /**
     * Handles the mouse click event on the complaints table.
     * Displays detailed information about the selected complaint and allows
withdrawal.
     */
    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(MyComplaint.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked method", ex);
        }

        // Getting the value of each column at the selected row.
        try {
            complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
            category = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
            description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
            date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
            location = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
            landmark = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
            urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
            status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(MyComplaint.class.getName())
                  .log(Level.SEVERE, "getValueAt method", ex);
        }


        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(MyComplaint.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked method", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {
            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };


            int option = JOptionPane.showOptionDialog(
                                this,
                                content,
                                "Complaint Details",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.PLAIN_MESSAGE,
                                null,
                                new String[]{"Withdraw", "Cancel"},
                                "Withdraw");




            if (option == JOptionPane.YES_OPTION) {

                if (!status.equals("New") && !status.equals("Under Review")) {

                JOptionPane.showMessageDialog(
                    this,
                    "Cannot withdraw. The complaint is being addressed or 
resolved.",
                    "Withdrawal Not Allowed",
                    JOptionPane.WARNING_MESSAGE);

                return;
                }

                int confirmOption = JOptionPane.showConfirmDialog(
                                            this,
                                            "Are you sure you want to withdraw 
this complaint?",
                                            "Confirm Withdrawal",
                                            JOptionPane.YES_NO_OPTION);

                if (confirmOption == JOptionPane.YES_OPTION) {

                    String updateQuery =
                            "UPDATE [UserComplaint] " +
                            "SET [Status] = 'Withdrawn' " +
                            "WHERE [ComplaintNo] = ?";
                    try (PreparedStatement pst =
connection.prepareStatement(updateQuery)) {

                        pst.setInt(1, complaintNo);
                        pst.executeUpdate();

                        JOptionPane.showMessageDialog(
                                            this,
                                            "Complaint has been Withdrawn.",
                                            "Withdrawal Successful",
                                            JOptionPane.INFORMATION_MESSAGE);



                    } catch (SQLException ex) {
                        Logger.getLogger(MyComplaint.class.getName())
                              .log(Level.SEVERE, "complaintTableMouseClicked
method", ex);
                    }
                }
            }
        }


