A PROJECT REPORT

on

NetAdmin Pro: Centralized Network Management

System for Educational Institutions.

Submitted by

Mr. SONU KUMAR JHA

in partial fulfillment for the award of the degree

of

BACHELOR OF SCIENCE

in

COMPUTER SCIENCE

under the guidance of

Mr. Mahendra Kanojia

Department of Computer Science

SHETH L.U.J. COLLEGE OF ARTS & SIR M.V. COLLEGE OF SCIENCE

(Sem V)

(2023 – 2024)

❖PREFACE

Welcome to the documentation of **NetAdmin Pro: Centralized Network Management System for Educational Institutions**. This document is an in-depth guide designed to provide comprehensive insights into the development, functionality, and implementation of our network management system tailored for educational institutions.

Project Overview

NetAdmin Pro addresses the unique challenges faced by educational institutions in managing network resources. It aims to foster an ethical and focused learning environment by ensuring students utilize college resources for academic purposes. This system offers centralized control, surveillance, and restrictions to prevent unauthorized access and ensure the ethical use of college resources.

Documentation Highlights

- **Technical Details**: Delve into the technical aspects of NetAdmin Pro, exploring the architecture, design principles, and functionalities that make it a robust network management solution.
- **User Interactions**: Understand the user experience, from administrators managing the system to students navigating within the restricted environment.
- **Client Perspective**: Gain insights into the client's vision and expectations, driving the development and features of NetAdmin Pro.

We invite you to explore this documentation thoroughly to gain a comprehensive understanding of NetAdmin Pro and its role in creating a conducive learning environment.
Thank you for your interest and commitment to ethical network management.

Sincerely,
[Sonu Jha]

❖ACKNOWLEDGEMENT

As a solo developer, the creation of **NetAdmin Pro: Centralized Network Management System for Educational Institutions** has been a personal journey filled with challenges, learning, and dedication. In reflecting upon this endeavor, I would like to express my gratitude to those who have played a significant role in this project.

Self-Acknowledgment

I extend heartfelt appreciation to myself for the determination, countless hours, and passion invested in conceptualizing, designing, and implementing NetAdmin Pro. This project reflects personal growth, resilience, and a commitment to delivering a valuable solution.

Educational Institutions

I appreciate the educational institutions that inspired the development of NetAdmin Pro. Your vision for a secure and focused learning environment motivated me to embark on this project. NetAdmin Pro stands as a testament to the importance of ethical resource use in academic settings.

Online Platforms and Forums

I am thankful to the various online platforms and forums where knowledge sharing and troubleshooting discussions have taken place. These resources have been instrumental in overcoming challenges and expanding my understanding of network management systems.

Learning Opportunities

Every challenge faced during this project has been a learning opportunity. I express gratitude for the moments of frustration, as they led to deeper insights, problem-solving skills, and an enhanced understanding of software development.

This acknowledgment encapsulates a personal journey of creation, innovation, and growth. NetAdmin Pro is not just a project; it is a reflection of personal and professional development.

Sincerely

[**Sonu Jha**]

❖DECLARATION

I hereby declare that the project entitled, "**NetAdmin Pro: Centralized Network Management System for Educational Institutions**" done **Sheth L.U.J. College of Arts & Sir M.V college of Science & Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (COMPUTER SCIENCE)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENT

❖INTRODUCTION

In the dynamic landscape of educational institutions, maintaining a secure and focused learning environment is paramount. With the surge in digital resources and the omnipresence of the internet, ensuring ethical usage of college resources becomes a critical concern. To address this challenge, I embarked on a solo endeavor to develop **NetAdmin Pro: Centralized Network Management System for Educational Institutions**.

Surveillance of student activities within the college network, while a delicate subject, is approached in this project with the aim of fostering ethical resource utilization. NetAdmin Pro goes beyond traditional network management systems, offering a comprehensive solution designed specifically for educational environments.

Key Features

- **Resource Ethical Use**:
  NetAdmin Pro ensures that students can access only academic resources within the college network, promoting ethical usage and minimizing distractions during study hours.

- **Centralized Management:**
  The system provides centralized control, allowing administrators to monitor, manage, and regulate network activities seamlessly.

- **User-Friendly Interface:**
  With a user-friendly interface, NetAdmin Pro is designed to be accessible to administrators with varying levels of technical expertise, ensuring ease of use.

- **Real-time Surveillance:**

Real-time monitoring features empower administrators to stay informed about network activities, enabling timely intervention when necessary.

- **Customization and Flexibility:**
  Tailored to the unique needs of educational institutions, NetAdmin Pro offers customization options, providing flexibility to adapt to diverse academic environments.

Project Vision

The vision behind NetAdmin Pro is to create a secure and conducive digital ecosystem within educational institutions. By fostering an environment where students utilize digital resources ethically and focus on academic pursuits, NetAdmin Pro aims to contribute to the overall enhancement of the learning experience.

Guiding Principles

- **Ethical Resource Use:** NetAdmin Pro is rooted in the principles of promoting ethical resource use, ensuring that students leverage digital resources for academic purposes.

- **Innovation:** The project reflects a commitment to innovation in network management systems, aiming to set a new standard for centralized control in educational institutions.

- **Continuous Improvement:** NetAdmin Pro is not just a static solution; it is a commitment to continuous improvement. Feedback, collaboration, and future enhancements will drive the evolution of this project.

As we navigate through the documentation, the subsequent sections will delve into the technical aspects, stakeholder considerations, system architecture, and the journey of development. NetAdmin Pro stands as a testament to the fusion of technological innovation with a conscientious approach to education.

Welcome to the exploration of NetAdmin Pro, where secure networks and ethical resource use converge to create a transformative learning environment.

❖STAKEHOLDERS

Stakeholders in NetAdmin Pro

1. **Technical Stakeholders:**
   - **Software Developer:** The primary architect and developer of NetAdmin Pro, responsible for its design, coding, testing, and overall implementation.
   - **IT Administrators:** The individuals tasked with the day-to-day management of the system, including installation, maintenance, and troubleshooting.
   - **Network Security Specialists:** Professionals focused on ensuring the security and integrity of the network, actively involved in evaluating and enhancing the system's security measures.

2. **User Stakeholders:**
   - **College Administrators**: Decision-makers at the institutional level who oversee the implementation of NetAdmin Pro and guide its integration into the existing network infrastructure.
   - **Faculty Members:** Academic staff who benefit from the system's features for managing and monitoring student activities within the network.
   - **Students:** End-users whose network activities are monitored and regulated by NetAdmin Pro to ensure ethical use of college resources.

3. **Client Stakeholders:**
   - **College Management:** The individuals or entities responsible for overseeing the college's overall operations, including decisions related to technology adoption and resource management.

4. **External Stakeholders:**
   - **Regulatory Bodies:** Government bodies or agencies that may have regulations or guidelines concerning the ethical use of networks in educational institutions. NetAdmin Pro aligns with these standards to ensure compliance.
   - **Parents:** While not direct users, parents may have concerns about the ethical use of digital resources by their children. NetAdmin Pro indirectly addresses these concerns by promoting a secure and focused learning environment.

5. **Collaborative Stakeholders:**

- **Educational Technology Partners:** Companies or organizations providing educational technology solutions that may collaborate with NetAdmin Pro for integration or future feature enhancements.

Understanding the diverse needs and perspectives of these stakeholders is crucial for the successful development, implementation, and adoption of NetAdmin Pro. Throughout the project lifecycle, effective communication and collaboration with these stakeholders will play a vital role in shaping the system's impact and ensuring its alignment with the goals of educational institutions.

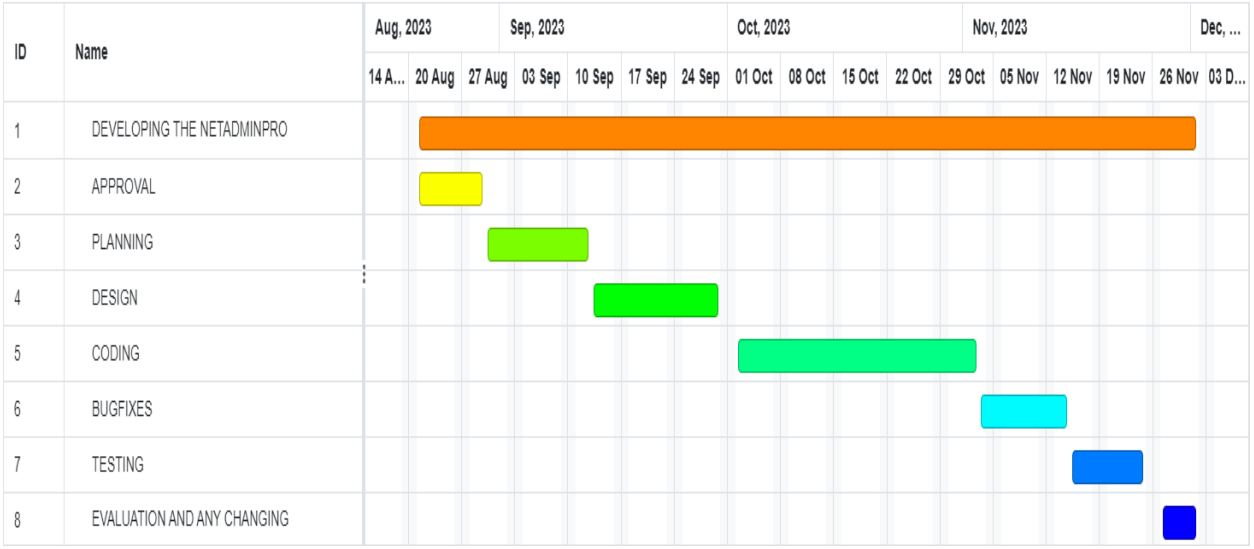
❖GANTT CHART

Planning and Scheduling Gantt Chart

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modem Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts are sometimes equated with bar charts. Gantt charts are usually created initially using an early start time approach, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.

