University Of Engineering And Technology Lahore

(New Campus)

Legal Link

Software Requirements Specification (SRS)

Submitted to:

Mr. Zeeshan Ramzan

Submitted by:

Messam Naqvi (2021-CS-622)

Muneeb Abbas (2021-CS-641)

Department:

Computer Science

Declaration

I declare that the work contained in this thesis is my own, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Members Signature:

Date: 13/12/2023

Acknowledgments

In the name of Allah, the most beneficent and the most merciful. Alhamdulillah we praise to Allah for the strength and his blessings in completing this project. Special appreciation goes to my teacher Sir Zeeshan Ramzan for the supervision and constant support. Their vibrant help of useful comments and suggestions during the tentative and proposal works have contributed to the success of this project and research. Our acknowledgment also goes to myself.Thanks to all my project members who directly contributed to this project research. Thank you very much.

Contents

List of figures

List of Tables

Abstract

Legal Link is innovative desktop app built using C# that is intended to improve and expedite the work of legal professionals. A full range of capabilities, such as case administration, client tracking, document organization, and billing connection, are available with this all-inclusive lawyer management system. Legal Link's user-friendly interface and data management features enable legal professionals to effectively handle intricate legal procedures, enhance client relationships, and boost overall productivity. This software, which is specifically designed to fulfil the requirements of legal professionals, is a major development in the management of legal practices. Legal Link raises the bar in the legal sector for productivity, accuracy, and customer happiness by smoothly incorporating technology into legal workflows.

Legal Link's impact ripples beyond individual case management. Its centralized data platform fosters seamless collaboration across teams, empowering legal professionals to leverage collective expertise and streamline complex investigations. Imagine legal researchers quickly sifting through vast document databases, while litigators craft compelling arguments based on real-time data visualizations. This integrated approach facilitates more informed decision-making, ultimately yielding superior legal outcomes for clients.

This unified intelligence spills over to client engagement, transcending traditional communication channels. Legal Link's built-in chat and video conferencing features bridge geographical distances, promoting real-time interactions that build trust and transparency. Clients become active participants in the legal process, empowered to ask questions, receive updates, and feel secure in their attorney's expertise. This interactive approach not only fosters deeper relationships but also empowers clients to make informed decisions, feel valued, and advocate for their own interests effectively. In a digital age where client retention is paramount, Legal Link's focus on engagement translates to a competitive edge for legal practices in a crowded marketplace.

Furthermore, Legal Link's focus on client engagement transcends traditional communication channels. Its in-built chat and video conferencing features promote real-time interactions, bolstering trust and transparency. Clients actively participate in the legal process, feeling empowered and informed every step of the way. This enhanced client experience translates to stronger relationships, repeat business, and a competitive edge in a dynamic legal landscape.

Chapter 1

Introduction

1.1 Introduction

Introducing "*Legal Link*" – an inventive C# desktop application designed to redefine the landscape of legal management. In a legal landscape that demands precision and streamlined client handling, Legal Link emerges as a modern solution poised to revolutionize lawyer management. Recognizing the evolving nature of the legal profession, this application addresses the challenges faced by legal practitioners in an increasingly intricate and data-driven environment.

Unlike traditional methods relying on manual record-keeping, Legal Link offers a comprehensive, intuitive platform tailored to the nuanced requirements of legal professionals. Existing software solutions often fall short, lacking the depth and integration needed for holistic legal practice management. Legal Link stands out by providing a complete solution, leveraging C# technology for an intuitive interface and unparalleled data management capabilities.

From robust case tracking to dynamic document management, Legal Link sets a new standard for efficiency and accuracy in legal practice, offering a seamless flow of information and tasks. Distinguished by its cutting-edge features, Legal Link introduces a groundbreaking approach to legal management. Our application transcends the conventional boundaries of legal software, incorporating advanced analytics, artificial intelligence, and machine learning algorithms. With a commitment to staying ahead of industry trends, Legal Link provides predictive insights, empowering legal practitioners to anticipate challenges, optimize workflows, and make informed decisions. This forward-looking technology not only enhances efficiency but also sets a precedent for the future of legal practice. Seamlessly integrating with your daily routines, Legal Link becomes a strategic partner in your legal endeavors, offering a sophisticated yet user-friendly platform that adapts to the evolving demands of the legal landscape.

In the ever-evolving legal landscape, efficiency and streamlined client handling are paramount. Legal Link, a cutting-edge C# desktop application, redefines the very concept of lawyer management. Recognizing the challenges faced by practitioners in today's complex and data-driven environment, Legal Link offers a comprehensive and intuitive platform tailored to their specific needs. Unlike traditional methods and fragmented software solutions, Legal Link provides a complete solution, built upon the foundation of C# technology and boasting an intuitive interface and unparalleled data

management capabilities. It goes beyond mere case tracking and document management, setting a new standard for efficiency and accuracy in legal practice. Legal Link empowers you to manage cases with unparalleled precision, cultivate stronger client relationships, streamline document organization, ensure accurate billing and financial management, and navigate the application with ease through a user-friendly interface that can be customized to your specific needs. Embrace the future of legal practice and achieve greater success than ever before with Legal Link.

1.2 Objectives

- ✓ To facilitate more efficient case management processes.
- ✓ To enhance client tracking and cultivate stronger relationships.
- ✓ To optimize document organization and accessibility for seamless workflow.
- ✓ To integrate precise billing and financial tracking capabilities.
- ✓ To develop a secure and efficient SQL database for managing case details, client info, documents and billing.
- ✓ Offer a calendar and scheduling system for managing appointments and deadline

1.3 Problem Statement

How can the legal profession effectively address the inefficiencies of manual record-keeping and the limitations of existing software solutions in managing cases, tracking clients, organizing documents, and providing comprehensive billing functionality for accurate and efficient financial management?

1.4 Assumptions and Constraints

1.4.1 Assumptions

1. All necessary internal and external resources will be available for the completion of the project.

2. Project team members will have access to the required resources, including equipment and software, to fulfill their assigned tasks promptly.

3. Personnel expenses are expected to remain consistent throughout the project lifecycle.

4. Other material and asset expenses will remain stable throughout the duration of the project.

5. The overall cost of routine activities is anticipated to remain unchanged.

6. All implemented enhancements will be fully operational throughout the project cycle.

7. The scope of the project will remain unchanged throughout its entire duration.

1.4.2 Constraints

1. Time for task completion or register cases may vary due to factors such as workload and complexity.

2. Device compatibility issues may arise, impacting the app's performance on certain devices.

1.5 Scope

The Scope of this project extends to:

- ✓ Development of a comprehensive desktop application using C# technology.
- ✓ Inclusion of features for case management, client tracking, document organization, billing integration, and case scheduling.
- ✓ Integration with existing legal databases and software, ensuring effiecient data transfer.
- ✓ Compatibility with major operating systems (Windows, MacOS, Linux).
- ✓ Implementation of a user-friendly interface with customizable settings to cater to individual preferences.

Chapter 2

Requirement Analysis

2.1 Literature Review

The legal profession, traditionally known for its reliance on paper-based systems and a slow pace of change, is undergoing a significant transformation. Technological advancements are driving this shift, with legal practice management software emerging as a key driver of efficiency and productivity. Legal Link, a desktop application built using C#, stands at the forefront of this evolution, offering a comprehensive suite of features designed to empower legal professionals.

Legal Link streamlines administrative tasks through automation and centralized data management. Case creation and management, document organization, and billing integration significantly reduce the time spent on routine tasks, allowing lawyers to focus on core legal work and client service. This focus on efficiency is supported by research from McKinsey & Company, which found that legal teams using technology can achieve up to 30% productivity gains[1].

Legal Link facilitates stronger client relationships through enhanced communication and organization. The platform provides a centralized platform for storing client information, documents, and communication history, allowing legal professionals to access and share information readily. This fosters transparency and trust, ultimately leading to improved client satisfaction. A 2022 study by Clio found that 83% of clients are more likely to continue working with a lawyer who uses technology to improve communication and collaboration[2].

Legal Link offers seamless integration with billing systems, enabling accurate and efficient time tracking and invoicing. This ensures timely payments and reduces administrative burden for both lawyers and clients. The importance of integrated billing is further emphasized by a 2021 study by Thomson Reuters, which found that law firms using integrated billing software collect invoices 25% faster than those using manual methods[3]. Legal Link leverages a secure and efficient SQL database for storing case details, client information, documents, and billing data. This ensures data integrity and accessibility, safeguarding sensitive information and complying with data privacy regulations.

American Bar Association emphasizes the ethical obligation of lawyers to protect client confidentiality and data integrity[4].Legal Link prioritizes user experience with its intuitive and user-friendly interface. The interface is customizable, allowing users to tailor settings and workflows to their individual preferences. This user-centric approach is crucial for maximizing adoption and ensuring that the software effectively meets the needs of its users. A 2020 study by LegalTech found that 75% of lawyers are more likely to adopt a legal software solution if it offers a user-friendly interface and customization options[5].

Legal Link represents a significant leap forward in legal practice management software. Its comprehensive features, coupled with its user-friendly interface and focus on data security, position it as a valuable tool for legal professionals seeking to improve efficiency, client service, and overall practice success. As the legal profession continues to embrace technology, Legal Link is poised to play a vital role in shaping the future of legal practice.

2.2 Requirement Elicitation

2.2.1 Stakeholders

The key stakeholders for Legal Link include:

1. *Lawyers:* The primary users of the application, who need features that support their daily legal work, such as case management, document organization, time tracking, and billing.
2. *Legal assistants:* They provide administrative support to lawyers and require functionalities that facilitate their tasks, such as scheduling appointments, managing client communication, and handling document preparation.
3. *Legal administrators:* They oversee the overall operations of the legal practice and need features that provide insights into practice performance, financial management, and resource allocation.
4. *IT personnel:* They are responsible for the technical infrastructure and need information about system requirements, security protocols, and data integration.

2.2.2 Elicitation Techniques

Several techniques can be used to elicit requirements from stakeholders:

1. *Interviews*: Individual or group interviews can be conducted to gather detailed information about stakeholders' needs, pain points, and desired features.
2. *Surveys*: Online or paper surveys can be used to collect quantitative data from a larger group of stakeholders.
3. *Workshops*: Interactive workshops can be facilitated to brainstorm ideas, prioritize features, and reach consensus on key requirements.
4. *Document analysis:* Existing documents, such as job descriptions, user manuals, and industry reports, can be reviewed to identify potential requirements.

2.2.3 Requirements Gathering and Prioritization

The gathered requirements need to be categorized, analyzed, and prioritized based on stakeholder importance, feasibility, and impact. The following are some key aspects to consider during prioritization:

1. *Essential features:* These features are critical for the core functionality of the application and must be included in the initial release.
2. *Desirable features:* These features enhance the user experience and add value to the application but can be implemented in later versions.
3. *Nice-to-have features*: These features are not essential for core functionality but can be considered for future development if resources allow.

2.2.4 Documentation and Management

Once requirements are gathered and prioritized, they need to be documented in a clear and concise manner. A requirements document should include the following information:

Functional requirements: What the application should do.

Non-functional requirements: How the application should perform.

User interface requirements: How the application should look and feel.

Data requirements: What data the application will need to store and manage.

Security requirements: How the application will protect sensitive data.

2.3 Functional Requirements

Index	Priority	Requirements
FR-01-01	1	The system shall allow the lawyers to log in.
FR-01-02	1	The system shall allow the lawyers to log in.
FR-01-03	1	The system shall allow the client to sign up
FR-01-04	1	The system shall allow the lawyer to sign up

Table 1: Sign Up

Table 2:Appointments

Index	Priority	Requirements
FR-07-01	1	The system shall offer a calendar and scheduling system, enabling lawyers to manage appointments and deadlines efficiently.

Index	Priority	Requirements
FR-03-01	1	The system shall allow the lawyers to create new cases, providing essential case details.
FR-03-02	1	The system shall provide a 'Save' button on the case creation form to save the entered case details.
FR-03-03	2	The system shall provide a option to cancel the case creation process using a 'Cancel' button.
FR-04-04	3	This system shall allow clients to request a lawyer to take their case by providing essential case details through a dedicated form.

Table 3: Case Creation

Index	Priority	Requirements
FR-04-01	1	The system shall allow the lawyers to view a list of their clients, including relevant details such as client names, case associations, and contact information.
FR-04-02	1	The system shall provide a option to remove and Update the client

Table 4: Clients

Index	Priority	Requirements
FR-05-01	1	The system shall optimize document organization, ensuring accessibility for a streamlined workflow.

Table 5: Document Organization

Index	Priority	Requirements
FR-06-01	1	The system shall allow lawyer to create invoices with service details and amounts.
FR-06-02	1	The system shall allow clients to make payments for bills or fees through the platform.

Table 6: Billing

2.4 Non-Functional Requirements

Index	Priority	Requirements
NFR-01-01	1	The system must provide a user-friendly interface.
NFR-01-02	1	The system must not require much knowledge about it to use it.
NFR-01-03	1	The system must provide visible and normal fonts to the user.
NFR-01-04	2	The system must have black and white color.

Table 7: Usablility

Index	Priority	Requirements
NFR-02-01	1	The system must provide 24/7 availability without any errors.
NFR-02-02	2	The backup of the records must be continuously maintained.

Table 8:Reliability

Index	Priority	Requirements
NFR-03-01	2	The system must require space according to device.

Table 9:Scalability

Index	Priority	Requirements
NFR-04-01	2	The system must take 0.5 seconds to respond a request.
NFR-04-02	2	The system must take 2 seconds to start app.

Index	Priority	Requirements
NFR-05-01	2	The system must require a update for its maintainability.
NFR-05-02	2	The system must be accessed any time anywhere.

Table 10:Supportability

Index	Priority	Requirements
NFR-06-01	1	The system must only access to authorized user only.
NFR-06-02	3	The system must have a secure billing method.

Table 11:Security

2.5 Use case Elaboration

Use Case ID:	**01**
Title:	Log in.
Goal:	As a lawyer or client, I want to log in to the Legal Link system.
Trigger:	When a user wants to access the Legal Link platform.
Primary Actor:	Lawyer or Client
Preconditions:	• The user shall have valid credentials. • Stable internet connection. • Sign up shall have done.
Postconditions:	• The user successfully logs in to the Legal Link system.
Basic Flow:	1. User clicks on the Legal Link login button. 2. User enters their username and password. 3. User submits the login form. 4. The system validates the user's credentials. 5. If credentials are valid, the system logs the user into the Legal Link platform.
Exceptions:	• Internet does not be available. • User does not have valid username.
Alternate Flow:	[Not Available]
Quality:	• User shall login in 1 sec.

Use Case ID:	**02**
Title:	Create New Case
Goal:	As a lawyer, I want to initiate the creation of a new legal case within the Legal Link system.
Trigger:	When a lawyer decides to add a new case.
Primary Actor:	Lawyer
Preconditions:	• The lawyer shall be logged into the Legal Link system. • The lawyer must have the necessary information related to the new case. • Stable internet connection.
Postconditions:	• A new legal case is successfully created and added to the system.
Basic Flow:	1 Lawyer navigates to the "New Cases" button of the Legal Link platform. 2 Lawyer fills out the new case form with essential details (e.g., title, type). 3 Lawyer clicks on the "Register" button to save the entered case details. 4 The system validates the information(i.e Client Name) and creates a unique case ID. 5 The system adds the new case to the lawyer's case list.
Exceptions:	• Internet does not be available. • lawyer cancels the case creation process.
Alternate Flow:	[Not Available]
Quality:	• The system shall complete the new case creation process efficiently to ensure a smooth workflow for the lawyer.

Use Case ID:	**03**
Title:	Client List.
Goal:	As a lawyer, I want to access the list of clients associated with my legal practice within the Legal Link system.
Trigger:	When a lawyer needs to view their list of clients.
Primary Actor:	Lawyer
Preconditions:	• The lawyer shall be logged into the Legal Link system.
Postconditions:	• The lawyer successfully viewed the client list.
Basic Flow:	1 Lawyer clicks on the "Client List" section of the Legal Link platform. 2 The system displays a list of clients associated with the lawyer's legal practice. 3 Lawyer can filter and sort the client list based on specific criteria (e.g., case type,priority). 4 Lawyer clicks on a client entry to view detailed information.
Exceptions:	• Internet does not be available.
Alternate Flow:	[Not Available]
Quality:	• The system shall provide a responsive and user-friendly interface, allowing the lawyer to efficiently manage their client list

Use Case ID:	**04**
Title:	Manage Documents.
Goal:	As a lawyer, I want to upload and access legal documents associated with my cases within the Legal Link system.
Trigger:	When a lawyer needs to manage documents related to their legal practice.
Primary Actor:	Lawyer
Preconditions:	• The lawyer shall be logged into the Legal Link system. • The lawyer must have legal documents ready for upload.
Postconditions:	• The lawyer successfully manages legal documents within the Legal Link system.
Basic Flow:	1 Lawyer clicks on the "Documents" section of the Legal Link platform. 2 The system displays a list of existing legal documents associated with the lawyer's cases. 3 Lawyer can upload new documents by selecting the appropriate content. 4 Lawyer can view any document from the list.
Exceptions:	• Internet does not be available. • Document does not upload.
Alternate Flow:	[Not Available]
Quality:	• The system shall provide a secure and efficient document management system, ensuring quick access and categorization for the lawyer.

Use Case ID:	**05**
Title:	Manage Billings
Goal:	As a user (lawyer or client), I want to access, review, and manage billings for legal services within the Legal Link system.
Trigger:	When a user needs to view or manage billing information.
Primary Actor:	User (Lawyer or Client)
Preconditions:	• The user shall be logged into the Legal Link system. • Billings for legal services must be generated and available.
Postconditions:	• The user successfully accesses and manages billing information within the Legal Link system.
Basic Flow:	1 User clicks to the "Billings" section of the Legal Link platform. 2 The system displays a list of billings for legal services, categorized by client or case. 3 User can view detailed information within each billing entry, including service descriptions and total amounts. 4 User can print bill.
Exceptions:	• Internet does not be available. • Amount does not retrieved.
Alternate Flow:	[Not Available]
Quality:	• The system shall provide a transparent and efficient billing system, ensuring accurate invoicing and convenient payment processing for both lawyers and clients.

Use Case ID:	**06**
Title:	Appointments
Goal:	As a lawyer, I want to schedule, view, and manage appointments with clients within the Legal Link system.
Trigger:	When a lawyer needs to organize and coordinate appointments.
Primary Actor:	Lawyer
Preconditions:	• The lawyer shall be logged into the Legal Link system. • Clients must be available for scheduling appointments.
Postconditions:	• The lawyer successfully schedules, views,appointments within the Legal Link system.
Basic Flow:	1 Lawyer navigates to the "Appointments" section of the Legal Link platform. 2 The system displays a calendar or list of upcoming appointments for the lawyer. 3 Lawyer views and manages existing appointments, including details like date, time, and purpose. 4 Lawyer can cancel appointments as needed.
Exceptions:	• Internet does not be available. • Client does not available.
Alternate Flow:	[Not Available]
Quality:	• The system shall provide a user-friendly and efficient appointment management system, allowing lawyers to easily schedule, view, and edit appointments within the Legal Link platform.