    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JLabel filterBy;
    private javax.swing.JButton refreshButton;
    private javax.swing.JComboBox<String> statusComboBox;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Create Complaint

```
package project.citizen;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Color;
import java.awt.Image;
import java.awt.event.FocusEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.ZoneId;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.SwingConstants;
import project.authentication.LoginFrame;
import project.concrete_class.ImageFilter;
import project.concrete_class.ComplaintStatus;
import project.database.Database;

/**
 * This class represents the Create Complaint panel of the application.
 * Users can file complaints using this panel.
 *
 * @since 4/27/2024
 */
public class CreateComplaint extends javax.swing.JPanel {

    private Connection connection;

    private String category;
    private String description;
    private String createdDate;
    private String location;
    private String landmark;
    private String urgencyLevel;
    private byte[] proofImageData;
    private String email;
    private int currentUser;

    private SimpleDateFormat dateFormat;

    public CreateComplaint() {
        initComponents();
        customizeComponent();

        try {
            connection = Database.getConnection();
        } catch (SQLException ex) {

Logger.getLogger(CreateComplaint.class.getName()).log(Level.ALL.SEVERE,
ex.getMessage(), ex);
        }

        this.email = LoginFrame.email;

        this.currentUser = getCurrentUser();

        this.dateFormat = new SimpleDateFormat("yyyy-MM-dd");

        descriptionTextAreaFocusLost(new FocusEvent(descriptionTextArea,
FocusEvent.FOCUS_LOST));
    }


     /**
     * Customizes the appearance of Swing components.
     * Sets styles and properties for various components.
     */
    private void customizeComponent() {
        categoryComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 3, 10, 3, 10;");

        categoryHelpButton.putClientProperty(FlatClientProperties.BUTTON_TYPE,
                FlatClientProperties.BUTTON_TYPE_HELP);

        descriptionTextArea.putClientProperty(FlatClientProperties.STYLE,
                "margin: 10, 10, 3, 10;");


locationTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
                "e.g. \"Mc Arthur Highway Malolos Bulacan\"");
        locationTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");


landmarkTextField.putClientProperty(FlatClientProperties.PLACEHOLDER_TEXT,
                "e.g. \"Near BSU Gate 1\"");
        landmarkTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

        urgencyLevelComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 3, 10, 3, 10;");


urgencyLevelHelpButton.putClientProperty(FlatClientProperties.BUTTON_TYPE,
                FlatClientProperties.BUTTON_TYPE_HELP);

    }

    // unused method
    private void loadProofLabelIcon() {

        ImageIcon icon = new
ImageIcon(getClass().getResource("/icon/upload.png"));
        Image uploadImage = icon.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(uploadImage);

        proofImageLabel.setIcon(scaledImageIcon);
        proofImageLabel.setHorizontalAlignment(SwingConstants.CENTER);
        proofImageLabel.setVerticalAlignment(SwingConstants.CENTER);

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        header = new javax.swing.JPanel();
        Title = new javax.swing.JLabel();
        Heading = new javax.swing.JLabel();
        content = new javax.swing.JPanel();
        leftPanel = new javax.swing.JPanel();
        Category = new javax.swing.JLabel();
        categoryComboBox = new javax.swing.JComboBox<>();
        Description = new javax.swing.JLabel();
        descriptionScrollPane = new javax.swing.JScrollPane();
        descriptionTextArea = new javax.swing.JTextArea();
        Date = new javax.swing.JLabel();
        dateChooser = new com.toedter.calendar.JDateChooser();
        Location = new javax.swing.JLabel();
        locationTextField = new javax.swing.JTextField();
        Landmark = new javax.swing.JLabel();
        landmarkTextField = new javax.swing.JTextField();
        urgencyLevelComboBox = new javax.swing.JComboBox<>();
        categoryHelpButton = new javax.swing.JButton();
        urgencyLevelHelpButton = new javax.swing.JButton();
        urgencyLevelComboBox1 = new javax.swing.JComboBox<>();
        Urgency1 = new javax.swing.JLabel();
        rightPanel = new javax.swing.JPanel();
        proofLabel = new javax.swing.JLabel();
        uploadFileButton = new javax.swing.JButton();
        proofImageLabel = new javax.swing.JLabel();
        footerSeparator = new javax.swing.JSeparator();
        footer = new javax.swing.JPanel();
        clearButton = new javax.swing.JButton();
        submitButton = new javax.swing.JButton();

        header.setLayout(new java.awt.GridBagLayout());

        Title.setFont(new java.awt.Font("Roboto", 0, 24)); // NOI18N
        Title.setForeground(new java.awt.Color(51, 51, 51));
        Title.setText("Welcome to Citizen Complaint Center!");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 5, 0);
        header.add(Title, gridBagConstraints);

        Heading.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Heading.setForeground(new java.awt.Color(51, 51, 51));
        Heading.setText("Please use this form to report any concerns or issues
you encounter in your community.");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        header.add(Heading, gridBagConstraints);

        java.awt.GridBagLayout leftPanelLayout = new java.awt.GridBagLayout();
        leftPanelLayout.columnWidths = new int[] {0, 20, 0, 20, 0, 20, 0};
        leftPanelLayout.rowHeights = new int[] {0, 15, 0, 15, 0, 15, 0, 15, 0,
15, 0};
        leftPanel.setLayout(leftPanelLayout);

        Category.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Category.setForeground(new java.awt.Color(51, 51, 51));
        Category.setText("Category");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 98, 0, 0);
        leftPanel.add(Category, gridBagConstraints);
        categoryComboBox.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        categoryComboBox.setForeground(new java.awt.Color(51, 51, 51));
        categoryComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "<html><font color='b6b6b6'>Select...</font></html>",
"Environment", "Infrastructure", "Utilities", "Public Services", "Government
Agencies" }));
        categoryComboBox.setPreferredSize(new java.awt.Dimension(81, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        leftPanel.add(categoryComboBox, gridBagConstraints);

        Description.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Description.setForeground(new java.awt.Color(51, 51, 51));
        Description.setText("Description");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 81, 0, 0);
        leftPanel.add(Description, gridBagConstraints);

        descriptionTextArea.setColumns(20);
        descriptionTextArea.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        descriptionTextArea.setForeground(new java.awt.Color(51, 51, 51));
        descriptionTextArea.setLineWrap(true);
        descriptionTextArea.setRows(5);
        descriptionTextArea.setToolTipText("");
        descriptionTextArea.setWrapStyleWord(true);
        descriptionTextArea.addFocusListener(new java.awt.event.FocusAdapter()
{
            public void focusGained(java.awt.event.FocusEvent evt) {
                descriptionTextAreaFocusGained(evt);
            }
            public void focusLost(java.awt.event.FocusEvent evt) {
                descriptionTextAreaFocusLost(evt);
            }
        });
        descriptionScrollPane.setViewportView(descriptionTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.weightx = 0.1;
        leftPanel.add(descriptionScrollPane, gridBagConstraints);

        Date.setText("Date");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        leftPanel.add(Date, gridBagConstraints);
        dateChooser.setToolTipText("YYYY-MM--DD");
        dateChooser.setDateFormatString("yyyy-MM-dd");
        dateChooser.setPreferredSize(new java.awt.Dimension(99, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        leftPanel.add(dateChooser, gridBagConstraints);

        Location.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Location.setForeground(new java.awt.Color(51, 51, 51));
        Location.setText("Location");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 99, 0, 0);
        leftPanel.add(Location, gridBagConstraints);

        locationTextField.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        locationTextField.setForeground(new java.awt.Color(51, 51, 51));
        locationTextField.setPreferredSize(new java.awt.Dimension(64, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        leftPanel.add(locationTextField, gridBagConstraints);

        Landmark.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Landmark.setForeground(new java.awt.Color(51, 51, 51));
        Landmark.setText("Landmark");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 89, 0, 0);
        leftPanel.add(Landmark, gridBagConstraints);

        landmarkTextField.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        landmarkTextField.setForeground(new java.awt.Color(51, 51, 51));
        landmarkTextField.setPreferredSize(new java.awt.Dimension(64, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        leftPanel.add(landmarkTextField, gridBagConstraints);

        urgencyLevelComboBox.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        urgencyLevelComboBox.setForeground(new java.awt.Color(51, 51, 51));
        urgencyLevelComboBox.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "<html><font
color='b6b6b6'>Select...</font></html>", "Low", "Medium", "High", "Emergency"
}));
        urgencyLevelComboBox.setPreferredSize(new java.awt.Dimension(81, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        leftPanel.add(urgencyLevelComboBox, gridBagConstraints);

        categoryHelpButton.setToolTipText("See category description");
        categoryHelpButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        categoryHelpButton.setPreferredSize(new java.awt.Dimension(25, 25));
        categoryHelpButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                categoryHelpButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridy = 0;
        gridBagConstraints.insets = new java.awt.Insets(0, 10, 0, 0);
        leftPanel.add(categoryHelpButton, gridBagConstraints);

        urgencyLevelHelpButton.setToolTipText("See Urgency Level
description");
        urgencyLevelHelpButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        urgencyLevelHelpButton.setPreferredSize(new java.awt.Dimension(25,
25));
        urgencyLevelHelpButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                urgencyLevelHelpButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridy = 10;
        gridBagConstraints.insets = new java.awt.Insets(0, 10, 0, 0);
        leftPanel.add(urgencyLevelHelpButton, gridBagConstraints);

        urgencyLevelComboBox1.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        urgencyLevelComboBox1.setForeground(new java.awt.Color(51, 51, 51));
        urgencyLevelComboBox1.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "<html><font
color='b6b6b6'>Select...</font></html>", "Low", "Medium", "High", "Emergency"
}));
        urgencyLevelComboBox1.setPreferredSize(new java.awt.Dimension(81,
40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        leftPanel.add(urgencyLevelComboBox1, gridBagConstraints);

        Urgency1.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        Urgency1.setForeground(new java.awt.Color(51, 51, 51));
        Urgency1.setText("Urgency level");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 68, 0, 0);
        leftPanel.add(Urgency1, gridBagConstraints);

        java.awt.GridBagLayout rightPanelLayout = new
java.awt.GridBagLayout();
        rightPanelLayout.columnWidths = new int[] {0, 20, 0, 20, 0, 20, 0, 20,
0, 20, 0};
        rightPanelLayout.rowHeights = new int[] {0, 16, 0, 16, 0};
        rightPanel.setLayout(rightPanelLayout);

        proofLabel.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        proofLabel.setForeground(new java.awt.Color(51, 51, 51));
        proofLabel.setText("Attach proof for your complaint");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        rightPanel.add(proofLabel, gridBagConstraints);

        uploadFileButton.setFont(new java.awt.Font("Roboto", 0, 16)); //
NOI18N
        uploadFileButton.setForeground(new java.awt.Color(51, 51, 51));
        uploadFileButton.setText("Upload");
        uploadFileButton.setToolTipText("Browse on file");
        uploadFileButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        uploadFileButton.setPreferredSize(new java.awt.Dimension(100, 40));
        uploadFileButton.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                uploadFileButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        rightPanel.add(uploadFileButton, gridBagConstraints);

        proofImageLabel.setBackground(new java.awt.Color(255, 255, 255));
        proofImageLabel.setToolTipText("Proof of complaint");
        proofImageLabel.setAlignmentX(0.5F);

proofImageLabel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(209, 209, 209)));
        proofImageLabel.setOpaque(true);
        proofImageLabel.setPreferredSize(new java.awt.Dimension(300, 250));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        rightPanel.add(proofImageLabel, gridBagConstraints);
        javax.swing.GroupLayout contentLayout = new
javax.swing.GroupLayout(content);
        content.setLayout(contentLayout);
        contentLayout.setHorizontalGroup(

contentLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(contentLayout.createSequentialGroup()
                .addComponent(leftPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(rightPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 422,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        contentLayout.setVerticalGroup(

contentLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(leftPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(rightPanel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );

        clearButton.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        clearButton.setForeground(new java.awt.Color(51, 51, 51));
        clearButton.setText("Clear");
        clearButton.setToolTipText("Clear");
        clearButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        clearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearButtonActionPerformed(evt);
            }
        });

        submitButton.setBackground(new java.awt.Color(0, 122, 255));
        submitButton.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        submitButton.setForeground(new java.awt.Color(255, 255, 255));
        submitButton.setText("Submit");
        submitButton.setToolTipText("Submit");
        submitButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        submitButton.setPreferredSize(new java.awt.Dimension(80, 40));
        submitButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                submitButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout footerLayout = new
javax.swing.GroupLayout(footer);
        footer.setLayout(footerLayout);
        footerLayout.setHorizontalGroup(

footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, 
footerLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE)
                .addComponent(clearButton, 
javax.swing.GroupLayout.PREFERRED_SIZE, 100, 
javax.swing.GroupLayout.PREFERRED_SIZE)
                
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(submitButton, 
javax.swing.GroupLayout.PREFERRED_SIZE, 100, 
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(93, 93, 93))
        );
        footerLayout.setVerticalGroup(
            
footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(footerLayout.createSequentialGroup()
                .addContainerGap()
                
.addGroup(footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                    .addComponent(clearButton, 
javax.swing.GroupLayout.PREFERRED_SIZE, 40, 
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(submitButton, 
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, 
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(104, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(header, javax.swing.GroupLayout.DEFAULT_SIZE, 
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(content, javax.swing.GroupLayout.Alignment.TRAILING, 
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE)
            .addComponent(footer, javax.swing.GroupLayout.DEFAULT_SIZE, 
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, 
layout.createSequentialGroup()
                .addGap(80, 80, 80)
                .addComponent(footerSeparator, 
javax.swing.GroupLayout.PREFERRED_SIZE, 896, 
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(80, 80, 80))
        );
        layout.setVerticalGroup(
            
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, 
layout.createSequentialGroup()
                .addComponent(header, javax.swing.GroupLayout.PREFERRED_SIZE, 
179, javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(content, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(footerSeparator,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(footer, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        );
    }// </editor-fold>

    String descriptionPlaceholder = "e.g. \"A plothole in main street\"";

    /**
     * Handles the event when the description text area loses focus.
     * If the text area is empty, sets its text to a placeholder and changes
its text color.
     *
     */
    private void descriptionTextAreaFocusLost(java.awt.event.FocusEvent evt) {

        if (descriptionTextArea.getText().isEmpty()) {
            descriptionTextArea.setText(descriptionPlaceholder);
            descriptionTextArea.setForeground(Color.decode("#b6b6b6"));
        }

    }

    /**
     * Handles the event when the description text area gains focus.
     * If the text area contains the placeholder, clears the text and resets
its text color.
     *
     */
    private void descriptionTextAreaFocusGained(java.awt.event.FocusEvent evt)
{

        if (descriptionTextArea.getText().equals(descriptionPlaceholder)) {
            descriptionTextArea.setText("");
            descriptionTextArea.setForeground(Color.decode("#333333"));
        }

    }


    /**
     * Handles the event when the "Upload File" button is clicked.
     * Opens a file chooser dialog to select an image file and displays it in
the proof image label.
     *
     */
    private void uploadFileButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        JFileChooser fc = new JFileChooser();
        fc.addChoosableFileFilter(new ImageFilter());
        fc.setAcceptAllFileFilterUsed(false);
        int returnValue = fc.showOpenDialog(this);

        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            Image image  = new ImageIcon(file.getAbsolutePath()).getImage();
            Image scaledImage =
image.getScaledInstance(proofImageLabel.getWidth(),

proofImageLabel.getHeight(),
                                                        Image.SCALE_SMOOTH);


            proofImageLabel.setIcon(new ImageIcon(scaledImage));

            try {
                this.proofImageData = readImageToByteArray(file);
            } catch (IOException ex) {

Logger.getLogger(CreateComplaint.class.getName()).log(Level.SEVERE,
"uploadFileButtonActionPerformed method", ex);
            }
        }

    }

    /**
     * Reads the contents of an image file and converts it to a byte array.
     *
     * @param file The image file to be read.
     * @return The byte array representing the image.
     * @throws IOException If an I/O error occurs while reading the file.
     */
    private byte[] readImageToByteArray(File file) throws IOException {

        try (FileInputStream fis = new FileInputStream(file)) {
            byte[] buffer = new byte[(int) file.length()];
            fis.read(buffer);
            return buffer;
        }

    }


    /**
     * Handles the event when the "Clear" button is clicked.
     * Clears all input fields and resets the proof image label.
     */
    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {

        clearInputFields();

    }

     /**
     * Clears all input fields on the form.
     */
    private void clearInputFields() {

        categoryComboBox.setSelectedIndex(0);
        descriptionTextArea.setText("");
        descriptionTextAreaFocusLost(new FocusEvent(descriptionTextArea,
FocusEvent.FOCUS_LOST));
        dateChooser.setDate(null);
        locationTextField.setText("");
        landmarkTextField.setText("");
        urgencyLevelComboBox.setSelectedIndex(0);
        proofImageLabel.setIcon(null);

    }

    /**
     * Retrieves the current user's ID from the database based on the logged-
in email.
     *
     * @return The ID of the current user.
     */
    private int getCurrentUser() {

        String selectQuery =
                "SELECT [UserID] FROM [User] " +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{

            pst.setString(1, this.email);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                return this.currentUser = rs.getInt("UserID");

            }

        } catch (SQLException ex) {
            Logger.getLogger(CreateComplaint.class.getName())
                  .log(Level.SEVERE, "getCurrentUser method", ex);
        }

        return -1;
    }


    /**
     * Handles the event when the "Submit" button is clicked.
     * Collects input data, validates it, and submits a complaint to the
database.
     */
    private void submitButtonActionPerformed(java.awt.event.ActionEvent evt) {

        this.category = categoryComboBox.getSelectedItem().toString();
        this.description = descriptionTextArea.getText();
        try {
            this.createdDate = dateChooser.getDate().toInstant()
```

```
                              .atZone(ZoneId.systemDefault())
                              .toLocalDate().toString();
        } catch (NullPointerException ex) {
            JOptionPane.showMessageDialog(
                        content,
                        "Fill out needed information!",
                        "Reminder",
                        JOptionPane.ERROR_MESSAGE);

            Logger.getLogger(CreateComplaint.class.getName())
                  .log(Level.INFO, ex.getMessage(), ex);

            return;
        }
        this.location = locationTextField.getText();
        this.landmark = landmarkTextField.getText();
        this.urgencyLevel = urgencyLevelComboBox.getSelectedItem().toString();

        try {

            if (isInputFieldEmpty() || isProofImageIconEmpty()) {

                JOptionPane.showMessageDialog(
                        content,
                        "Fill out needed information!",
                        "Reminder",
                        JOptionPane.ERROR_MESSAGE);

                return;
            }

        } catch (NullPointerException ex) {
            JOptionPane.showMessageDialog(
                content,
                "Fill out needed information!",
                "Reminder",
                JOptionPane.ERROR_MESSAGE);

            Logger.getLogger(CreateComplaint.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);

            return;
        }

        String userComplaintQuery =
                "INSERT INTO [UserComplaint] (" +
                    "[Category], [Description], [CreatedDate], " +
                    "[Location], [Landmark], [UrgencyLevel], " +
                    "[Status], [Proof], [UserID]) " +
                "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

        try (PreparedStatement complaint =
connection.prepareStatement(userComplaintQuery)) {

            // Complaint Table
            complaint.setString(1, category);
            complaint.setString(2, description);
            try {
                complaint.setDate(3, new
java.sql.Date(dateFormat.parse(this.createdDate).getTime()));
            } catch (ParseException ex) {
                Logger.getLogger(CreateComplaint.class.getName())
                      .log(Level.INFO, ex.getMessage(), ex);
            }
            complaint.setString(4, location);
            complaint.setString(5, landmark);
            complaint.setString(6, urgencyLevel);
            complaint.setString(7, ComplaintStatus.NEW.getDescription());
            complaint.setBytes(8, proofImageData);
            complaint.setInt(9, this.currentUser);
            complaint.executeUpdate();

        } catch (SQLException ex) {
            Logger.getLogger(CreateComplaint.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
            return;
        }

        JOptionPane.showMessageDialog(
            this,
            "Your complaint has been successfully submitted.",
            "Submission Successful",
            JOptionPane.INFORMATION_MESSAGE);

        clearInputFields();
    }

    /**
     * Checks if any of the input fields related to complaint details are
empty.
     *
     * @return true if any input field is empty, false otherwise.
     * @throws NullPointerException If any required field is null.
     */
    private boolean isInputFieldEmpty() throws NullPointerException {

        String placeholder = "<html><font
color='b6b6b6'>Select...</font></html>";

        return description.isBlank() ||
                createdDate == null ||
               location.isBlank() ||
               landmark.isBlank() ||
               category.equals(placeholder) ||
               urgencyLevel.equals(placeholder);

    }

    /**
     * Checks if the proofImageLabel has an image icon.
     *
     * @return true if the proofImageLabel has no image icon, false otherwise.
     * @throws NullPointerException If the proof image label is null.
     */
    private boolean isProofImageIconEmpty() throws NullPointerException {

        return proofImageLabel.getIcon() == null;
```

```
   }

    /**
     * Displays a popup with information about complaint categories.
     */
    private void categoryHelpButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        String popUpDetails =
                "<html>" +
                "<b>Please categorize your complaint according to the
following categories:</b><br><br>" +
                "1. <b>Environment:</b> Issues related to pollution, waste
management, natural resource conservation, etc.<br><br>" +
                "2. <b>Infrastructure:</b> Concerns about roads, bridges,
buildings, public transport, etc.<br><br>" +
                "3. <b>Utilities:</b> Problems with water supply, electricity,
gas, telecommunications, etc.<br><br>" +
                "4. <b>Public Services:</b> Issues with healthcare, education,
law enforcement, firefighting services, etc.<br><br>" +
                "5. <b>Government Agencies:</b> Complaints related to the
services provided by government departments and agencies." +
                "</html>";

        JOptionPane.showMessageDialog(categoryHelpButton, popUpDetails,
"Category Guide", JOptionPane.INFORMATION_MESSAGE);

    }


    /**
     * Displays a popup with information about complaint urgency levels.
     */
    private void
urgencyLevelHelpButtonActionPerformed(java.awt.event.ActionEvent evt) {

        String popUpDetails =
            "<html>" +
            "<b>Priority Levels for Complaints:</b><br><br>" +
            "<b>Low:</b><br>" +
            "Description: Non-urgent issues that do not require immediate
attention.<br>" +
            "Examples: Minor road damage, graffiti in public areas.<br><br>" +
            "<b>Medium:</b><br>" +
            "Description: Issues that need attention soon but are not
emergencies.<br>" +
            "Examples: Moderate traffic congestion, streetlights out in a
residential area.<br><br>" +
            "<b>High:</b><br>" +
            "Description: Urgent issues requiring prompt action or
intervention.<br>" +
            "Examples: Major road accidents, burst water mains, large-scale
environmental hazards.<br><br>" +
            "<b>Critical/Emergency:</b><br>" +
            "Description: Extremely urgent issues requiring immediate action
to prevent serious harm or damage.<br>" +
            "Examples: Major natural disasters, large-scale public health
emergencies." +
            "</html>";

            JOptionPane.showMessageDialog(urgencyLevelHelpButton,
popUpDetails, "Urgency Level Guide", JOptionPane.INFORMATION_MESSAGE);

    }





    // Variables declaration - do not modify
    private javax.swing.JLabel Category;
    private javax.swing.JLabel Date;
    private javax.swing.JLabel Description;
    private javax.swing.JLabel Heading;
    private javax.swing.JLabel Landmark;
    private javax.swing.JLabel Location;
    private javax.swing.JLabel Title;
    private javax.swing.JLabel Urgency1;
    private javax.swing.JComboBox<String> categoryComboBox;
    private javax.swing.JButton categoryHelpButton;
    private javax.swing.JButton clearButton;
    private javax.swing.JPanel content;
    private com.toedter.calendar.JDateChooser dateChooser;
    private javax.swing.JScrollPane descriptionScrollPane;
    private javax.swing.JTextArea descriptionTextArea;
    private javax.swing.JPanel footer;
    private javax.swing.JSeparator footerSeparator;
    private javax.swing.JPanel header;
    private javax.swing.JTextField landmarkTextField;
    private javax.swing.JPanel leftPanel;
    private javax.swing.JTextField locationTextField;
    private javax.swing.JLabel proofImageLabel;
    private javax.swing.JLabel proofLabel;
    private javax.swing.JPanel rightPanel;
    private javax.swing.JButton submitButton;
    private javax.swing.JButton uploadFileButton;
    private javax.swing.JComboBox<String> urgencyLevelComboBox;
    private javax.swing.JComboBox<String> urgencyLevelComboBox1;
    private javax.swing.JButton urgencyLevelHelpButton;
    // End of variables declaration
}
```