On a Gantt chart you can easily see:

- The start date of the project
- What the project tasks are
- When tasks start and finish
- How long each task will take
- How tasks group together, overlap and link with each other
- The finish date of the project.



❖DATABASE DESCRIPTION

SQLite:

SQLite is a lightweight, serverless relational database management system. It operates as a self-contained, file-based engine, requiring no separate server setup. Key features include cross-platform support, minimal resource consumption, and SQL compatibility. SQLite is commonly used in embedded systems, mobile applications, and desktop software due to its simplicity and versatility. It offers transaction support, is open-source, and is widely adopted for prototyping and development.

Database Name: Account.db

Introduction:

The Account.db database is an integral component of the NetAdminPro software, responsible for managing user registration and authentication. This relational database is built using SQLite, a lightweight and efficient database engine. It stores essential user information, such as usernames, emails, mobile numbers, and hashed passwords, ensuring a secure and organized environment for user management.

Database Description:

- **UsersInfo Table:**

  The UsersInfo table serves as the primary repository for user data.

  It is designed to store unique user information and support the authentication process.

- **Columns:**
    1. **UserName (TEXT):** Primary key representing the unique username chosen by the user during registration.
    2. **Email (TEXT):** User's email address, allowing for communication and additional account recovery options.
    3. **MobileNum (INT):** User's mobile number, used for SMS-based verification during registration.
    4. **Password (TEXT):** Securely hashed password for user authentication.

Operations and Features:

1. **User Registration:**

- The database supports the registration process by storing user details securely.
- During registration, the system checks for the uniqueness of the chosen username and verifies the email domain against a predefined list.
- Mobile numbers are used for SMS-based verification, enhancing security.
- A six-digit verification code is generated and sent to the user's mobile number using the Twilio API.

2. **User Authentication:**
    - The system performs user authentication by comparing entered credentials with stored data.
    - If a user attempts to register with an existing username, the system prompts the user to choose a different username.
    - Passwords are securely hashed to prevent unauthorized access.

Security Measures:

1. **Hashed Passwords:**
    - User passwords are stored as secure hashes, providing an additional layer of security against data breaches.

2. **Email Domain Verification:**
   - The system validates user-provided email addresses against a predefined list of allowed email domains.
3. **Mobile Number Verification:**
   - SMS-based verification using Twilio ensures the validity of the provided mobile number during registration.

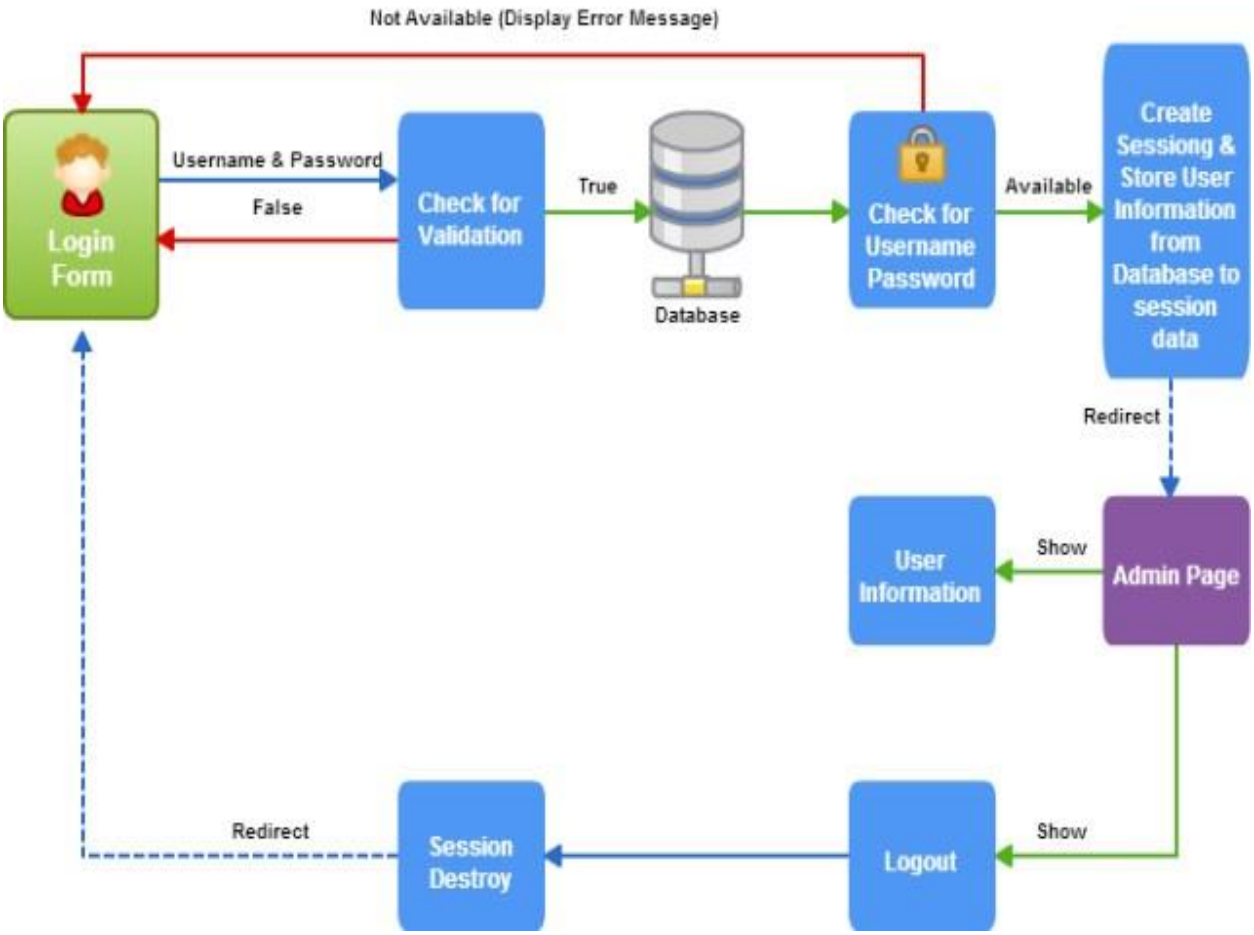
Usage Guidelines:

1. **Data Integrity**:
   - The database ensures data integrity by enforcing a unique constraint on usernames.
2. **Error Handling**:
   - Error handling mechanisms are implemented to address potential issues during registration and authentication.

Conclusion:

The Account.db database plays a crucial role in NetAdminPro, managing user data securely and supporting key registration and authentication processes. Its design prioritizes data integrity, security, and efficient user management.

Database Diagram



❖SYSTEM FLOWCHART & ER DIAGRAM

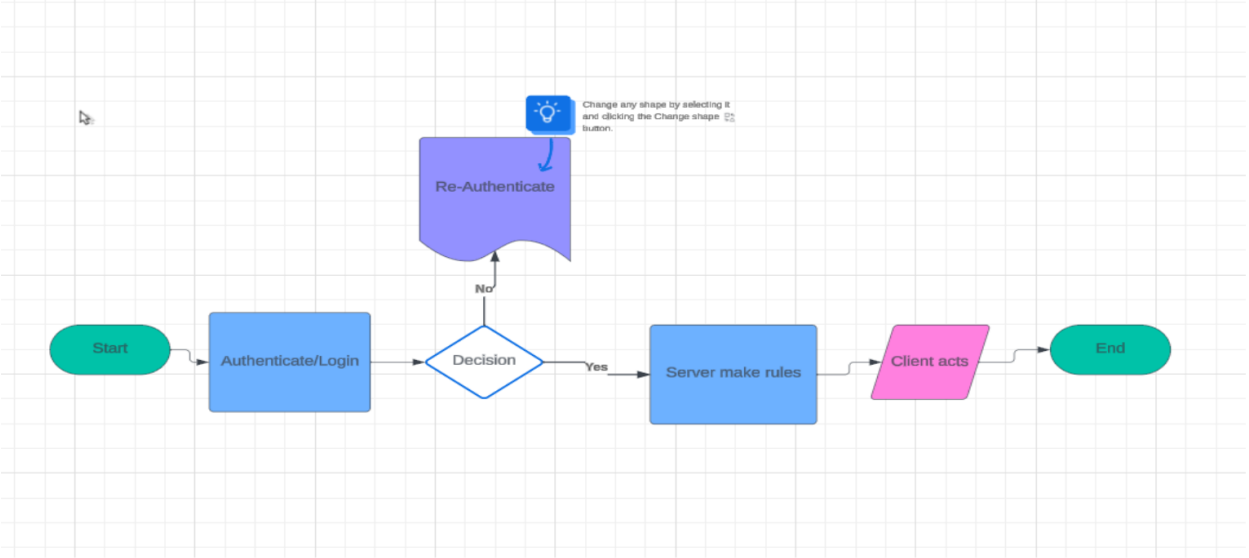
1. System FlowChart

A system flowchart is a visual representation or diagram that illustrates the sequence of operations and interactions within a system or process. It provides a high-level overview of how data, information, or materials move through different components or stages within a system. System flowcharts are commonly used in systems analysis and design to document, understand, and communicate the logical flow of activities in a system.