2.2 Use Case Diagram

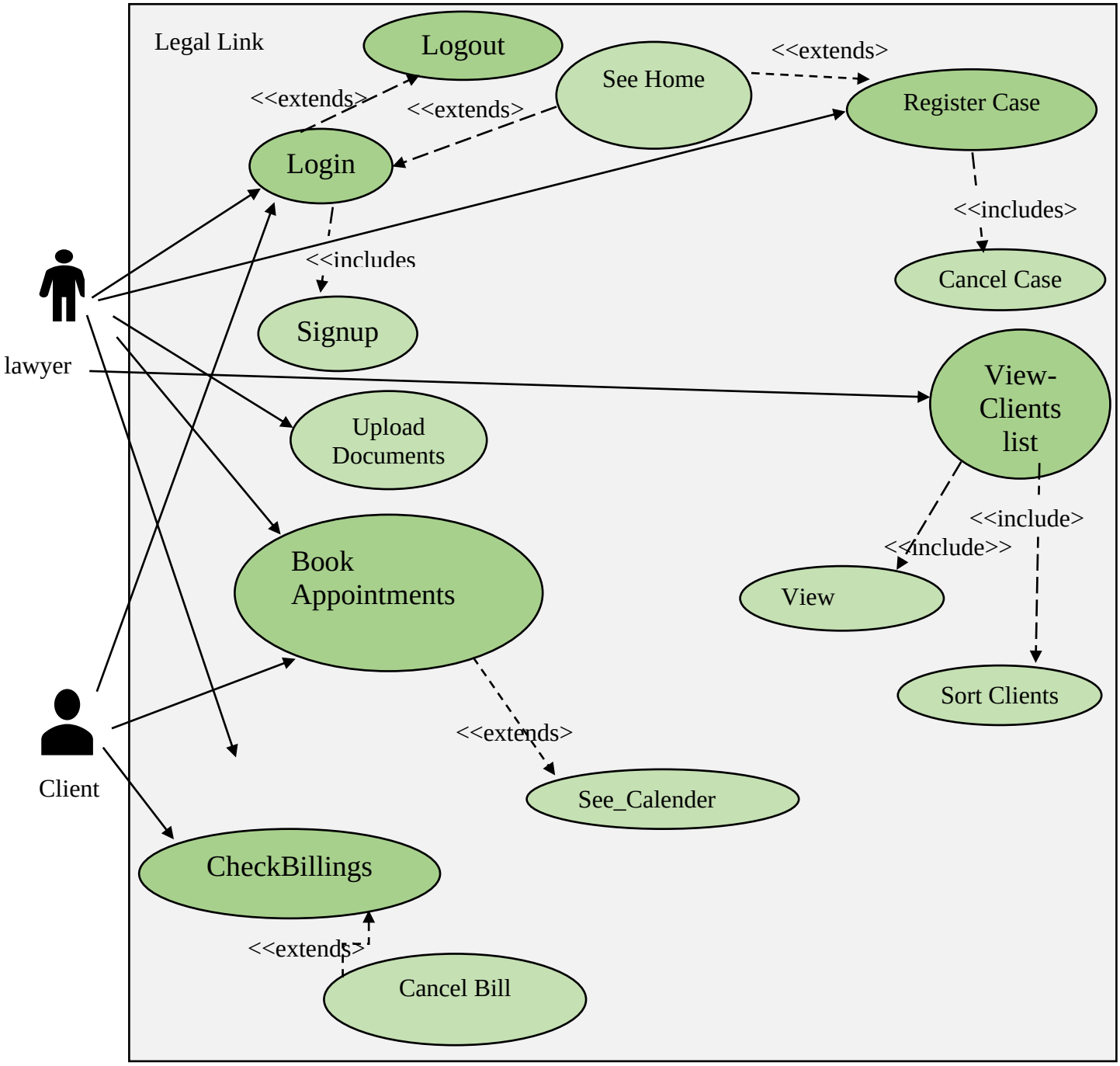


Figure 1: Usecase Diagram

Chapter 3

System Design Analysis

3.1 Component Diagram

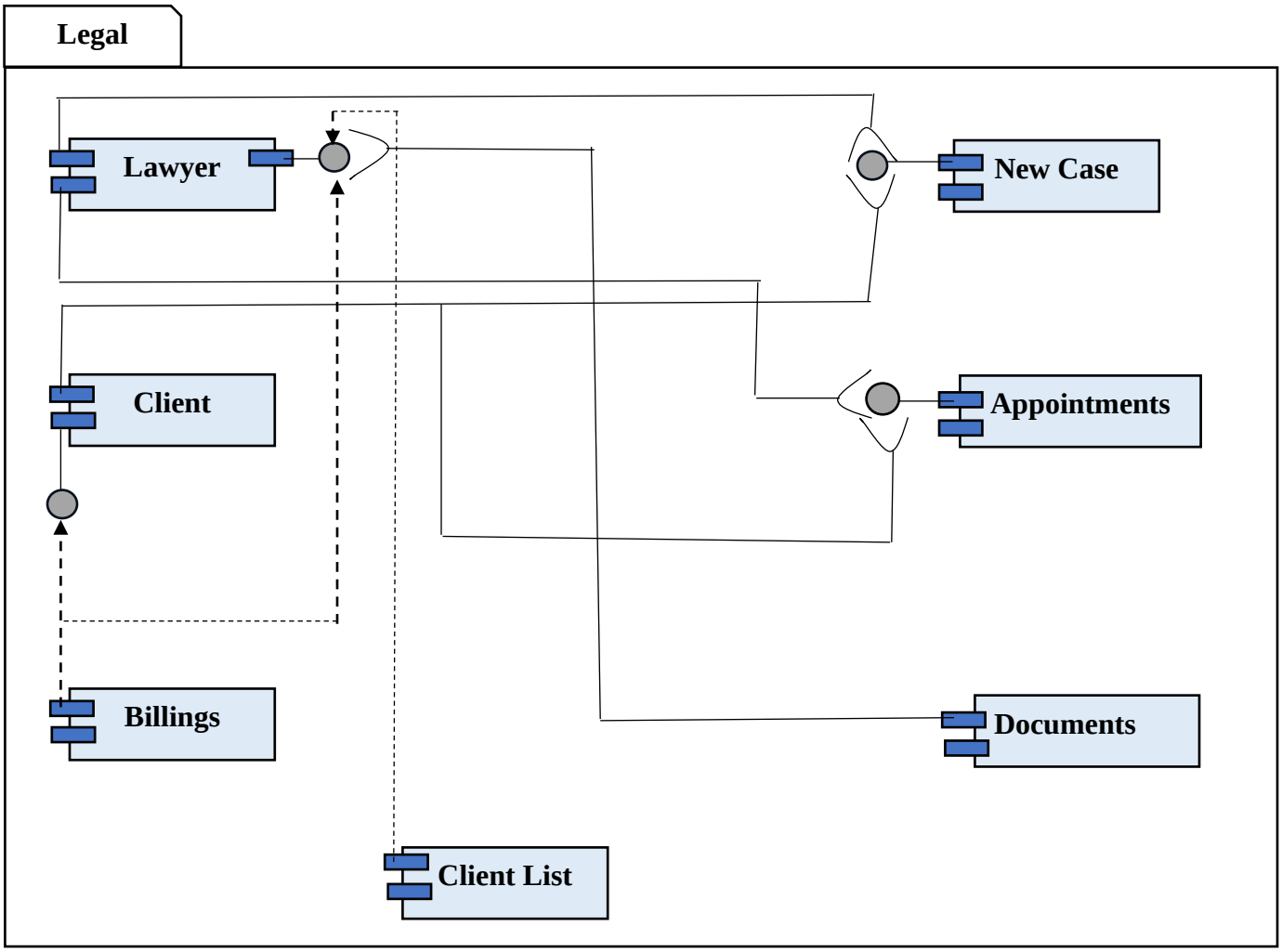


Figure 2: Component Diagram

3.2 Class Diagram

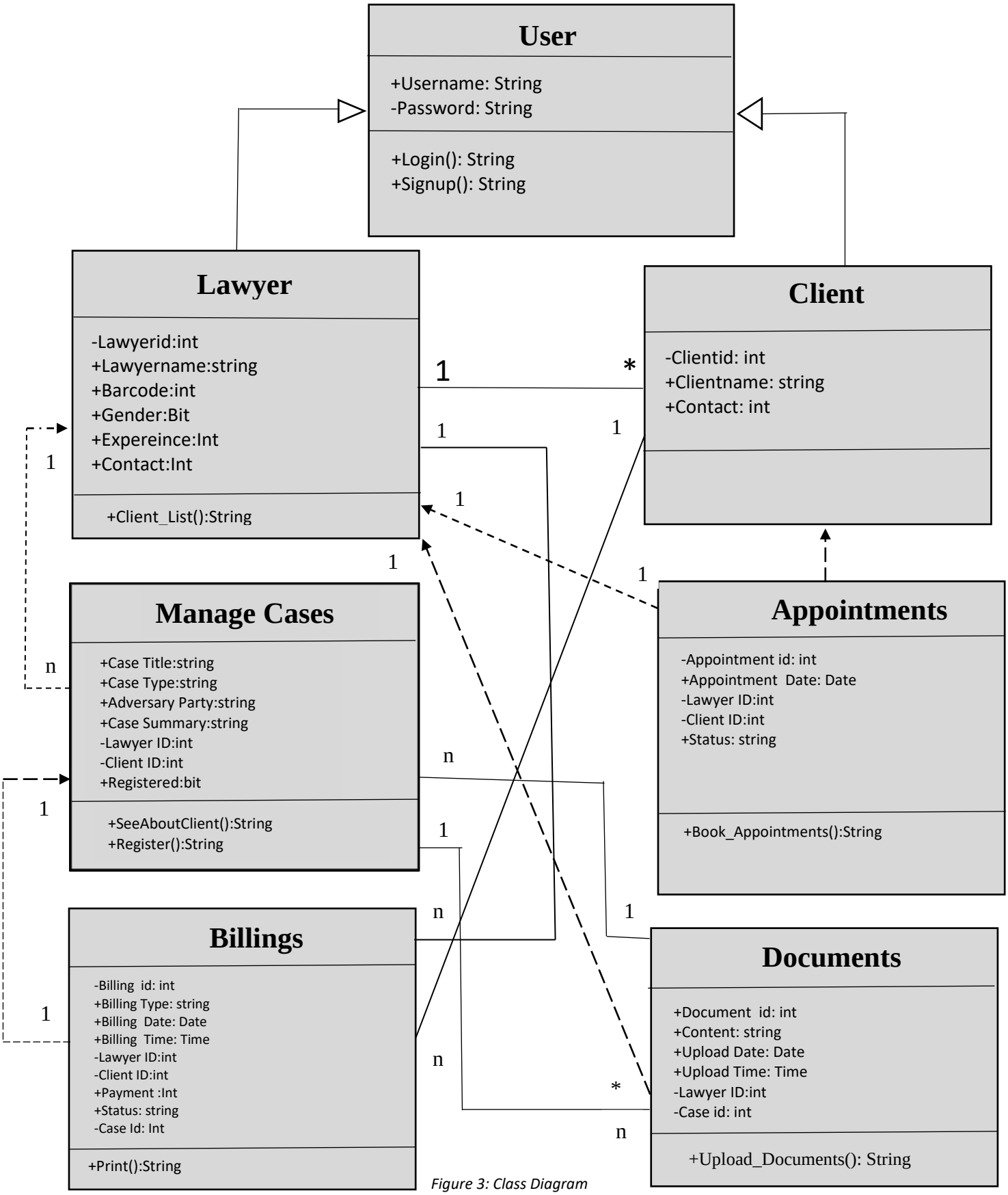


Figure 3: Class Diagram

3.3 Object Diagram

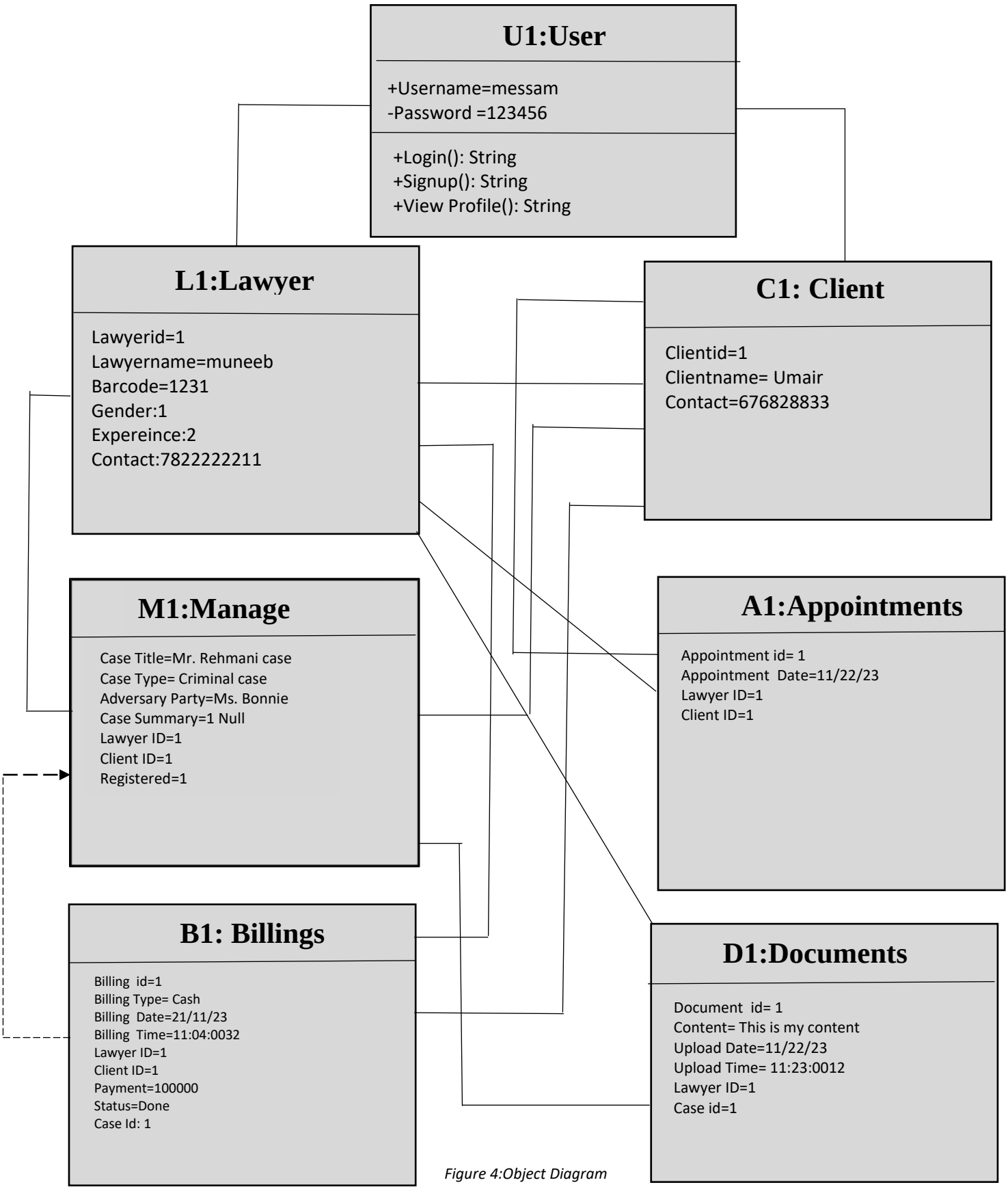


Figure 4:Object Diagram

3.4 Deployment Diagram

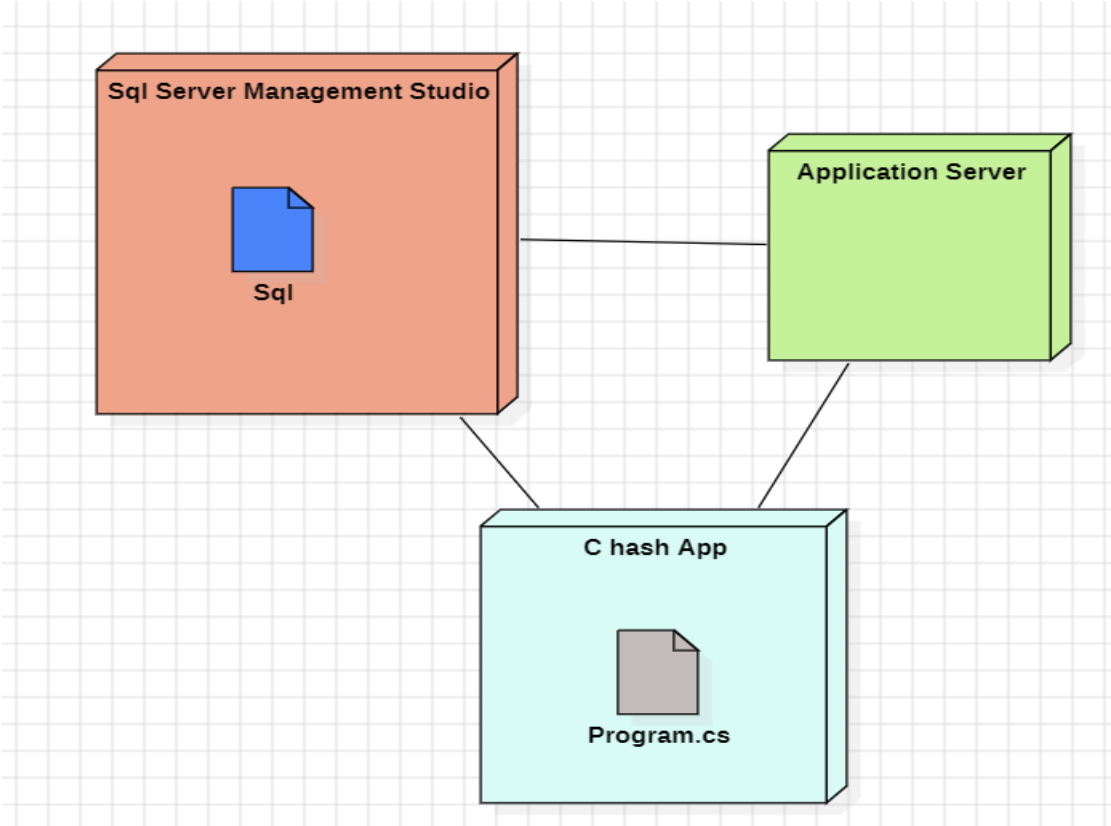


Figure 5:Deployement Diagram

3.5 Package Diagram

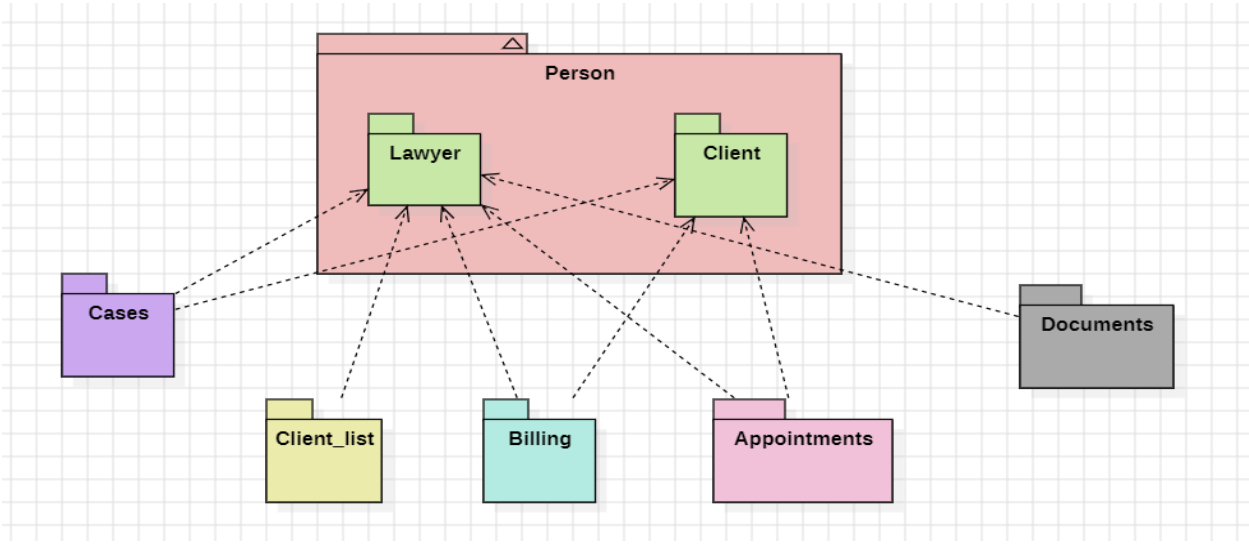


Figure 6: Package Diagram

3.6 Sequence Diagram

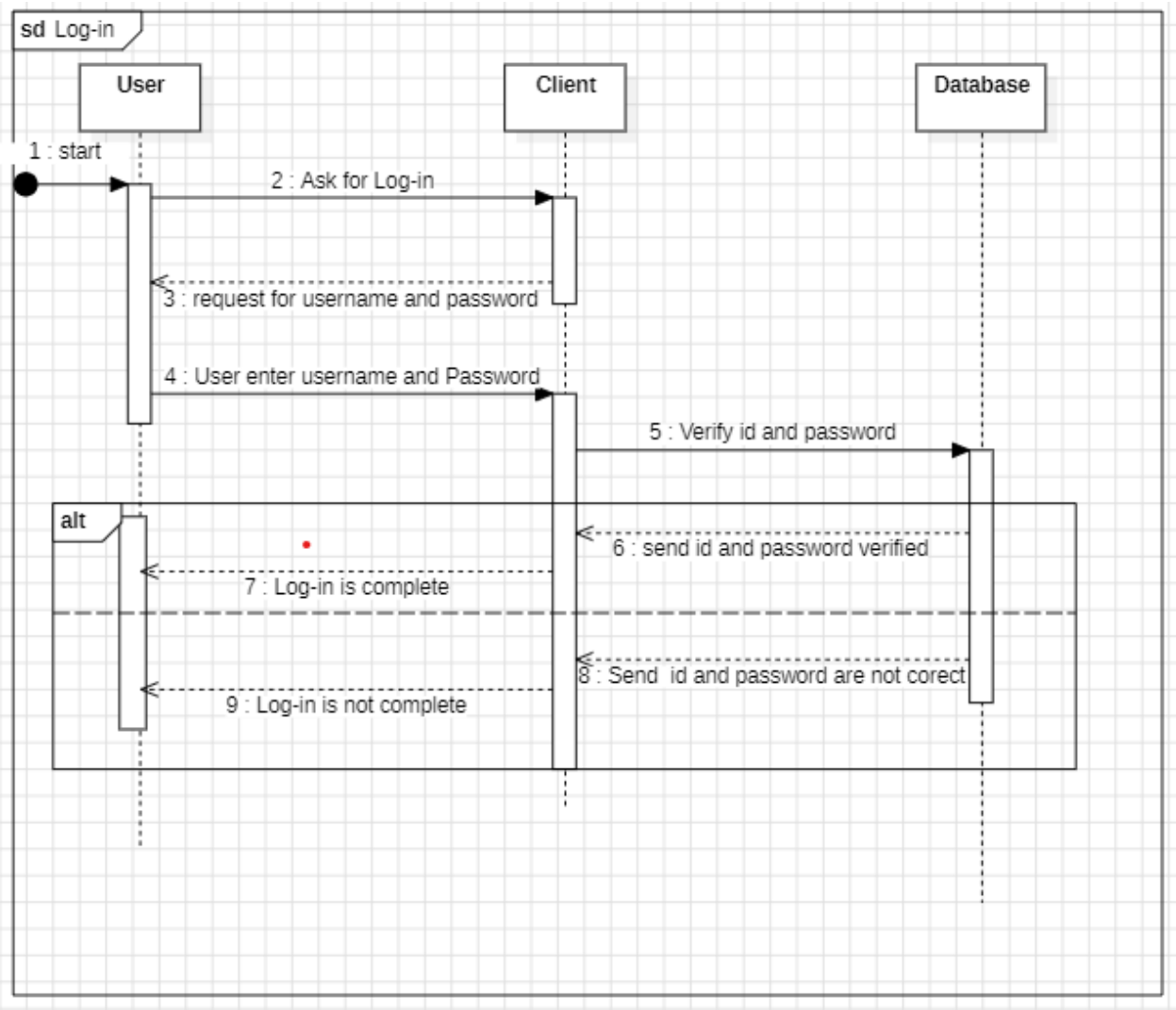


Figure 7:Login Sequence Diagram

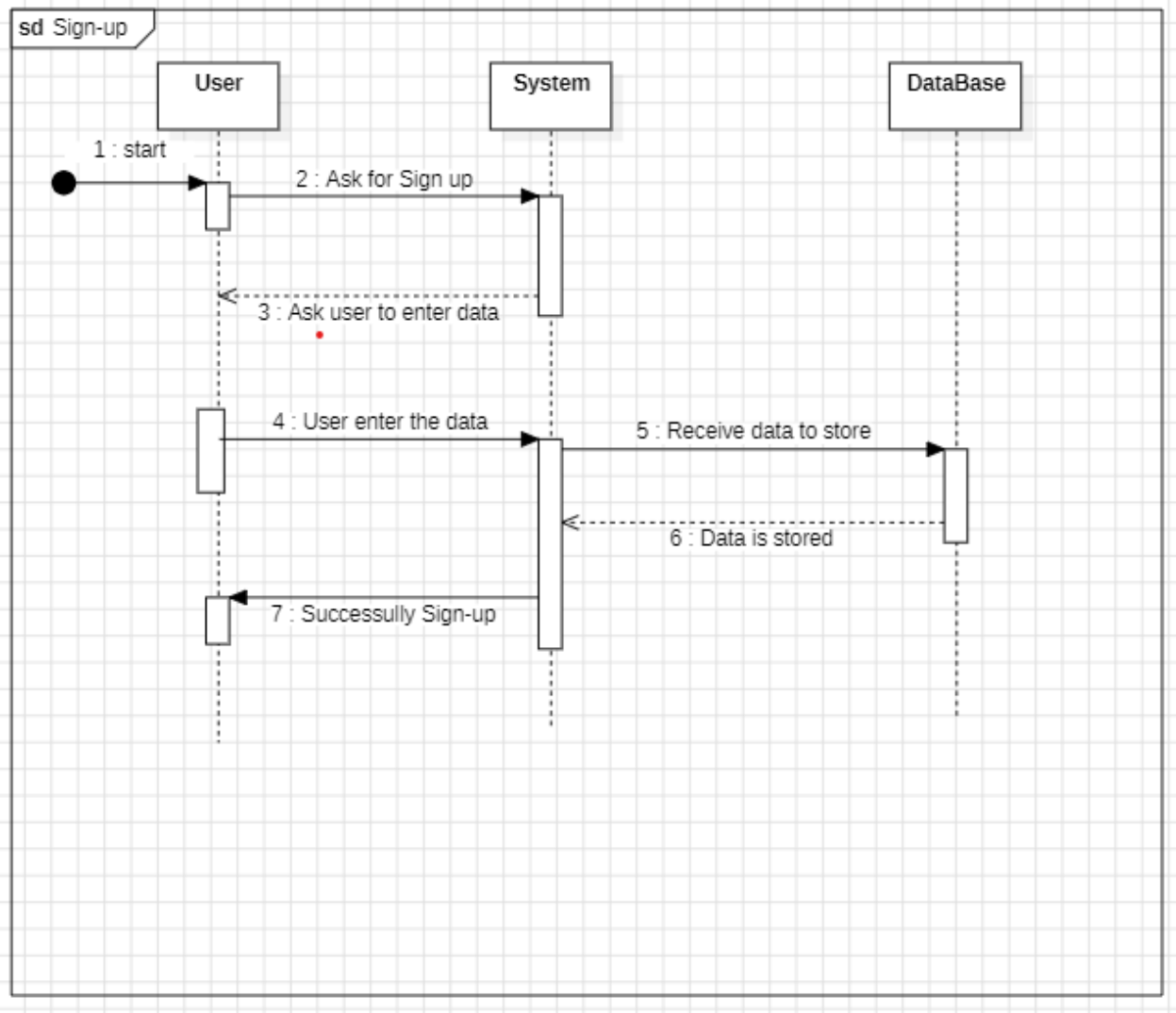


Figure 8:Signup Sequence Diagram

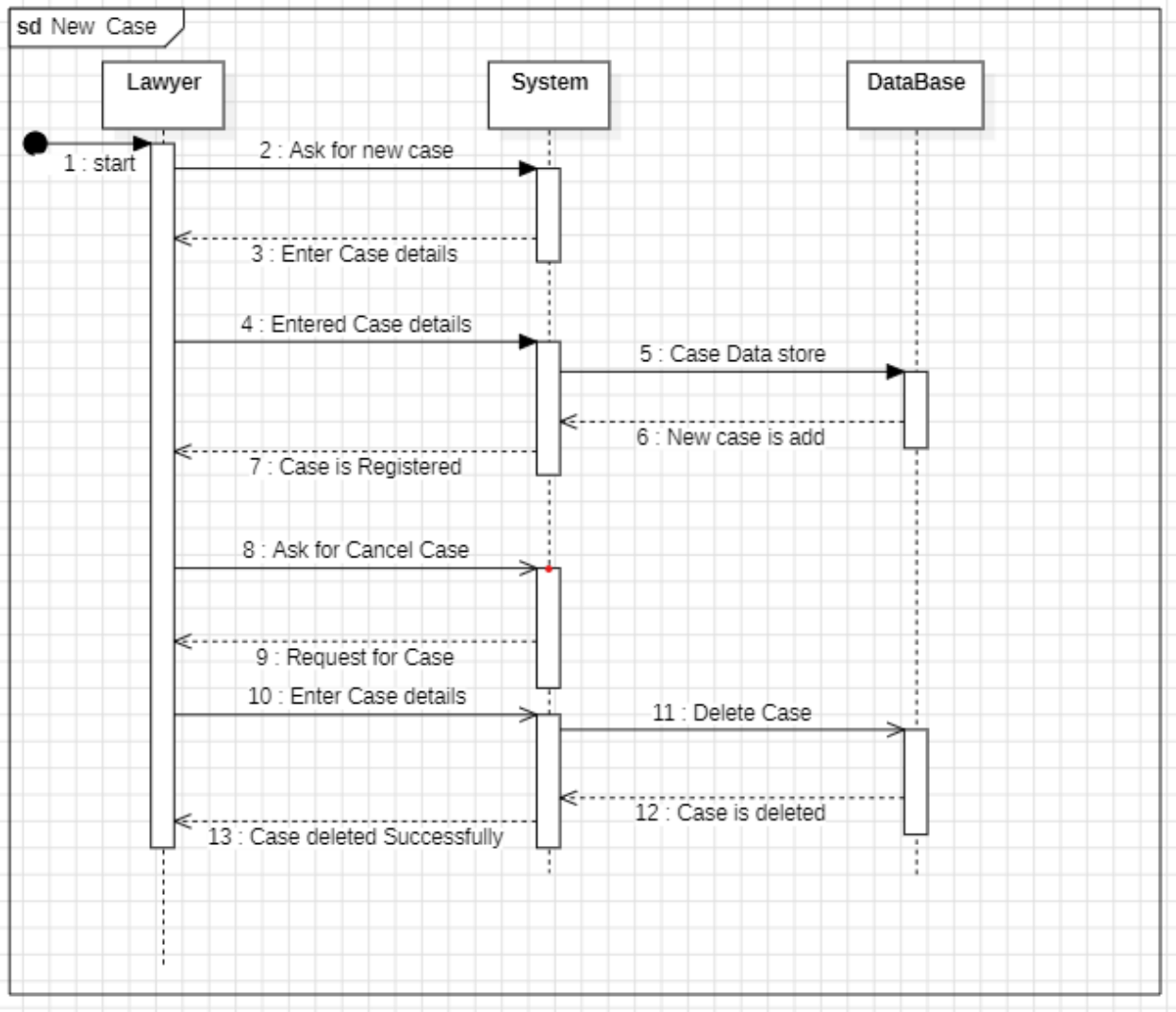


Figure 9: New Case Sequence Diagram

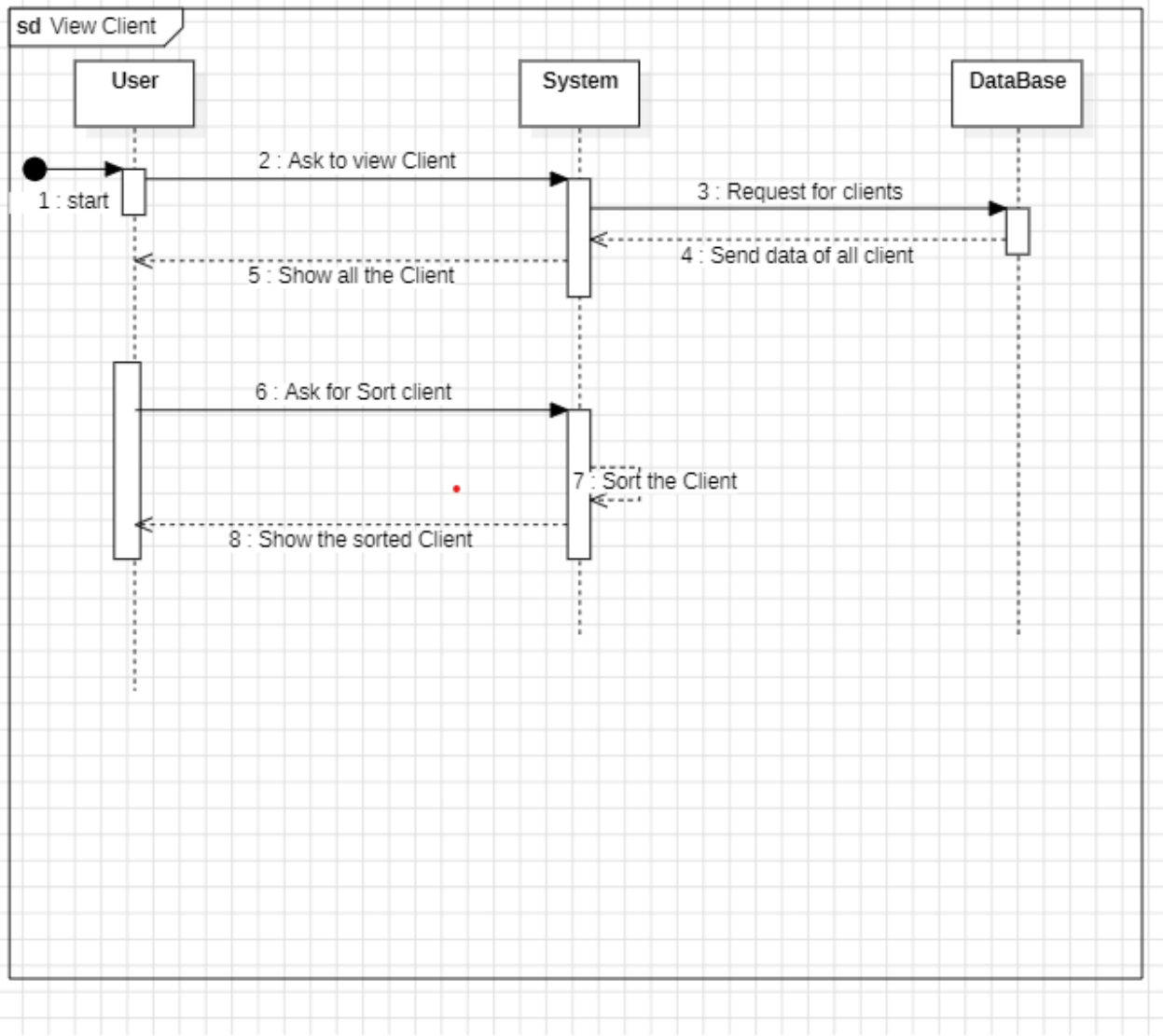


Figure 10: view client sequence Diagram

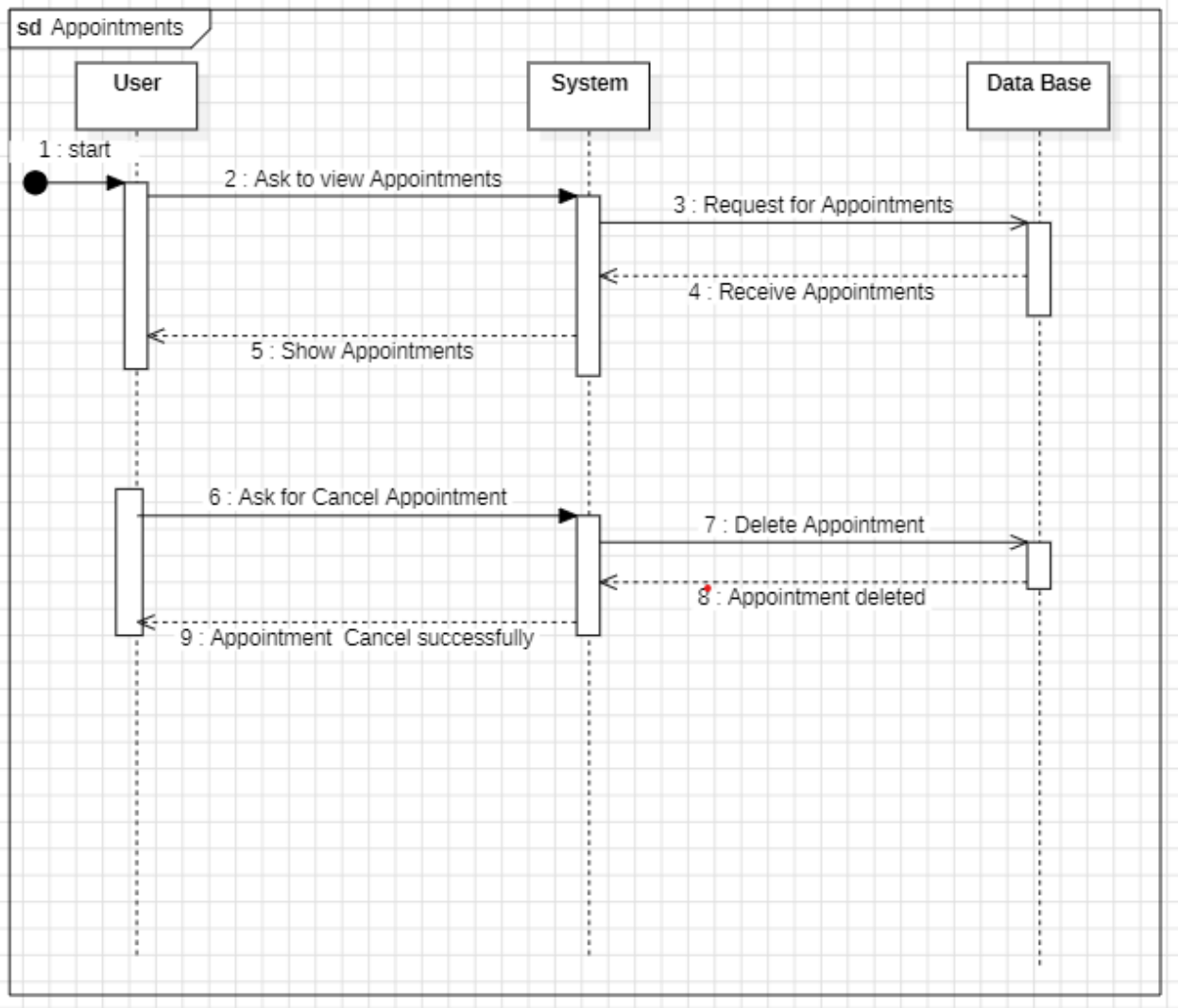


Figure 11: Appointment Sequence Diagram

3.7 Activity Diagram

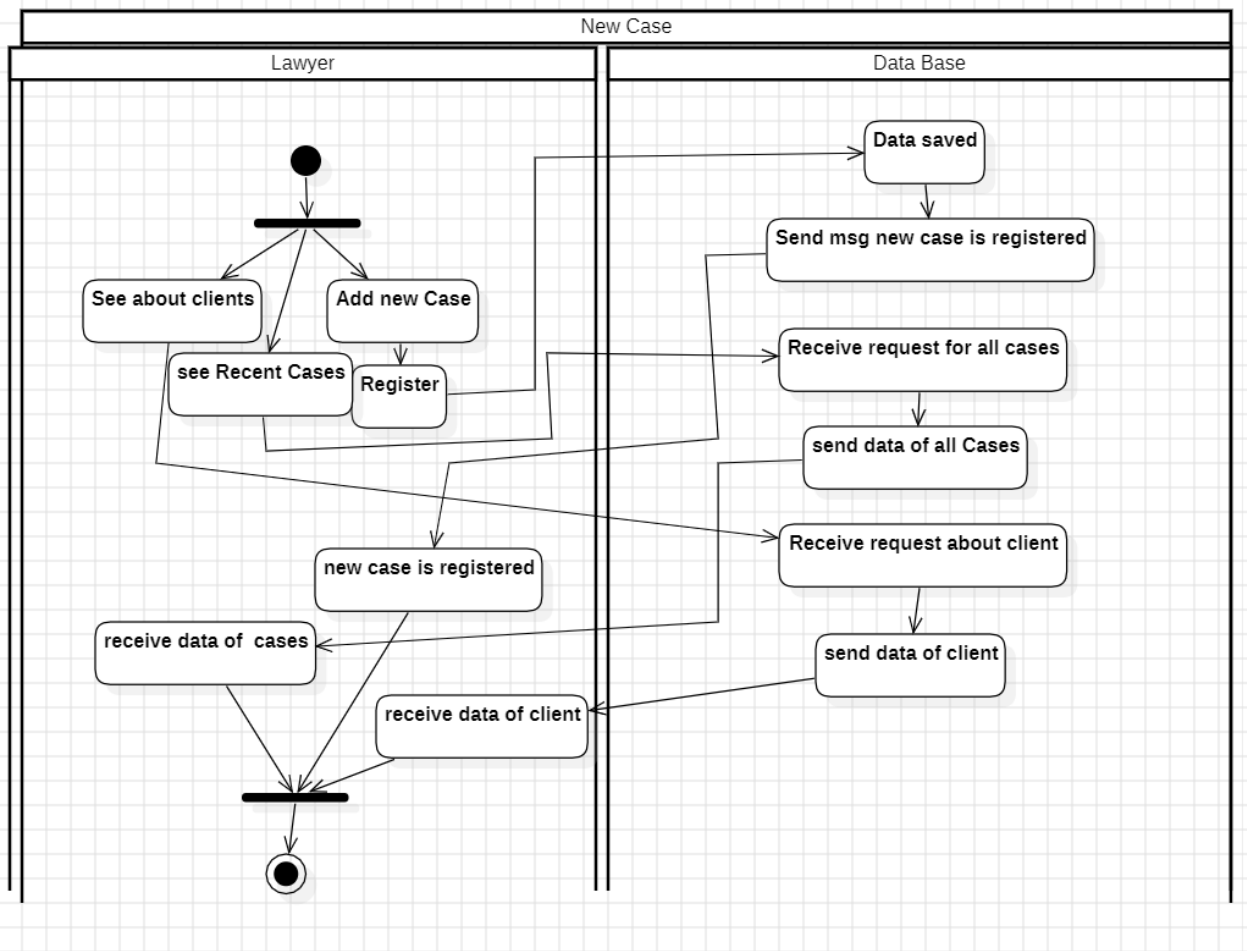


Figure 12: New Case Activity Diagram

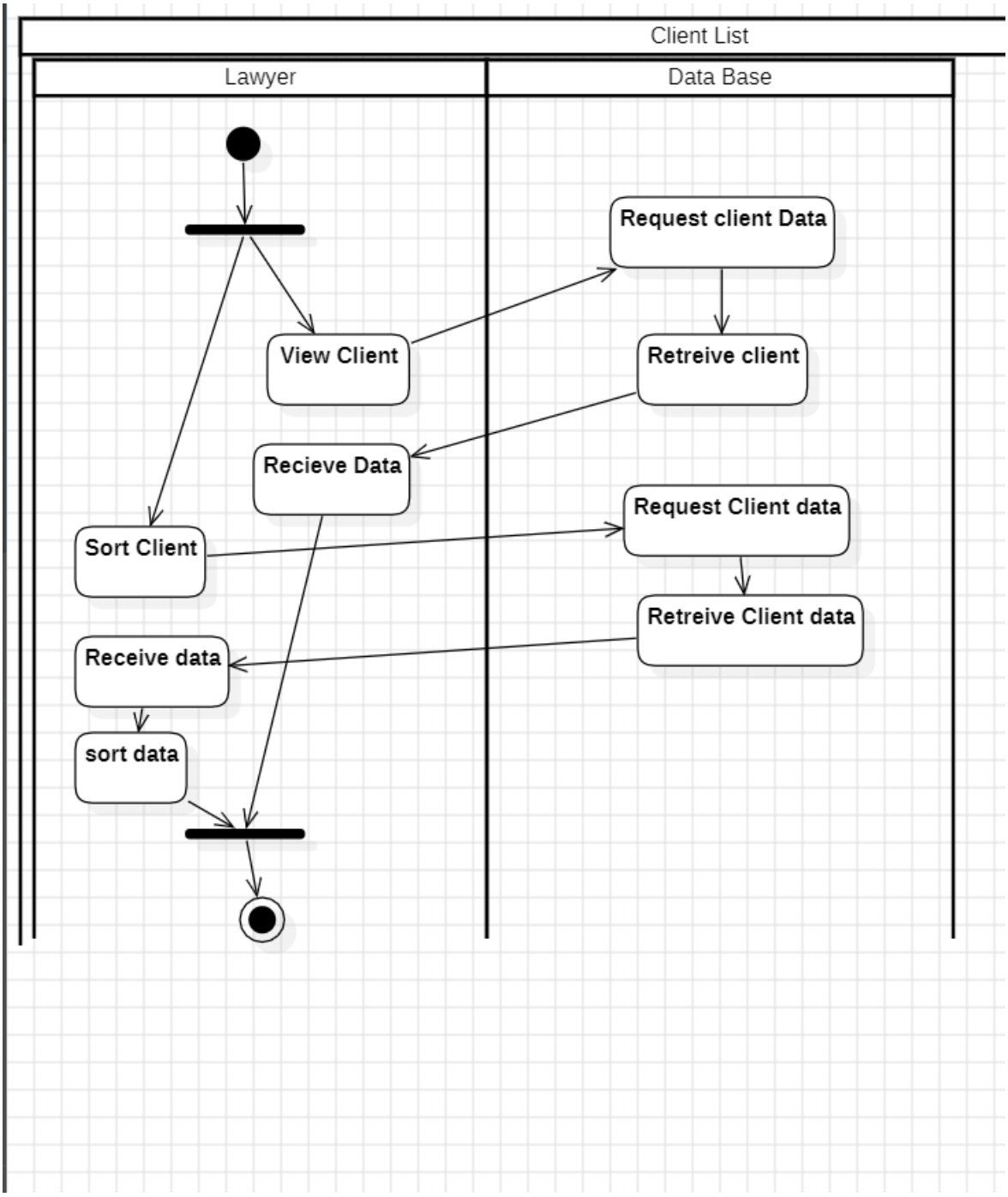


Figure 13: Client List Activity Diagram

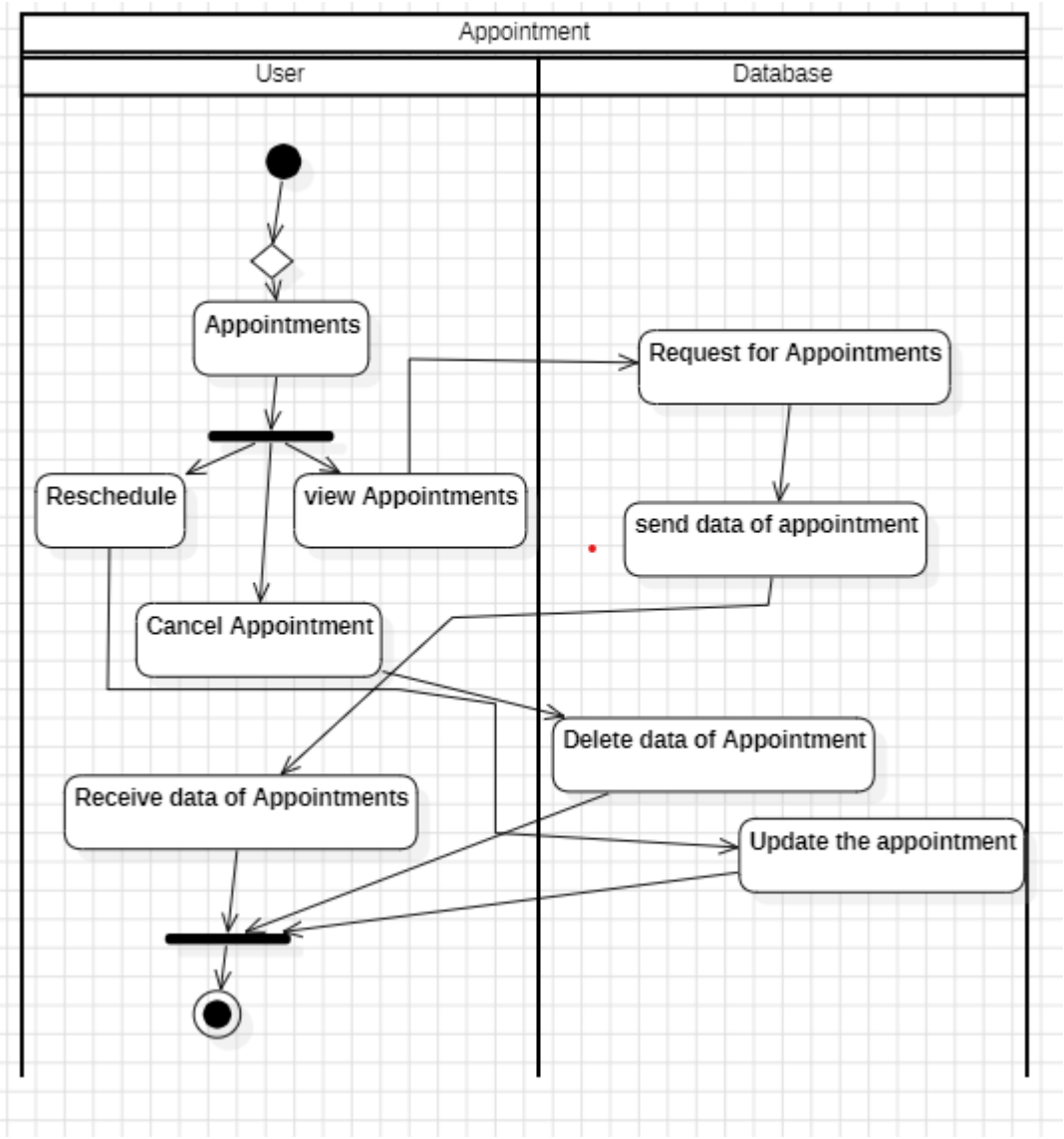


Figure 14: Appointment Activity Diagram

3.8 Collaboration Diagram

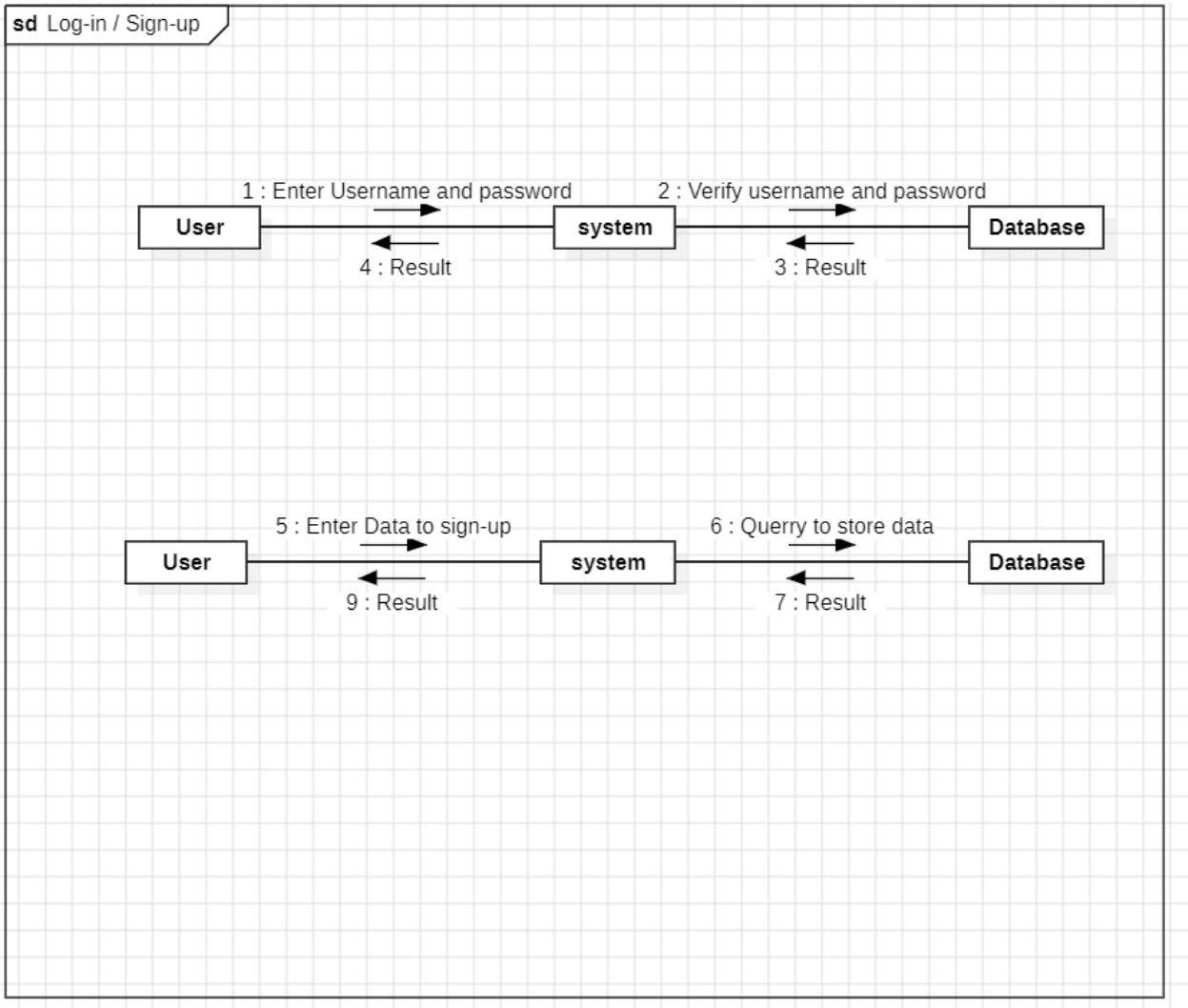