Account Setting

```
package project.citizen;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Image;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
```

```java
import java.time.ZoneId;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import project.authentication.LoginFrame;
import project.concrete_class.ImageFilter;
import project.database.Database;

/**
 * This class represents the Account Setting panel of the application.
 * Users can view and update their account information and change their
password using this panel.
 *
 */
public class AccountSetting extends javax.swing.JPanel {

    // Account Info
    private String email;

    // User Info
    private byte[] profilePictureData;
    private String firstName;
    private String lastName;
    private String contactNo;
    private String birthdate;
    private String street;
    private String barangay;
    private String city;
    private String province;

    // Change password Info
    private String currentPassword;
    private String newPassword;
    private String confirmPassword;

    Connection connection;

    public AccountSetting() {
        initComponents();
        customizeComponent();
        loadDefaultProfileImage();
        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(AccountSetting.class.getName())
                    .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.email = LoginFrame.email;

        initIntanceVariable();

        try {
            fetchDataIntoInputField();
        } catch (IllegalArgumentException ex) {
            Logger.getLogger(AccountSetting.class.getName())
```

```
                    .log(Level.INFO, "fetchDataIntoInputField method", ex);
        }


    }


    /**
     * Customizes the appearance of Swing components.
     * Sets styles and properties for password fields.
     */
    private void customizeComponent() {

        currentPasswordField.putClientProperty(FlatClientProperties.STYLE,
                "showRevealButton: true;" +
                "showCapsLock: true;" +
                "margin: 3, 10, 3, 10;");

        newPasswordField.putClientProperty(FlatClientProperties.STYLE,
                "showRevealButton: true;" +
                "showCapsLock: true;" +
                "margin: 3, 10, 3, 10;");

        confirmPasswordField.putClientProperty(FlatClientProperties.STYLE,
                "showRevealButton: true;" +
                "showCapsLock: true;" +
                "margin: 3, 10, 3, 10;");

    }

    private void loadDefaultProfileImage() {
        ImageIcon defaultProfile = new
ImageIcon(getClass().getResource("/profile-picture/default-profile-
picture.png"));
        Image scaledImage = defaultProfile.getImage().getScaledInstance(70,
                                                                        70,

Image.SCALE_SMOOTH);

        profilePictureLabel.setIcon(new ImageIcon(scaledImage));
        profilePictureLabel.setHorizontalAlignment(JLabel.CENTER);
        profilePictureLabel.setVerticalAlignment(JLabel.CENTER);
    }

    /**
     * Fetches user data from the database and populates input fields with
retrieved data.
     * Throws IllegalArgumentException if any retrieved data is null.
     */
    private void fetchDataIntoInputField() throws IllegalArgumentException {

        firstNameTextField.setText(this.firstName);
        lastNameTextField.setText(this.lastName);
        emailTextField.setText(this.email);
        contactNoTextField.setText(this.contactNo);
        streetTextField.setText(this.street);
        barangayTextField.setText(this.barangay);
        cityTextField.setText(this.city);
        provinceTextField.setText(this.province);
        try {


profilePictureLabel.setIcon(scaledImageIcon(this.profilePictureData));

            birthdateChooser.setDate(java.sql.Date.valueOf(this.birthdate));

        } catch (NullPointerException ex) {
            Logger.getLogger(AccountSetting.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

    }


    /**
     * Scales the profile picture image to fit a 100x100 pixel area.
     *
     * @param pictureData The byte array data of the profile picture.
     * @return The scaled ImageIcon of the profile picture.
     * @throws NullPointerException If the profile picture data is null.
     */
    private ImageIcon scaledImageIcon(byte[] pictureData) throws
NullPointerException {

        Image imageIcon = new ImageIcon(profilePictureData).getImage()

.getScaledInstance(100,

100,

Image.SCALE_SMOOTH);

        return new ImageIcon(imageIcon);
    }

    /**
     * Retrieves user data from the database based on the logged-in email and
     * initializes instance variables.
     */
    private void initIntanceVariable() {

        String selectQuery =
                "SELECT [Firstname], [Lastname], [ContactNo], [Birthdate],
[Street], [Barangay], [City], [Province], [ProfilePicture] " +
                "FROM [User] " +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{

            pst.setString(1, this.email);

            ResultSet rs = pst.executeQuery();
            while (rs.next()) {

                this.firstName = rs.getString("Firstname");
                this.lastName = rs.getString("Lastname");
                this.contactNo = rs.getString("ContactNo");
                this.birthdate = rs.getString("Birthdate");
                this.street = rs.getString("Street");
                this.barangay = rs.getString("Barangay");
                this.city = rs.getString("City");
                this.province = rs.getString("Province");
                this.profilePictureData = rs.getBytes("ProfilePicture");

            }

        } catch (SQLException ex) {
            Logger.getLogger(AccountSetting.class.getName()).log(Level.SEVERE,
"initIntanceVariable method", ex);
        }

    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        personalInformationPanel = new javax.swing.JPanel();
        descriptionPanel = new javax.swing.JPanel();
        titleLabel = new javax.swing.JLabel();
        header1Label = new javax.swing.JLabel();
        profilePanel = new javax.swing.JPanel();
        profilePictureLabel = new javax.swing.JLabel();
        uploadButton = new javax.swing.JButton();
        inputFieldsPanel = new javax.swing.JPanel();
        firstNameTextField = new javax.swing.JTextField();
        lastNameTextField = new javax.swing.JTextField();
        emailTextField = new javax.swing.JTextField();
        contactNoTextField = new javax.swing.JTextField();
        birthdateChooser = new com.toedter.calendar.JDateChooser();
        streetTextField = new javax.swing.JTextField();
        barangayTextField = new javax.swing.JTextField();
        cityTextField = new javax.swing.JTextField();
        provinceTextField = new javax.swing.JTextField();
        editButton = new javax.swing.JButton();
        saveButton = new javax.swing.JButton();
        changePasswordPanel = new javax.swing.JPanel();
        descriptionPanel2 = new javax.swing.JPanel();
        titleLabel1 = new javax.swing.JLabel();
        header1Label1 = new javax.swing.JLabel();
        inputFieldsPanel2 = new javax.swing.JPanel();
        changePasswordSaveButton = new javax.swing.JButton();
        currentPasswordField = new javax.swing.JPasswordField();
        newPasswordField = new javax.swing.JPasswordField();
        confirmPasswordField = new javax.swing.JPasswordField();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        personalInformationPanel.setBackground(new java.awt.Color(255, 255,
255));

personalInformationPanel.setBorder(javax.swing.BorderFactory.createLineBorder(
new java.awt.Color(209, 209, 209)));
```

```
        descriptionPanel.setOpaque(false);
        java.awt.GridBagLayout descriptionPanelLayout = new
java.awt.GridBagLayout();
        descriptionPanelLayout.columnWidths = new int[] {0};
        descriptionPanelLayout.rowHeights = new int[] {0, 5, 0};
        descriptionPanel.setLayout(descriptionPanelLayout);

        titleLabel.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        titleLabel.setForeground(new java.awt.Color(51, 51, 51));
        titleLabel.setText("Personal Information");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.insets = new java.awt.Insets(3, 15, 0, 0);
        descriptionPanel.add(titleLabel, gridBagConstraints);

        header1Label.setForeground(new java.awt.Color(51, 51, 51));
        header1Label.setText("Update your personal details here.");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.insets = new java.awt.Insets(0, 15, 236, 0);
        descriptionPanel.add(header1Label, gridBagConstraints);

        profilePanel.setOpaque(false);
        java.awt.GridBagLayout profilePanelLayout = new
java.awt.GridBagLayout();
        profilePanelLayout.columnWidths = new int[] {0};
        profilePanelLayout.rowHeights = new int[] {0, 15, 0};
        profilePanel.setLayout(profilePanelLayout);

        profilePictureLabel.setBackground(new java.awt.Color(255, 255, 255));

profilePictureLabel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(209, 209, 209)));
        profilePictureLabel.setOpaque(true);
        profilePictureLabel.setPreferredSize(new java.awt.Dimension(100,
100));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        profilePanel.add(profilePictureLabel, gridBagConstraints);

        uploadButton.setBackground(new java.awt.Color(245, 245, 245));
        uploadButton.setText("Upload");
        uploadButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        uploadButton.setEnabled(false);
        uploadButton.setPreferredSize(new java.awt.Dimension(79, 40));
        uploadButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                uploadButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 119, 0);
        profilePanel.add(uploadButton, gridBagConstraints);

        inputFieldsPanel.setOpaque(false);
        java.awt.GridBagLayout inputFieldsPanelLayout = new
java.awt.GridBagLayout();
        inputFieldsPanelLayout.columnWidths = new int[] {0, 20, 0, 20, 0, 20,
0};
        inputFieldsPanelLayout.rowHeights = new int[] {0, 10, 0, 10, 0, 10, 0,
10, 0};
        inputFieldsPanel.setLayout(inputFieldsPanelLayout);

        firstNameTextField.setForeground(new java.awt.Color(51, 51, 51));

firstNameTextField.setHorizontalAlignment(javax.swing.JTextField.LEFT);

firstNameTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(ja
vax.swing.BorderFactory.createTitledBorder(null, "First Name",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        firstNameTextField.setEnabled(false);
        firstNameTextField.setOpaque(true);
        firstNameTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(firstNameTextField, gridBagConstraints);

        lastNameTextField.setForeground(new java.awt.Color(51, 51, 51));

lastNameTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(jav
ax.swing.BorderFactory.createTitledBorder(null, "Last Name",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        lastNameTextField.setEnabled(false);
        lastNameTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(lastNameTextField, gridBagConstraints);

        emailTextField.setForeground(new java.awt.Color(51, 51, 51));

emailTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(javax.
swing.BorderFactory.createTitledBorder(null, "Email",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        emailTextField.setEnabled(false);
        emailTextField.setPreferredSize(new java.awt.Dimension(400, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(emailTextField, gridBagConstraints);

        contactNoTextField.setForeground(new java.awt.Color(51, 51, 51));

contactNoTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(ja
vax.swing.BorderFactory.createTitledBorder(null, "Contact No",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        contactNoTextField.setEnabled(false);
        contactNoTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(contactNoTextField, gridBagConstraints);

        birthdateChooser.setBackground(new java.awt.Color(255, 255, 255));

birthdateChooser.setBorder(javax.swing.BorderFactory.createCompoundBorder(java
x.swing.BorderFactory.createTitledBorder(null, "Birthdate",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0))); // NOI18N
        birthdateChooser.setForeground(new java.awt.Color(51, 51, 51));
        birthdateChooser.setEnabled(false);
        birthdateChooser.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.LINE_START;
        inputFieldsPanel.add(birthdateChooser, gridBagConstraints);

        streetTextField.setForeground(new java.awt.Color(51, 51, 51));

streetTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(javax
.swing.BorderFactory.createTitledBorder(null, "Street",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        streetTextField.setEnabled(false);
        streetTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(streetTextField, gridBagConstraints);

        barangayTextField.setForeground(new java.awt.Color(51, 51, 51));
barangayTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(jav
ax.swing.BorderFactory.createTitledBorder(null, "Barangay",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        barangayTextField.setEnabled(false);
        barangayTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(barangayTextField, gridBagConstraints);

        cityTextField.setForeground(new java.awt.Color(51, 51, 51));

cityTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(javax.s
wing.BorderFactory.createTitledBorder(null, "City",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        cityTextField.setEnabled(false);
        cityTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(cityTextField, gridBagConstraints);

        provinceTextField.setForeground(new java.awt.Color(51, 51, 51));

provinceTextField.setBorder(javax.swing.BorderFactory.createCompoundBorder(jav
ax.swing.BorderFactory.createTitledBorder(null, "Province",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        provinceTextField.setEnabled(false);
        provinceTextField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
        inputFieldsPanel.add(provinceTextField, gridBagConstraints);

        editButton.setBackground(new java.awt.Color(245, 245, 245));
        editButton.setForeground(new java.awt.Color(51, 51, 51));
        editButton.setText("Edit");
        editButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        editButton.setPreferredSize(new java.awt.Dimension(95, 40));
        editButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                editButtonActionPerformed(evt);
            }
        });
        saveButton.setBackground(new java.awt.Color(0, 122, 255));
        saveButton.setForeground(new java.awt.Color(255, 255, 255));
        saveButton.setText("Save");
        saveButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        saveButton.setEnabled(false);
        saveButton.setPreferredSize(new java.awt.Dimension(95, 40));
        saveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout personalInformationPanelLayout = new
javax.swing.GroupLayout(personalInformationPanel);
        personalInformationPanel.setLayout(personalInformationPanelLayout);
        personalInformationPanelLayout.setHorizontalGroup(

personalInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
personalInformationPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(descriptionPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 300,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(89, 89, 89)
                .addComponent(profilePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(inputFieldsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 522, Short.MAX_VALUE)
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
personalInformationPanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(editButton,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(saveButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 92,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(55, 55, 55))
        );
        personalInformationPanelLayout.setVerticalGroup(

personalInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(personalInformationPanelLayout.createSequentialGroup()
                .addGap(15, 15, 15)

.addGroup(personalInformationPanelLayout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
```

```
                    .addComponent(inputFieldsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 333, Short.MAX_VALUE)