Key Elements of a System Flowchart:

1. **Shapes and Symbols:**
   - Different shapes and symbols represent various elements in the system, such as processes, decision points, inputs, outputs, and storage.
2. **Processes:**
   - Represented by rectangles, processes depict specific actions or operations performed within the system.
3. **Arrows:**
   - Arrows indicate the flow or direction of data, information, or materials between different components in the system.
4. **Decision Points:**
   - Diamonds or other shapes with conditional statements represent decision points where the flow of the process diverges based on specific conditions.

5. **Inputs and Outputs:**
   - Parallelograms represent inputs or outputs, indicating the sources of input data and where the processed data is directed.
6. **Connectors:**
   - Connectors join different parts of the flowchart, showing the relationships and continuity between processes.



2. ER DIAGRAM:

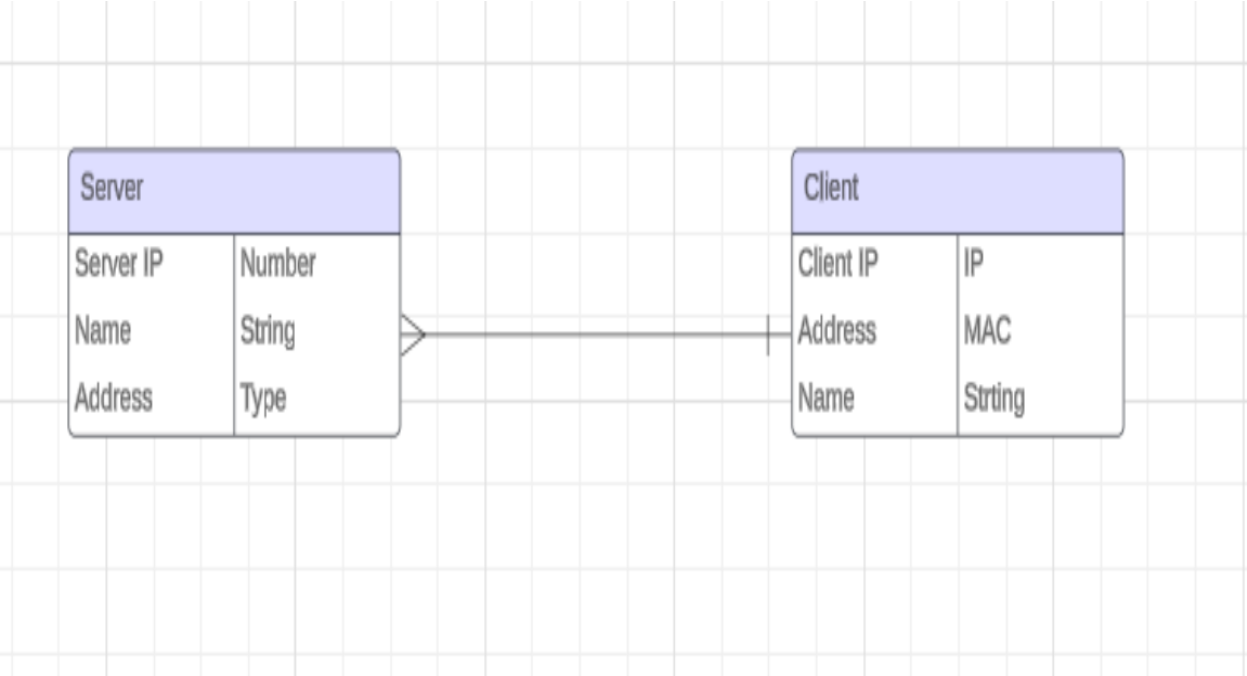
An entity relationship diagram shows the relationship of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have a similar that define its properties.

- **Entities**: They are represented by rectangle. An entity is an object or concept about which you want to store information.
- **Weak entity**: A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified.
- **Actions**: It shows how two entities share information in a relationship.
- **Attributes**: They are represented by ovals. A key attribute is the unique distinguishing characteristic of an entity.

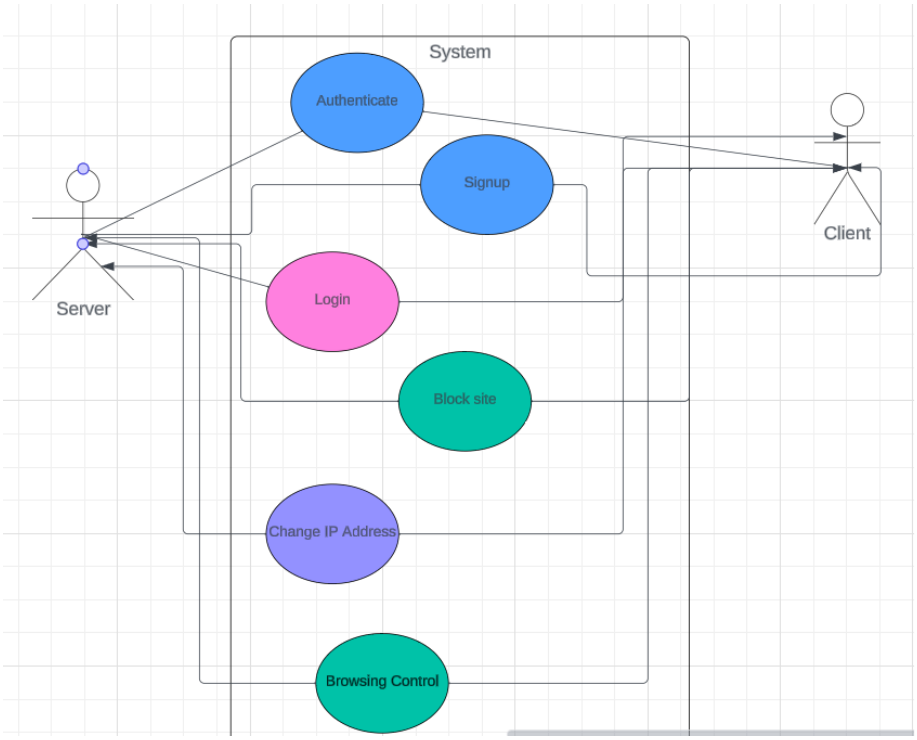
Benefits of ER Diagram:

The ER diagram is highly popular as it carries multiple benefits which include:

- **Efficient communication**: It allows the readers to understand the relationship among different fields in an effective manner. Symbols are utilized to represent information effectively and they also help in comprehending the working of the database.
- **Visual representation**: The data-flow diagrams along with ER diagrams can be used effectively for visual representation of the layout
- **Easy understanding**: Due to ease of understanding associated with design using ER diagrams, it can be represented to the concerned people for confirmation. Also, changes can be made effectively on the basis of the suggestions.
- **Highly Flexible**: The ER diagram can be effectively utilized for establishing and deriving relationships from the existing ones. Mathematical formulas and relational tables can be utilized for performing this operation.


Server
Server IP
Number
Name
String
Address
Type
Client
Client IP
IP
Address
MAC
Name
Strting

3] USE CASE DIAGRAM



4] Class Diagram :

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

Benefits of class diagrams

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation-independent description of types used in a system that are later passed between its components.

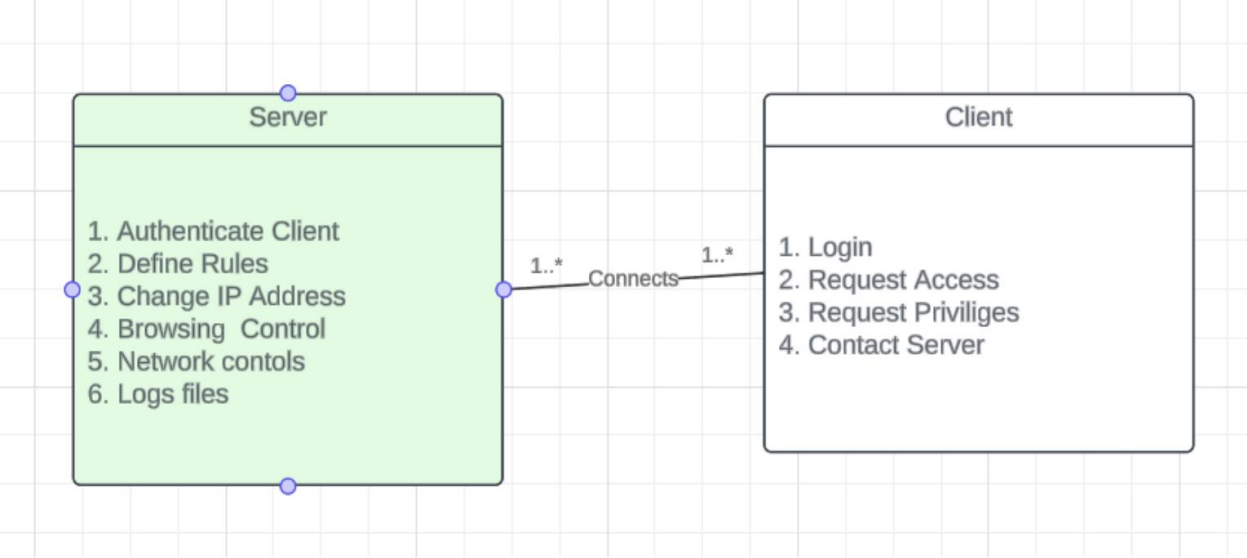
Basic components of a class diagram

The standard class diagram is composed of three sections: -

- **Upper section**: - Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middlesection**: - Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom section**: - Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed

modelling, the classes of the conceptual design are often split into a number of subclasses.