Figure 15: Login Collaboration Diagram

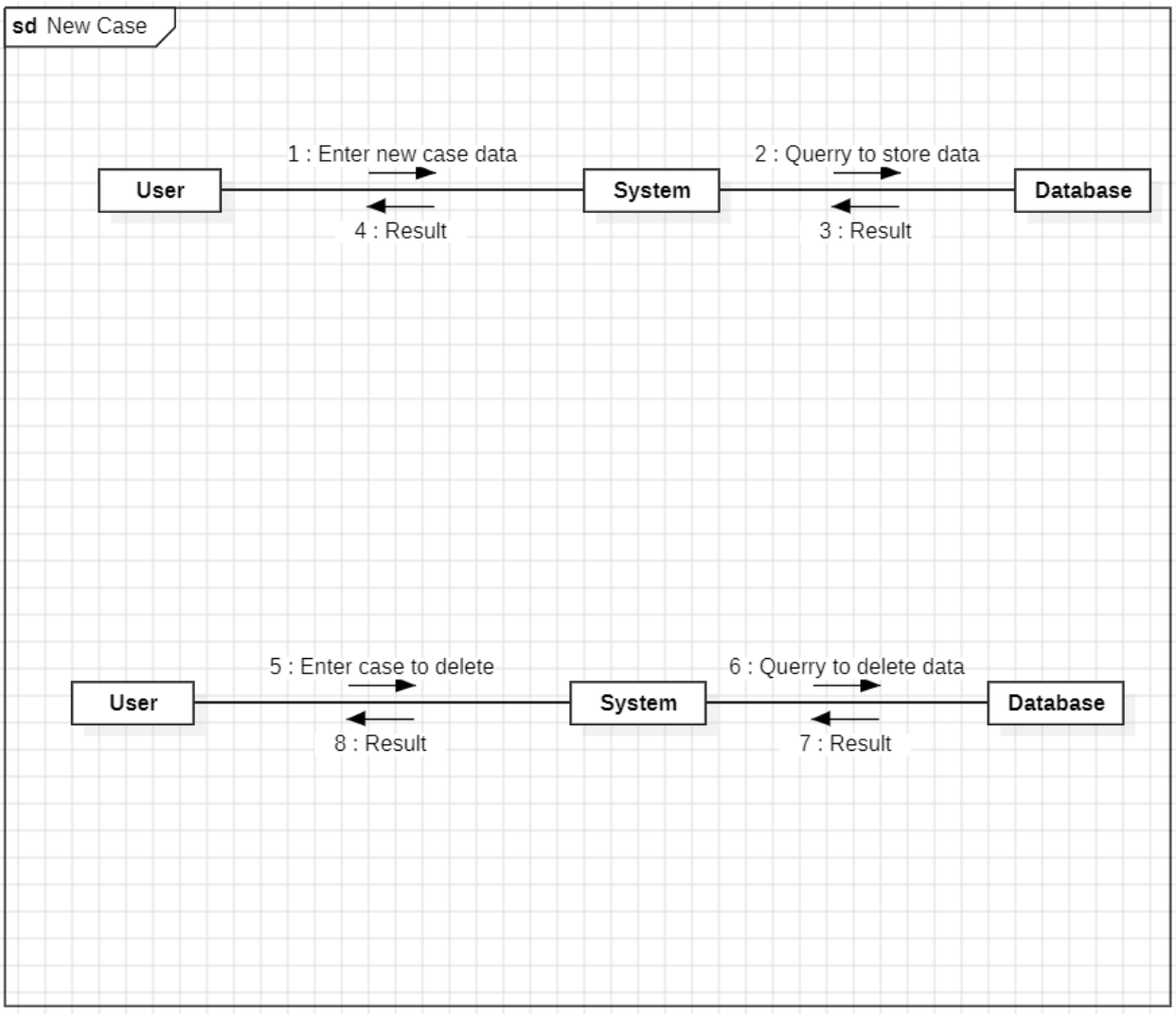


Figure 16:: New case Collaboration Diagram

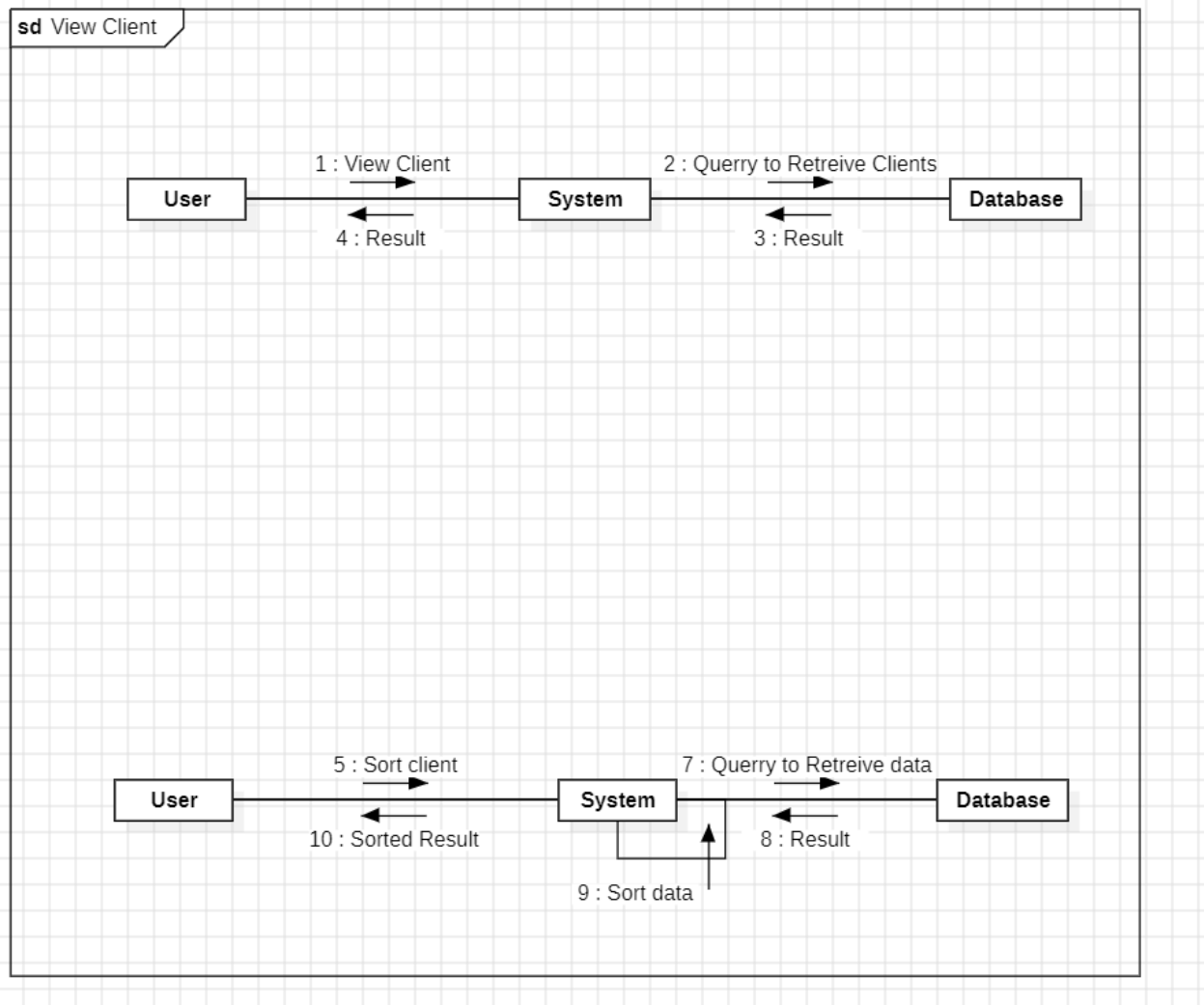


Figure 17: View client Collaboration Diagram

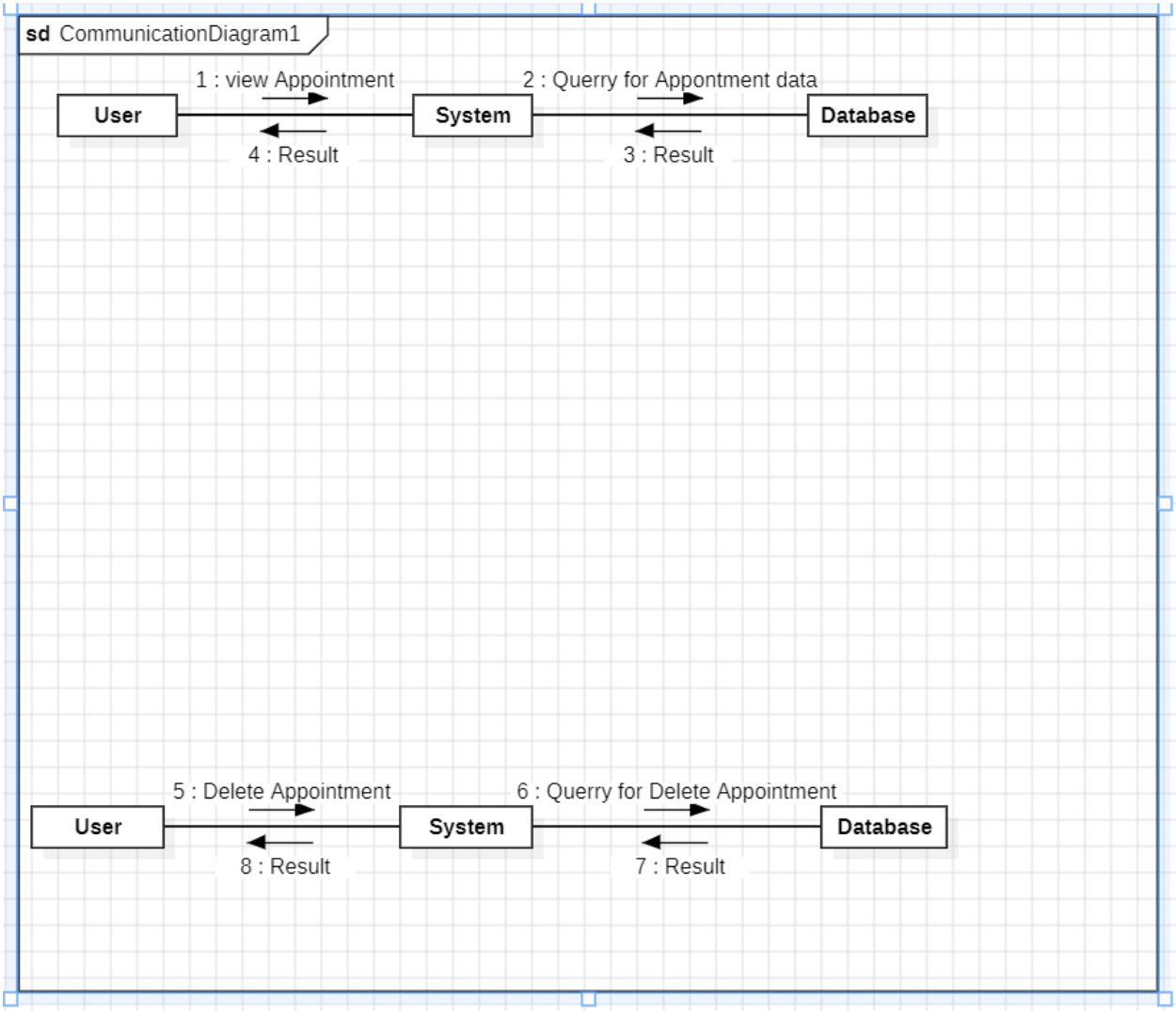


Figure 18: Appointment Collaboration Diagram

3.9 State Diagram

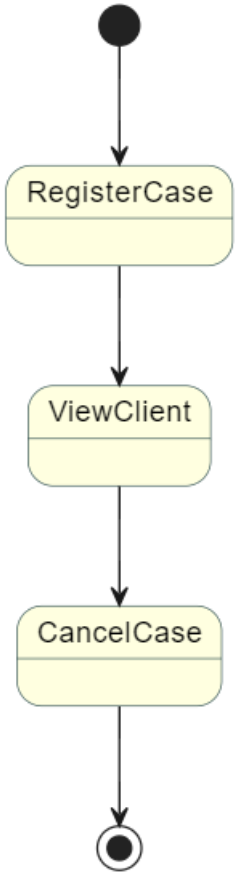


Figure 19: New Case State Diagram

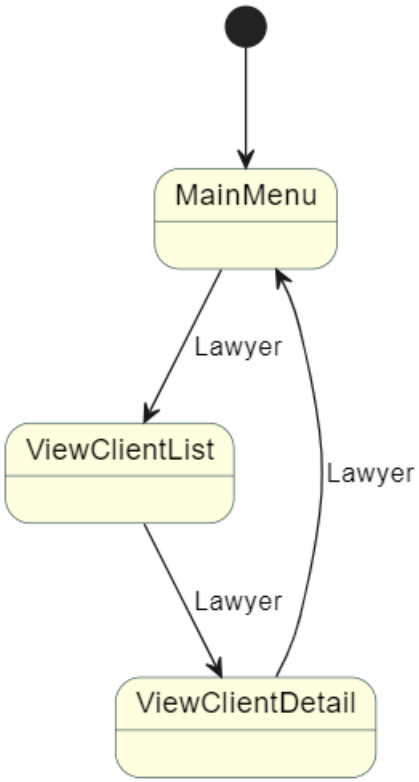


Figure 20: View Client State Diagram

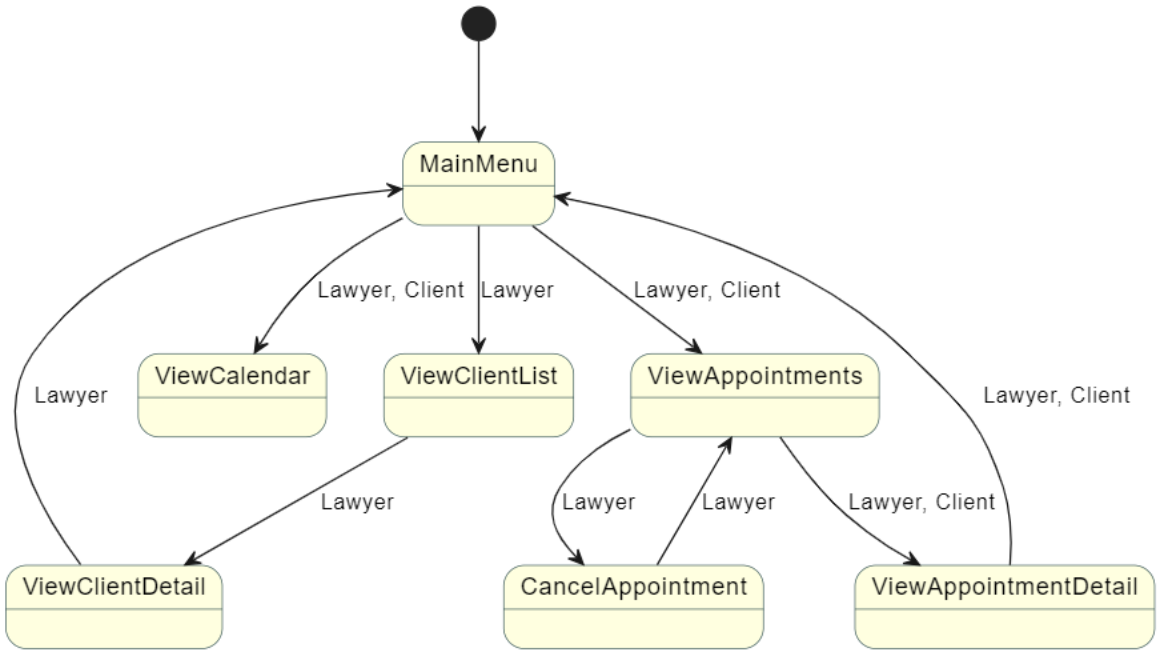
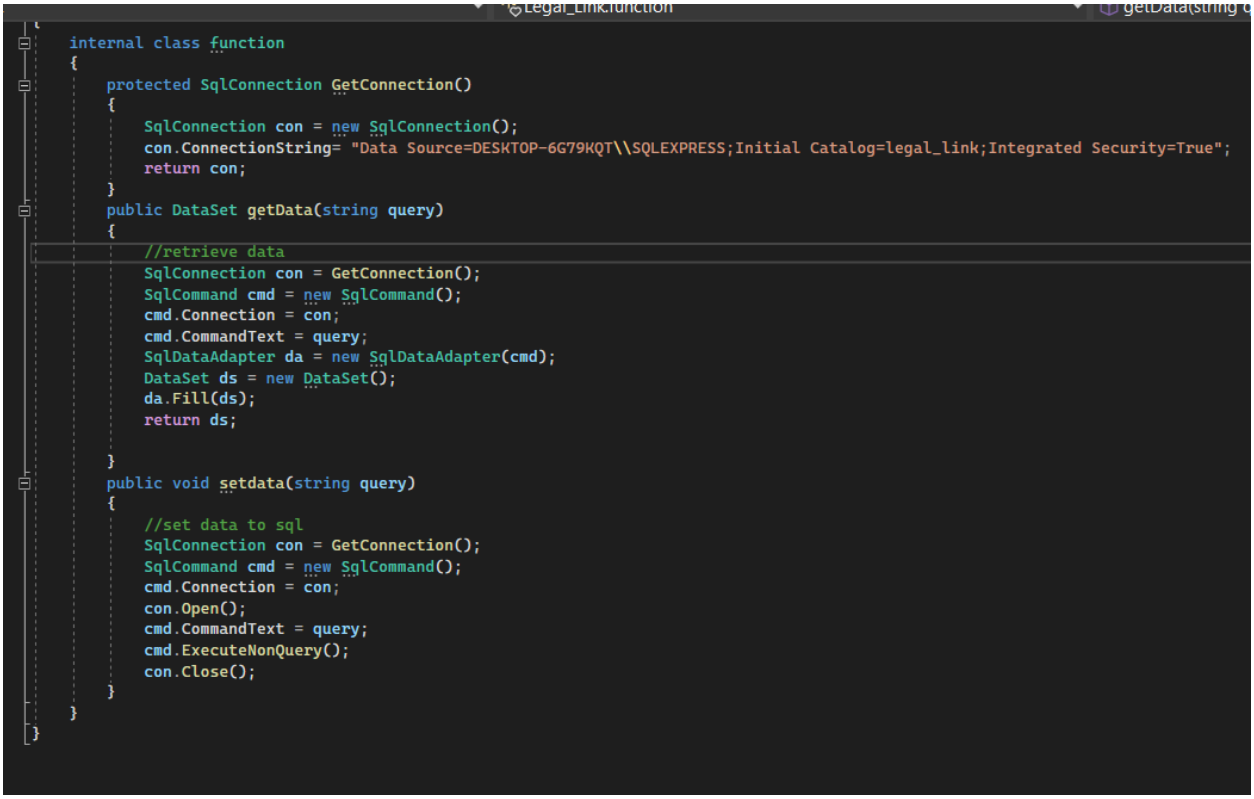


Figure 21: Appointment State Diagram

Chapter 4

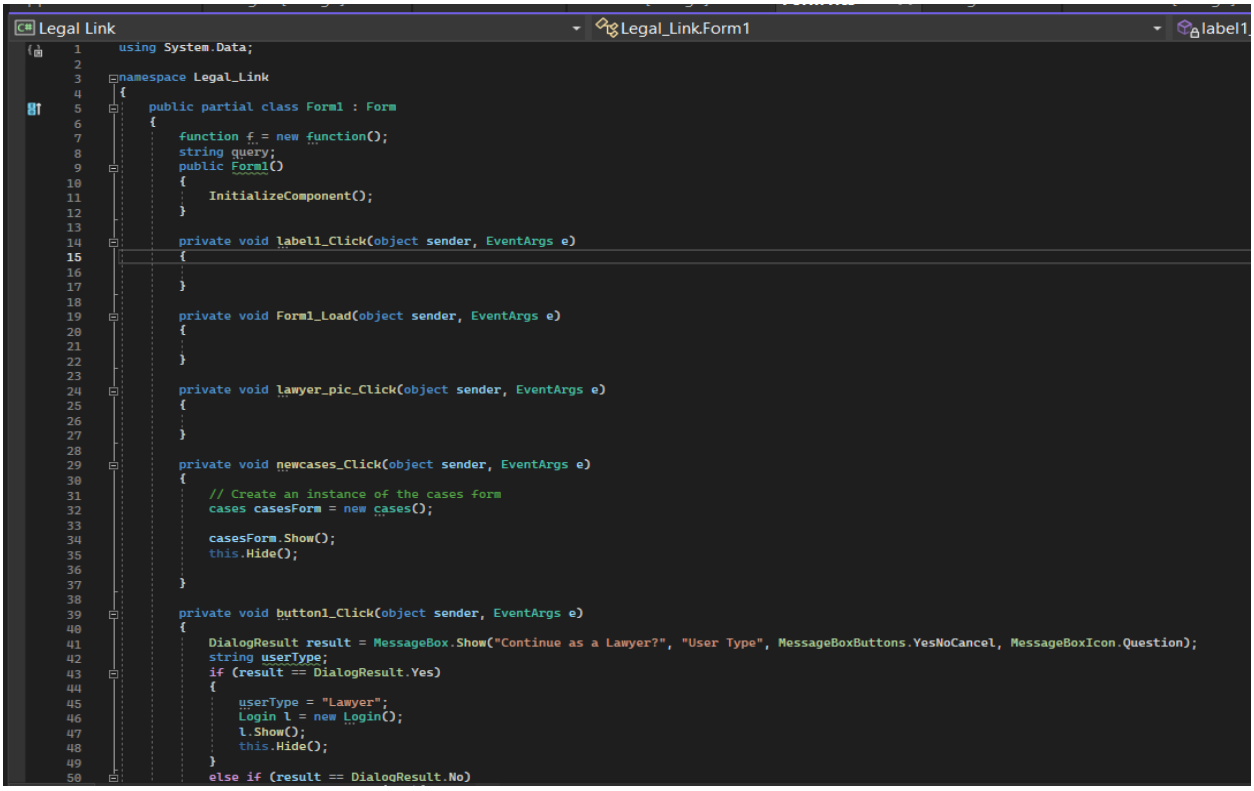
Implementation

4.1 Sql Connection Function

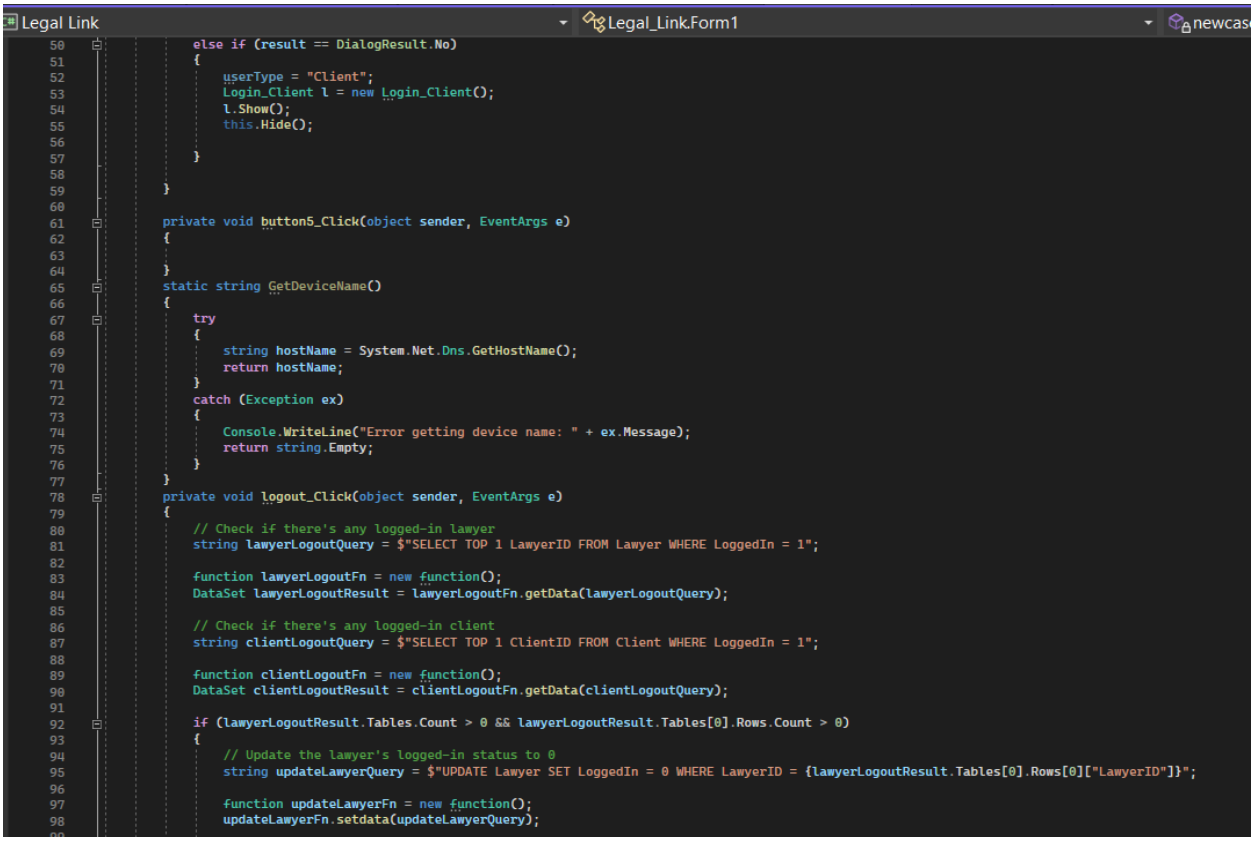


```
internal class function
{
    protected SqlConnection GetConnection()
    {
        SqlConnection con = new SqlConnection();
        con.ConnectionString= "Data Source=DESKTOP-6G79KQT\\SQLEXPRESS;Initial Catalog=legal_link;Integrated Security=True";
        return con;
    }
    public DataSet getData(string query)
    {
        //retrieve data
        SqlConnection con = GetConnection();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        cmd.CommandText = query;
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();
        da.Fill(ds);
        return ds;

    }
    public void setdata(string query)