                    .addComponent(descriptionPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(profilePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(personalInformationPanelLayout.createParallelGroup(javax.swing.Group
Layout.Alignment.BASELINE)
                    .addComponent(editButton,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(saveButton,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(29, 29, 29))
        );

        changePasswordPanel.setBackground(new java.awt.Color(255, 255, 255));

changePasswordPanel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(209, 209, 209)));

        descriptionPanel2.setOpaque(false);
        java.awt.GridBagLayout descriptionPanel2Layout = new
java.awt.GridBagLayout();
        descriptionPanel2Layout.columnWidths = new int[] {0};
        descriptionPanel2Layout.rowHeights = new int[] {0, 5, 0};
        descriptionPanel2.setLayout(descriptionPanel2Layout);

        titleLabel1.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        titleLabel1.setForeground(new java.awt.Color(51, 51, 51));
        titleLabel1.setText("Change Password");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        descriptionPanel2.add(titleLabel1, gridBagConstraints);

        header1Label1.setForeground(new java.awt.Color(51, 51, 51));
        header1Label1.setText("<html>Update your password associated </br
>with your account.</html>\n\n");
        header1Label1.setPreferredSize(new java.awt.Dimension(250, 40));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 150, 0);
        descriptionPanel2.add(header1Label1, gridBagConstraints);

        inputFieldsPanel2.setOpaque(false);
        java.awt.GridBagLayout inputFieldsPanel1Layout = new
java.awt.GridBagLayout();
        inputFieldsPanel1Layout.columnWidths = new int[] {0, 20, 0};
        inputFieldsPanel1Layout.rowHeights = new int[] {0, 10, 0, 10, 0};
        inputFieldsPanel2.setLayout(inputFieldsPanel1Layout);
        changePasswordSaveButton.setBackground(new java.awt.Color(0, 122,
255));
        changePasswordSaveButton.setForeground(new java.awt.Color(255, 255,
255));
        changePasswordSaveButton.setText("Save");
        changePasswordSaveButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        changePasswordSaveButton.setPreferredSize(new java.awt.Dimension(95,
40));
        changePasswordSaveButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                changePasswordSaveButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        gridBagConstraints.insets = new java.awt.Insets(16, 0, 35, 0);
        inputFieldsPanel2.add(changePasswordSaveButton, gridBagConstraints);

        currentPasswordField.setForeground(new java.awt.Color(51, 51, 51));

currentPasswordField.setBorder(javax.swing.BorderFactory.createCompoundBorder(
javax.swing.BorderFactory.createTitledBorder(null, "Current Password",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        currentPasswordField.setPreferredSize(new java.awt.Dimension(200,
50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        inputFieldsPanel2.add(currentPasswordField, gridBagConstraints);

        newPasswordField.setForeground(new java.awt.Color(51, 51, 51));

newPasswordField.setBorder(javax.swing.BorderFactory.createCompoundBorder(java
x.swing.BorderFactory.createTitledBorder(null, "New Password",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        newPasswordField.setPreferredSize(new java.awt.Dimension(200, 50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        inputFieldsPanel2.add(newPasswordField, gridBagConstraints);

        confirmPasswordField.setForeground(new java.awt.Color(51, 51, 51));

confirmPasswordField.setBorder(javax.swing.BorderFactory.createCompoundBorder(
javax.swing.BorderFactory.createTitledBorder(null, "Confirm Password",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Segoe
UI", 0, 12), new java.awt.Color(51, 51, 51)),
javax.swing.BorderFactory.createEmptyBorder(0, 8, 5, 0))); // NOI18N
        confirmPasswordField.setPreferredSize(new java.awt.Dimension(200,
50));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 2;
        inputFieldsPanel2.add(confirmPasswordField, gridBagConstraints);

        javax.swing.GroupLayout changePasswordPanelLayout = new
javax.swing.GroupLayout(changePasswordPanel);
        changePasswordPanel.setLayout(changePasswordPanelLayout);
        changePasswordPanelLayout.setHorizontalGroup(

changePasswordPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(changePasswordPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(descriptionPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(145, 145, 145)
                .addComponent(inputFieldsPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addContainerGap())
        );
        changePasswordPanelLayout.setVerticalGroup(

changePasswordPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(changePasswordPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(changePasswordPanelLayout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING)
                    .addComponent(inputFieldsPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(descriptionPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 269, Short.MAX_VALUE))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addComponent(changePasswordPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
                    .addComponent(personalInformationPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(15, 15, 15))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(15, 15, 15)
                .addComponent(personalInformationPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(changePasswordPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(15, Short.MAX_VALUE))
        );
    }// </editor-fold>

     /**
     * Enables input fields for editing user information.
     * Enables save button and disables edit button.
     */
    private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {

        enableInputFields(true);
        saveButton.setEnabled(true);
        editButton.setEnabled(false);

    }



    /**
     * Saves the edited user information to the database.
     * Disables input fields and save button, and enables edit button.
     */
    private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {

        enableInputFields(false);
        saveButton.setEnabled(false);
        editButton.setEnabled(true);

        firstName = firstNameTextField.getText();
        lastName = lastNameTextField.getText();
        contactNo = contactNoTextField.getText();
        try {
            birthdate = birthdateChooser.getDate().toInstant()
                                   .atZone(ZoneId.systemDefault())
                                   .toLocalDate().toString();
        } catch (NullPointerException ex) {
            Logger.getLogger(AccountSetting.class.getName()).log(Level.SEVERE,
"saveButtonActionPerformed method", ex);
        }
        street = streetTextField.getText();
        barangay = barangayTextField.getText();
        city = cityTextField.getText();
        province = provinceTextField.getText();

        String updateQuery =
                "UPDATE [User] " +
                "SET " +
                    "[Firstname] = ?, [Lastname] = ?, [ContactNo] = ?, " +
                    "[BirthDate] = ?, [Street] = ?, [Barangay] = ?, " +
                    "[City] = ?, [Province] = ?, [ProfilePicture] = ? " +
                "WHERE [Email] = ?";

        try (PreparedStatement userTable =
connection.prepareStatement(updateQuery)) {

            userTable.setString(1, this.firstName);
            userTable.setString(2, this.lastName);
            userTable.setString(3, this.contactNo);
            userTable.setString(4, this.birthdate);
            userTable.setString(5, this.street);
            userTable.setString(6, this.barangay);
            userTable.setString(7, this.city);
            userTable.setString(8, this.province);
            userTable.setBytes(9, this.profilePictureData);
            userTable.setString(10, this.email);
            userTable.executeUpdate();

            System.out.println("Successful");
            JOptionPane.showMessageDialog(
                    this,
                    "Your changes have been successfully saved.",
                    "Account Settings Updated",
                    JOptionPane.INFORMATION_MESSAGE);

        } catch (SQLException ex) {
            Logger.getLogger(AccountSetting.class.getName())
                  .log(Level.SEVERE, "saveButtonActionPerformed method", ex);
        }
    }


        /**
     * Enables or disables input fields for editing user information.
     *
     * @param isEnabled If true, input fields are enabled; if false, input
fields are disabled.
     */
    private void enableInputFields(boolean isEnabled) {

        firstNameTextField.setEnabled(isEnabled);
        lastNameTextField.setEnabled(isEnabled);
        contactNoTextField.setEnabled(isEnabled);
        birthdateChooser.setEnabled(isEnabled);
        streetTextField.setEnabled(isEnabled);
        barangayTextField.setEnabled(isEnabled);
        cityTextField.setEnabled(isEnabled);
        provinceTextField.setEnabled(isEnabled);
        uploadButton.setEnabled(isEnabled);

    }
    /**
     * Allows the user to select and upload a profile picture.
     */
    private void uploadButtonActionPerformed(java.awt.event.ActionEvent evt) {

        JFileChooser fc = new JFileChooser();
        fc.addChoosableFileFilter(new ImageFilter());
        fc.setAcceptAllFileFilterUsed(false);
        int returnValue = fc.showOpenDialog(this);

        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            Image image  = new ImageIcon(file.getAbsolutePath()).getImage();
            Image scaledImage =
image.getScaledInstance(profilePictureLabel.getWidth(),

profilePictureLabel.getHeight(),
                                                        Image.SCALE_SMOOTH);


            profilePictureLabel.setIcon(new ImageIcon(scaledImage));

            try {
                this.profilePictureData = readImageToByteArray(file);
            } catch (IOException ex) {

Logger.getLogger(AccountSetting.class.getName()).log(Level.SEVERE,
"readImageToByteArray method", ex);
            }
        }

    }


     /**
     * Saves the newly set password to the database after validating the
current password.
     */
    private void
changePasswordSaveButtonActionPerformed(java.awt.event.ActionEvent evt) {

        this.currentPassword =
String.valueOf(currentPasswordField.getPassword());
        this.newPassword = String.valueOf(newPasswordField.getPassword());
        this.confirmPassword =
String.valueOf(confirmPasswordField.getPassword());

        if (isPaswordFieldEmpty()) {

            JOptionPane.showMessageDialog(
                            this,
                            "Fill out needed information.",
                            "Change Password",
                            JOptionPane.ERROR_MESSAGE);
            return;

        }
        String selectQuery =
                "SELECT [Password] FROM [Account] " +
                "WHERE [Email] = ?";

        String updatePasswordQuery =
                "UPDATE [Account] " +
                "SET [Password] = ?" +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery);
             PreparedStatement updatePassword =
connection.prepareStatement(updatePasswordQuery)) {

            pst.setString(1, this.email);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                String dbCurrentPassword = rs.getString("Password");

                if (currentPasswordMatch(dbCurrentPassword)) {

                    if (newAndConfirmPasswordMatch()) {

                        updatePassword.setString(1, this.newPassword);
                        updatePassword.setString(2, this.email);
                        updatePassword.executeUpdate();

                        JOptionPane.showMessageDialog(
                                this,
                                "Password changed successful.",
                                "Change Password",
                                JOptionPane.INFORMATION_MESSAGE);

                        clearPasswordInputFields();

                        return;
                    }
                    else {

                        JOptionPane.showMessageDialog(
                                this,
                                "New Password and Confirm Password doens't
match.",
                                "Change Password",
                                JOptionPane.ERROR_MESSAGE);

                        return;
                    }
                }

                JOptionPane.showMessageDialog(
                        this,
                        "Current Password doesn't match.",
                        "Change Password",
                        JOptionPane.ERROR_MESSAGE);
                return;
            }

        } catch (SQLException ex) {
            Logger.getLogger(AccountSetting.class.getName())
                   .log(Level.SEVERE, "changePasswordSaveButtonActionPerformed
method", ex);
        }

    }

    /**
     * Checks if any of the password fields is empty.
     *
     * @return true if any of the password fields is empty, false otherwise.
     */
    private boolean isPaswordFieldEmpty() {

        return currentPassword.isBlank() || newPassword.isBlank() ||
confirmPassword.isBlank();

    }

    /**
     * Clears the input fields for the current password, new password, and
confirm password.
     */
    private void clearPasswordInputFields() {

        currentPasswordField.setText("");
        newPasswordField.setText("");
        confirmPasswordField.setText("");

    }

    /**
     * Checks if the entered current password matches the password stored in
the database.
     *
     * @param dbCurrentPassword The password retrieved from the database.
     * @return true if the current password matches, false otherwise.
     */
    private boolean currentPasswordMatch(String dbCurrentPassword) {

        return this.currentPassword.equals(dbCurrentPassword);

    }


    /**
     * Checks if the new password matches the confirmed password.
     *
     * @return true if the new password matches the confirmed password, false
otherwise.
     */
    private boolean newAndConfirmPasswordMatch() {

        return this.newPassword.equals(this.confirmPassword);
    }

    /**
     * Reads an image file and converts it into a byte array.
     *
     * @param file The image file to be converted.
     * @return The byte array representing the image.
     * @throws IOException If an I/O error occurs.
     */

    private byte[] readImageToByteArray(File file) throws IOException {

        try (FileInputStream fis = new FileInputStream(file)) {
            byte[] buffer = new byte[(int) file.length()];
            fis.read(buffer);
            return buffer;
        }

    }



    // Variables declaration - do not modify
    private javax.swing.JTextField barangayTextField;
    private com.toedter.calendar.JDateChooser birthdateChooser;
    private javax.swing.JPanel changePasswordPanel;
    private javax.swing.JButton changePasswordSaveButton;
    private javax.swing.JTextField cityTextField;
    private javax.swing.JPasswordField confirmPasswordField;
    private javax.swing.JTextField contactNoTextField;
    private javax.swing.JPasswordField currentPasswordField;
    private javax.swing.JPanel descriptionPanel;
    private javax.swing.JPanel descriptionPanel2;
    private javax.swing.JButton editButton;
    private javax.swing.JTextField emailTextField;
    private javax.swing.JTextField firstNameTextField;
    private javax.swing.JLabel header1Label;
    private javax.swing.JLabel header1Label1;
    private javax.swing.JPanel inputFieldsPanel;
    private javax.swing.JPanel inputFieldsPanel2;
    private javax.swing.JTextField lastNameTextField;
    private javax.swing.JPasswordField newPasswordField;
    private javax.swing.JPanel personalInformationPanel;
    private javax.swing.JPanel profilePanel;
    private javax.swing.JLabel profilePictureLabel;
    private javax.swing.JTextField provinceTextField;
    private javax.swing.JButton saveButton;
    private javax.swing.JTextField streetTextField;
    private javax.swing.JLabel titleLabel;
    private javax.swing.JLabel titleLabel1;
    private javax.swing.JButton uploadButton;
    // End of variables declaration
}
```