❖MODULE DESCRIPTION

1. **User Authentication and Authorization:**
   - Administrators must log in with secure credentials.
   - User roles (administrator, support staff) with appropriate permissions should be defined.

2. **Centralized Dashboard:**
   - Provide an intuitive dashboard for administrators to manage network resources and access key features.

3. **Remote Access and Support:**
   - Enable administrators to initiate secure remote connections to client computers.
   - Allow real-time screen sharing, remote control, and file transfer during remote sessions.

4. **Shared File Transfer:**
   - Implement a shared folder where administrators can upload and download files to/from client computers.
   - Ensure secure file transfer with proper access controls.

5. **Network Configuration Management:**
   - Allow administrators to view and modify network configurations, including IP addresses and DNS settings.
   - Provide a user-friendly interface for network configuration adjustments.

6. **Browsing Control and Filtering:**
   - Implement URL filtering and website blocking mechanisms.
   - Enable administrators to define browsing policies and control access to specific websites.

7. **Logging and Reporting:**
   - Log detailed information about network activities, remote sessions, file transfers, and configuration changes.

- Generate reports and visualizations for administrators to monitor network usage and performance.

8. Remote Troubleshooting Tools:
   - Offer tools for remote diagnostics, such as command-line access, system information retrieval, and remote task management.

9. User Management:
   - Allow administrators to manage user accounts, assign roles, and reset passwords.

10. Security Measures:
    - Implement encryption for data transmission between client and server.
    - Ensure that all sensitive data, including user credentials and network configurations, is stored securely.

11. Notification and Alerts:
    - Send notifications and alerts to administrators for critical events, such as unauthorized access attempts or configuration changes.

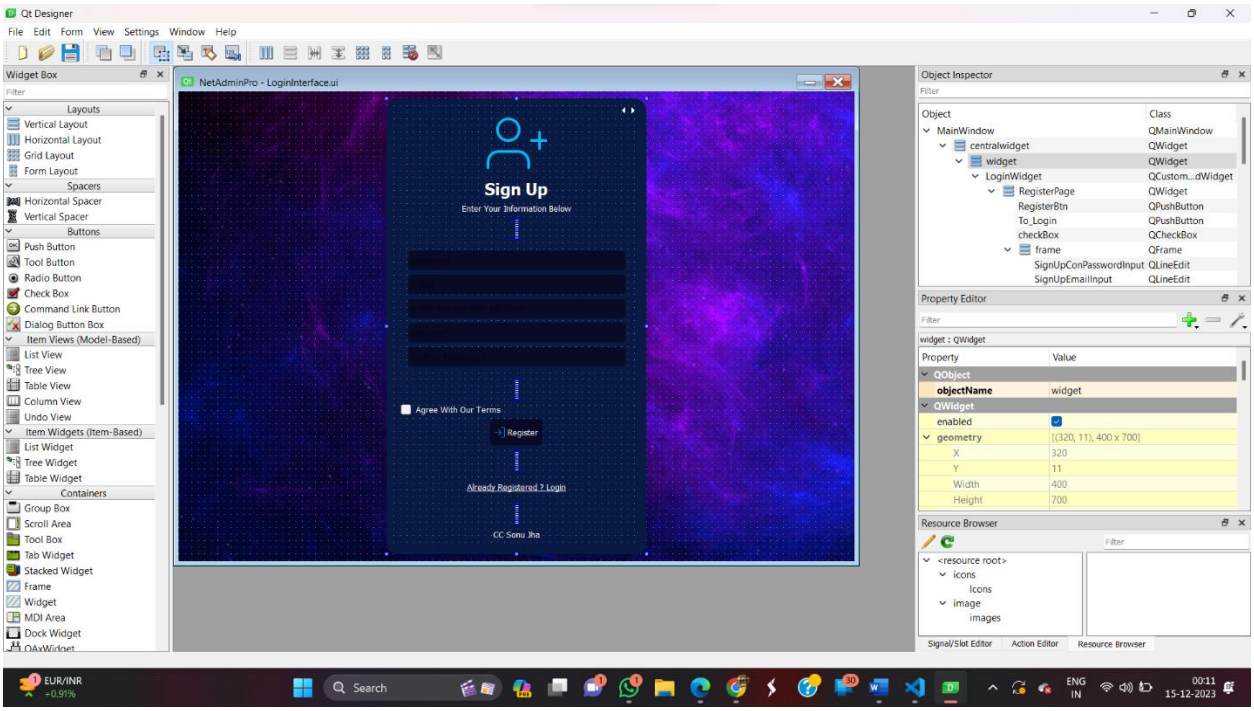
12. Documentation and Help Resources:
    - Provide user manuals and help resources for administrators to understand and use the application effectively.

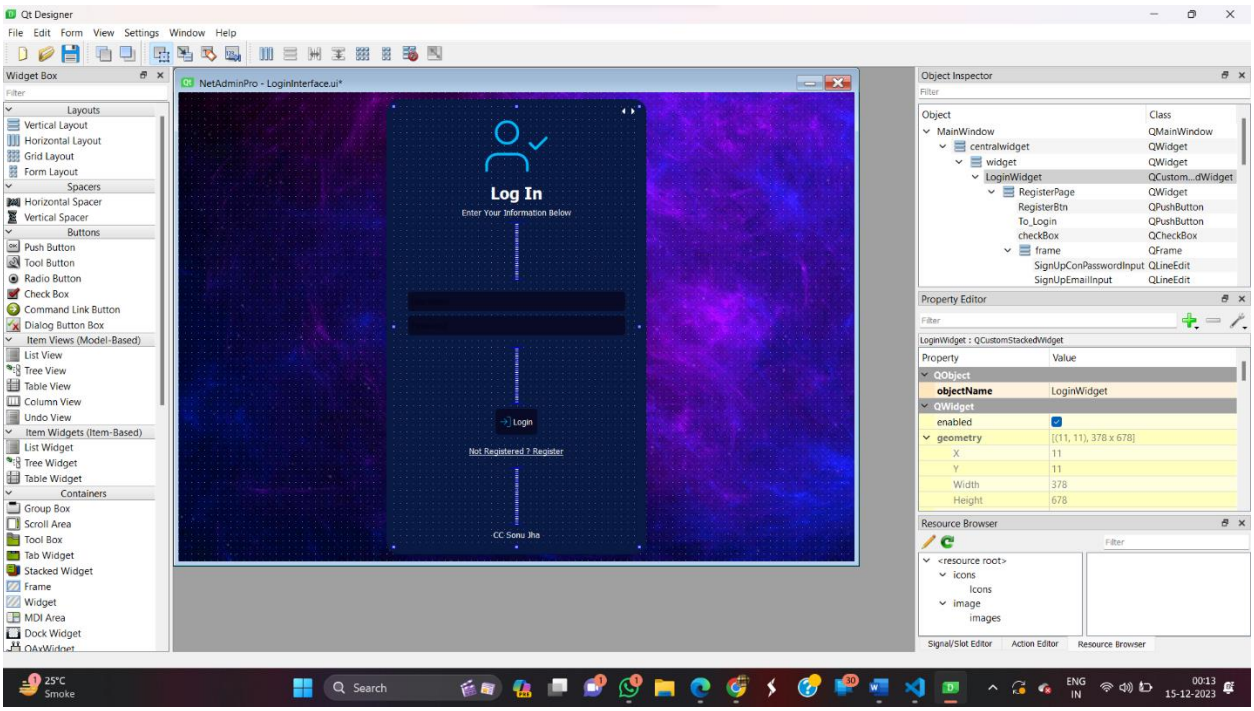
13. Data Backup and Recovery:
    - Implement mechanisms to regularly back up application data and configurations to prevent data loss.