    {
        //set data to sql
        SqlConnection con = GetConnection();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        con.Open();
        cmd.CommandText = query;
        cmd.ExecuteNonQuery();
        con.Close();
    }
}
```

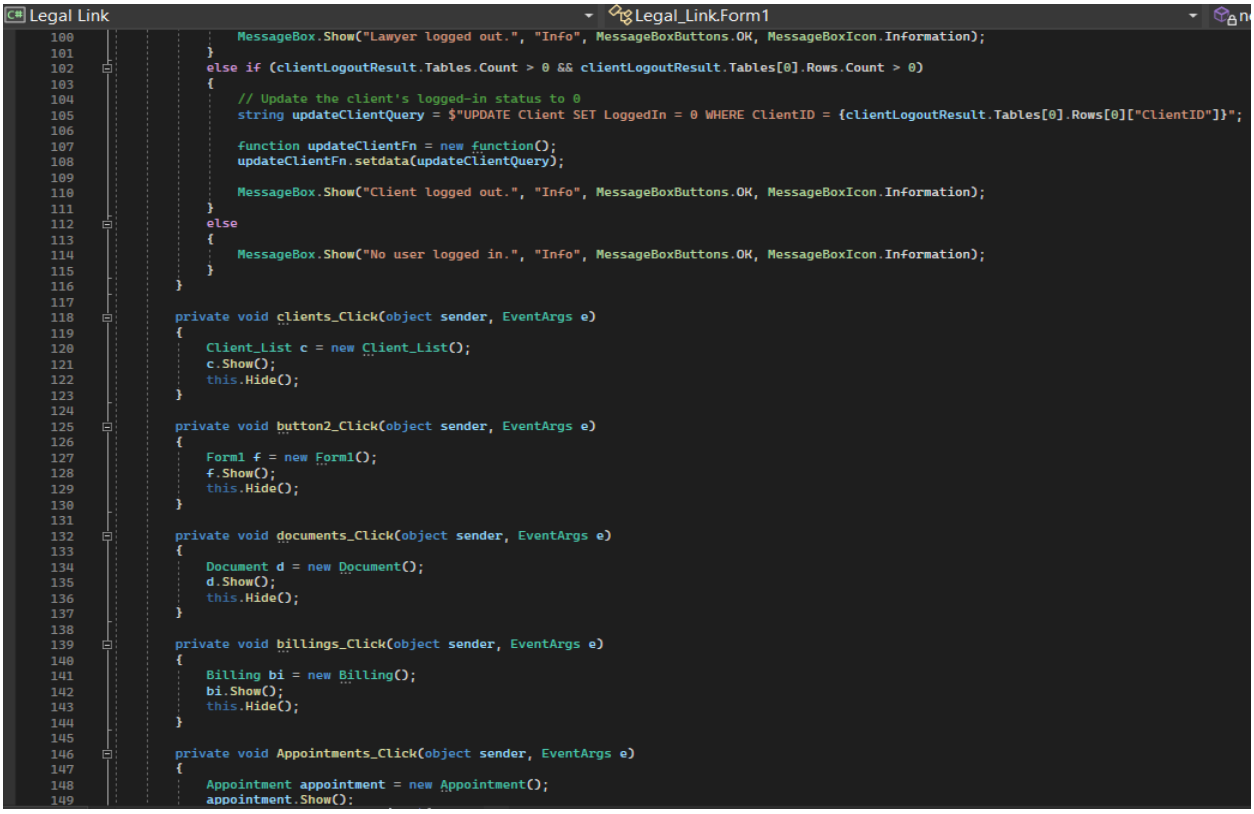
4.2 Home



```
Legal Link                                  Legal_Link.Form1                label1

using System.Data;

namespace Legal_Link
{
    public partial class Form1 : Form
    {
        function f = new function();
        string query;
        public Form1()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void lawyer_pic_Click(object sender, EventArgs e)
        {

        }

        private void newcases_Click(object sender, EventArgs e)
        {
            // Create an instance of the cases form
            cases casesForm = new cases();

            casesForm.Show();
            this.Hide();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Continue as a Lawyer?", "User Type", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
            string userType;
            if (result == DialogResult.Yes)
            {
                userType = "Lawyer";
                Login l = new Login();
                l.Show();
                this.Hide();
            }
            else if (result == DialogResult.No)
```