Admin

Admin Frame

```
package project.admin;

import com.formdev.flatlaf.FlatClientProperties;
import com.formdev.flatlaf.FlatLaf;
import com.formdev.flatlaf.fonts.roboto.FlatRobotoFont;
import com.formdev.flatlaf.themes.FlatMacLightLaf;
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Image;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import javax.swing.tree.DefaultMutableTreeNode;
import project.authentication.LoginFrame;
import project.database.Database;


public class AdminFrame extends javax.swing.JFrame {
    CardLayout cl;

    private Connection connection;
    private String email;
    private byte[] profilePictureData;
    private String fullName;

    public AdminFrame() {
        initComponents();
        customizeComponent();

        cl = (CardLayout) contentPanel.getLayout();

        try {
            connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(AdminFrame.class.getName()).log(Level.SEVERE,
                    ex.getMessage(), ex);
        }

        this.email = LoginFrame.email;

        initProfileInformation();

    }

    private void customizeComponent() {

        menuTree.putClientProperty(FlatClientProperties.STYLE,
                "rootVisible: false;" +
                "selectionBackground: #007AFF;" +
                "selectionInactiveBackground: #007AFF;" +
                "selectionInactiveForeground: #FFFFFF;" +
                "selectionArc: 8;");

    }

    private void initProfileInformation() {
        this.email = LoginFrame.email;

        String selectQuery =
                "SELECT [Firstname] + ' ' + [Lastname] AS Fullname,
[ProfilePicture] " +
                "FROM [User]" +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectQuery))
{

            pst.setString(1, this.email);

            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                this.fullName = rs.getString("Fullname");
                this.profilePictureData = rs.getBytes("ProfilePicture");
            }

        } catch (SQLException ex) {
            Logger.getLogger(AdminFrame.class.getName()).log(Level.SEVERE,
                    " initProfileInformation method ", ex);
        }

        nameLabel.setText(this.fullName);

        try {

this.profilePictureLabel.setIcon(scaledImageIcon(profilePictureData));
        } catch (NullPointerException ex) {
            Logger.getLogger(AdminFrame.class.getName()).log(Level.SEVERE,
                    " scaledImageIcon method ", ex);
        }


    }

    private ImageIcon scaledImageIcon(byte[] pictureData) throws
NullPointerException {

        Image imageIcon = new ImageIcon(profilePictureData).getImage()

.getScaledInstance(profilePictureLabel.getWidth(),

profilePictureLabel.getHeight(),

Image.SCALE_SMOOTH);

        return new ImageIcon(imageIcon);
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        mainPanel = new javax.swing.JPanel();
        menuPanel = new javax.swing.JPanel();
        profilePanel = new javax.swing.JPanel();
        profilePictureLabel = new javax.swing.JLabel();
        nameLabel = new javax.swing.JLabel();
        emailLabel = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        logoutButton = new javax.swing.JButton();
        menuTreeScrollPane = new javax.swing.JScrollPane();
        menuTree = new javax.swing.JTree();
        contentPanel = new javax.swing.JPanel();
        manageComplaints1 = new project.admin.ManageComplaints();
        manageUsers1 = new project.admin.ManageUsers();
        environmentReport1 = new
project.admin.report.category.EnvironmentReport();
        infrastructureReport1 = new
project.admin.report.category.InfrastructureReport();
        utilitiesReport1 = new
project.admin.report.category.UtilitiesReport();
        publicServices1 = new project.admin.report.category.PublicServices();
        governmentAgenciesReport1 = new
project.admin.report.category.GovernmentAgenciesReport();
        lowReport1 = new project.admin.report.urgency_level.LowReport();
        mediumReport1 = new project.admin.report.urgency_level.MediumReport();
        highReport1 = new project.admin.report.urgency_level.HighReport();
        emergencyReport1 = new
project.admin.report.urgency_level.EmergencyReport();
        newReport1 = new project.admin.report.status.NewReport();
        underReviewReport1 = new
project.admin.report.status.UnderReviewReport();
        assignedReport1 = new project.admin.report.status.AssignedReport();
        resolvedReport1 = new project.admin.report.status.ResolvedReport();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        mainPanel.setPreferredSize(new java.awt.Dimension(1300, 800));

        menuPanel.setBackground(new java.awt.Color(255, 255, 255));

        profilePanel.setBackground(new java.awt.Color(255, 255, 255));
        profilePanel.setPreferredSize(new java.awt.Dimension(240, 170));
        profilePanel.setLayout(new java.awt.GridBagLayout());

        profilePictureLabel.setPreferredSize(new java.awt.Dimension(70, 70));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 0;
        profilePanel.add(profilePictureLabel, gridBagConstraints);

        nameLabel.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        nameLabel.setForeground(new java.awt.Color(51, 51, 51));
        nameLabel.setText("Full Name");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(20, 0, 0, 0);
        profilePanel.add(nameLabel, gridBagConstraints);

        emailLabel.setFont(new java.awt.Font("Roboto", 0, 12)); // NOI18N
        emailLabel.setForeground(new java.awt.Color(128, 128, 128));
        emailLabel.setText("admin");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
        profilePanel.add(emailLabel, gridBagConstraints);

        jSeparator1.setForeground(new java.awt.Color(235, 235, 235));
        jSeparator1.setPreferredSize(new java.awt.Dimension(230, 10));

        logoutButton.setBackground(java.awt.Color.red);
        logoutButton.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N
        logoutButton.setForeground(new java.awt.Color(255, 255, 255));
        logoutButton.setText("Log out");
        logoutButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                logoutButtonActionPerformed(evt);
            }
        });

        menuTreeScrollPane.setBorder(null);

        menuTree.setFont(new java.awt.Font("Roboto", 0, 16)); // NOI18N
        javax.swing.tree.DefaultMutableTreeNode treeNode1 = new
javax.swing.tree.DefaultMutableTreeNode("root");
        javax.swing.tree.DefaultMutableTreeNode treeNode2 = new
javax.swing.tree.DefaultMutableTreeNode("Manage Complaints");
        treeNode1.add(treeNode2);
        treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Manage
Users");
        treeNode1.add(treeNode2);
        treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Reports");
        javax.swing.tree.DefaultMutableTreeNode treeNode3 = new
javax.swing.tree.DefaultMutableTreeNode("Category");
        javax.swing.tree.DefaultMutableTreeNode treeNode4 = new
javax.swing.tree.DefaultMutableTreeNode("Environment");
        treeNode3.add(treeNode4);
        treeNode4 = new
javax.swing.tree.DefaultMutableTreeNode("Infrastructure");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Utilities");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Public
Services");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Government
Agencies");
        treeNode3.add(treeNode4);
        treeNode2.add(treeNode3);
        treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("Urgency
Level");
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Low");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Medium");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("High");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Emergency");
        treeNode3.add(treeNode4);
        treeNode2.add(treeNode3);
        treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("Status");
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("New");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Under
Review");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Assigned");
        treeNode3.add(treeNode4);
        treeNode4 = new javax.swing.tree.DefaultMutableTreeNode("Resolved");
        treeNode3.add(treeNode4);
        treeNode2.add(treeNode3);
        treeNode1.add(treeNode2);
        menuTree.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
        menuTree.setRootVisible(false);
        menuTree.setRowHeight(40);
        menuTree.setSelectionRows(new int[] {0});
        menuTree.setShowsRootHandles(true);
        menuTree.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
                menuTreeMousePressed(evt);
            }
        });
        menuTreeScrollPane.setViewportView(menuTree);

        javax.swing.GroupLayout menuPanelLayout = new
javax.swing.GroupLayout(menuPanel);
        menuPanel.setLayout(menuPanelLayout);
        menuPanelLayout.setHorizontalGroup(

menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
menuPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING)
                    .addComponent(menuTreeScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 220, Short.MAX_VALUE)
                    .addComponent(logoutButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jSeparator1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addComponent(profilePanel,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
                .addContainerGap())
        );
        menuPanelLayout.setVerticalGroup(
menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(menuPanelLayout.createSequentialGroup()
                .addComponent(profilePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(menuTreeScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(logoutButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        contentPanel.setBackground(new java.awt.Color(204, 204, 255));
        contentPanel.setPreferredSize(new java.awt.Dimension(800, 600));
        contentPanel.setLayout(new java.awt.CardLayout());
        contentPanel.add(manageComplaints1, "Card1");
        contentPanel.add(manageUsers1, "Card2");
        contentPanel.add(environmentReport1, "Card3");
        contentPanel.add(infrastructureReport1, "Card4");
        contentPanel.add(utilitiesReport1, "Card5");
        contentPanel.add(publicServices1, "Card6");
        contentPanel.add(governmentAgenciesReport1, "Card7");
        contentPanel.add(lowReport1, "Card8");
        contentPanel.add(mediumReport1, "Card9");
        contentPanel.add(highReport1, "Card10");
        contentPanel.add(emergencyReport1, "Card11");
        contentPanel.add(newReport1, "Card12");
        contentPanel.add(underReviewReport1, "Card13");
        contentPanel.add(assignedReport1, "Card14");
        contentPanel.add(resolvedReport1, "Card15");

        javax.swing.GroupLayout mainPanelLayout = new
javax.swing.GroupLayout(mainPanel);
        mainPanel.setLayout(mainPanelLayout);
        mainPanelLayout.setHorizontalGroup(

mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addComponent(menuPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(contentPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1062, Short.MAX_VALUE))
        );
        mainPanelLayout.setVerticalGroup(
mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addComponent(contentPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 750,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addComponent(menuPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
756, Short.MAX_VALUE)
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void menuTreeMousePressed(java.awt.event.MouseEvent evt) {

        int selectedRow = menuTree.getClosestRowForLocation(evt.getX(),
evt.getY());
        DefaultMutableTreeNode node =
(DefaultMutableTreeNode)menuTree.getPathForRow(selectedRow).getLastPathCompone
nt();
        String selectedNode = node.toString();

        switch(selectedNode) {

            // Main menu
            case "Manage Complaints": cl.show(contentPanel, "Card1"); break;
            case "Manage Users": cl.show(contentPanel, "Card2"); break;

            // Category sub-menu
            case "Environment": cl.show(contentPanel, "Card3"); break;
            case "Infrastructure": cl.show(contentPanel, "Card4"); break;
            case "Utilities": cl.show(contentPanel, "Card5"); break;
            case "Public Services": cl.show(contentPanel, "Card6"); break;
            case "Government Agencies": cl.show(contentPanel, "Card7"); break;

            // Urgency Level sub-menu
            case "Low": cl.show(contentPanel, "Card8"); break;
            case "Medium": cl.show(contentPanel, "Card9"); break;
            case "High": cl.show(contentPanel, "Card10"); break;
            case "Emergency": cl.show(contentPanel, "Card11"); break;
            // Status sub-menu
            case "New": cl.show(contentPanel, "Card12"); break;
            case "Under Review": cl.show(contentPanel, "Card13"); break;
            case "Assigned": cl.show(contentPanel, "Card14"); break;
            case "Resolved": cl.show(contentPanel, "Card15"); break;

        }

    }

    private void logoutButtonActionPerformed(java.awt.event.ActionEvent evt) {

        int response = JOptionPane.showConfirmDialog(
                this,
                "Are you sure you want to log out?",
                "Log out Confirmation",
                JOptionPane.YES_NO_OPTION);

        final int YES = 0;
        if (response == YES) {
            new LoginFrame().setVisible(true);
            dispose();
        }

    }



    public static void main(String args[]) {

        // Overall UI Customization
        FlatLaf.registerCustomDefaultsSource("theme");
        FlatMacLightLaf.setup();

        FlatRobotoFont.install();

        // TitlePane Customization
        UIManager.put("TitlePane.unifiedBackground", false);

        // Table Customization
        UIManager.put("TableHeader.separatorColor", Color.decode("#ffffff"));
        UIManager.put("TableHeader.hoverBackground", Color.decode("#f2f2f2"));
        UIManager.put("TableHeader.height", 40);
        UIManager.put("Table.selectionBackground", Color.decode("#c2e7ff"));
        UIManager.put("Table.selectionForeground", Color.decode("#333333"));
        UIManager.put("Table.showCellFocusIndicator", true);
        //UIManager.put("Table.alternateRowColor", Color.decode("#f2f2f2"));
        //UIManager.put("Table.cellFocusColor", Color.decode("#B0E2FF"));

        java.awt.EventQueue.invokeLater(() -> {
            new AdminFrame().setVisible(true);
        });
    }

    // Variables declaration - do not modify
    private project.admin.report.status.AssignedReport assignedReport1;
    private javax.swing.JPanel contentPanel;
    private javax.swing.JLabel emailLabel;
```

```
    private project.admin.report.urgency_level.EmergencyReport
emergencyReport1;
    private project.admin.report.category.EnvironmentReport
environmentReport1;
    private project.admin.report.category.GovernmentAgenciesReport
governmentAgenciesReport1;
    private project.admin.report.urgency_level.HighReport highReport1;
    private project.admin.report.category.InfrastructureReport
infrastructureReport1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JButton logoutButton;
    private project.admin.report.urgency_level.LowReport lowReport1;
    private javax.swing.JPanel mainPanel;
    private project.admin.ManageComplaints manageComplaints1;
    private project.admin.ManageUsers manageUsers1;
    private project.admin.report.urgency_level.MediumReport mediumReport1;
    private javax.swing.JPanel menuPanel;
    private javax.swing.JTree menuTree;
    private javax.swing.JScrollPane menuTreeScrollPane;
    private javax.swing.JLabel nameLabel;
    private project.admin.report.status.NewReport newReport1;
    private javax.swing.JPanel profilePanel;
    private javax.swing.JLabel profilePictureLabel;
    private project.admin.report.category.PublicServices publicServices1;
    private project.admin.report.status.ResolvedReport resolvedReport1;
    private project.admin.report.status.UnderReviewReport underReviewReport1;
    private project.admin.report.category.UtilitiesReport utilitiesReport1;
    // End of variables declaration
}
```