❖RUNTIME OUTPUT & SCREENSHOT

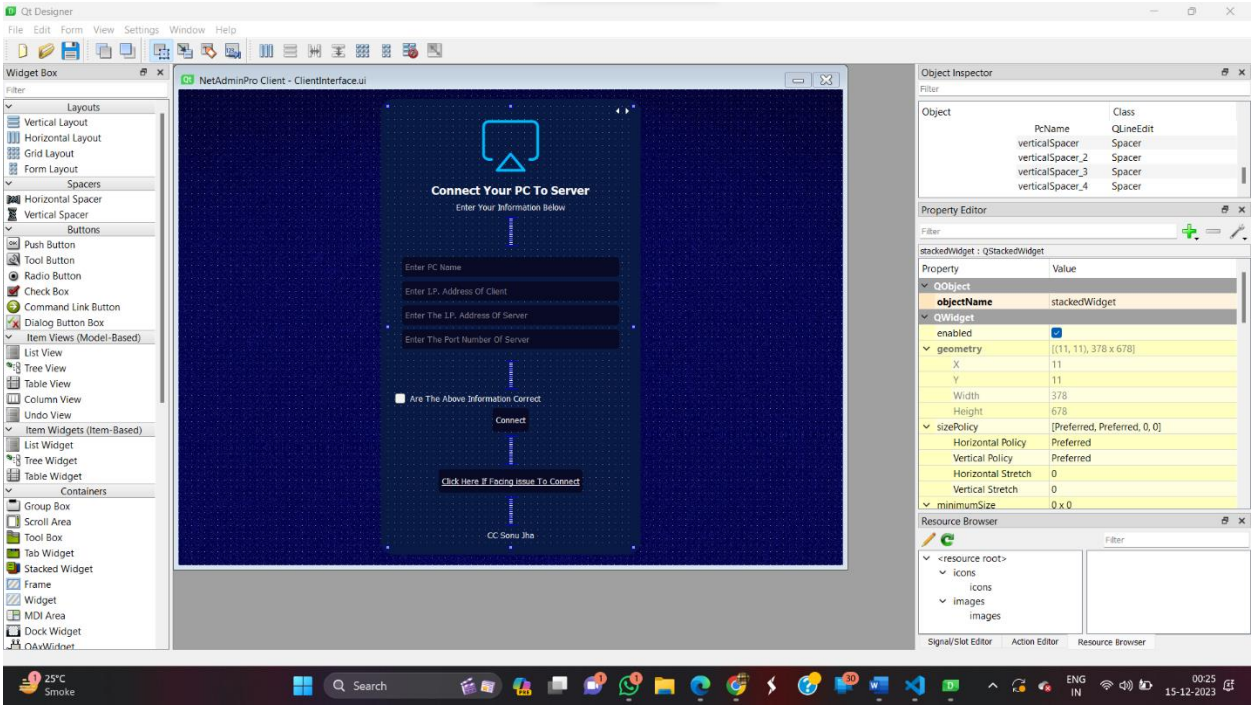
1. DESIGN IN UT DESIGNER

LOGIN UI

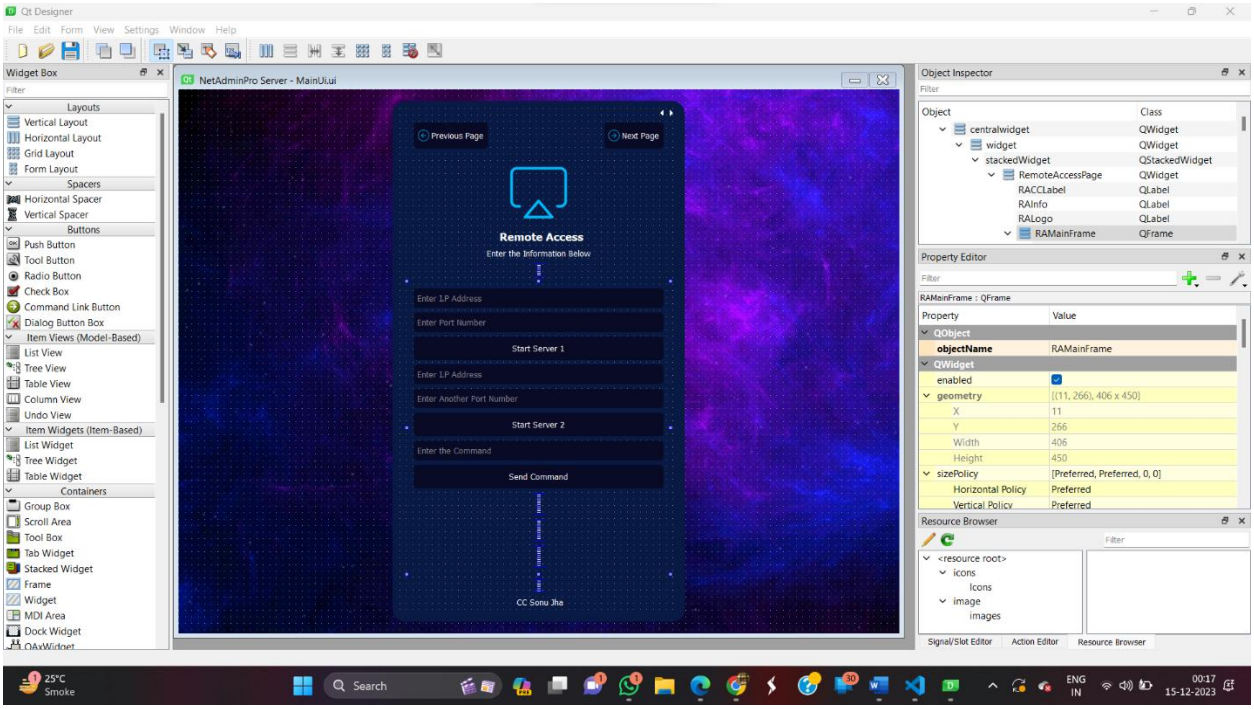


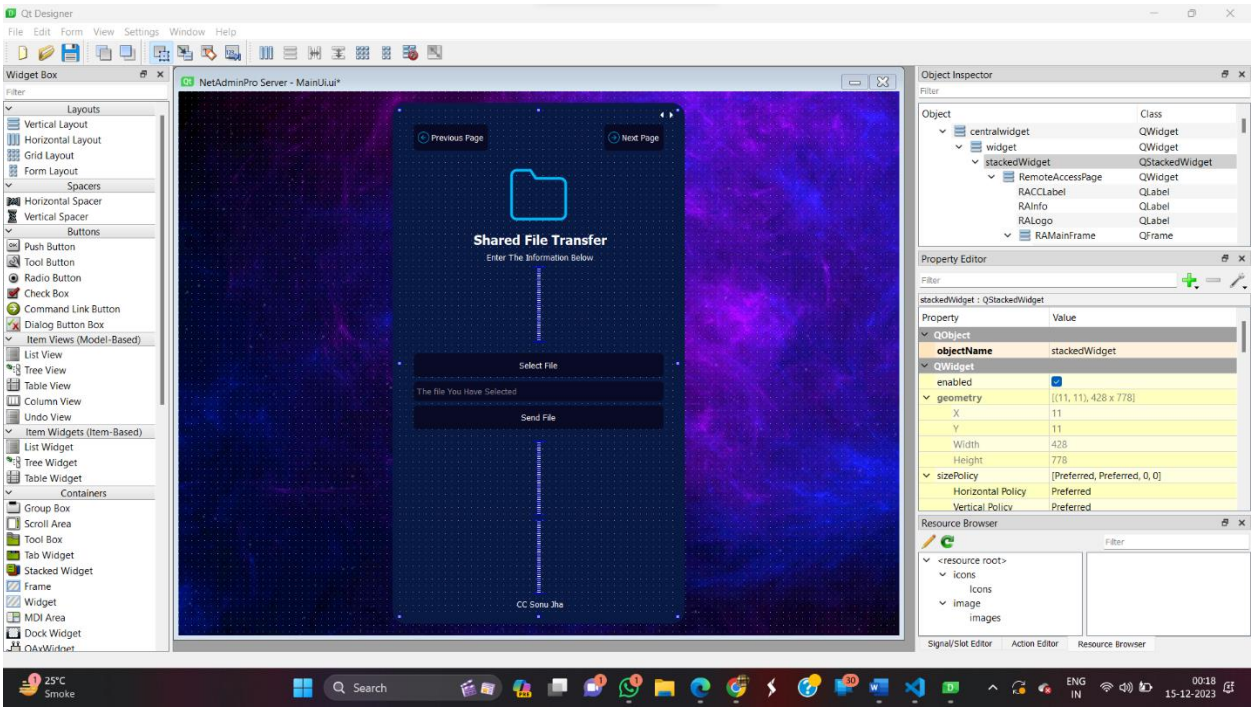

Qt Designer
File Edit Form View Settings Window Help
Widget Box
NetAdminPro - LoginInterface.ui*
Log In
Enter Your Information Below
Login
Not Registered ? Register
CC Sonu Jha
Object Inspector
Property Editor
Resource Browser
Signal/Slot Editor
Action Editor
Resource Browser