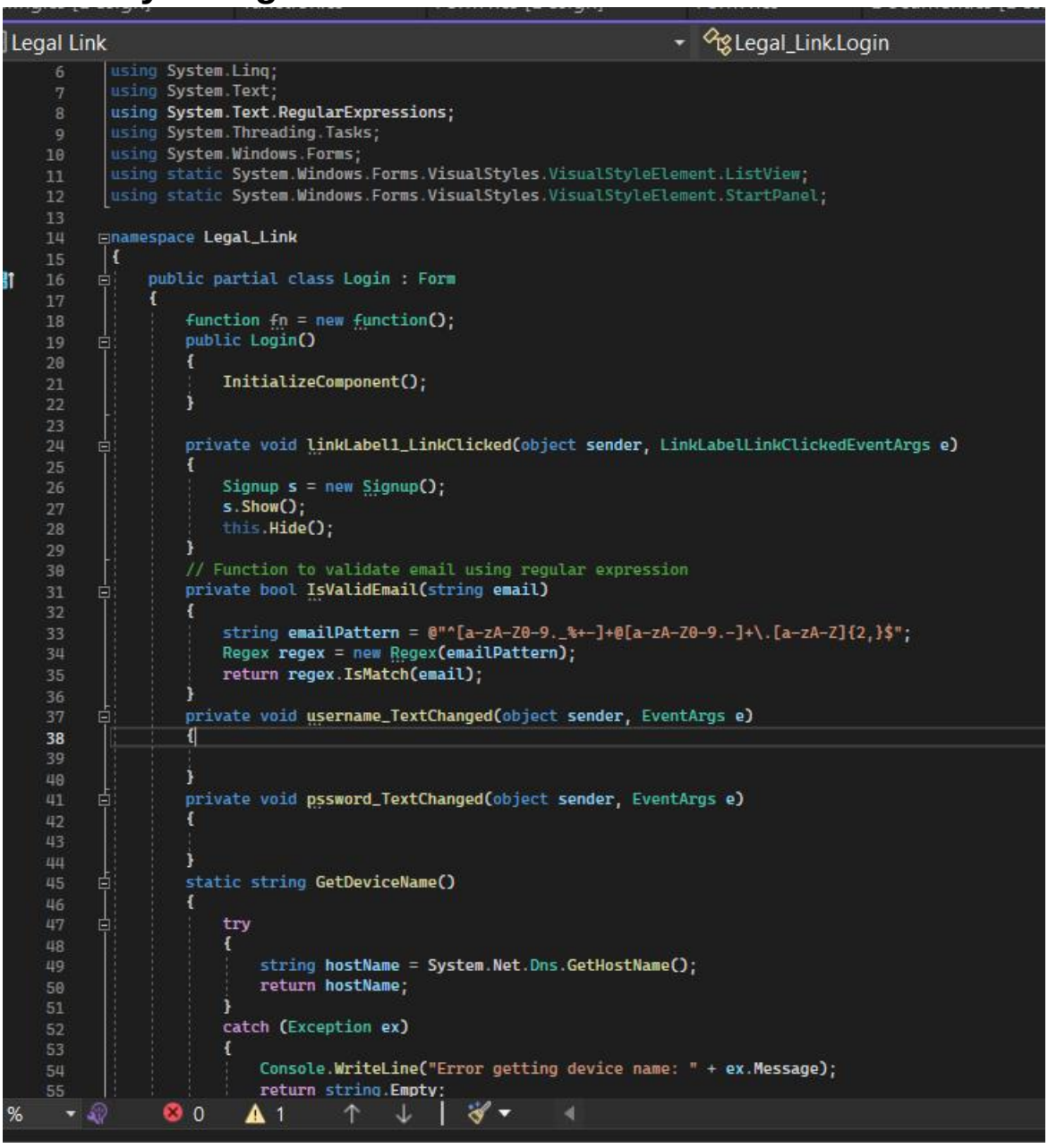


```
Legal Link                                  Legal_Link.Form1                newcas

            else if (result == DialogResult.No)
            {
                userType = "Client";
                Login_Client l = new Login_Client();
                l.Show();
                this.Hide();

            }
        }

        private void button5_Click(object sender, EventArgs e)
        {

        }
        static string GetDeviceName()
        {
            try
            {
                string hostName = System.Net.Dns.GetHostName();
                return hostName;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error getting device name: " + ex.Message);
                return string.Empty;
            }
        }
        private void logout_Click(object sender, EventArgs e)
        {
            // Check if there's any logged-in lawyer
            string lawyerLogoutQuery = $"SELECT TOP 1 LawyerID FROM Lawyer WHERE LoggedIn = 1";

            function lawyerLogoutFn = new function();
            DataSet lawyerLogoutResult = lawyerLogoutFn.getData(lawyerLogoutQuery);

            // Check if there's any logged-in client
            string clientLogoutQuery = $"SELECT TOP 1 ClientID FROM Client WHERE LoggedIn = 1";

            function clientLogoutFn = new function();
            DataSet clientLogoutResult = clientLogoutFn.getData(clientLogoutQuery);

            if (lawyerLogoutResult.Tables.Count > 0 && lawyerLogoutResult.Tables[0].Rows.Count > 0)
            {
                // Update the lawyer's logged-in status to 0
                string updateLawyerQuery = $"UPDATE Lawyer SET LoggedIn = 0 WHERE LawyerID = {lawyerLogoutResult.Tables[0].Rows[0]["LawyerID"]}";

                function updateLawyerFn = new function();
                updateLawyerFn.setdata(updateLawyerQuery);
```

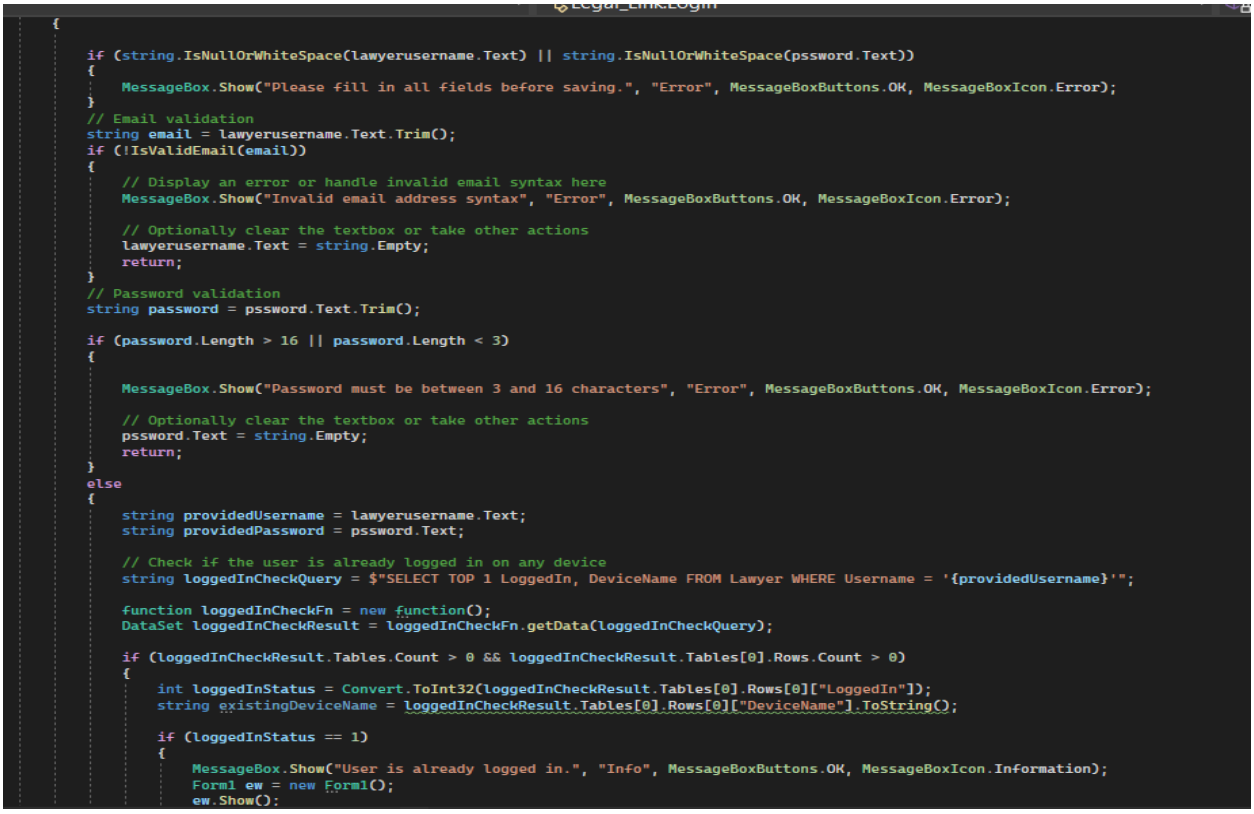



```
            MessageBox.Show("Lawyer logged out.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else if (clientLogoutResult.Tables.Count > 0 && clientLogoutResult.Tables[0].Rows.Count > 0)
        {
            // Update the client's logged-in status to 0
            string updateClientQuery = $"UPDATE Client SET LoggedIn = 0 WHERE ClientID = {clientLogoutResult.Tables[0].Rows[0]["ClientID"]}";

            function updateClientFn = new function();
            updateClientFn.setdata(updateClientQuery);

            MessageBox.Show("Client logged out.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("No user logged in.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void clients_Click(object sender, EventArgs e)
    {
        Client_List c = new Client_List();
        c.Show();
        this.Hide();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form1 f = new Form1();
        f.Show();
        this.Hide();
    }

    private void documents_Click(object sender, EventArgs e)
    {
        Document d = new Document();
        d.Show();
        this.Hide();
    }

    private void billings_Click(object sender, EventArgs e)
    {
        Billing bi = new Billing();
        bi.Show();
        this.Hide();
    }

    private void Appointments_Click(object sender, EventArgs e)
    {
        Appointment appointment = new Appointment();
        appointment.Show();
```

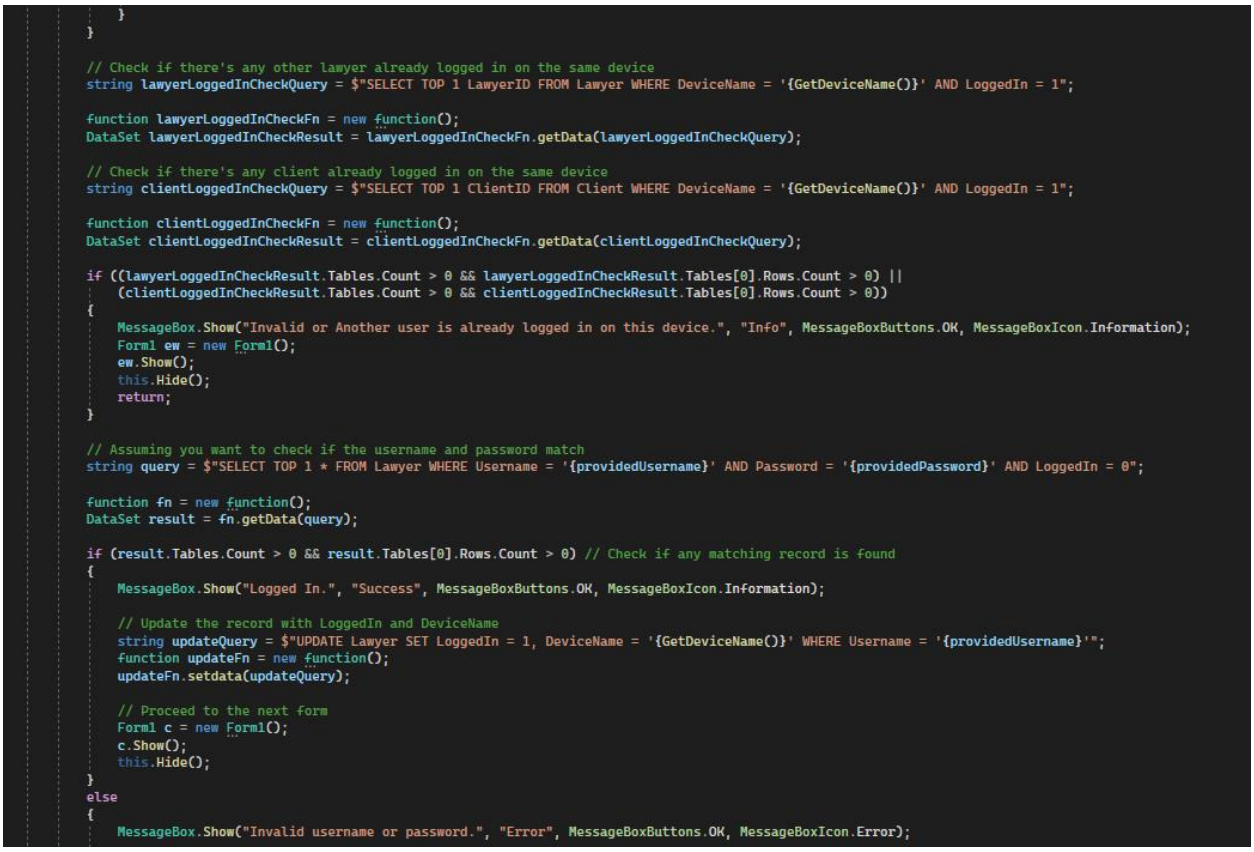
4.3 Lawyer Login




```
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace Legal_Link
{
    public partial class Login : Form
    {
        function fn = new function();
        public Login()
        {
            InitializeComponent();
        }

        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Signup s = new Signup();
            s.Show();
            this.Hide();
        }
        // Function to validate email using regular expression
        private bool IsValidEmail(string email)
        {
            string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
            Regex regex = new Regex(emailPattern);
            return regex.IsMatch(email);
        }
        private void username_TextChanged(object sender, EventArgs e)
        {

        }
        private void password_TextChanged(object sender, EventArgs e)
        {

        }
        static string GetDeviceName()
        {
            try
            {
                string hostName = System.Net.Dns.GetHostName();
                return hostName;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error getting device name: " + ex.Message);
                return string.Empty;
```



```
{
    if (string.IsNullOrWhiteSpace(lawyerusername.Text) || string.IsNullOrWhiteSpace(pssword.Text))
    {
        MessageBox.Show("Please fill in all fields before saving.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    // Email validation
    string email = lawyerusername.Text.Trim();
    if (!IsValidEmail(email))
    {
        // Display an error or handle invalid email syntax here
        MessageBox.Show("Invalid email address syntax", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

        // Optionally clear the textbox or take other actions
        lawyerusername.Text = string.Empty;
        return;
    }
    // Password validation
    string password = pssword.Text.Trim();

    if (password.Length > 16 || password.Length < 3)
    {

        MessageBox.Show("Password must be between 3 and 16 characters", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

        // Optionally clear the textbox or take other actions
        pssword.Text = string.Empty;
        return;
    }
    else
    {
        string providedUsername = lawyerusername.Text;
        string providedPassword = pssword.Text;

        // Check if the user is already logged in on any device
        string loggedInCheckQuery = $"SELECT TOP 1 LoggedIn, DeviceName FROM Lawyer WHERE Username = '{providedUsername}'";

        function loggedInCheckFn = new function();
        DataSet loggedInCheckResult = loggedInCheckFn.getData(loggedInCheckQuery);

        if (loggedInCheckResult.Tables.Count > 0 && loggedInCheckResult.Tables[0].Rows.Count > 0)
        {
            int loggedInStatus = Convert.ToInt32(loggedInCheckResult.Tables[0].Rows[0]["LoggedIn"]);
            string existingDeviceName = loggedInCheckResult.Tables[0].Rows[0]["DeviceName"].ToString();

            if (loggedInStatus == 1)
            {
                MessageBox.Show("User is already logged in.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
                Form1 ew = new Form1();
                ew.Show();
```



```
            }
        }

        // Check if there's any other lawyer already logged in on the same device
        string lawyerLoggedInCheckQuery = $"SELECT TOP 1 LawyerID FROM Lawyer WHERE DeviceName = '{GetDeviceName()}' AND LoggedIn = 1";

        function lawyerLoggedInCheckFn = new function();
        DataSet lawyerLoggedInCheckResult = lawyerLoggedInCheckFn.getData(lawyerLoggedInCheckQuery);

        // Check if there's any client already logged in on the same device
        string clientLoggedInCheckQuery = $"SELECT TOP 1 ClientID FROM Client WHERE DeviceName = '{GetDeviceName()}' AND LoggedIn = 1";

        function clientLoggedInCheckFn = new function();
        DataSet clientLoggedInCheckResult = clientLoggedInCheckFn.getData(clientLoggedInCheckQuery);

        if ((lawyerLoggedInCheckResult.Tables.Count > 0 && lawyerLoggedInCheckResult.Tables[0].Rows.Count > 0) ||
            (clientLoggedInCheckResult.Tables.Count > 0 && clientLoggedInCheckResult.Tables[0].Rows.Count > 0))
        {
            MessageBox.Show("Invalid or Another user is already logged in on this device.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
            Form1 ew = new Form1();
            ew.Show();
            this.Hide();
            return;
        }

        // Assuming you want to check if the username and password match
        string query = $"SELECT TOP 1 * FROM Lawyer WHERE Username = '{providedUsername}' AND Password = '{providedPassword}' AND LoggedIn = 0";

        function fn = new function();
        DataSet result = fn.getData(query);

        if (result.Tables.Count > 0 && result.Tables[0].Rows.Count > 0) // Check if any matching record is found
        {
            MessageBox.Show("Logged In.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

            // Update the record with LoggedIn and DeviceName
            string updateQuery = $"UPDATE Lawyer SET LoggedIn = 1, DeviceName = '{GetDeviceName()}' WHERE Username = '{providedUsername}'";
            function updateFn = new function();
            updateFn.setdata(updateQuery);

            // Proceed to the next form
            Form1 c = new Form1();
            c.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Invalid username or password.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                    // Proceed to the next form
                    Form1 c = new Form1();
                    c.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Invalid username or password.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }
            }



        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 f = new Form1();
            f.Show();
            this.Hide();
        }
    }
}
```

4.3 Client Login

```
namespace Legal_Link
{
    public partial class Login_Client : Form
    {
        public Login_Client()
        {
            InitializeComponent();
        }
        // Function to validate email using regular expression
        private bool IsValidEmail(string email)
        {
            string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
            Regex regex = new Regex(emailPattern);
            return regex.IsMatch(email);
        }
        private void us_TextChanged(object sender, EventArgs e)
        {

        }

        private void password_TextChanged(object sender, EventArgs e)
        {


        }
        static string GetDeviceName()
        {
            try
            {
                string hostName = System.Net.Dns.GetHostName();
                return hostName;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error getting device name: " + ex.Message);
                return string.Empty;
            }
        }
        private void button1_Click(object sender, EventArgs e) //register
        {
            //client login
            if (string.IsNullOrWhiteSpace(us.Text) || string.IsNullOrWhiteSpace(password.Text))
            {
```

```
        string email = us.Text.Trim();
        if (!IsValidEmail(email))
        {

            MessageBox.Show("Invalid email address syntax", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);


            us.Text = string.Empty;
        }
        // Password validation
        string password = pssword.Text.Trim();

        if (password.Length > 16 || password.Length < 3)
        {

            MessageBox.Show("Password must be between 3 and 16 characters", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            // Optionally clear the textbox or take other actions
            pssword.Text = string.Empty;
        }
        else
        {
            string providedUsername = us.Text;
            string providedPassword = pssword.Text;

            // Check if the user is already logged in on any device
            string loggedInCheckQuery = $"SELECT TOP 1 LoggedIn, DeviceName FROM Client WHERE Username = '{providedUsername}'";

            function loggedInCheckFn = new function();
            DataSet loggedInCheckResult = loggedInCheckFn.getData(loggedInCheckQuery);

            if (loggedInCheckResult.Tables.Count > 0 && loggedInCheckResult.Tables[0].Rows.Count > 0)
            {
                int loggedInStatus = Convert.ToInt32(loggedInCheckResult.Tables[0].Rows[0]["LoggedIn"]);
                string existingDeviceName = loggedInCheckResult.Tables[0].Rows[0]["DeviceName"].ToString();

                if (loggedInStatus == 1)
                {
                    MessageBox.Show("User is already logged in.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    Form1 ew = new Form1();
                    ew.Show();
                    this.Hide();
                    return;
                }
            }
            // Check if there's any other lawyer already logged in on the same device
            string lawyerLoggedInCheckQuery = $"SELECT TOP 1 LawyerID FROM Lawyer WHERE DeviceName = '{GetDeviceName()}' AND LoggedIn = 1";

            function lawyerLoggedInCheckFn = new function();
            DataSet lawyerLoggedInCheckResult = lawyerLoggedInCheckFn.getData(lawyerLoggedInCheckQuery);
```

```
117                     MessageBox.Show("Invalid or Another user is already logged in on this device.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
118                     Form1 ew = new Form1();
119                     ew.Show();
120                     this.Hide();
121                     return;
122                 }
123
124                 // Assuming you want to check if the username and password match
125                 string query = $"SELECT TOP 1 * FROM Client WHERE Username = '{providedUsername}' AND Password = '{providedPassword}' AND LoggedIn = 0";
126
127                 function fn = new function();
128                 DataSet result = fn.getData(query);
129
130                 if (result.Tables.Count > 0 && result.Tables[0].Rows.Count > 0) // Check if any matching record is found
131                 {
132                     MessageBox.Show("Logged In.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
133
134                     // Update the record with LoggedIn and DeviceName
135                     string updateQuery = $"UPDATE Client SET LoggedIn = 1, DeviceName = '{GetDeviceName()}' WHERE Username = '{providedUsername}'";
136                     function updateFn = new function();
137                     updateFn.setdata(updateQuery);
138
139                     // Proceed to the next form or perform other actions
140                     Form1 c = new Form1();
141                     c.Show();
142                     this.Hide();
143                 }
144                 else
145                 {
146                     MessageBox.Show("Invalid username or password.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
147                 }
148             }
149         }
150
151         private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
152         {
153             Signup l = new Signup();
154             l.Show();
155             this.Hide();
156         }
157
158         private void button2_Click(object sender, EventArgs e)
159         {
160             Form1 f = new Form1();
161             f.Show();
162             this.Hide();
163         }
164     }
165 }
```

4.4 Sign up

```
        }
        // Password validation
        string ps = password1.Text.Trim();

        if (ps.Length > 16 || ps.Length < 3)
        {

            MessageBox.Show("Password must be between 3 and 16 characters", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            password1.Text = string.Empty;
            return;
        }
        string enteredName = User_Full_name.Text.Trim();

        // Define a regular expression pattern to allow only English letters
        string pattern = "^[a-zA-Z]+$";

        if (!Regex.IsMatch(enteredName, pattern))
        {
            // Display an error message or take appropriate action
            MessageBox.Show("Invalid name. Please use only English letters.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            // Optionally clear the textbox or take other actions
            User_Full_name.Text = string.Empty;
            return;
        }
        //contct
        string input = Contactnum.Text;

        // Check if the input contains only numeric characters
        if (!string.IsNullOrEmpty(input) && !input.All(char.IsDigit))
        {
            MessageBox.Show("Please enter only numeric characters for the contact number.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            // Remove non-numeric characters from the input
            Contactnum.Text = new string(input.Where(char.IsDigit).ToArray());
            Contactnum.SelectionStart = Contactnum.Text.Length; // Set cursor at the end
            return;
        }

        // Check if the length is less than 11 or more than 11 characters
        if (!string.IsNullOrEmpty(input) && (input.Length < 11 || input.Length > 11))
```

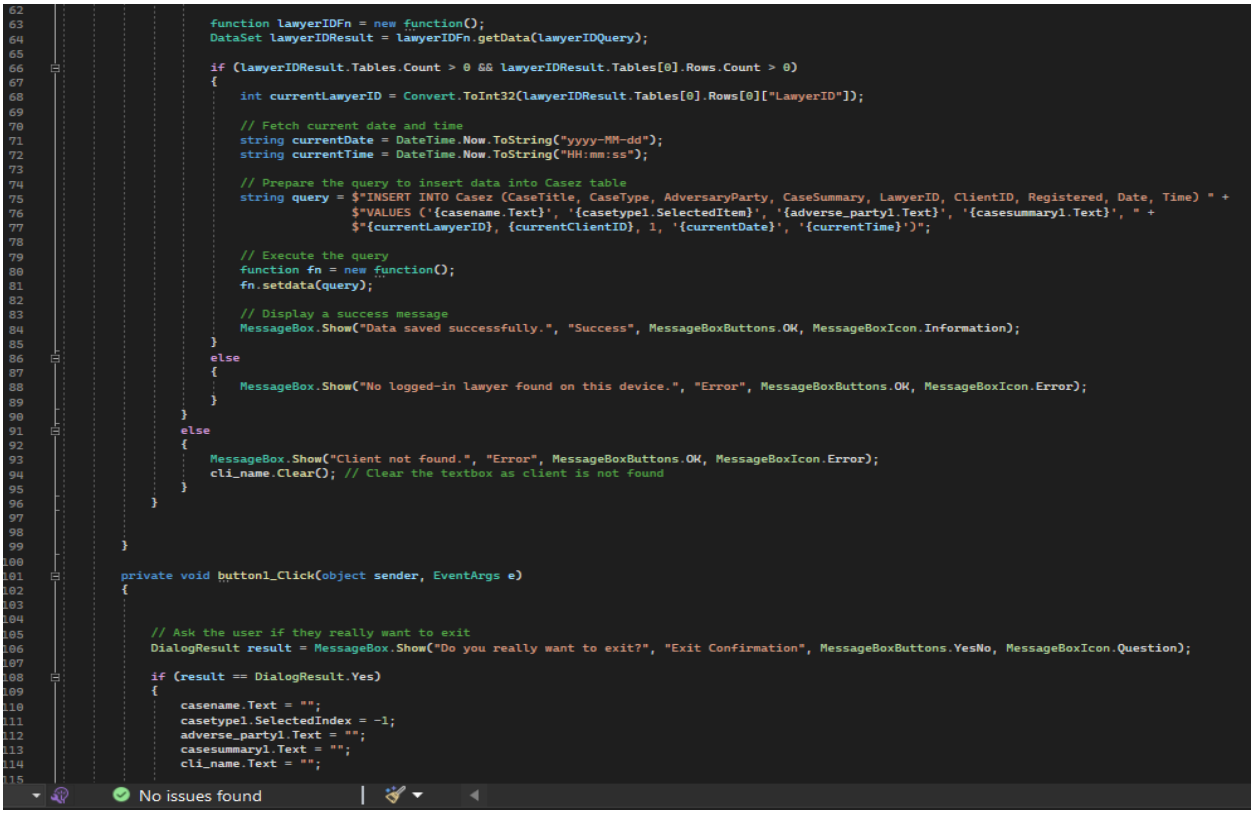
```
        {
            MessageBox.Show("Contact number should be exactly 11 digits.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            // Clear the input
            Contactnum.Clear();
            return;
        }
        else
        {
            string userType = User_type.SelectedItem.ToString();
            if (userType == "Lawyer")
            {
                query = "insert into lawyer (LawyerName, Contact, BarCode, Gender, Experience, Username, Password) values('" + User_Full_name.Text + "' , '" + Contactnum.Text + "' , NULL ,NULL,NULL, '" + email.Text + "','" + passwo
                fn.setdata(query);
                MessageBox.Show("Data saved successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                Login l = new Login();
                l.Show();
                this.Hide();
            }

            else if (userType == "Client")
            {
                query = "insert into Client (ClientName,Contact,UserName,Password) values('" + User_Full_name.Text + "' , '" + Contactnum.Text + "' , '" + email.Text + "','" + password1.Text + "')";
                fn.setdata(query);
                MessageBox.Show("Data saved successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                Login_Client l = new Login_Client();
                l.Show();
                this.Hide();
            }



        }
    }
    private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        if (string.IsNullOrWhiteSpace(Us
        {
            MessageBox.Show("Please fill user type before continuing.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else
        {
            string userType = User_type.SelectedItem.ToString();
            if (userType == "Lawyer")
            {
                Login l = new Login();
                l.Show();
                this.Hide();
            }

            else if (userType == "Client")
```

class System.Object
Supports all classes in the .NET class hierarchy and provides low-level services to derived classes. This is the ultimate base class of all .NET classes; it is the root of the type hierarchy.
GitHub Examples and Documentation (Ctrl+Alt+)

```
            }

            else
            {
                string userType = User_type.SelectedItem.ToString();
                if (userType == "Lawyer")
                {
                    Login l = new Login();
                    l.Show();
                    this.Hide();
                }

                else if (userType == "Client")
                {
                    Login_Client l = new Login_Client();
                    l.Show();
                    this.Hide();
                }

            }

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 f = new Form1();
            f.Show();
            this.Hide();
        }
    }

}
```