Manage Users

```
package project.admin;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.MultilineTableCellRenderer;
import project.concrete_class.UserColumn;
import project.database.Database;
```

```
/**
 * The MyComplaint class represents a panel for displaying and managing user
complaints.
 * It provides functionalities to populate, filter, and interact with the
complaints table.
 * This panel includes options to view complaint details, withdraw complaints,
and refresh the table.
 *
 * The complaints are fetched from the database based on the currently logged-
in user's email.
 * Users can filter complaints by their status (e.g., New, Assigned,
Resolved).
 * Double-clicking on a complaint row displays detailed information about the
complaint,
 * including its category, description, date, location, landmark, urgency
level, and status.
 * Additionally, users can withdraw complaints by selecting the respective
option from the pop-up window.
 */
public class ManageUsers extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] profilePictureData;

    /**
     * Constructs a new MyComplaint panel.
     * Initializes GUI components, sets up table header, customizes
components,
     * establishes database connection, and populates the complaints table.
     */
    public ManageUsers() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeComponents();
        customizeCellRender();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                    .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) userTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }

    /**
     * Populates the complaints table with user-specific complaints retrieved
from the database.
```

```
     */
    private void populateTable() {

        tableModel.setRowCount(0);

        String selectQuery =
                "SELECT " +
                       "u.[Firstname] + ' ' + u.[Lastname], " +
                       "u.[Email], u.[ContactNo], " +
                       "u.[Street] + ' ' + u.[Barangay] + ' ' + " +
                       "u.[City] + ' ' + u.[Province], " +
                       "a.[Status] " +
                "FROM [User] AS u " +
                "INNER JOIN [Account] AS a " +
                    "ON u.[Email] = a.[Email] " +
                "WHERE a.[Role] != 'Admin'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();
            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " populateTable method ", ex);
        }
    }


    /**
     * Initializes the table header with a specific font and alignment.
     */
    private void initTableHeader() {

        JTableHeader header = userTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {
        final int STATUS_COLUMN = 4;

        TableColumn complaintNoColumn =
userTable.getColumnModel().getColumn(STATUS_COLUMN);
        complaintNoColumn.setPreferredWidth(20);

    }


    /**
     * Customizes components, such as setting padding for the status combo
box.
     */
    private void customizeComponents() {
        statusComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 3, 10, 3, 10;");

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");
    }


    /**
     * Customizes cell rendering for the complaints table to support multiline
text.
     */
    private void customizeCellRender() {

        for (int i = 0; i < userTable.getColumnCount(); i++) {
            userTable.getColumnModel()
                              .getColumn(i)
                              .setCellRenderer(new MultilineTableCellRenderer());
        }
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        updateStatusButton = new javax.swing.JButton();
        statusComboBox = new javax.swing.JComboBox<>();
        refreshButton = new javax.swing.JButton();
        printButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        userTableScrollPane = new javax.swing.JScrollPane();
        userTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0, 10,
0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);
        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setForeground(new java.awt.Color(51, 51, 51));
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 18;
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setForeground(new java.awt.Color(51, 51, 51));
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 18;
        topPanel.add(searchTextField, gridBagConstraints);

        updateStatusButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        updateStatusButton.setForeground(new java.awt.Color(51, 51, 51));
        updateStatusButton.setText("Update");
        updateStatusButton.setToolTipText("Update status of selected user.");
        updateStatusButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateStatusButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        topPanel.add(updateStatusButton, gridBagConstraints);

        statusComboBox.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        statusComboBox.setForeground(new java.awt.Color(51, 51, 51));
        statusComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "<html><font color='b6b6b6'>Status</font></html>", "Active",
"Suspended" }));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 8;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(statusComboBox, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setForeground(new java.awt.Color(51, 51, 51));
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 225);
        topPanel.add(refreshButton, gridBagConstraints);

        printButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        printButton.setForeground(new java.awt.Color(51, 51, 51));
        printButton.setText("Print");
        printButton.setToolTipText("Print user information");
        printButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 10;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        topPanel.add(printButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


userTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));

        userTable.setAutoCreateRowSorter(true);
        userTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        userTable.setForeground(new java.awt.Color(51, 51, 51));
        userTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "Name", "Email", "ContactNo", "Address", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        userTable.setFillsViewportHeight(true);
        userTable.setRowHeight(40);
        userTable.setShowHorizontalLines(true);
        userTable.getTableHeader().setReorderingAllowed(false);
        userTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                userTableMouseClicked(evt);
            }
        });
        userTableScrollPane.setViewportView(userTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(userTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(userTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(11, Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    /**
     * Refreshes the complaints table by clearing existing data and
repopulating it.
     */
    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        statusComboBox.setSelectedIndex(0);
        tableModel.setRowCount(0);
        populateTable();

    }



    /**
     * Handles the mouse click event on the complaints table.
     * Displays detailed information about the selected complaint and allows
withdrawal.
     */
    private void userTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = userTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " complaintTableMouseClicked method ",
ex);
        }

        // Getting the value of each column at the selected row.
        String fullName = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.NAME.ordinal()));
        String email = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.EMAIL.ordinal()));
        String contactNo = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.CONTACT_NO.ordinal()));
        String address = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.ADDRESS.ordinal()));
        String status = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [ProfilePicture] " +
                "FROM [User] " +
                "WHERE [Email] = ?";
        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setString(1, email);

            ResultSet rs = pst.executeQuery();
            while (rs.next()) {

                this.profilePictureData = rs.getBytes("ProfilePicture");

            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " complaintTableMouseClicked method ",
ex);
        }

        final int CLICKED_1_TIMES = 1;
        if (evt.getClickCount() == CLICKED_1_TIMES) {
            statusComboBox.setSelectedItem(status);
        }

        // Resizing the image
        ImageIcon imageIcon;
        Image scaledImage;
        ImageIcon scaledImageIcon = null;
        try {
            imageIcon = new ImageIcon(this.profilePictureData);
            scaledImage = imageIcon.getImage().getScaledInstance(250, 250,
Image.SCALE_SMOOTH);
            scaledImageIcon = new ImageIcon(scaledImage);
        } catch (NullPointerException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " complaintTableMouseClicked method ",
ex);
        }


        final int CLICKED_2_TIMES = 2;
        if (evt.getClickCount() == CLICKED_2_TIMES) {
            String userInformation =
                    "<html>" +
                    "Name:<b> " + fullName + "</b><br>" +
                    "Email:<b> " + email + "</b><br>" +
                    "Contact No:<b> " + contactNo + "</b><br>" +
                    "Address:<b> " + address + "</b><br>" +
                    "Status:<b> " + status + "</b><br><br>" +
                    "</html>";

            Object[] content = {
                new JLabel(userInformation),
                new JLabel(scaledImageIcon)
            };

            int option = JOptionPane.showOptionDialog(
                                this,
                                content,
                                "User Information",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.PLAIN_MESSAGE,
                                null,
                                new String[]{"Delete", "Cancel"},
                                "Delete");


            if (option == JOptionPane.YES_OPTION) {

                int confirmOption = JOptionPane.showConfirmDialog(
                                            this,
                                            "Are you sure you want to delete
this user?",
                                            "Confirm Deletion",
                                            JOptionPane.YES_NO_OPTION);

                if (confirmOption == JOptionPane.YES_OPTION) {

                    String userTableQuery =
                            "DELETE FROM [User] " +
                            "WHERE [Email] = ?";

                    String accountTableQuery =
                            "DELETE FROM [Account] " +
                            "WHERE [Email] = ?";

                    try (PreparedStatement deleteUser =
connection.prepareStatement(userTableQuery);
                         PreparedStatement deleteAccount =
connection.prepareStatement(accountTableQuery)) {

                        connection.setAutoCommit(false);

                        deleteUser.setString(1, email);
                        deleteUser.executeUpdate();

                        deleteAccount.setString(1, email);
                        deleteAccount.executeUpdate();

                        connection.commit();

                        JOptionPane.showMessageDialog(
                                            this,
                                            "The user has been successfully
deleted.",
                                            "Deletion Completed",
                                            JOptionPane.INFORMATION_MESSAGE);

                    } catch (SQLException ex) {
                        Logger.getLogger(ManageComplaints.class.getName())
                              .log(Level.SEVERE, " userTableMouseClicked
method ", ex);
                    }
                }
            }

        }
```

```
    }


    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                       "u.[Firstname] + ' ' + u.[Lastname], " +
                       "u.[Email], u.[ContactNo], " +
                       "u.[Street] + ' ' + u.[Barangay] + ' ' + " +
                       "u.[City] + ' ' + u.[Province], " +
                       "a.[Status] " +
                "FROM [User] AS u " +
                "INNER JOIN [Account] AS a " +
                    "ON u.[Email] = a.[Email] " +
                "WHERE " +
                    "(u.[Firstname] + ' ' + u.[Lastname] LIKE ? " +
                    "OR u.[Email] LIKE ? " +
                    "OR u.[ContactNo] LIKE ? " +
                    "OR u.[Street] LIKE ? " +
                    "OR u.[Barangay] LIKE ? " +
                    "OR u.[City] LIKE ? " +
                    "OR u.[Province] LIKE ? " +
                    "OR a.[Status] LIKE ?) " +
                    "AND a.[Role] != 'Admin'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);
            pst.setString(5, searchPattern);
            pst.setString(6, searchPattern);
            pst.setString(7, searchPattern);
            pst.setString(8, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {
                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void updateStatusButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        String selectedStatus = statusComboBox.getSelectedItem().toString();
        int selectedRow = userTable.getSelectedRow();

        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(
                    userTable,
                    "Please select a row.",
                    "No Row Selected",
                    JOptionPane.WARNING_MESSAGE);
        }

        String email = String.valueOf(userTable.getValueAt(selectedRow,
UserColumn.EMAIL.ordinal()));

        String updateQuery =
                "UPDATE [Account] " +
                "SET [Status] = ? " +
                "WHERE [Email] = ?";

        try (PreparedStatement pst = connection.prepareStatement(updateQuery))
{

            pst.setString(1, selectedStatus);
            pst.setString(2, email);
            pst.executeUpdate();

            populateTable();

            statusComboBox.setSelectedIndex(0);

        } catch (SQLException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " updateStatusButtonActionPerformed
method ", ex);
        }
    }

    private void printButtonActionPerformed(java.awt.event.ActionEvent evt) {

        MessageFormat header = new MessageFormat("Users Information Report");

        try {

            userTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(ManageUsers.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
```

```
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JButton printButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JComboBox<String> statusComboBox;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    private javax.swing.JButton updateStatusButton;
    private javax.swing.JTable userTable;
    private javax.swing.JScrollPane userTableScrollPane;
    // End of variables declaration
}
```