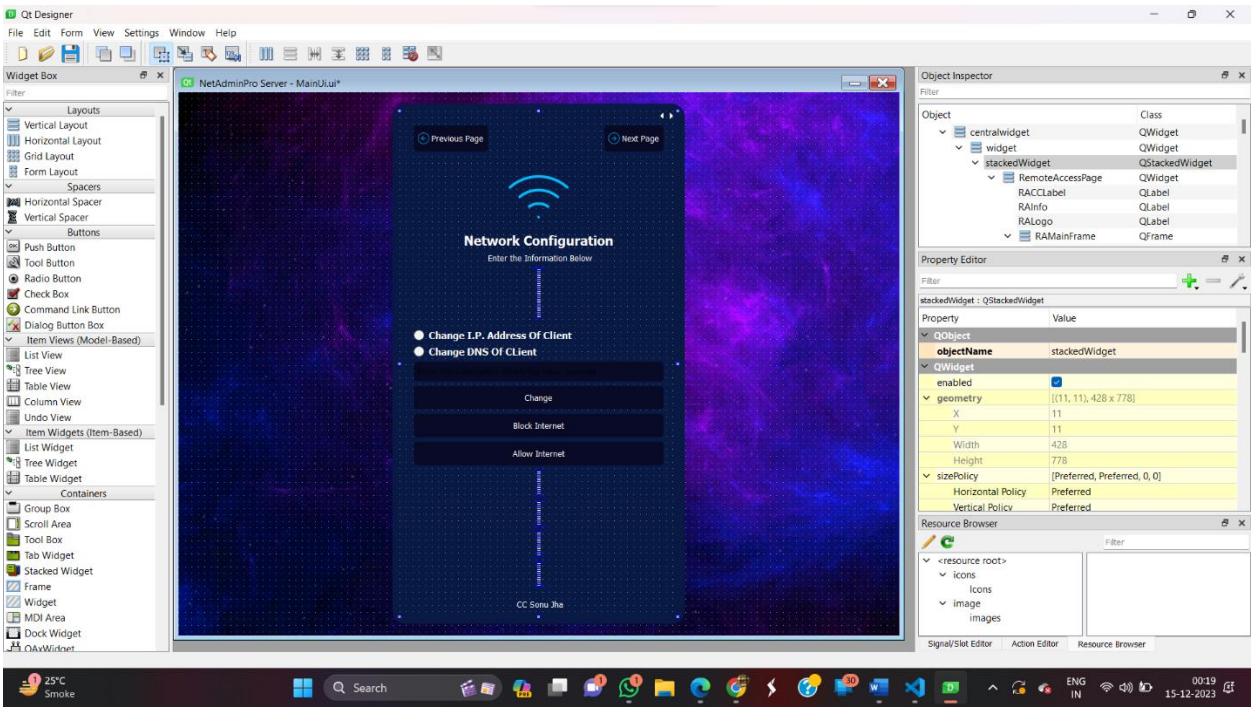
CLIENT UI

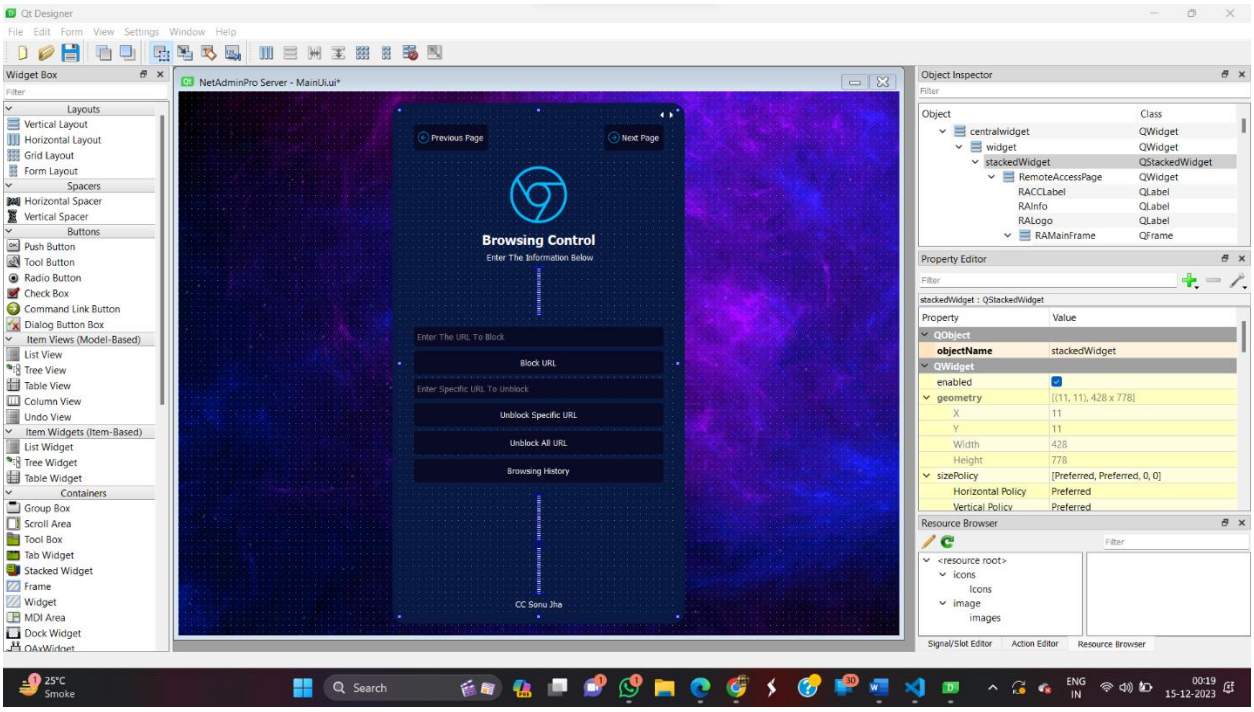

Qt Designer
File Edit Form View Settings Window Help
Widget Box
NetAdminPro Client - ClientInterface.ui
Connect Your PC To Server
Enter Your Information Below
Enter PC Name
Enter I.P. Address Of Client
Enter The I.P. Address Of Server
Enter The Port Number Of Server
Are The Above Information Correct
Connect
Click Here If Facing Issue To Connect
CC Sonu Jha
Object Inspector
Property Editor
Resource Browser
Signal/Slot Editor
Action Editor
Resource Browser

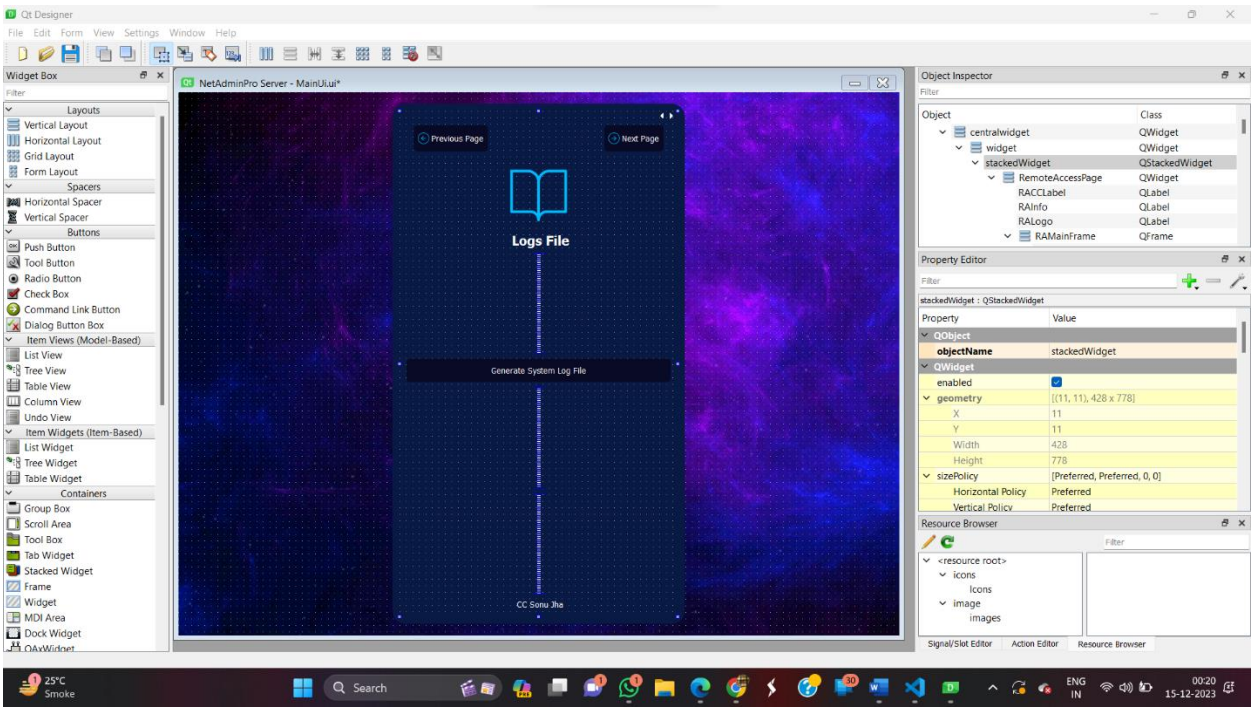
SERVER UI

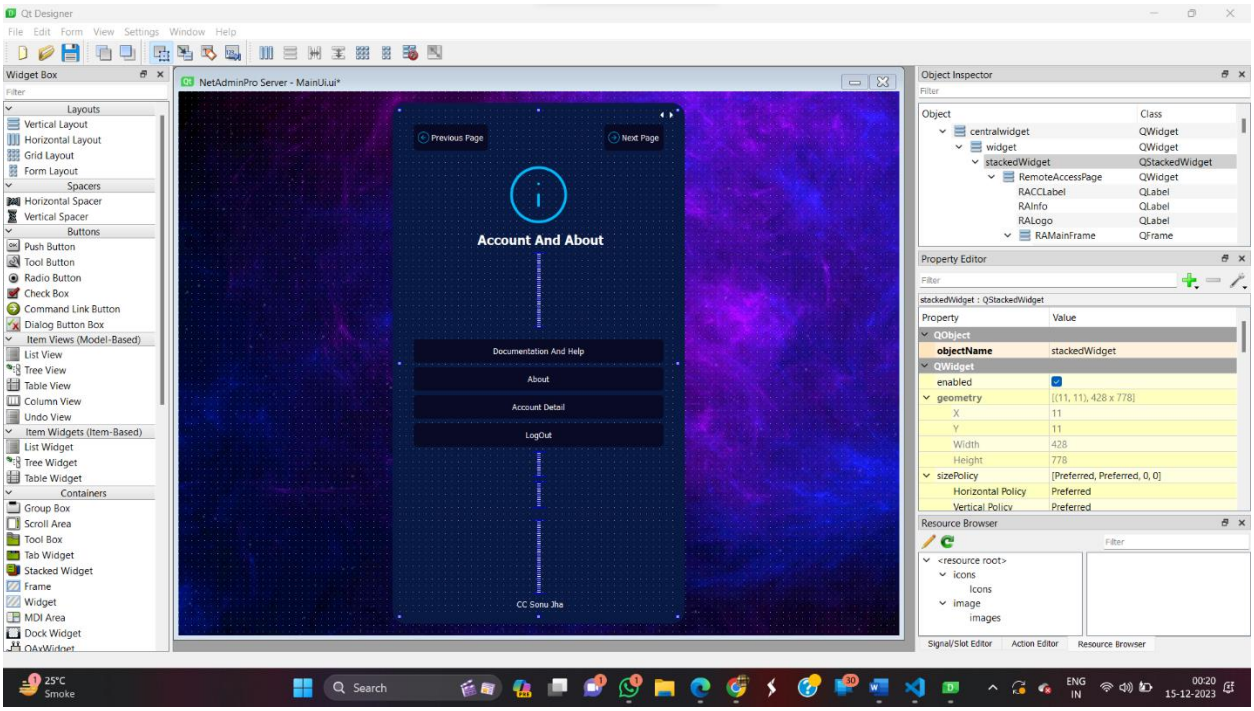

Qt Designer
NetAdminPro Server - MainUi.ui
Previous Page
Next Page
Remote Access
Enter the Information Below
Enter LP Address
Enter Port Number
Start Server 1
Enter LP Address
Enter Another Port Number
Start Server 2
Enter the Command
Send Command
CC Sonu Jha