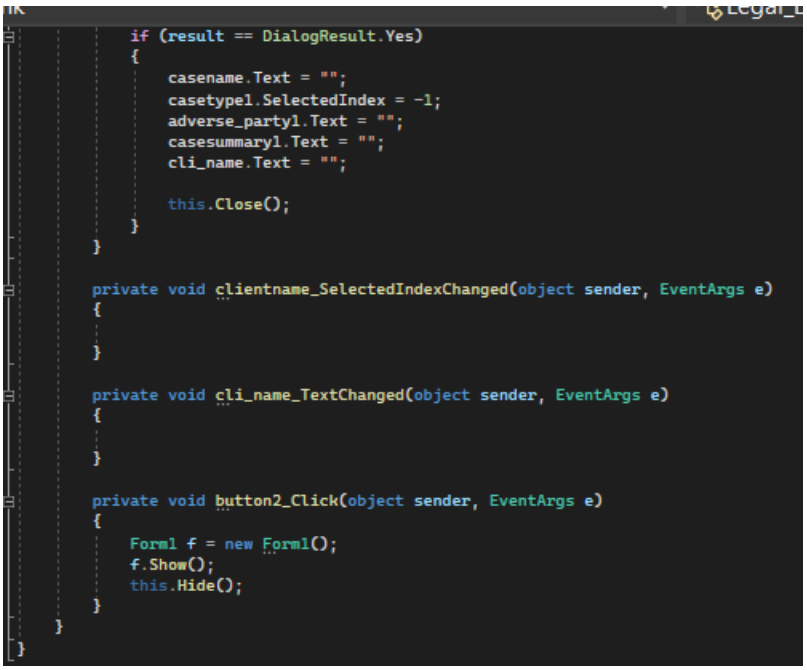
4.5 New Cases

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Legal_Link
{
    public partial class cases : Form
    {
        public cases()
        {
            InitializeComponent();
        }


        static string GetDeviceName()
        {
            try
            {
                string hostName = System.Net.Dns.GetHostName();
                return hostName;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error getting device name: " + ex.Message);
                return string.Empty;
            }
        }
        private void save_Click(object sender, EventArgs e)
        {
            // Check if any of the textboxes is empty
            if (string.IsNullOrEmpty(casename.Text) ||
                casetype1.SelectedIndex == -1 || // Assuming casetype1 is a ComboBox
                string.IsNullOrEmpty(adverse_party1.Text) ||
                string.IsNullOrEmpty(casesummary1.Text) ||
                string.IsNullOrEmpty(cli_name.Text))
            {
                // Display an error message
                MessageBox.Show("Please fill in all fields before saving or Enter Valid Data.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error
            }
            else
            {
                // Fetch current client ID based on the client name
                string clientName = cli_name.Text;
                string clientIDQuery = $"SELECT TOP 1 ClientID FROM Client WHERE ClientName = '{clientName}'";

                function clientIDFn = new function();
                DataSet clientIDResult = clientIDFn.getData(clientIDQuery);

                if (clientIDResult.Tables.Count > 0 && clientIDResult.Tables[0].Rows.Count > 0)
                {
                    int currentClientID = Convert.ToInt32(clientIDResult.Tables[0].Rows[0]["ClientID"]);
```



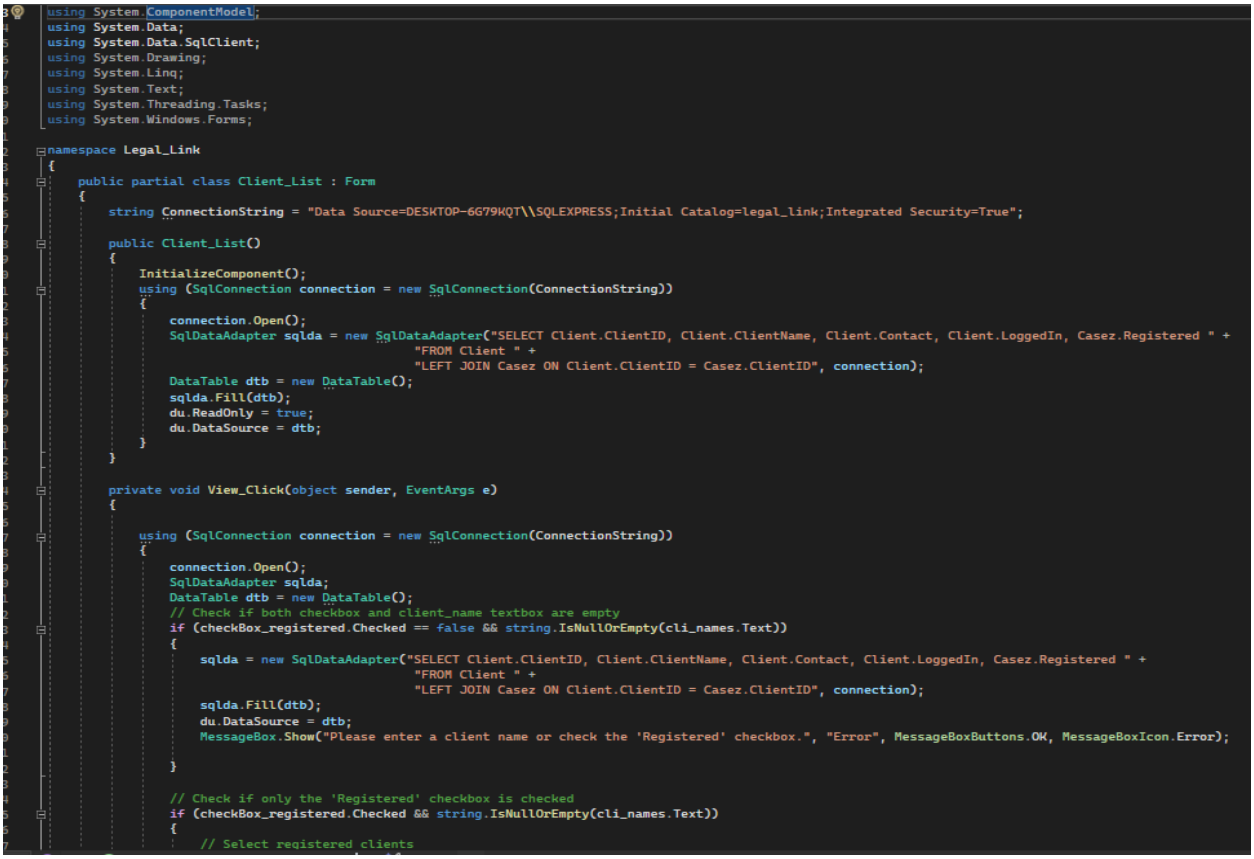
```
            function lawyerIDFn = new function();
            DataSet lawyerIDResult = lawyerIDFn.getData(lawyerIDQuery);

            if (lawyerIDResult.Tables.Count > 0 && lawyerIDResult.Tables[0].Rows.Count > 0)
            {
                int currentLawyerID = Convert.ToInt32(lawyerIDResult.Tables[0].Rows[0]["LawyerID"]);

                // Fetch current date and time
                string currentDate = DateTime.Now.ToString("yyyy-MM-dd");
                string currentTime = DateTime.Now.ToString("HH:mm:ss");

                // Prepare the query to insert data into Casez table
                string query = $"INSERT INTO Casez (CaseTitle, CaseType, AdversaryParty, CaseSummary, LawyerID, ClientID, Registered, Date, Time) " +
                               $"VALUES ('{casename.Text}', '{casetype1.SelectedItem}', '{adverse_party1.Text}', '{casesummary1.Text}', " +
                               $"{currentLawyerID}, {currentClientID}, 1, '{currentDate}', '{currentTime}')";

                // Execute the query
                function fn = new function();
                fn.setdata(query);

                // Display a success message
                MessageBox.Show("Data saved successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("No logged-in lawyer found on this device.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show("Client not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            cli_name.Clear(); // Clear the textbox as client is not found
        }
    }


}

private void button1_Click(object sender, EventArgs e)
{


    // Ask the user if they really want to exit
    DialogResult result = MessageBox.Show("Do you really want to exit?", "Exit Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        casename.Text = "";
        casetype1.SelectedIndex = -1;
        adverse_party1.Text = "";
        casesummary1.Text = "";
        cli_name.Text = "";
```

No issues found

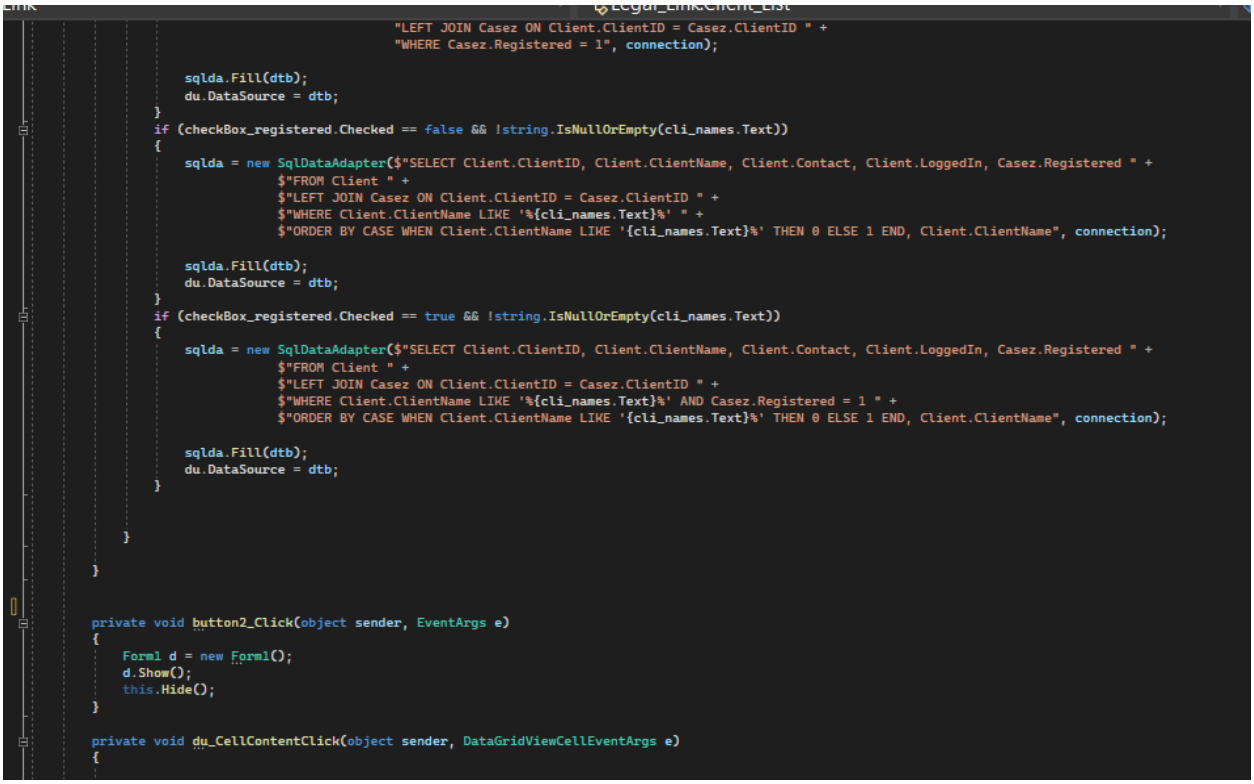


```
        if (result == DialogResult.Yes)
        {
            casename.Text = "";
            casetype1.SelectedIndex = -1;
            adverse_party1.Text = "";
            casesummary1.Text = "";
            cli_name.Text = "";

            this.Close();
        }
    }

    private void clientname_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void cli_name_TextChanged(object sender, EventArgs e)
    {

    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form1 f = new Form1();
        f.Show();
        this.Hide();
    }
}
}
```

4.6 Client List



```
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Legal_Link
{
    public partial class Client_List : Form
    {
        string ConnectionString = "Data Source=DESKTOP-6G79KQT\\SQLEXPRESS;Initial Catalog=legal_link;Integrated Security=True";

        public Client_List()
        {
            InitializeComponent();
            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();
                SqlDataAdapter sqlda = new SqlDataAdapter("SELECT Client.ClientID, Client.ClientName, Client.Contact, Client.LoggedIn, Casez.Registered " +
                                                "FROM Client " +
                                                "LEFT JOIN Casez ON Client.ClientID = Casez.ClientID", connection);
                DataTable dtb = new DataTable();
                sqlda.Fill(dtb);
                du.ReadOnly = true;
                du.DataSource = dtb;
            }
        }

        private void View_Click(object sender, EventArgs e)
        {

            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();
                SqlDataAdapter sqlda;
                DataTable dtb = new DataTable();
                // Check if both checkbox and client_name textbox are empty
                if (checkBox_registered.Checked == false && string.IsNullOrEmpty(cli_names.Text))
                {
                    sqlda = new SqlDataAdapter("SELECT Client.ClientID, Client.ClientName, Client.Contact, Client.LoggedIn, Casez.Registered " +
                                                "FROM Client " +
                                                "LEFT JOIN Casez ON Client.ClientID = Casez.ClientID", connection);
                    sqlda.Fill(dtb);
                    du.DataSource = dtb;
                    MessageBox.Show("Please enter a client name or check the 'Registered' checkbox.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

                // Check if only the 'Registered' checkbox is checked
                if (checkBox_registered.Checked && string.IsNullOrEmpty(cli_names.Text))
                {
                    // Select registered clients
```



```
                                            "LEFT JOIN Casez ON Client.ClientID = Casez.ClientID " +
                                            "WHERE Casez.Registered = 1", connection);

                    sqlda.Fill(dtb);
                    du.DataSource = dtb;
                }
                if (checkBox_registered.Checked == false && !string.IsNullOrEmpty(cli_names.Text))
                {
                    sqlda = new SqlDataAdapter($"SELECT Client.ClientID, Client.ClientName, Client.Contact, Client.LoggedIn, Casez.Registered " +
                                $"FROM Client " +
                                $"LEFT JOIN Casez ON Client.ClientID = Casez.ClientID " +
                                $"WHERE Client.ClientName LIKE '%{cli_names.Text}%' " +
                                $"ORDER BY CASE WHEN Client.ClientName LIKE '{cli_names.Text}%' THEN 0 ELSE 1 END, Client.ClientName", connection);

                    sqlda.Fill(dtb);
                    du.DataSource = dtb;
                }
                if (checkBox_registered.Checked == true && !string.IsNullOrEmpty(cli_names.Text))
                {
                    sqlda = new SqlDataAdapter($"SELECT Client.ClientID, Client.ClientName, Client.Contact, Client.LoggedIn, Casez.Registered " +
                                $"FROM Client " +
                                $"LEFT JOIN Casez ON Client.ClientID = Casez.ClientID " +
                                $"WHERE Client.ClientName LIKE '%{cli_names.Text}%' AND Casez.Registered = 1 " +
                                $"ORDER BY CASE WHEN Client.ClientName LIKE '{cli_names.Text}%' THEN 0 ELSE 1 END, Client.ClientName", connection);

                    sqlda.Fill(dtb);
                    du.DataSource = dtb;
                }


            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            Form1 d = new Form1();
            d.Show();
            this.Hide();
        }

        private void du_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
```

4.7 Documents

```
using System.Windows.Forms;

namespace Legal_Link
{
    public partial class Document : Form
    {
        string ConnectionString = "Data Source=DESKTOP-6G79KQT\\SQLEXPRESS;Initial Catalog=legal_link;Integrated Security=True";
        public Document()
        {
            InitializeComponent();
            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();
                SqlDataAdapter sqlda = new SqlDataAdapter("SELECT DocumentID, DocumentTitle, Content, UploadDate, UploadTime, LawyerName FROM Documents", connection);
                DataTable dtb = new DataTable();
                sqlda.Fill(dtb);
                du.ReadOnly = true;
                du.DataSource = dtb;
            }
        }


        private void Upload_Click(object sender, EventArgs e)
        {
            //upload btn
            if (string.IsNullOrEmpty(title.Text))
            {
                MessageBox.Show("Title cannot be empty. Please fill in the title.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Check if title exists in the Casez table
            string titleCheckQuery = $"SELECT TOP 1 CaseID FROM Casez WHERE CaseTitle = '{title.Text}'";
            function checkTitleFn = new function();
            DataSet titleCheckResult = checkTitleFn.getData(titleCheckQuery);

            int caseID = -1;

            if (titleCheckResult.Tables.Count > 0 && titleCheckResult.Tables[0].Rows.Count > 0)
            {
                // Case with the provided title already exists, retrieve its CaseID
                caseID = Convert.ToInt32(titleCheckResult.Tables[0].Rows[0]["CaseID"]);
            }
            else
            {
                MessageBox.Show("A case with the provided title does not exist. Please create the case first.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (string.IsNullOrEmpty(lawyer_name.Text))
            {
                MessageBox.Show("Lawyer Name cannot be empty. Please fill in the lawyer name.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
```

```
                MessageBox.Show("A case with the provided title does not exist. Please create the case first.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (string.IsNullOrEmpty(lawyer_name.Text))
            {
                MessageBox.Show("Lawyer Name cannot be empty. Please fill in the lawyer name.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            string lawyerCheckQuery = $"SELECT TOP 1 1 FROM Lawyer WHERE LawyerName = '{lawyer_name.Text}'";
            function checkFn = new function();
            DataSet lawyerCheckResult = checkFn.getData(lawyerCheckQuery);

            if (lawyerCheckResult.Tables.Count == 0 || lawyerCheckResult.Tables[0].Rows.Count == 0)
            {
                MessageBox.Show("Lawyer with the provided name does not exist.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (string.IsNullOrEmpty(content.Text))
            {
                MessageBox.Show("Content cannot be empty. Please fill in the content.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            // Fetch current date and time
            DateTime currentDate = DateTime.Now;
            TimeSpan currentTime = DateTime.Now.TimeOfDay;

            // Insert values into the Documents table
            string insertQuery = $"INSERT INTO Documents (DocumentTitle, Content, UploadDate, UploadTime, LawyerName, CaseID) " +
                                 $"VALUES ('{title.Text}', '{content.Text}', '{currentDate.ToString("yyyy-MM-dd")}', '{currentTime}', '{lawyer_name.Text}', {caseID})";

            function insertFn = new function();
            insertFn.setdata(insertQuery);

            // Display success message
            MessageBox.Show("Document uploaded successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);


        }

        private void du_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 dr = new Form1();
            dr.Show();
            this.Close();
        }
    }
```

4.8 Billings

```
namespace Legal_Link
{
    public partial class Billing : Form
    {
        string ConnectionString = "Data Source=DESKTOP-6G79KQT\\SQLEXPRESS;Initial Catalog=legal_link;Integrated Security=True";

        public Billing()
        {
            InitializeComponent();
            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();

                SqlDataAdapter sqlda = new SqlDataAdapter("SELECT " +
                    "Lawyer.LawyerName, " +
                    "Client.ClientName, " +
                    "Payments.Amount, " +
                    "Payments.PaymentType, " +
                    "Payments.PaymentDate, " +
                    "Payments.Status " +
                    "FROM Payments " +
                    "INNER JOIN Lawyer ON Payments.LawyerID = Lawyer.LawyerID " +
                    "INNER JOIN Client ON Payments.ClientID = Client.ClientID", connection);

                DataTable dtb = new DataTable();
                sqlda.Fill(dtb);
                du.ReadOnly = true;
                du.DataSource = dtb;
            }
        }

        private void du_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 f = new Form1();
            f.Show();
            this.Close();
        }

        private void title_TextChanged(object sender, EventArgs e)
        {
            //lawyer name text box
        }

        private void client_name_TextChanged(object sender, EventArgs e)
        {
            //client name btn text box
        }
```

```
        private void check_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(title.Text) && string.IsNullOrWhiteSpace(client_name.Text) && string.IsNullOrWhiteSpace(date.Text))
            {
                MessageBox.Show("Something missing", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Check if any of lawyer name, client name, or date textboxes are filled
            if (!string.IsNullOrEmpty(title.Text) || !string.IsNullOrEmpty(client_name.Text) || !string.IsNullOrEmpty(date.Text))
            {
                // Construct the query based on the filled textboxes
                string query = $"SELECT Payments.PaymentID, Payments.Amount, Payments.PaymentType, " +
                               $"Lawyer.LawyerName, Client.ClientName, Payments.PaymentTime, Payments.PaymentDate, Payments.Status " +
                               $"FROM Payments " +
                               $"INNER JOIN Lawyer ON Payments.LawyerID = Lawyer.LawyerID " +
                               $"INNER JOIN Client ON Payments.ClientID = Client.ClientID " +
                               $"WHERE 1 = 1";

                if (!string.IsNullOrEmpty(title.Text))
                {
                    query += $" AND Lawyer.LawyerName LIKE '%{title.Text}%'";
                }

                if (!string.IsNullOrEmpty(client_name.Text))
                {
                    query += $" AND Client.ClientName LIKE '%{client_name.Text}%'";
                }

                if (!string.IsNullOrEmpty(date.Text))
                {
                    query += $" AND Payments.PaymentDate = '{date.Text}'";
                }

                // Use the constructed query to fetch and display data in the data grid (du)
                using (SqlConnection connection = new SqlConnection(ConnectionString))
                {
                    connection.Open();
                    SqlDataAdapter sqlda = new SqlDataAdapter(query, connection);
                    DataTable dtb = new DataTable();
                    sqlda.Fill(dtb);
                    du.DataSource = dtb;
                }
            }
            else
            {
                // Display an error message if none of the textboxes are filled
                MessageBox.Show("Please enter at least one search criteria.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

4.9 Appointments

```
namespace Legal_Link
{
    public partial class Appointment : Form
    {
        private function sqlFunction;
        public Appointment()
        {
            string ConnectionString = "Data Source=DESKTOP-6G79KQT\\SQLEXPRESS;Initial Catalog=legal_link;Integrated Security=True";

            InitializeComponent();
            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();

                SqlDataAdapter sqlda = new SqlDataAdapter("SELECT " +
                    "Lawyer.LawyerName AS 'Lawyer Name', " +
                    "Client.ClientName AS 'Client Name', " +
                    "Ap.appointment_date AS 'Appointment Date', " +
                    "Ap.appointment_time AS 'Appointment Time', " +
                    "Ap.status AS 'Status' " +
                    "FROM Ap " +
                    "INNER JOIN Lawyer ON Ap.LawyerID = Lawyer.LawyerID " +
                    "INNER JOIN Client ON Ap.ClientID = Client.ClientID", connection);

                DataTable dtb = new DataTable();
                sqlda.Fill(dtb);
                dataGridView1.ReadOnly = true;
                dataGridView1.DataSource = dtb;
            }

        }

        private void Appointment_Load(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 form1 = new Form1();
            form1.Show();
            this.Hide();
        }

        private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }

        private void lawname_TextChanged(object sender, EventArgs e)
        {
            //lawyer name
```

```
        }

        private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }

        private void lawname_TextChanged(object sender, EventArgs e)
        {
            //lawyer name
        }

        private void client_name_TextChanged(object sender, EventArgs e)
        {
            //client name
        }

        private void date_TextChanged(object sender, EventArgs e)
        {
            //date
        }

        private void bok_appoint_Click(object sender, EventArgs e)
        {
            // Validate that all text boxes are filled
        if (string.IsNullOrWhiteSpace(lawname.Text) || string.IsNullOrWhiteSpace(client_name.Text) || string.IsNullOrWhiteSpace(date.Text))
        {
            MessageBox.Show("Please fill in all fields before booking the appointment.", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }



        MessageBox.Show("Appointment booked successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);


        }
    }
}
```

Chapter 5

Experimentation and testing

Functional vs. Non-functional Testing

The objective of employing diverse testing methodologies throughout your development lifecycle is to ensure the effective operation of your software across various environments and diverse platforms. This categorization typically revolves around the differentiation between functional and non-functional testing. Functional testing revolves around validating the application against the specified business requirements, encompassing all test types intended to ensure each component of the software functions as anticipated based on the use cases outlined by the design team or business analyst. These testing methodologies are typically executed sequentially and include:

- Unit testing
- Integration testing
- System testing
- Acceptance testing

Non-functional testing methods incorporate all test types focused on the operational aspects of a piece of software. These include:

- Performance testing
- Security testing
- Usability testing
- Compatibility testing

1.1 Unit Testing

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

1.1.1 Sign-in Testing

The module sign in screen is used to sign in into the system if the user is registered into the system.The components of sign in screen are:

- Username Text Field
- Password Text Field
- Log in Button
- Continue Sign-up?

1.1.1.0 Username Text Field

Some of the test values are entered into the username text box component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	post	Email must be rejected.	Email is rejected.	Pass
2.	(empty)	Email must be rejected.	Email is rejected.	Pass
3.	Messam1@gmail.com	Email must be Accepted.	Email is accepted.	Pass
4.	@gmail	Email must be rejected.	Email is rejected.	Pass

Table 12: Username Text Field

1.1.1.0 Password Text Field

Some of the test values are entered into the password text box component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Mess	Password must be accepted.	Password is accepted.	Pass
2.	(empty)	Password must be rejected.	Password is rejected.	Pass
3.	12	Password must be rejected.	Password is accepted.	Pass
4.	Qwertyuiop123456987	Password must be rejected.	Password is rejected.	Pass

Table 13: Password Text Field

1.1.1.0 Sign-in Button

On clicking sign in button user must be logged in by providing valid information.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Username and password are empty.	Clicking sign-in button must display an error.	Clicking sign-in button displays an error.	Pass
2.	(empty)	Clicking sign-in button must display an error.	Clicking sign-in button displays an error.	Pass
3.	Username and Password exist.	User must be logged in.	User is logged in.	Pass
4.	Invalid values entered.	Clicking sign-in button must display an error.	Clicking sign-in button displays an error.	Pass
5.	User already logged in.	Error message must be displayed.	Error message is displayed.	Pass

Table 14: Sign-in button

1.1.1.0 Continue Sign-up

User must be navigate to sign up screen.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click on sign-up	User must be navigate to sign-up.	User is navigated to sign-up page.	Pass

Table 15: Continue Sign up

1.1.2 Sign up Screen Testing

The module sign-up screen is used to sign up into the system with provided valid name, email and password user can create his/her account into the system.
The components of sign-up screen are:

- User Type Drop Down
- Name Text Field
- Email Text Field
- Password Text Field
- Contact Number Text Field
- Sign up button
- Already have an account?