Manage Complaint

```
package project.admin;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


/**
 * The MyComplaint class represents a panel for displaying and managing user
complaints.
 * It provides functionalities to populate, filter, and interact with the
complaints table.
 * This panel includes options to view complaint details, withdraw complaints,
and refresh the table.
 *
 * The complaints are fetched from the database based on the currently logged-
in user's email.
 * Users can filter complaints by their status (e.g., New, Assigned,
Resolved).
```

```
 * Double-clicking on a complaint row displays detailed information about the
complaint,
 * including its category, description, date, location, landmark, urgency
level, and status.
 * Additionally, users can withdraw complaints by selecting the respective
option from the pop-up window.
 */
public class ManageComplaints extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;
    String fullName;

    /**
     * Constructs a new MyComplaint panel.
     * Initializes GUI components, sets up table header, customizes
components,
     * establishes database connection, and populates the complaints table.
     */
    public ManageComplaints() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCumponent();
        customizeCellRender();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {
        statusComboBox.putClientProperty(FlatClientProperties.STYLE,
                "padding: 3, 10, 3, 10;");

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");
    }


    /**
     * Populates the complaints table with user-specific complaints retrieved
from the database.
     */
    private void populateTable() {
        tableModel.setRowCount(0);

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint]";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();
            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    rowData[i-1] = rs.getObject(i);
                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, " populateTable method ", ex);
        }
    }


    /**
     * Initializes the table header with a specific font and alignment.
     */
    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

    }
    /**
     * Customizes cell rendering for the complaints table to support multiline
text.
     */
    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());

        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        updateStatusButton = new javax.swing.JButton();
        statusComboBox = new javax.swing.JComboBox<>();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0, 10,
0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setForeground(new java.awt.Color(51, 51, 51));
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 18;
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setForeground(new java.awt.Color(51, 51, 51));
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 18;
        topPanel.add(searchTextField, gridBagConstraints);

        updateStatusButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        updateStatusButton.setForeground(new java.awt.Color(51, 51, 51));
        updateStatusButton.setText("Update");
        updateStatusButton.setToolTipText("Update status of selected
complaint.");
        updateStatusButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateStatusButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        topPanel.add(updateStatusButton, gridBagConstraints);

        statusComboBox.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        statusComboBox.setForeground(new java.awt.Color(51, 51, 51));
        statusComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "<html><font color='b6b6b6'>Status</font></html>", "New", "Under
Review", "Assigned", "Resolved" }));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 8;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        topPanel.add(statusComboBox, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setForeground(new java.awt.Color(51, 51, 51));
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 315);
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(11, Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    /**
     * Refreshes the complaints table by clearing existing data and
repopulating it.
     */
    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        statusComboBox.setSelectedIndex(0);
        tableModel.setRowCount(0);
        populateTable();

    }


    /**
     * Handles the mouse click event on the complaints table.
     * Displays detailed information about the selected complaint and allows
withdrawal.
     */
    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, " complaintTableMouseClicked method ",
ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT " +
                    "c.[Proof], " +
                    "u.[Firstname] + ' ' + u.[Lastname] AS [Fullname]" +
                "FROM [UserComplaint] AS c " +
                "INNER JOIN [User] AS u " +
                    "ON c.[UserID] = u.[UserID] " +
                "WHERE c.[ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");
                this.fullName = rs.getString("Fullname");

            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, " complaintTableMouseClicked method ",
ex);
```

```
        }

        // clicked a row
        final int CLICKED_1_TIMES = 1;
        if (evt.getClickCount() == CLICKED_1_TIMES) {
            statusComboBox.setSelectedItem(status);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        // Clicked a row 2 times
        final int CLICKED_2_TIMES = 2;
        if (evt.getClickCount() == CLICKED_2_TIMES) {
            String complaintDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "Complainant:<b> " + this.fullName + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(complaintDetails),
                new JLabel(scaledImageIcon)
            };

            // here na ko
            int option = JOptionPane.showOptionDialog(
                                this,
                                content,
                                "Complaint Details",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.PLAIN_MESSAGE,
                                null,
                                new String[]{"Delete", "Cancel"},
                                "Delete");


            if (option == JOptionPane.YES_OPTION) {

                int confirmOption = JOptionPane.showConfirmDialog(
                                      this,
                                      "Are you sure you want to delete
this complaint?",
                                      "Confirm Deletion",
                                      JOptionPane.YES_NO_OPTION);

                if (confirmOption == JOptionPane.YES_OPTION) {

                    String updateQuery =
                            "DELETE FROM [UserComplaint] " +
                            "WHERE [ComplaintNo] = ?";

                    try (PreparedStatement pst =
connection.prepareStatement(updateQuery)) {

                        pst.setInt(1, complaintNo);
                        pst.executeUpdate();

                        JOptionPane.showMessageDialog(
                                            this,
                                            "The selected complaint has been
deleted successfully.",
                                            "Deletion Completed",
                                            JOptionPane.INFORMATION_MESSAGE);

                    } catch (SQLException ex) {

Logger.getLogger(ManageComplaints.class.getName()).log(Level.SEVERE, null,
ex);
                    }
                }
            }
        }
    }


    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "[Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ? OR " +
                    "[CreatedDate] LIKE ?";


        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);
            pst.setString(5, searchPattern);

            ResultSet rs = pst.executeQuery();
            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void updateStatusButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        String selectedStatus = statusComboBox.getSelectedItem().toString();
        int selectedRow = complaintTable.getSelectedRow();

        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(
                    tablePanel,
                    "Please select a row.",
                    "No Row Selected",
                    JOptionPane.WARNING_MESSAGE);
        }

        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));

        String updateQuery =
                "UPDATE [UserComplaint] " +
                "SET [Status] = ? " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(updateQuery))
{

            pst.setString(1, selectedStatus);
            pst.setInt(2, complaintNo);
            pst.executeUpdate();

            JOptionPane.showMessageDialog(
                    tablePanel,
                    "Complaint status has been successfully updated.",
                    "Complaint Status Updated",
                    JOptionPane.INFORMATION_MESSAGE);

            populateTable();
            statusComboBox.setSelectedIndex(0);

        } catch (SQLException ex) {
            Logger.getLogger(ManageComplaints.class.getName())
                  .log(Level.SEVERE, " updateStatusButtonActionPerformed
method ", ex);
        }
    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JComboBox<String> statusComboBox;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    private javax.swing.JButton updateStatusButton;
    // End of variables declaration
}
```

Report

Category

Environment Report

```
package project.admin.report.category;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;
```

```
public class EnvironmentReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public EnvironmentReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(EnvironmentReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Category] = 'Environment'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];
                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(EnvironmentReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;
        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Environment Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);
        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(EnvironmentReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);
            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {

Logger.getLogger(EnvironmentReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Category] = 'Environment'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(EnvironmentReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(EnvironmentReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
```

```
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Government Agencies Report

```
package project.admin.report.category;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class GovernmentAgenciesReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public GovernmentAgenciesReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(GovernmentAgenciesReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Category] = 'Government Agencies'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }
        } catch (SQLException ex) {
            Logger.getLogger(GovernmentAgenciesReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();
        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Government Agencies Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();
    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(GovernmentAgenciesReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
```

```
Logger.getLogger(GovernmentAgenciesReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Category] = 'Government Agencies'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(GovernmentAgenciesReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(GovernmentAgenciesReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Infrastructure Report

```
package project.admin.report.category;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class InfrastructureReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public InfrastructureReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(InfrastructureReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }
        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Category] = 'Infrastructure'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(InfrastructureReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));
        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Infrastructure Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);
        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(InfrastructureReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {

Logger.getLogger(InfrastructureReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);
        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Category] = 'Infrastructure'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(InfrastructureReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(InfrastructureReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Public Services Report

```
package project.admin.report.category;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class PublicServices extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public PublicServices() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(PublicServices.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }
    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Category] = 'Public Services'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(PublicServices.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {
        TableColumn complaintNoColumn = 
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn = 
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Public Services Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(PublicServices.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
```

```
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(PublicServices.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Category] = 'Public Services'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];
```

```
                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(PublicServices.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(PublicServices.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Utilities Report

```
package project.admin.report.category;

import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class UtilitiesReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public UtilitiesReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(UtilitiesReport.class.getName())
                    .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Category] = 'Utilities'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(UtilitiesReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }
    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Utilities Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
```

```
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)
                
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(UtilitiesReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
```

```
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {

Logger.getLogger(UtilitiesReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };
            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Category] = 'Utilities'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(UtilitiesReport.class.getName())
```

```
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(UtilitiesReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Status

New

```
package project.admin.report.status;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class NewReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public NewReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(NewReport.class.getName())
                    .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Status] = 'New'";
        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(NewReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
```

```
                        .getColumn(i)
                        .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("New Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
```

```
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(NewReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));
```

```
        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(NewReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }
```

```
    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Status] = 'New'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(NewReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(NewReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Assigned

```
package project.admin.report.status;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatCslientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;
```

```
public class AssignedReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public AssignedReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(AssignedReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Status] = 'Assigned'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];
                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(AssignedReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;
        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Assigned Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);
        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(AssignedReport.class.getName())
                    .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);
            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(AssignedReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Status] = 'Assigned'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(AssignedReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(AssignedReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }
    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Under Reviewed

```
package project.admin.report.status;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class UnderReviewReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public UnderReviewReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(UnderReviewReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Status] = 'Under Review'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(UnderReviewReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));
        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Under Review Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(
            
tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)
                
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)
                
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();
    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(UnderReviewReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo = 
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category = 
String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.CATEGORY.ordinal()));
        String description = 
String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.DATE.ordinal()));
        String location = 
String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.LOCATION.ordinal()));
        String landmark = 
String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel = 
String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow, 
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage)) 
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
```

```
Logger.getLogger(UnderReviewReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Status] = 'Under Review'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(UnderReviewReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(UnderReviewReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Resolved

```
package project.admin.report.status;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class ResolvedReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public ResolvedReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(ResolvedReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [Status] = 'Resolved'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ResolvedReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Resolved Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(ResolvedReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(ResolvedReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);
        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[Status] = 'Resolved'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(ResolvedReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(ResolvedReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Urgency Level

Lows

```
package project.admin.report.urgency_level;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class LowReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public LowReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(LowReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();
    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [UrgencyLevel] = 'Low'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(LowReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }
    private void initTableColumn() {

        TableColumn complaintNoColumn = 
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn = 
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Low Level Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));
```

```
        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(LowReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(LowReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[UrgencyLevel] = 'Low'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();
            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(LowReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(LowReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Medium

```
package project.admin.report.urgency_level;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;s
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class MediumReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public MediumReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(MediumReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }
    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [UrgencyLevel] = 'Medium'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(MediumReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);
        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Medium Level Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
```

```
                  .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                  .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(MediumReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
```

```
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(MediumReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";
            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[UrgencyLevel] = 'Medium'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(MediumReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(MediumReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

High

```
package project.admin.report.urgency_level;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class HighReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public HighReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(HighReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [UrgencyLevel] = 'High'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(HighReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {
            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("High Level Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);
        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(HighReport.class.getName())
                  .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
```

```
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";

        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {
            Logger.getLogger(HighReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);
        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);

        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[UrgencyLevel] = 'High'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(HighReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }
    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);