Qt Designer
NetAdminPro Server - MainUi.ui*
Previous Page
Next Page
Shared File Transfer
Enter The Information Below
Select File
The file You Have Selected
Send File
CC Sonu Jha

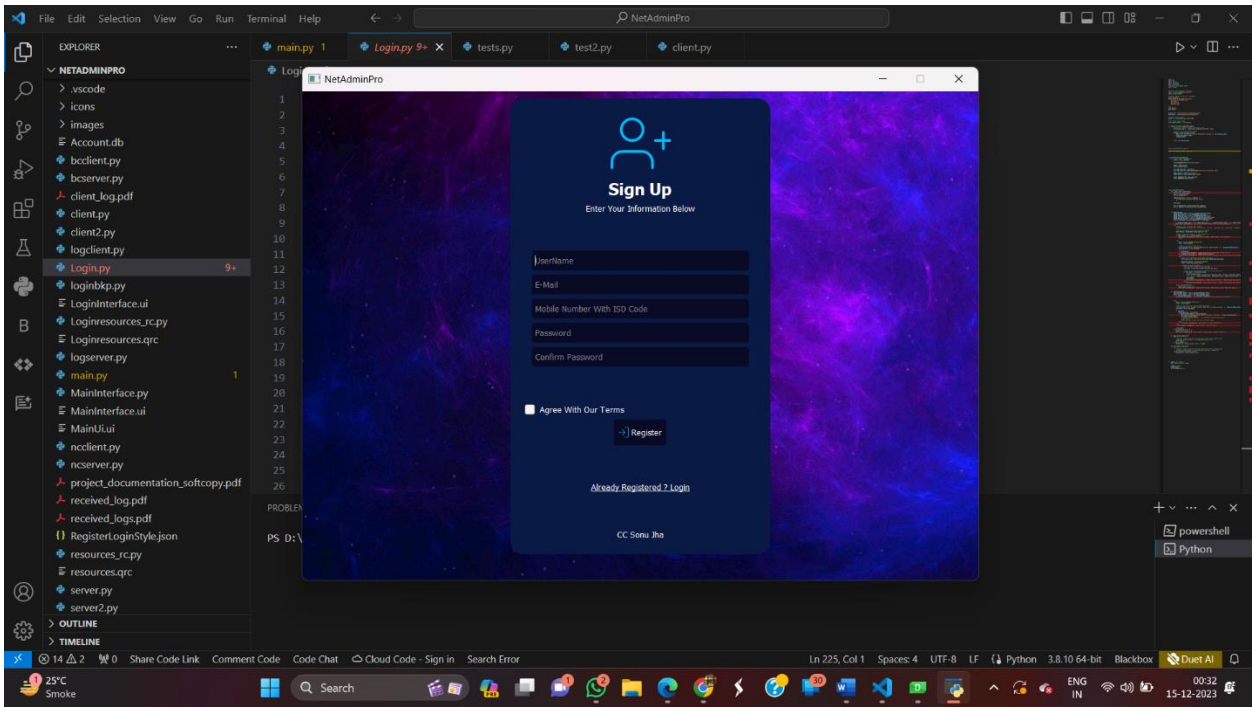

Qt Designer
File Edit Form View Settings Window Help
Widget Box
NetAdminPro Server - MainUi.ui*
Previous Page
Next Page
Network Configuration
Enter the Information Below
Change I.P. Address Of Client
Change DNS Of CLient
Change
Block Internet
Allow Internet
CC Sonu Jha
Object Inspector
Property Editor
Resource Browser
Signal/Slot Editor
Action Editor
Resource Browser

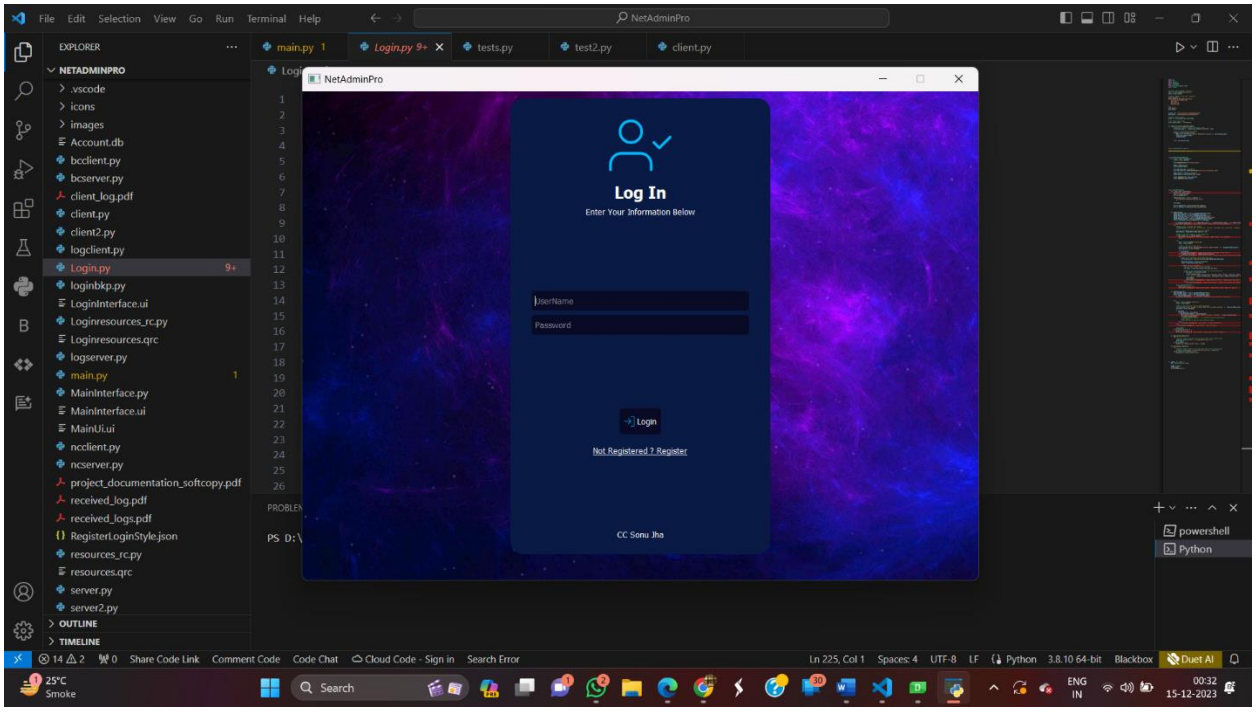

Qt Designer
File Edit Form View Settings Window Help
Widget Box
NetAdminPro Server - MainUi.ui*
Previous Page
Next Page
Browsing Control
Enter The Information Below
Enter The URL To Block
Block URL
Enter Specific URL To Unblock
Unblock Specific URL
Unblock All URL
Browsing History
CC Sonu Jha
Object Inspector
Property Editor
Resource Browser
Signal/Slot Editor
Action Editor
Resource Browser