1.1.2.1.0 User Type Drop Down

Some of the test values are entered into the user type drop down component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Ywuqhdb	User Type must be rejected.	User type is rejected.	Pass
2.	(empty)	User Type must be rejected.	User Type is rejected.	Pass
3.	Lawyer	User Type must be accepted.	User Type is accepted.	Pass
4.	Client	User Type must be accepted.	User Type is accepted.	Pass

Table 16: UserType Dropdown

1.1.2.2.0 Name Text Field

Some of the test values are entered into the Name text field down component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Mom5	name must be rejected.	name is accepted.	Fail
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	Messam	name must be accepted.	name is accepted.	Pass
4.	peedios	name must be accepted.	name is accepted.	Pass

Table 17:Name Text Field

1.1.2.2.1 Name Text Field

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Mom5	name must be rejected.	name is rejected.	Pass

1.1.3.0 Email Text Field

Some of the test values are entered into the email text box component to check weather it accepts the valid input data form the user and rejects invalid data and register the email.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	post	Email must be rejected.	Email is rejected.	Pass
2.	(empty)	Email must be rejected.	Email is rejected.	Pass
3.	me1@gmail.com	Email must be Accepted.	Email is accepted.	Pass
4.	@gmail	Email must be rejected.	Email is rejected.	Pass

Table 18: Email Text Field

1.1.4.0 Password Text Field

Some of the test values are entered into the email text box component to check weather it accepts the valid input data form the user and rejects invalid data and register the email.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Smeks1	Password must be accepted.	Password is accepted.	Pass
2.	(empty)	Password must be rejected.	Password is rejected.	Pass
3.	12	Password must be rejected.	Password is rejected.	Pass
4.	Qwertyuiop123456987	Password must be rejected.	Password is rejected.	Pass

Table 19:Sign up Password Text Field

1.1.5.0 Contact Text Field

Some of the test values are entered into the email text box component to test it.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	12345678901	Contact must be accepted.	Contact is accepted.	Pass
2.	(empty)	Contact must be rejected.	Contact is rejected.	Pass
3.	12	Contact must be rejected.	Contact is accepted.	Fail
4.	Qwer6987	Contact must be rejected.	Contact is accepted.	Fail

Table 20: Contact Text Field

1.1.5.1 Contact Text Field

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	12	Contact must be rejected.	Contact is rejected.	Pass
2.	Qwer6987	Contact must be rejected.	Contact is rejected.	Pass

1.1.6.0 Sign-up Button

On click on sign-up user must be able to create a new account by providing valid information required to create the account.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Usertype, Name, Email, Contact are empty.	Clicking sign-up button must display an error.	Clicking sign-up button displays an error.	Pass
2.	(empty)	Clicking sign-up button must display an error.	Clicking sign-up button displays an error.	Pass
3.	All Fields exists	User must jump to sign-in page.	User is jumped to sign-in page.	Pass
4.	Invalid values entered.	Clicking sign-up button must display an error.	Clicking sign-up button displays an error.	Pass
5.	User already logged in.	Error message must be displayed.	Error message is displayed.	Pass

Table 21: Sign up button

1.1.2.7.0 Already have an account?

User must be navigate to sign in screen.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click on sign-in	User must be navigate to sign-in.	User is navigated to sign-in page.	Pass

Table 22: Already Have an account?

1.1.3 Home Screen Testing

In Home Screen module user must be able to login, view about legal link system and navigate to see his bookings, cases, lists, payments and Documents
The components of home screen are:
• Login Navigation
• New Cases Navigation
• Client List Navigation
• Documents Navigation
• Billings Navigation
• Appointments Navigation
• Log out button

1.1.3.1.0 Login Navigation

User must be able to navigate to Login screen to see sign in to system.

.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click login button	User must navigate to sign-in screen.	User is navigated to sign-in page.	Pass

Table 23: Login Navigation

1.1.3.2.0 New Cases Navigation

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click new cases button	User must navigate to new cases screen.	User is navigated to new cases page.	Pass

Table 24: New cases navigation

1.1.3.3.0 Client list Navigation

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click client list button	User must navigate to client list screen.	User is navigated to client list page.	Pass

Table 25: Client list navigation

1.1.3.4.0 Documents Navigation

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click documents button	User must navigate to documents screen.	User is navigated to documents page.	Pass

Table 26: Documents Navigation

1.1.3.5.0 Billings Navigation

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click billings button	User must navigate to billings screen.	User is navigated to billings page.	Pass

Table 27: Billings Navigation

1.1.3.6.0 Appointments Navigation

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click Appointments button	User must navigate to Appointments screen.	User is navigated to Appointments page.	Pass

Table 28: Appointment Navigation

1.1.3.7.0 Logout Button

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click logout button	User must logout.	User is logged out.	Pass

Table 29:Logout Button

1.1.4 New Cases Screen Testing

In New cases Screen module user must be able to register a new case and cancel the case.
The components of new case screen are:

- Client name text box
- Case name text box
- Case type drop down
- Case Summary text box
- Adversary Party name text box
- Register button
- Cancel button

1.1.4.1.0 Client name text box

Some of the test values are entered into the client name text field down component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Registered client name entered	client name must be accepted.	client name is accepted.	Pass
2.	(empty)	client name must be rejected.	client name is rejected.	Pass
3.	Un-Registered client name entered	client name must be rejected.	client name is accepted.	Fail
4.	Invalid characters entered.	client name must be rejected.	client name is accepted.	Fail

Table 30: Client name text box

1.1.4.1.1 Client name text box

Sr. No	Test Data	Expected Result	Actual Result	Test Result
3.	Un-Registered client name entered	client name must be rejected.	client name is rejected.	Pass
4.	Invalid characters entered.	client name must be rejected.	client name is rejected.	Pass

1.1.4.2.0 Case Name text box

Some of the test values are entered into the case name text field down component to check weather it accepts the valid input data form the user and rejects invalid data.

Table 31:Case Name Text Box

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Registered case name entered	case name must be accepted.	Case name is accepted.	Pass
2.	(empty)	case name must be rejected.	case name is rejected.	Pass
3.	Un-Registered case name entered	case name must be rejected.	case name is rejected.	Pass
4.	Invalid characters entered.	case name must be rejected.	case name is rejected.	Pass

1.1.4.3.0 Case type text box

Some of the test values are entered into the case type drop down field down component to check
weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	given case type entered from drop down	case type must be accepted.	Case type is accepted.	Pass
2.	(empty)	case type must be rejected.	case type is rejected.	Pass
3.	Random case type entered	case type must be rejected.	case type is rejected.	Pass
4.	Invalid characters entered.	case type must be rejected.	case type is rejected.	Pass

1.1.4.4.0 Case Summary text box

Some of the test values are entered into the case summary component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Valid Case summary is Entered	case summary must be accepted.	Case summary is accepted.	Pass
2.	(empty)	Case summary must be rejected.	Case summary is rejected.	Pass
3.	6278egdhsj	Case summary must be rejected.	Case summary is rejected.	Pass

1.1.4.5.0 Adversary Party name text box

Some of the test values are entered into the adversary party text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Messam	Adversary party name must be accepted.	Adversary party name is accepted.	Pass
2.	(empty)	Adversary party name must be rejected.	Adversary party name is rejected.	Pass
3.	30dbge!	Adversary party name must be rejected.	Adversary party name is accepted.	Fail

1.1.4.5.1 Adversary Party name text box

Some of the test values are entered *again* into the adversary party text name component to check weather it rejects the invalid input data form the user.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	30dbge!	Adversary party name must be rejected.	Adversary party name is rejected.	Pass

1.1.4.6.0 Register button

Some of the test values are entered into the register button component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	All fields exists	Clicking register button will show success message.	Clicking register button shows success message.	Pass
2.	(empty)	Clicking register button must display an error.	Clicking register button displays an error.	Pass

1.1.4.7.0 Cancel button

Some of the test values are entered into the cancel button component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Click cancel button	Clicking cancel button must show an decision making prompt.	Clicking cancel button shows an decision making prompt.	Pass

1.1.5 Client list Screen Testing

In client list Screen module user must be able see client lists,and sort it.
The components of client list screen are:

- Client list data grid
- is_registered check box
- Client name
- Sort Button

1.1.5.1.0 Client list data grid

Some of the test values are entered into the client list data grid component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the client list data grid is displayed upon opening the client list page.	The client list data grid must be visible and populated with client data.	The client list data grid is visible and populated with client data.	Pass
2.	Verify that the client list data grid displays the correct client information, including client name, email address, phone number.	The client list data grid must display accurate and up-to-date client information.	The client list data grid displays accurate and up-to-date client information.	Pass

1.1.5.2.0 is_registered check box

Some of the test values are entered into the is_registered check box component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the is_registered checkbox is initially unchecked.	The is_registered checkbox must be unchecked by default.	The is_registered checkbox is unchecked by default.	Pass
2.	Verify that clicking the sort button enables the is_registered checkbox	Clicking the sort button must enable and make the is_registered checkbox interactive.	Clicking the sort button must enable and make the is_registered checkbox interactive.	

1.1.5.3.0 Client name text box

Some of the test values are entered into the client name text field down component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Registered client name entered	client name must be accepted.	client name is accepted.	Pass
2.	(empty)	client name must be rejected.	client name is rejected.	Pass
3.	Un-Registered client name entered	client name must be accepted.	client name is accepted.	Pass
4.	Yrg47e	client name must be rejected.	client name is rejected.	Pass

1.1.6 Document Screen Testing

In Document Screen module user must be able to upload a Document of the Case.
The components of client list screen are:

- Name
- Title
- Content
- Document list data grid

1.1.6.1.0 Your Name text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Messam	Name must be accepted.	Name is accepted.	Pass
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	30dbge!	Name must be rejected.	Name is rejected.	Pass

1.1.6.2.0 Title Text Field

Some of the test values are entered into the username text box component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Hello	Title must be rejected.	Title is rejected.	Pass
2.	(empty)	Title must be rejected.	Title is rejected.	Pass
3.	Criminal Case	Title must be Accepted.	Title is accepted.	Pass
4.	@	Title must be rejected.	Title is rejected.	Pass

1.1.6.3.0 Content Text Field

Some of the test values are entered into the content text feild component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Ghgysuwjsn syuw	Content must be accepted.	Content is accepted.	Pass
2.	(empty)	Context must be rejected.	Context is rejected.	Pass
3.	6278egdhsj	Context must be rejected.	Context is rejected.	Pass

1.1.6.4.0 Document list data grid

Some of the test values are entered into the Document list data grid component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the Document list data grid is displayed upon opening the Document page.	The Document list data grid should be visible and populated with Document data.	The Document list data grid is visible and populated with Document data.	Pass
2.	Verify that the Document list data grid displays the correct Document information, including Case name, Title, Context.	The Document list data grid should display accurate and up-to-date Document information.	The Document list data grid displays accurate and up-to-date Document information.	Pass

1.1.6.5.0 Upload button

Some of the test values are entered into the Upload button component to check whether it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	All fields exists	Clicking Upload button will show success message and show the data in grid veiw.	Clicking Upload button shows success message and show the data in grid veiw.	Pass
2.	(empty)	Clicking Upload button must display an error.	Clicking Upload button displays an error.	Pass

1.1.7 Billing Screen Testing

In Billing Screen module user must be able to make payment.
The components of client list screen are:

- Lawyer Name
- Client Name
- Specific Date
- Document list data grid
- View Date
- Check Button

1.1.7.1.0 Lawyer Name text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Messam	Name must be accepted.	Name is accepted.	Pass
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	30dbge!	Name must be rejected.	Name is rejected.	Pass

1.1.7.2.0 Client Name text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Muneeb	Name must be accepted.	Name is accepted.	Pass
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	30dbge!	Name must be rejected.	Name is rejected.	Pass

1.1.7.2.0 Specific Date text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	12/11/2023	Date must be accepted.	Date is accepted.	Pass
2.	(empty)	Date must be rejected.	Date is rejected.	Pass
3.	30dbge!	Date must be rejected.	Date is rejected.	Pass

1.1.7.3.0 Payment list data grid

Some of the test values are entered into the Payment list data grid component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the Payment list data grid is displayed upon opening the Billing page.	The Payment list data grid should be visible and populated with payment data.	The Payment list data grid is visible and populated with Payment data.	Pass
2.	Verify that the Payment list data grid displays the correct Payment information, including Lawyer name, Client Name, Date, Amount paid.	The Payment list data grid should display accurate and up-to-date Payment information.	The Payment list data grid displays accurate and up-to-date Payment information.	Pass

1.1.7.4.0 Check button

Some of the test values are entered into the check button component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	All fields exists	Clicking check button will show success message and show the data in grid veiw.	Clicking check button shows success message and show the data in grid veiw.	Pass
2.	(empty)	Clicking check button must display an error.	Clicking check button displays an error.	Pass

1.1.7.4.0 Print button

Click the print button component to check whether it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	All fields exists	Clicking print button will show success message.	Clicking print button shows success message.	Pass
2.	(empty)	Clicking print button must display an error.	Clicking print button displays an error.	Pass

1.1.7 Appointments Screen Testing

In Appointment Screen module user must be able to make Appointment.
The components of client list screen are:

- Lawyer Name
- Client Name
- Specific Date
- Document list data grid
- Book Appointment Button

1.1.7.1.0 Lawyer Name text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Messam	Name must be accepted.	Name is accepted.	Pass
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	30dbge!	Name must be rejected.	Name is rejected.	Pass

1.1.7.2.0 Client Name text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Muneeb	Name must be accepted.	Name is accepted.	Pass
2.	(empty)	Name must be rejected.	Name is rejected.	Pass
3.	30dbge!	Name must be rejected.	Name is rejected.	Pass

1.1.7.2.0 Specific Date text box

Some of the test values are entered into the Your name text name component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	12/11/2023	Date must be accepted.	Date is accepted.	Pass
2.	(empty)	Date must be rejected.	Date is rejected.	Pass
3.	30dbge!	Date must be rejected.	Date is rejected.	Pass

1.1.7.3.0 Appointment list data grid

Some of the test values are entered into the Appointment list data grid component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the Appointment list data grid is displayed upon opening the Appointment page.	The Appointment list data grid should be visible and populated with Appointments data.	The Appointment list data grid is not visible and populated with Appointment data.	Fail
2.	Verify that the Appointment list data grid displays the correct Appointment information, including Lawyer name, Client Name, Date.	The Appointment list data grid should display accurate and up-to-date Appointment information.	The Appointment list data grid displays accurate and up-to-date Appointment information.	Pass

1.1.7.3.1 Appointment list data grid

Some of the test values are entered again into the Appointment list data grid component.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that the Appointment list data grid is displayed upon opening the Appointment page.	The Appointment list data grid should be visible and populated with Appointments data.	The Appointment list data grid is visible and populated with Appointment data.	Pass

1.1.7.4.0 Book Appointment button

Some of the test values are entered into the Book Appointment button component to check weather it accepts the valid input data form the user and rejects invalid data.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	All fields exists	Clicking Book Appointment button will show success message.	Clicking Book Appointment button shows success message.	Pass
2.	(empty)	Clicking Book Appointment button must display an error.	Clicking Book Appointment button displays an error.	Pass

1.2 Integration Testing

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities.

These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

1.2.1 Sign in and Sign up

User must be able to signup and sign up properly.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Verify that valid sign-in credentials allow access to the client list page.	Entering correct username and password must redirect the user to home page.	Entering correct username and password redirects the user to the home page.	Pass
2.	Verify that invalid sign-in credentials prevent access and display an appropriate error message.	Entering incorrect username or password must display an error message and prevent access to the client list page.	Entering incorrect username or password displays an error message and prevents access to the client list page.	Pass
3.	Verify that a new user can successfully sign up.	Entering valid sign-up information must create a new user account.	Entering valid sign-up information creates a new user account and sends a confirmation email.	Pass
4.	Verify that an existing email address cannot be used for sign-up and displays an appropriate error message.	Attempting to sign up with an already registered email address must display an error message indicating the email is already in use.	Attempting to sign up with an already registered email address displays an error message indicating the email is already in use.	Pass

1.2.2 Search client

User must be able to search client from client list.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Access the Client List module.	The user must be able to successfully access the Client List module.	The user successfully accesses the Client List module.	Pass
2.	Client list sorting by attribute	Client list must be sorted by is_registered or client name order based on the selected attribute.	Client list is sorted by is_registered or client name order based on the selected attribute.	Pass
3.	Validate empty search results.	An error message must be displayed.	An error message is displayed.	Pass

1.2.3 Register a new case

User must be able to initiate or register a new case.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Access New Cases module	The user must be able to successfully access the New Cases module.	The user successfully accesses the New Cases module.	Pass
2.	Validate client name and other fields	The system must allow the user to enter valid information for client name, case type, case name, adversary party, and case summary.	The system allows the user to enter valid information for all required fields.	Pass
3.	Register new case	Upon clicking the "Register Case" button, the system must validate the entered information and register the new case.	The system validates the entered information and registers the new case.	Pass
4.	Verify successful case registration	The system must display a confirmation message indicating that the case has been successfully registered.	The system displays a confirmation message indicating successful case registration.	Pass
5.	Validate cancel functionality	Upon clicking the "Cancel" button, the system must redirect the user back to the home page.	The system redirects the user back to the home page.	Pass

1.2.4 Check Payments

User must be able to check all his payments done.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Access Check billings or payments module	The user must be able to successfully access the Check Payments module.	The user successfully accesses the Check Payments module.	Pass
2.	Filter payments by any of lawyer, client, and date attribute.	The user must be able to filter payments based on lawyer name, client name, and date range attributes.	The system allows filtering payments based on specified criteria.	Pass
3.	Select and print specific payments	The user must be able to generate a printable representation after filtering.	The user can generate a printable document after filtering.	Pass
4.	Verify data consistency and dynamic updates	Printed data must match the data grid information, and the data grid must dynamically update based on search criteria.	Data consistency and dynamic updates are observed.	Pass

1.2.5 Upload document of Case

User must be able to upload documents for only registered cases.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Access Upload Document module	The user must be able to successfully access the Upload Document module.	The user successfully accesses the Upload Document module.	Pass
2.	Upload documents with valid case title and lawyer selection	The system must allow the user to upload documents with a valid case title and select a valid lawyer.	The system allows valid document uploads and lawyer selection.	Pass
3.	Verify document upload confirmation and association	The system must display a confirmation message upon successful document upload, and the document must be correctly associated with the specified case and lawyer.	Confirmation message is displayed, and document association is verified.	Pass
4.	Validate data grid information and document download	The data grid must display relevant data for each uploaded document	Data grid displays accurate data, and document upload is successful.	Pass

1.2.4 Book a appointment

User must be able to book his appointments.

Sr. No	Test Data	Expected Result	Actual Result	Test Result
1.	Access Book Appointment module	The user must be able to successfully access the Book Appointment module.	The user successfully accesses the Book Appointment module.	Pass
2.	Validate lawyer and client selection, date selection	The system must allow valid lawyer and client selection, date selection and display available appointment slots.	Valid selections and slot availability are observed.	Pass
3.	Book appointments	The user must be able to select appointment slots and book appointments, receiving confirmation message.	Appointment booking and confirmation are successful.	Pass

1.3 System Testing

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end.

1.3.1 Create New Account Testing

Test Case ID	ST-1.3.1.0	**Test Case Description**	Test the Create new account Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	UserType= Lawyer
2	Stable Internet Connection	2	Name =danyal
		3	Email=dn@gmail.com
		4	Password=0987654321
		5	Contact Number=03134755888

Test Scenario	Verify user able to create new account.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Click continue sign up link label	Sign up screen shown	As Expected	Pass
2	Enter name, email ,Usertype ,contact number and password	Credential can be entered	As Expected	Pass
3	Click signup	User is signed up	As Expected	Pass
4				

1.3.2 Sign in Testing

Test Case ID	ST-1.3.2.0	**Test Case Description**	Test the sign in Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Email=dn@gmail.com
2	Stable Internet Connection	2	Password=0987654321

Test Scenario	Verify user able to login.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter username and password	Credential can be entered	As Expected	Pass
2	Click signin	User is signed in.	As Expected	Pass

1.3.3 Register Case Testing

Test Case ID	ST-1.3.3.0	**Test Case Description**	Test the register case Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Client name= Mia Jones
2	Stable Internet Connection	2	Case title= Mr Jones Fam.
			Case Type= Employment Case
			Case summary= personal case about Mr.jones
			Adversery Party= Mr. Rehami

Test Scenario	Verify lawyer able register case.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter client name,case title,case summary,case type,adversary party	Credential can be entered	As Expected	Pass
2	Click Register	Case can be registered	As Expected	Pass
3	Click Cancel	Case data can be removed	As Expected	Pass

1.3.4 Client list Testing

Test Case ID	ST-1.3.3.0	**Test Case Description**	Test the client list Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Client name= Mia Jones
2	Stable Internet Connection	2	Is_registered=null

Test Scenario	Verify lawyer able to sort client list.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter client name and fill is_registered check box	Values can be entered	As Expected	Pass
2	Click Sort	Client list can be sorted	As Expected	Pass

1.3.5 Document Upload Testing

Test Case ID	ST-1.3.4.0	**Test Case Description**	Test the document upload Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Case title= Probate of Will
2	Stable Internet Connection	2	Name= daniyal
		3	Content: Case is finished

Test Scenario	Verify lawyer able to upload content for document.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter case title,name,content	Values can be entered	As Expected	Pass
2	Click Upload	Document can be uploaded	As Expected	Pass

1.3.6 Billing list testing

Test Case ID	ST-1.3.5.0	**Test Case Description**	Test the Billing list Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Lawyer name=Null
2	Stable Internet Connection	2	Client name= Daniel Wilson
		3	Specific date=null

Test Scenario	Verify lawyer able to Sort Billing list.			

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter lawyer name,client name,Specific date	Values can be entered	As Expected	Pass
2	Click Check	Payement list can be sorted	As Expected	Pass
3	Click print	Payement can be printed	As Expected	Pass

1.3.7 Appointment testing

Test Case ID	ST-1.3.6.0	**Test Case Description**	Test the Appointment Functionality in Legal link		
Created By	Messam	**Reviewed By**	Muneeb	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Lawyer name=Null
2	Stable Internet Connection	2	Client name= Daniel Wilson
		3	Specific date=null

Test Scenario	Verify lawyer able to see Sorted Billing list.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter lawyer and client selection, date selection and availability	Values can be entered	As Expected	Pass
2	Click Book Appointment	Appointment list can be sorted	As Expected	Pass

1.4 Acceptance testing

Acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance with all of the original business criteria and that it meets the end user‘s needs. Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. This requires the product be tested both internally and externally, meaning you‘ll need to get it into the hands of your end users for beta testing. Beta testing is key to getting real feedback from potential customers and can address any final usability concerns.

However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing) But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario. Acceptance testing is the squeezing of all the testing processes that have done previously

- Sign in Testing
- Sign up Testing
- Register case Testing
- Client list Testing
- Document Upload Testing
- Billing list Testing
- Appointment Testing

1.4.1 Sign up Testing

Test Case ID	ST-1.3.1.0	**Test Case Description**	Test the Create new account Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	UserType= Lawyer
2	Stable Internet Connection	2	Name =danyal
		3	Email=dn@gmail.com
		4	Password=0987654321
		5	Contact Number=03134755888

Test Scenario	Verify user able to create new account.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Click continue sign up link label	Sign up screen shown	As Expected	Pass
2	Enter name, email ,Usertype ,contact number and password	Credential can be entered	As Expected	Pass
3	Click signup	User is signed up	As Expected	Pass
4				

1.4.2 Sign in Testing

Test Case ID	ST-1.3.2.0	**Test Case Description**	Test the sign in Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Email=dn@gmail.com
2	Stable Internet Connection	2	Password=0987654321

Test Scenario	Verify user able to login.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter username and password	Credential can be entered	As Expected	Pass
2	Click signin	User is signed in.	As Expected	Pass

1.4.3 Register Case Testing

Test Case ID	ST-1.3.3.0	**Test Case Description**	Test the register case Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Client name= Mia Jones
2	Stable Internet Connection	2	Case title= Mr Jones Fam.
			Case Type= Employment Case
			Case summary= personal case about Mr.jones
			Adversery Party= Mr. Rehami

Test Scenario	Verify lawyer able register case.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter client name,case title,case summary,case type,adversary party	Credential can be entered	As Expected	Pass
2	Click Register	Case can be registered	As Expected	Pass
3	Click Cancel	Case data can be removed	As Expected	Pass

1.4.4 Client list Testing

Test Case ID	ST-1.3.4.0	**Test Case Description**	Test the client list Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Client name= Mia Jones
2	Stable Internet Connection	2	Is_registered=null

Test Scenario	Verify lawyer able to sort client list.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter client name and fill is_registered check box	Values can be entered	As Expected	Pass
2	Click Sort	Client list can be sorted	As Expected	Pass

1.4.5 Document Upload Testing

Test Case ID	ST-1.3.5.0	**Test Case Description**	Test the document upload Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Case title= Probate of Will
2	Stable Internet Connection	2	Name= daniyal
		3	Content: Case is finished

Test Scenario	Verify lawyer able to upload content for document.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter case title,name,content	Values can be entered	As Expected	Pass
2	Click Upload	Document can be uploaded	As Expected	Pass

1.4.6 Billing list testing

Test Case ID	ST-1.3.6.0	**Test Case Description**	Test the Billing list Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Lawyer name=Null
2	Stable Internet Connection	2	Client name= Daniel Wilson
		3	Specific date=null

Test Scenario	Verify lawyer able to Sort Billing list.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter lawyer name,client name,Specific date	Values can be entered	As Expected	Pass
2	Click Check	Payement list can be sorted	As Expected	Pass
3	Click print	Payement can be printed	As Expected	Pass

1.4.7 Appointment testing

Test Case ID	ST-1.3.7.0	**Test Case Description**	Test the Appointment Functionality in Legal link		
Created By	Muneeb	**Reviewed By**	Messam	**Version**	1.0
QA Tester's Log	Review comments from Muneeb incorporate in version 2.1				
Tester's Name	Messam	**Date Tested**	12 Dec 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	Lawyer name=Null
2	Stable Internet Connection	2	Client name= Daniel Wilson
		3	Specific date=null

Test Scenario	Verify lawyer able to see Sorted Billing list.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enter lawyer and client selection, date selection	Values can be entered	As Expected	Pass
2	Click Book Appointment	Appointment list can be sorted	As Expected	Pass

1.5 Performance Testing

Performance testing is a non-functional testing technique used to determine how an application will behave under various conditions. The goal is to test its respon siveness and stability in real user situations. Performance testing can be broken down into four types:

- Load testing
- Stress testing
- Endurance testing
- Spike testing

1.5.1 Load Testing

We done this testing under the heavy load of the data and system can take 400 - 600 ms to be processed and initiate the system. After initiating it takes 10 - 30 ms to settle all the rest application and then start working properly and made applications functional. Estimated initiating time for application 1200 - 200 ms in heavy data loaded environment.

Sr. no.	Functionality	Estimated Load Time (seconds)
1.	Sign in	0.2 to 0.5
2.	Sign up	0.2 to 0.5
3.	Home	0.1 to 0.3
4.	New Cases	0.3 to 0.5
5.	Client list	0.31 to 39
6.	Billings	0.2 to 0.5
7.	Appointments	0.1 to 0.2
8.	Documents	0.2 to 0.5
9.	Logout	0.1 to 0.2

1.5.2 Stress Testing

We have done stress testing under heavy load of data and putting that load to the system and load data more than it‘s capacity. By increasing desired load into the system then the system takes the approx. time given below.