        } catch (PrinterException ex) {
            Logger.getLogger(HighReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Emergency

```
package project.admin.report.urgency_level;

import project.admin.report.category.*;
import com.formdev.flatlaf.FlatClientProperties;
import java.awt.Font;
import java.awt.Image;
import java.awt.print.PrinterException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import project.authentication.LoginFrame;
import project.concrete_class.ComplaintColumn;
import project.concrete_class.MultilineTableCellRenderer;
import project.database.Database;


public class EmergencyReport extends javax.swing.JPanel {

    private Connection connection;
    private DefaultTableModel tableModel;

    private String email;
    private byte[] proofImageData;

    public EmergencyReport() {
        initComponents();
        initTableHeader();
        initTableColumn();
        customizeCellRender();
        customizeCumponent();

        try {
            this.connection = Database.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(EmergencyReport.class.getName())
                  .log(Level.SEVERE, ex.getMessage(), ex);
        }

        this.tableModel = (DefaultTableModel) complaintTable.getModel();

        this.email = LoginFrame.email;

        populateTable();

    }


    private void customizeCumponent() {

        searchTextField.putClientProperty(FlatClientProperties.STYLE,
                "margin: 3, 10, 3, 10;");

    }

    private void populateTable() {

        String selectQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE [UrgencyLevel] = 'Emergency'";

        try (PreparedStatement complaint =
connection.prepareStatement(selectQuery)) {

            ResultSet rs = complaint.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();
            while (rs.next()) {

                Object[] rowData = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {

                    rowData[i-1] = rs.getObject(i);

                }

                tableModel.addRow(rowData);
            }

        } catch (SQLException ex) {
            Logger.getLogger(EmergencyReport.class.getName())
                  .log(Level.SEVERE, "populateTable method", ex);
        }
    }


    private void initTableHeader() {

        JTableHeader header = complaintTable.getTableHeader();
        header.setFont(new Font("Roboto", Font.BOLD, 15));

        DefaultTableCellRenderer renderer = (DefaultTableCellRenderer)
header.getDefaultRenderer();
        renderer.setHorizontalAlignment(JLabel.LEFT);

    }


    private void initTableColumn() {

        TableColumn complaintNoColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.COMPLAINT_NO.ordinal
());
        complaintNoColumn.setPreferredWidth(5);

        TableColumn dateColumn =
complaintTable.getColumnModel().getColumn(ComplaintColumn.DATE.ordinal());
        dateColumn.setPreferredWidth(30);

    }


    private void customizeCellRender() {

        for (int i = 0; i < complaintTable.getColumnCount(); i++) {

            complaintTable.getColumnModel()
                          .getColumn(i)
                          .setCellRenderer(new MultilineTableCellRenderer());
        }

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        topPanel = new javax.swing.JPanel();
        reportTitle = new javax.swing.JLabel();
        searchLabel = new javax.swing.JLabel();
        searchTextField = new javax.swing.JTextField();
        printReportButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        tablePanel = new javax.swing.JPanel();
        complaintTableScrollPane = new javax.swing.JScrollPane();
        complaintTable = new javax.swing.JTable();

        setPreferredSize(new java.awt.Dimension(1056, 750));

        java.awt.GridBagLayout topPanelLayout = new java.awt.GridBagLayout();
        topPanelLayout.columnWidths = new int[] {0, 10, 0, 10, 0, 10, 0};
        topPanelLayout.rowHeights = new int[] {0, 5, 0, 5, 0, 5, 0, 5, 0, 5,
0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0, 5, 0};
        topPanel.setLayout(topPanelLayout);

        reportTitle.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
        reportTitle.setText("Emergency Level Complaint Report");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = 7;
        topPanel.add(reportTitle, gridBagConstraints);

        searchLabel.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchLabel.setText("Search");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.insets = new java.awt.Insets(0, 20, 0, 0);
        topPanel.add(searchLabel, gridBagConstraints);

        searchTextField.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        searchTextField.setPreferredSize(new java.awt.Dimension(300, 40));
        searchTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                searchTextFieldKeyReleased(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 22;
        topPanel.add(searchTextField, gridBagConstraints);

        printReportButton.setFont(new java.awt.Font("Roboto", 0, 15)); //
NOI18N
        printReportButton.setText("Print");
        printReportButton.setToolTipText("Print table content");
        printReportButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                printReportButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 20);
        topPanel.add(printReportButton, gridBagConstraints);

        refreshButton.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        refreshButton.setText("Refresh");
        refreshButton.setToolTipText("Refresh table");
        refreshButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        refreshButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshButtonActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 22;
        gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 0.1;
        topPanel.add(refreshButton, gridBagConstraints);

        tablePanel.setBackground(new java.awt.Color(255, 255, 255));


complaintTableScrollPane.setBorder(javax.swing.BorderFactory.createEmptyBorder
(1, 1, 1, 1));

        complaintTable.setAutoCreateRowSorter(true);
        complaintTable.setFont(new java.awt.Font("Roboto", 0, 15)); // NOI18N
        complaintTable.setForeground(new java.awt.Color(51, 51, 51));
        complaintTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "No", "Category", "Description", "Date", "Location",
"Landmark", "Urgency Level", "Status"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        complaintTable.setFillsViewportHeight(true);
        complaintTable.setRowHeight(40);
        complaintTable.setShowHorizontalLines(true);
        complaintTable.getTableHeader().setReorderingAllowed(false);
        complaintTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                complaintTableMouseClicked(evt);
            }
        });
        complaintTableScrollPane.setViewportView(complaintTable);

        javax.swing.GroupLayout tablePanelLayout = new
javax.swing.GroupLayout(tablePanel);
        tablePanel.setLayout(tablePanelLayout);
        tablePanelLayout.setHorizontalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(tablePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(complaintTableScrollPane)
                .addContainerGap())
        );
        tablePanelLayout.setVerticalGroup(

tablePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
tablePanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(complaintTableScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 604,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G, false)
                    .addComponent(topPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 1029, Short.MAX_VALUE)
                    .addComponent(tablePanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(0, 15, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(topPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(tablePanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 580,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20))
        );
    }// </editor-fold>


    private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt)
{

        searchTextField.setText("");
        tableModel.setRowCount(0);
        populateTable();

    }


    private void complaintTableMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = 0;

        try {
            selectedRow = complaintTable.getSelectedRow();
        } catch (IndexOutOfBoundsException ex) {
            Logger.getLogger(EmergencyReport.class.getName())
                    .log(Level.SEVERE, "complaintTableMouseClicked", ex);
        }

        // Getting the value of each column at the selected row.
        int complaintNo =
Integer.parseInt(String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.COMPLAINT_NO.ordinal())));
        String category =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.CATEGORY.ordinal()));
        String description =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DESCRIPTION.ordinal()));
        String date = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.DATE.ordinal()));
        String location =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LOCATION.ordinal()));
        String landmark =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.LANDMARK.ordinal()));
        String urgencyLevel =
String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.URGENCY_LEVEL.ordinal()));
        String status = String.valueOf(complaintTable.getValueAt(selectedRow,
ComplaintColumn.STATUS.ordinal()));

        // Retrieving the associated image of the selected row in the table.
        String selectImage =
                "SELECT [Proof] " +
                "FROM [UserComplaint] " +
                "WHERE [ComplaintNo] = ?";
        try (PreparedStatement pst = connection.prepareStatement(selectImage))
{

            pst.setInt(1, complaintNo);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                this.proofImageData = rs.getBytes("Proof");

            }

        } catch (SQLException ex) {

Logger.getLogger(EmergencyReport.class.getName()).log(Level.SEVERE,
"complaintTableMouseClicked", ex);
        }

        // Resizing the image
        ImageIcon imageIcon = new ImageIcon(proofImageData);
        Image scaledImage = imageIcon.getImage().getScaledInstance(350, 300,
Image.SCALE_SMOOTH);
        ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

        final int CLICKED_2_TIMES = 2;

        if (evt.getClickCount() == CLICKED_2_TIMES) {

            String popUpDetails =
                    "<html>" +
                    "Category:<b> " + category + "</b><br>" +
                    "Description:<b> " + description + "</b><br>" +
                    "Created Date:<b> " + date + "</b><br>" +
                    "Location:<b> " + location + "</b><br>" +
                    "Landmark:<b> " + landmark + "</b><br>" +
                    "Urgency Level:<b> " + urgencyLevel + "</b><br>" +
                    "Status:<b> " + status + "</b><br>" + "<br>" +
                    "</html>";

            Object[] content = {
                new JLabel(popUpDetails),
                new JLabel(scaledImageIcon)
            };

            // get the selected option
            JOptionPane.showMessageDialog(
                    complaintTable,
                    content,
                    "Complaint Details",
                    JOptionPane.PLAIN_MESSAGE);

        }

    }

    private void searchTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

        tableModel.setRowCount(0);
        String searchInput = searchTextField.getText();

        String searchQuery =
                "SELECT " +
                    "[ComplaintNo], [Category], " +
                    "[Description], [CreatedDate], " +
                    "[Location], [Landmark], " +
                    "[UrgencyLevel], [Status] " +
                "FROM [UserComplaint] " +
                "WHERE " +
                    "([Category] LIKE ? OR " +
                    "[Location] LIKE ? OR " +
                    "[UrgencyLevel] LIKE ? OR " +
                    "[Status] LIKE ?) AND " +
                    "[UrgencyLevel] = 'Emergency'";

        try (PreparedStatement pst = connection.prepareStatement(searchQuery))
{

            String searchPattern = "%" + searchInput + "%";

            pst.setString(1, searchPattern);
            pst.setString(2, searchPattern);
            pst.setString(3, searchPattern);
            pst.setString(4, searchPattern);

            ResultSet rs = pst.executeQuery();

            int columnCount = rs.getMetaData().getColumnCount();

            while (rs.next()) {

                Object[] row = new Object[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }

                tableModel.addRow(row);
            }

        } catch (SQLException ex) {
            Logger.getLogger(EmergencyReport.class.getName())
                  .log(Level.SEVERE, " searchTextFieldKeyReleased method ",
ex);
        }

    }

    private void printReportButtonActionPerformed(java.awt.event.ActionEvent
evt) {

        MessageFormat header = new MessageFormat(reportTitle.getText());

        try {

            complaintTable.print(JTable.PrintMode.FIT_WIDTH, header, null);
        } catch (PrinterException ex) {
            Logger.getLogger(EmergencyReport.class.getName())
                  .log(Level.SEVERE, " printReportButtonActionPerformed method
", ex);
        }

    }


    // Variables declaration - do not modify
    private javax.swing.JTable complaintTable;
    private javax.swing.JScrollPane complaintTableScrollPane;
    private javax.swing.JButton printReportButton;
    private javax.swing.JButton refreshButton;
    private javax.swing.JLabel reportTitle;
    private javax.swing.JLabel searchLabel;
    private javax.swing.JTextField searchTextField;
    private javax.swing.JPanel tablePanel;
    private javax.swing.JPanel topPanel;
    // End of variables declaration
}
```

Concrete Class

Account Status

```
package project.concrete_class;

public enum AccountStatus {
    ACTIVE("Active"),
    SUSPENDED("Under Review");

    private final String description;

    private AccountStatus(String description) {
        this.description = description;
    }

    public String getDescription() {
        return this.description;
    }

}
```

Complaint Column

```
package project.concrete_class;

public enum ComplaintColumn {
    COMPLAINT_NO,
    CATEGORY,
    DESCRIPTION,
    DATE,
    LOCATION,
    LANDMARK,
    URGENCY_LEVEL,
    STATUS
}
```

Complaint Status

```
package project.concrete_class;

public enum ComplaintStatus {
    NEW("New"),
    UNDER_REVIEW("Under Review"),
    ASSIGNED("Assigned"),
    RESOLVED("Resolved"),
    ON_HOLD("On Hold"),
    WITHDRAWN("Withdrawn");

    private final String description;

    private ComplaintStatus(String description) {
        this.description = description;
    }

    public String getDescription() {
        return this.description;
    }

}
```

Image Filter

```
package project.concrete_class;

import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;

/* ImageFilter.java is used by FileChooserDemo2.java. */
public class ImageFilter extends FileFilter {

    //Accept all directories and all gif, jpg, tiff, or png files.
    @Override
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = Utils.getExtension(f);
        if (extension != null) {
            if (extension.equals(Utils.tiff) ||
                extension.equals(Utils.tif) ||
                extension.equals(Utils.gif) ||
                extension.equals(Utils.jpeg) ||
                extension.equals(Utils.jpg) ||
                extension.equals(Utils.png)) {
                    return true;
            } else {
                return false;
            }
        }

        return false;
    }
    //The description of this filter
    @Override
    public String getDescription() {
        return "Just Images";
    }
}
```

Multiline Table Cell Renderer

```
package project.concrete_class;

import java.awt.Color;
import java.awt.Component;
import java.awt.Insets;
import javax.swing.BorderFactory;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.border.Border;
import javax.swing.table.TableCellRenderer;

/**
 *
 * This class is used to modify
 * table cell so that as cell resize
 * the content shrink line by line
 */

public class MultilineTableCellRenderer extends JTextArea
                                        implements TableCellRenderer{

    public MultilineTableCellRenderer() {
        setWrapStyleWord(true);
        setLineWrap(true);
        setOpaque(true);
        setRows(2);
        Border bottomColorBorder = BorderFactory.createMatteBorder(0, 0, 1, 0,
Color.decode("#ebebeb"));
        setBorder(bottomColorBorder);

    }

    @Override
    public Component getTableCellRendererComponent(
                            JTable table, Object value,
                            boolean isSelected, boolean hasFocus,
                            int row, int column) {

        if (isSelected) {
            setForeground(table.getSelectionForeground());
            setBackground(table.getSelectionBackground());
        } else {
            setForeground(table.getForeground());
            setBackground(table.getBackground());
        }

        setFont(table.getFont());
        setText((value == null) ? "" : value.toString());
        setSize(table.getColumnModel().getColumn(column).getWidth(),
                getPreferredSize().height);
        if (table.getRowHeight(row) != getPreferredSize().height) {
            table.setRowHeight(row, getPreferredSize().height);
        }


        return this;

    }

}
```

Use Column

```
package project.concrete_class;

public enum UserColumn {

    NAME,
    EMAIL,
    CONTACT_NO,
    ADDRESS,
    STATUS

}
```

Utils

```
package project.concrete_class;

import java.io.File;
import javax.swing.ImageIcon;

/* Utils.java is used by FileChooserDemo2.java. */
public class Utils {
    public final static String jpeg = "jpeg";
    public final static String jpg = "jpg";
    public final static String gif = "gif";
    public final static String tiff = "tiff";
    public final static String tif = "tif";
    public final static String png = "png";

    /*
     * Get the extension of a file.
     */
    public static String getExtension(File file) {

        String fileName = file.getName();
        int dotIndex = fileName.lastIndexOf('.');

        if (dotIndex >= 0 && dotIndex < fileName.length() - 1) {
            return fileName.substring(dotIndex + 1).toLowerCase();
        } else {
            return null; // No extension found
        }
    }
    /** Returns an ImageIcon, or null if the path was invalid. */
    public static ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = Utils.class.getResource(path);
        if (imgURL != null) {
            return new ImageIcon(imgURL);
        } else {
            System.err.println("Couldn't find file: " + path);
            return null;
        }
    }
}
```

Database

```
package project.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Database {

    private static final String DB_URL = "jdbc:sqlserver://localhost:1433;"
            + "databaseName=CitizenComplaintManagementSystem;"
            + "encrypt=true;"
            + "trustServerCertificate=true;";

    private static final String DB_USERNAME = "romeo";
    private static final String DB_PASSWORD = "211708";
    private static Connection connection;

    public static Connection getConnection() throws SQLException {

        return connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);

    }

}
```