Sr. no.	Functionality	Estimated Load Time (seconds)
1.	Sign in	0.6 to 0.7
2.	Sign up	0.2 to 0.5
3.	Home	0.2 to 0.3
4.	New Cases	0.1 to 0.2
5.	Client list	0.4 to 0.5
6.	Billings	0.1 to 0.4
7.	Appointments	0.1 to 0.2
8.	Documents	0.1 to 0.3
9.	Logout	0.1 to 0.2

1.5.3 Endurance Testing

We done this testing under the heavy load of the data and putting that load Over a Longer amount of time to understand how your system will behave under sustained use, making it a longer process than load or stress testing.

Sr. no	Endurance Test
1	Register many cases so system not able to handle as many cases.
2	Upload many documents so that system sustained to upload document
3	Load many appointments so system cannot able to load properly.
4	Request many payments so user cannot handle.

1.5.4 Spike Testing

We done this testing under the heavy load of the data and putting that load of same amount of data over a longer amount of time to understand how your system will behave when the load is suddenly and drastically increase.

Sr. no	Spike Test
1	Enter many wrong email and passwords again and again.

1.6 Security Testing

With the rise of cloud-based testing platforms and cyber-attacks, there is a grow concern and need for the security of data being used and stored in software. Security testing is a non-functional software testing technique used to determine if the information and data in a system is protected. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses. There are multiple types of this testing method, each of which aimed at verifying five basic principles of security:

- Integrity
- Confidentiality
- Authentication
- Authorization
- Availability

1.6.1 Integrity

We managed integrity of our system by user access controls. User must have to sign up and the sign in by giving his/her details i.e. name, email, and password to enter in the system.

1.6.2 Confidentiality

To avoid unauthorized persons we demand password from the user when he/she sign in to our system and make sure to do it confidentially with the help of encryption method and asks user to change his/her password after an year to make confidentiality more strong.

1.6.3 Authentication

User cannot able to sign in into the system if he/she does not enter the email and password that is register into the system, so we can make sure it is the right person and when user sign in his/her account into a different device.

1.6.4 Authorization

To recognize the authorized person we ask for sign in to the system. It assures that only authorized person will be entered to the system and handle all the rest procedure of handling all the system. We focus on the security of our application because it contains all the data information of the system.

1.6.5 Availability

System must be available 24/7 and provider services to the users.

1.7 Usability Testing

Usability testing is a testing method that measures an system ease-of-use from the end-user perspective and is often performed during the system or acceptance testing stages. The goal is to determine whether or not the visible design and aesthetics of an application meet the intended work flow for various processes, such as logging into an system. Our systems UI design is easy to use for the lawyers and work flow can be easily understand able by the user .We performed acceptance testing that shows that user can‘t go wrong and didn‘t face any distraction in our system design.

Sr. no	System Flow
1	After sign in into the system, Home screen shown to user and able to select different services.
2	User able to navigate to new cases, Documents, client lists and appointments.

3	User chooses new cases and register new case.
4	See all client lists available
5	Select any client.
6	Upload content for a specific case.
7	Click on Billings to view payments lists.
8	User moves to Home screen.
9	User selects Appointments.
10	Able to see his/herAppointments.
11	Click on logout.

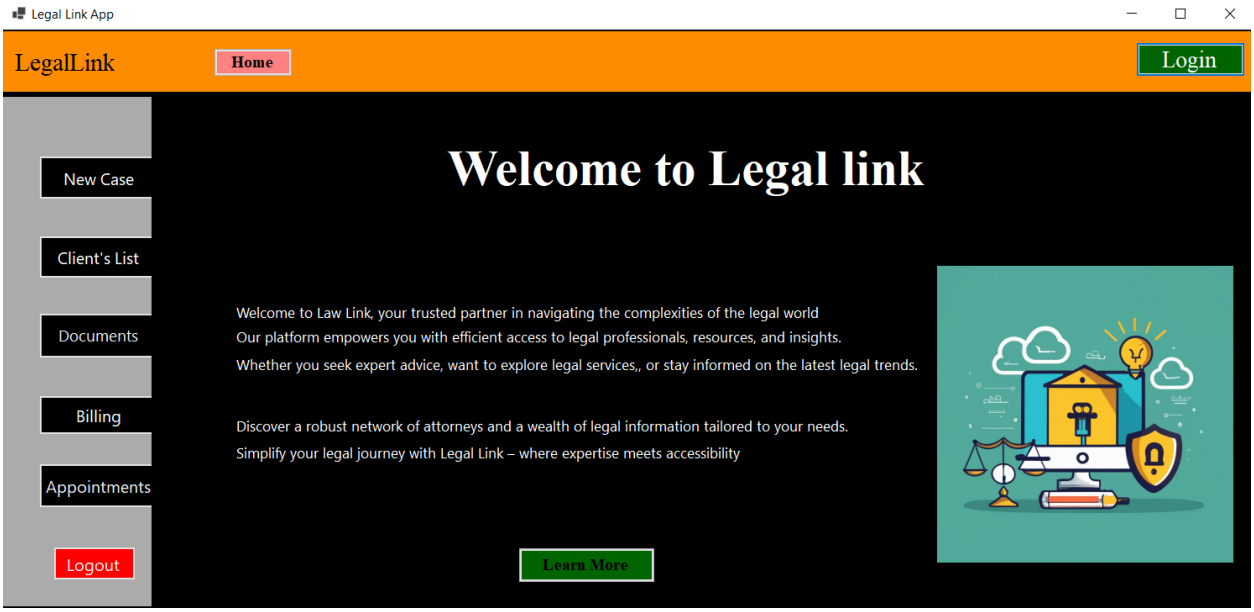
1.8 Compatibility Testing

Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that our system functionality is consistently supported across any environment you expect your end users to be using. We perform compatible testing on our project by running it on different systems.

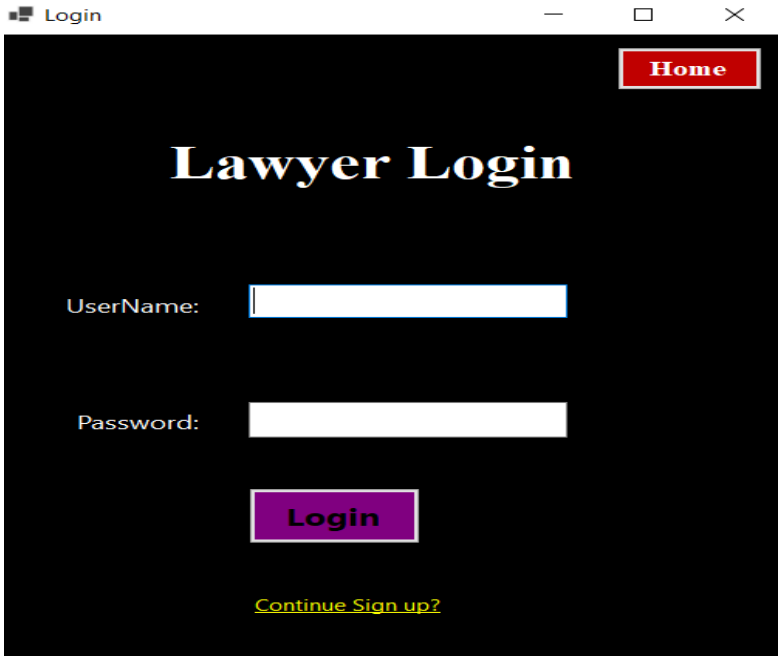
Chapter 6

Results and Outputs

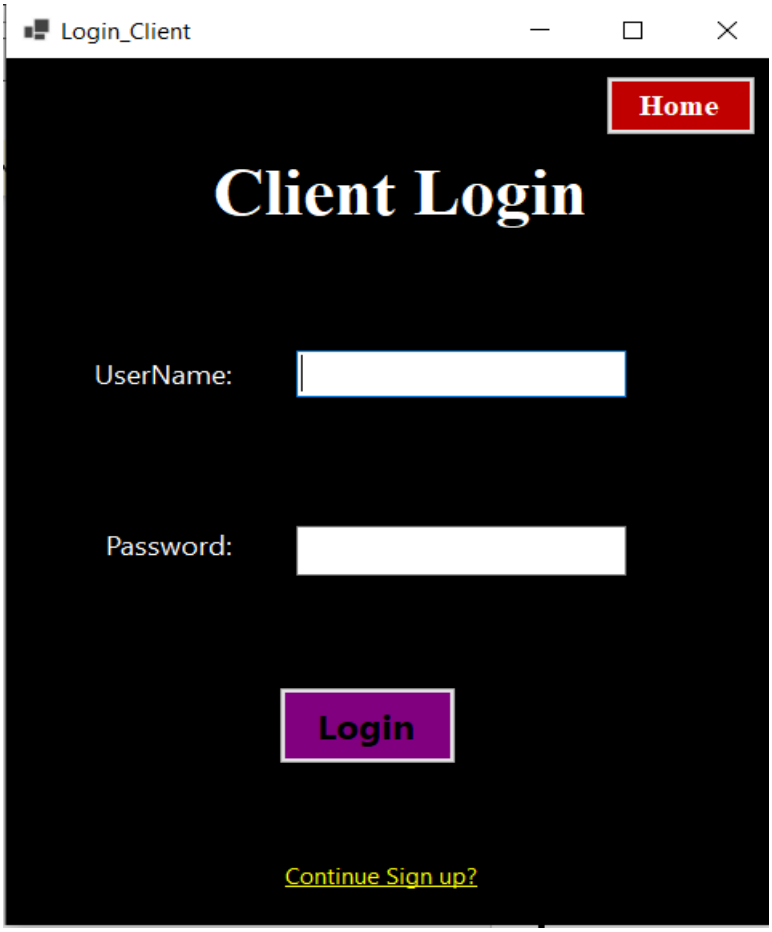
6.1 Legal Link Home Screen



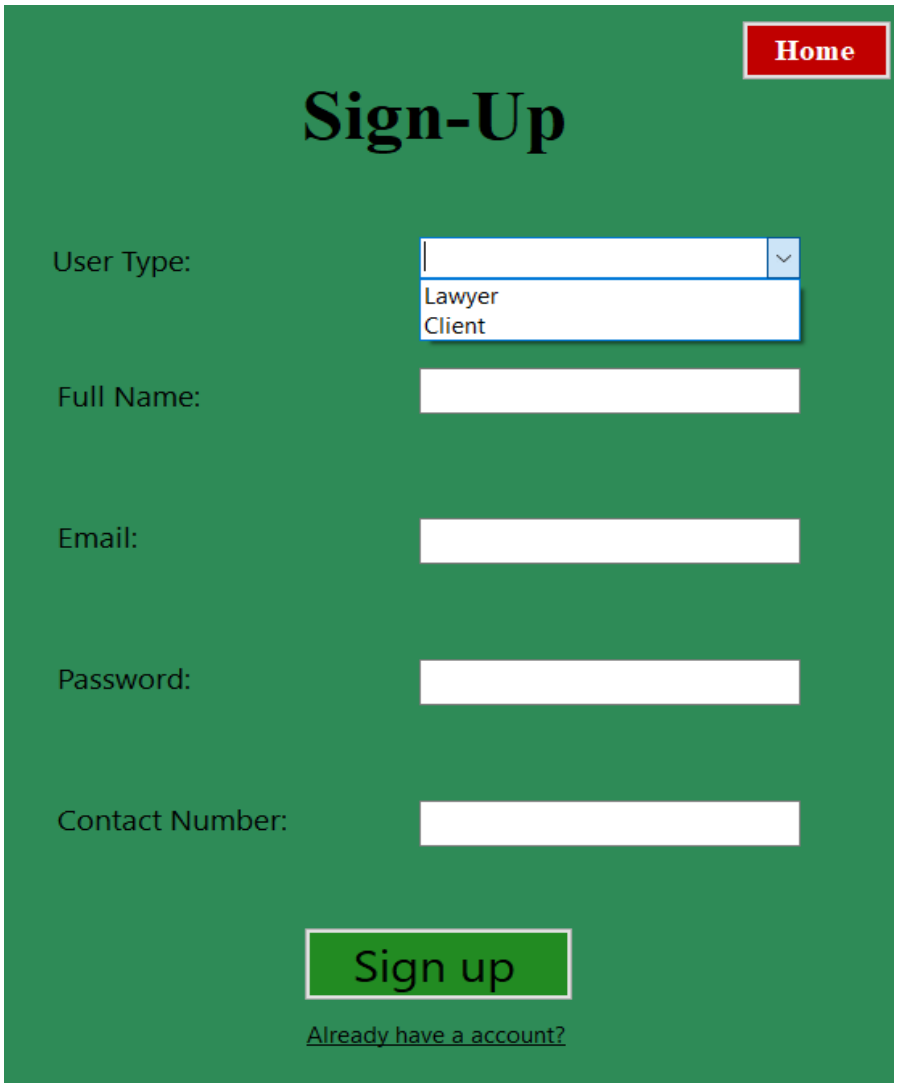
6.2 Lawyer Login Screen



6.3 Client Login

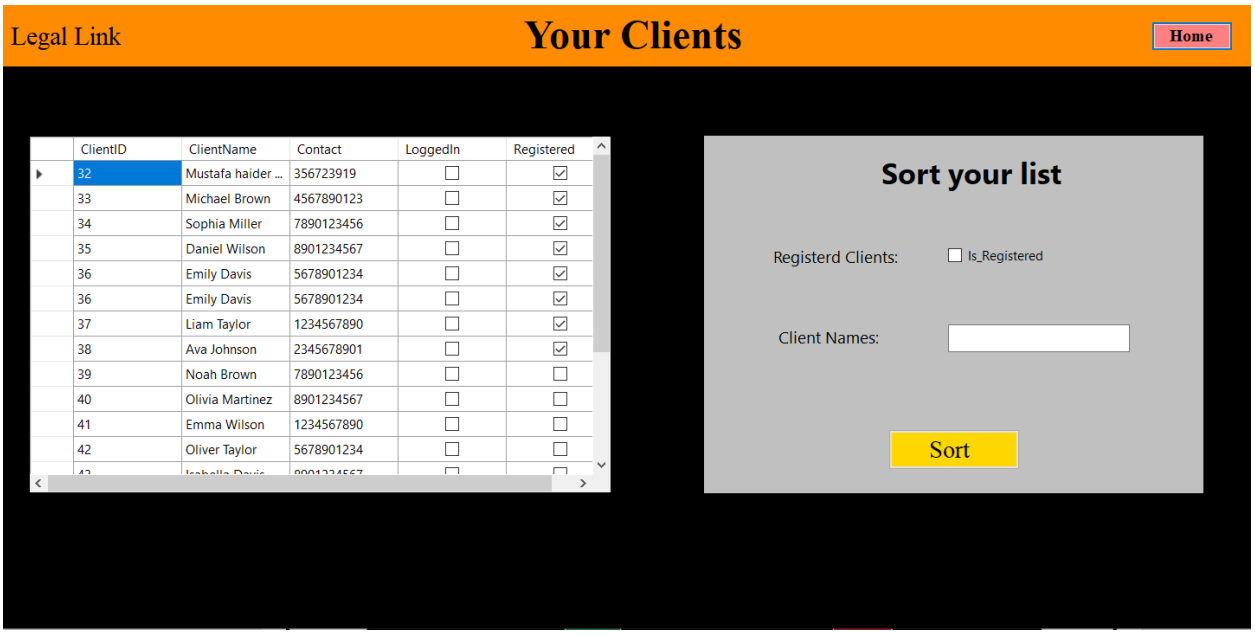


6.4 Signup Screen

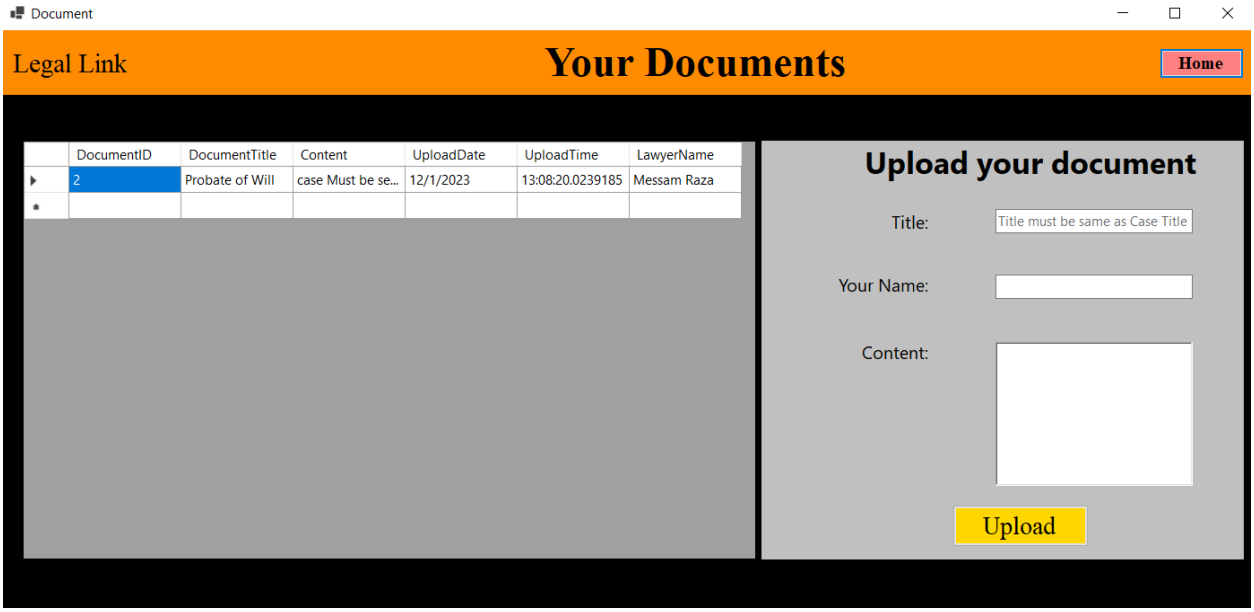


6.5 New Cases Screen

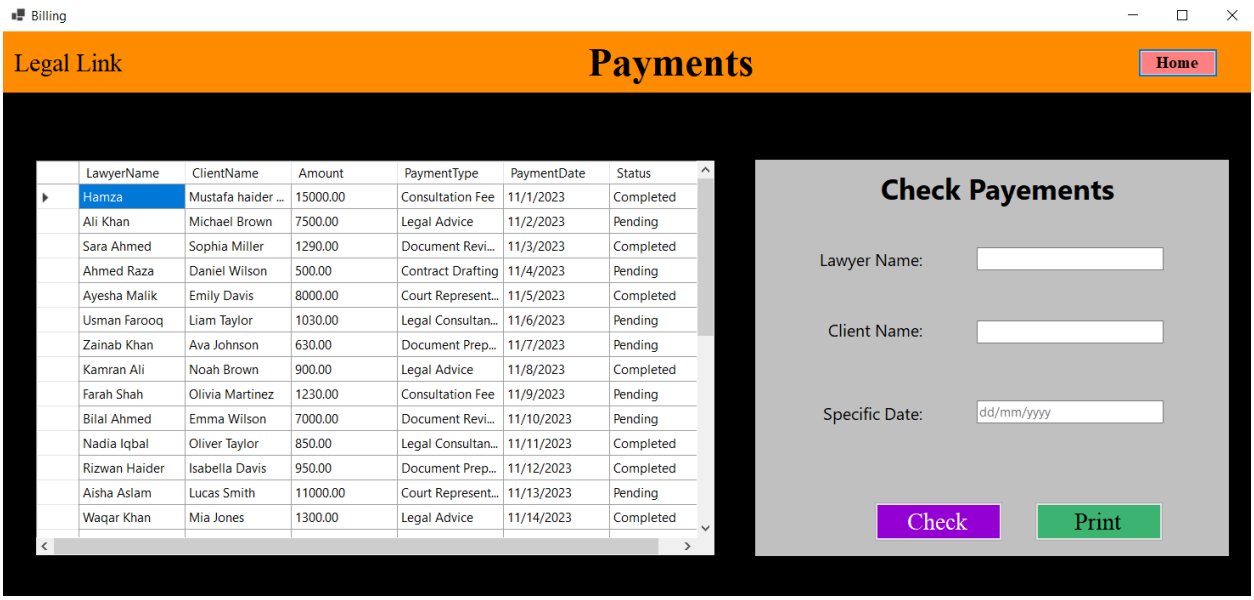
6.6 Client List Screen



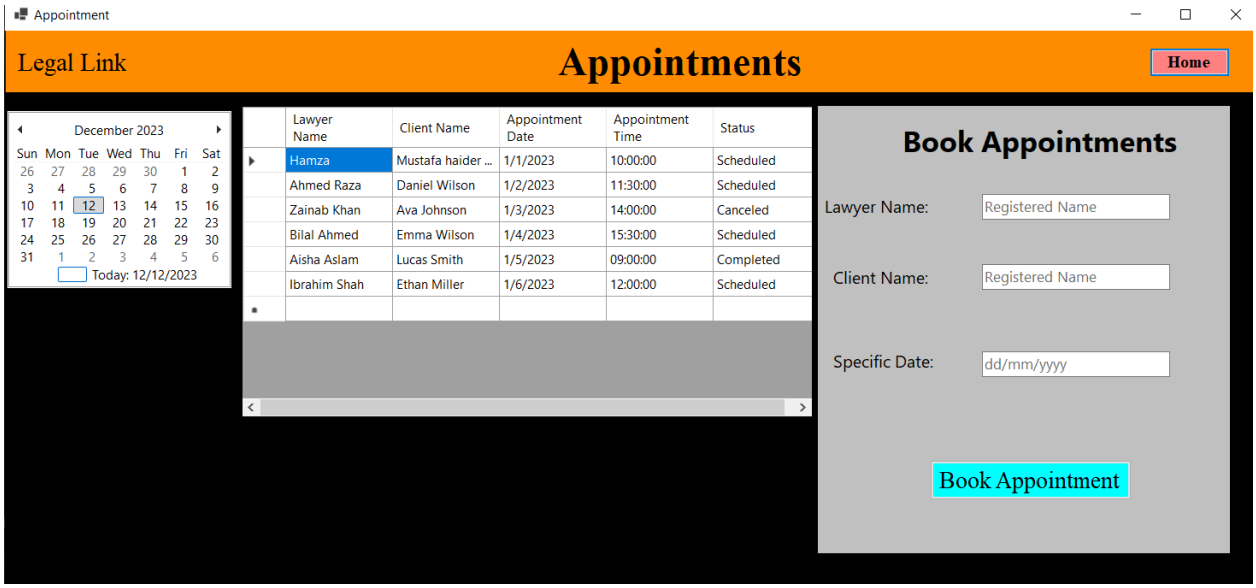
6.7 Documents Screen



6.8 Billings Screen



6.9 Appointments Screen



Chapter 7

Conclusion and Future Direction

7.1 Conclusion

The Legal Link app represents a significant advancement in legal practice management. This innovative C# desktop application offers a comprehensive and user-friendly solution for case management, client tracking, document organization, and billing integration. There are seven sections in this report,Section 1 describes our objectives and the scope of the project. Section 2 includes use cases that explain the functional and non-functional requirements. Section 3 examines system analysis design. Section 4 describes how we design our database and server using project code. Section 5 contains the whole test document (Test cases, unit testing, integration testing, Acceptance testing, Performance testing, Performance testing, Security testing, Usability testing and Compatibility testing). Section 6 describes our project's results and output. Section 7 concludes with a discussion on Future Directions.

7.2 Future Direction

Looking ahead, Legal Link is poised to evolve even further, continuously adapting to meet the evolving needs of legal professionals. Here are some exciting avenues for future development:

1. Artificial Intelligence Integration: Integrating AI capabilities could significantly enhance Legal Link's efficiency and accuracy. AI algorithms could be employed for automated document analysis, legal research assistance, and even predictive analytics to anticipate potential legal issues and prepare proactive solutions.

2. Blockchain Integration: Blockchain technology has the potential to revolutionize the legal industry by offering secure and immutable data storage and transaction tracking. Legal Link could leverage blockchain to enhance document security, expedite contract agreements, and streamline legal processes, fostering greater transparency and trust within the legal system.

3. Mobile Application Development: Expanding Legal Link's functionality to a mobile platform would provide legal professionals with greater accessibility and flexibility. A mobile app would allow access to critical case information, client communication, and billing management on the go, further boosting productivity and responsiveness.

4. Custom Integrations and API Development: Legal Link can be further empowered by offering custom integrations with existing legal software solutions and databases. This would eliminate the need for data silos and allow legal professionals to seamlessly integrate Legal Link into their established workflows. Additionally, developing an API would allow third-party developers to create custom applications that further extend Legal Link's functionality and cater to specific needs within the legal community.

5. Enhanced Collaboration Features: Legal Link can evolve to facilitate stronger collaboration between legal professionals. Integrating features like secure document sharing, real-time co-editing, and online discussion forums would encourage teamwork and streamline legal proceedings, leading to faster and more effective case resolution.

6. International Expansion: Legal Link's potential extends beyond geographical borders. Adapting the application to comply with international legal standards and regulations would allow legal professionals to operate seamlessly across jurisdictions, facilitating global collaboration and legal service delivery.

7. Continuous User Feedback and Improvement: Legal Link's development should remain driven by user feedback and needs. Implementing an effective feedback mechanism and actively incorporating user suggestions will ensure that the application evolves to address the changing demands of the legal profession and remains at the forefront of legal technology innovation.

By actively pursuing these future directions, Legal Link can solidify its position as an indispensable tool for legal professionals, empowering them to achieve greater efficiency, accuracy, and success in their endeavors.

Chapter 8

References

[1] McKinsey & Company. (2023)., “McKinsey & Company,” 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://www.reuters.com/legal/legalindustry/how-will-leveraging-ai-change-future-legal-services-2023-08-23/

[2] Clio, “Clio,” 2022. Accessed: Dec. 12, 2023. [Online]. Available: https://www.clio.com/about/press/2022-legal-trends-report/

[3] Thomson Reuters., “Thomson Reuters. (2021). ,” 2021. Accessed: Dec. 12, 2023. [Online]. Available: https://www.americanbar.org/groups/law_practice/resources/legal-technology-resource-center/2021-survey/

[4] American Bar Association, “American Bar Association,” 2021. Accessed: Dec. 12, 2023. [Online]. Available: https://www.americanbar.org/groups/professional_responsibility/publications/model_rules_of_professional_conduct/model_rules_of_professional_conduct_table_of_contents/

[5] legals, “legal tech,” 2021. Accessed: Dec. 12, 2023. [Online]. Available: https://www.legaltechtrends.com/