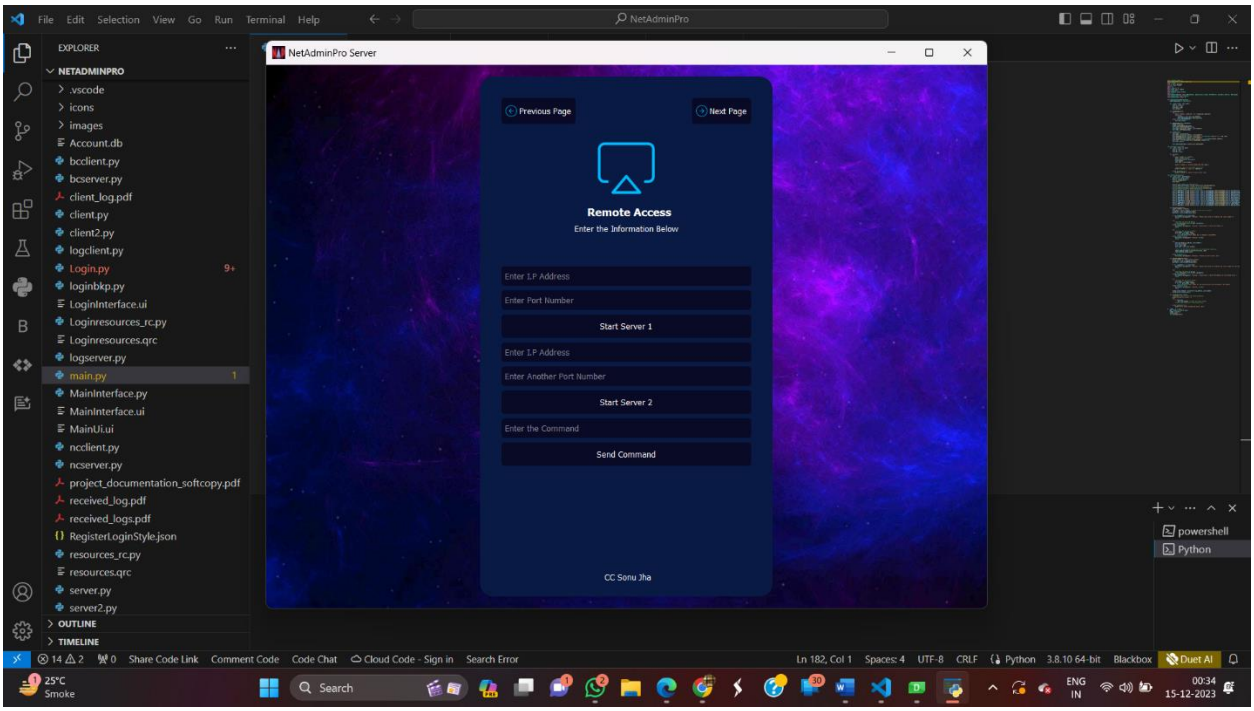

Qt Designer
File Edit Form View Settings Window Help
Widget Box
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Dock Widget
NetAdminPro Server - MainUi.ui*
Previous Page
Next Page
Logs File
Generate System Log File
CC Sonu Jha
Object Inspector
Object
Class
centralwidget
QWidget
widget
QWidget
stackedWidget
QStackedWidget
RemoteAccessPage
QWidget
RACCLabel
QLabel
RAInfo
QLabel
RALogo
QLabel
RAMainFrame
QFrame
Property Editor
stackedWidget : QStackedWidget
Property
Value
QObject
objectName
stackedWidget
QWidget
enabled
geometry
[(11, 11), 428 x 778]
X
11
Y
11
Width
428
Height
778
sizePolicy
[Preferred, Preferred, 0, 0]
Horizontal Policy
Preferred
Vertical Policy
Preferred
Resource Browser
<resource root>
icons
Icons
image
images
Signal/Slot Editor
Action Editor
Resource Browser
25°C
Smoke
Search
ENG
IN
00:20
15-12-2023

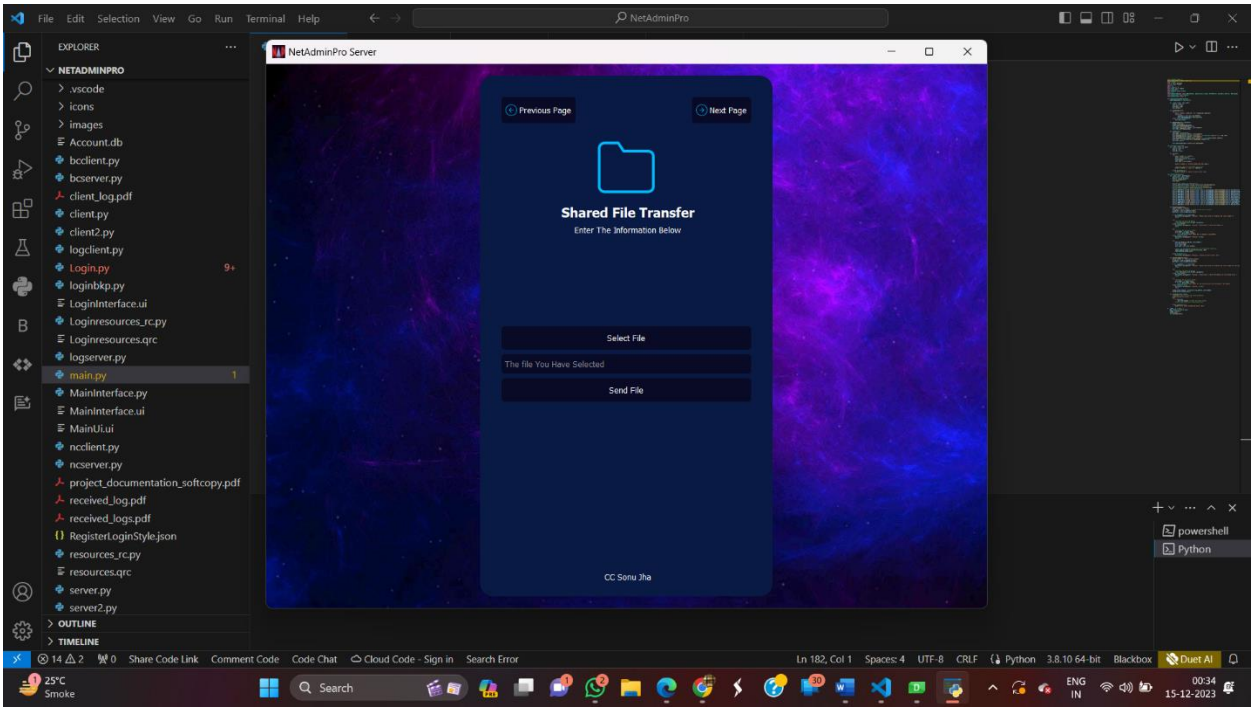

Qt Designer
File Edit Form View Settings Window Help
Widget Box
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Dock Widget
NetAdminPro Server - MainUi.ui*
Previous Page
Next Page
Account And About
Documentation And Help
About
Account Detail
LogOut
CC Sonu Jha
Object Inspector
Object
Class
centralwidget
QWidget
widget
QWidget
stackedWidget
QStackedWidget
RemoteAccessPage
QWidget
RACCLabel
QLabel
RAInfo
QLabel
RALogo
QLabel
RAMainFrame
QFrame
Property Editor
stackedWidget : QStackedWidget
Property
Value
QObject
objectName
stackedWidget
QWidget
enabled
geometry
[(11, 11), 428 x 778]
X
11
Y
11
Width
428
Height
778
sizePolicy
[Preferred, Preferred, 0, 0]
Horizontal Policy
Preferred
Vertical Policy
Preferred
Resource Browser
<resource root>
icons
Icons
image
images
Signal/Slot Editor
Action Editor
Resource Browser
25°C
Smoke
Search
ENG
IN
00:20
15-12-2023

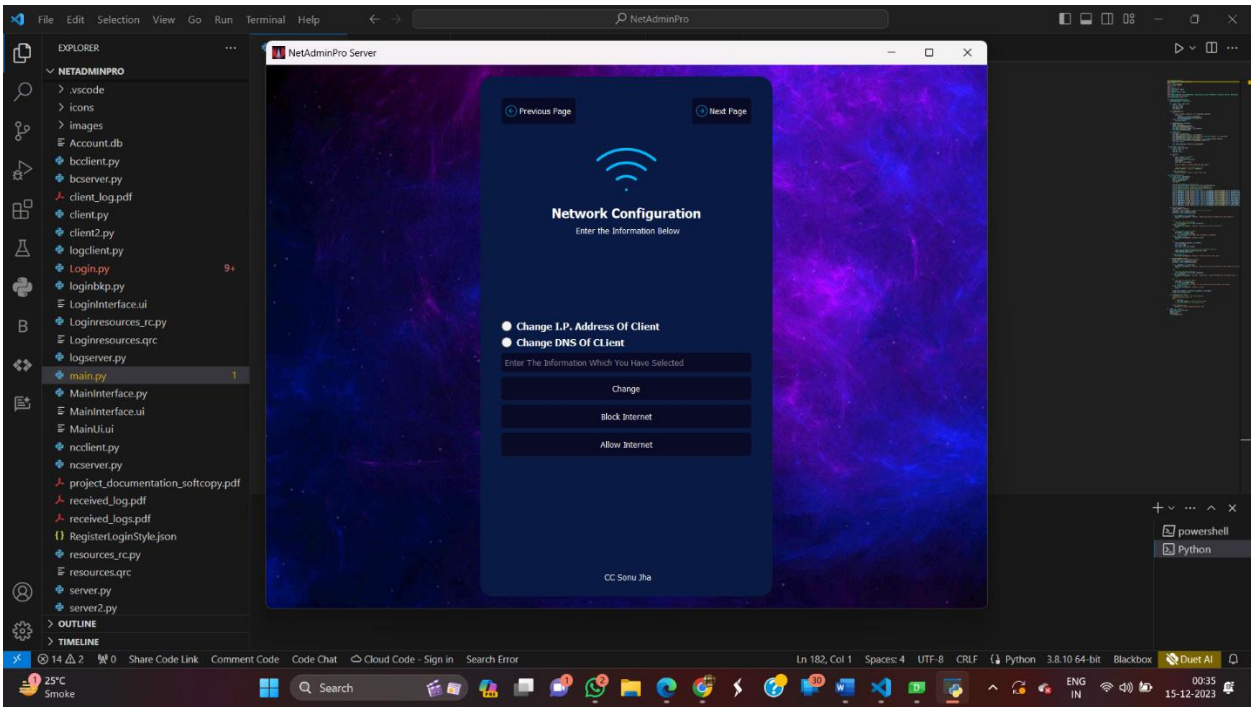
2. RUNTIME DESIGN OUTPUT

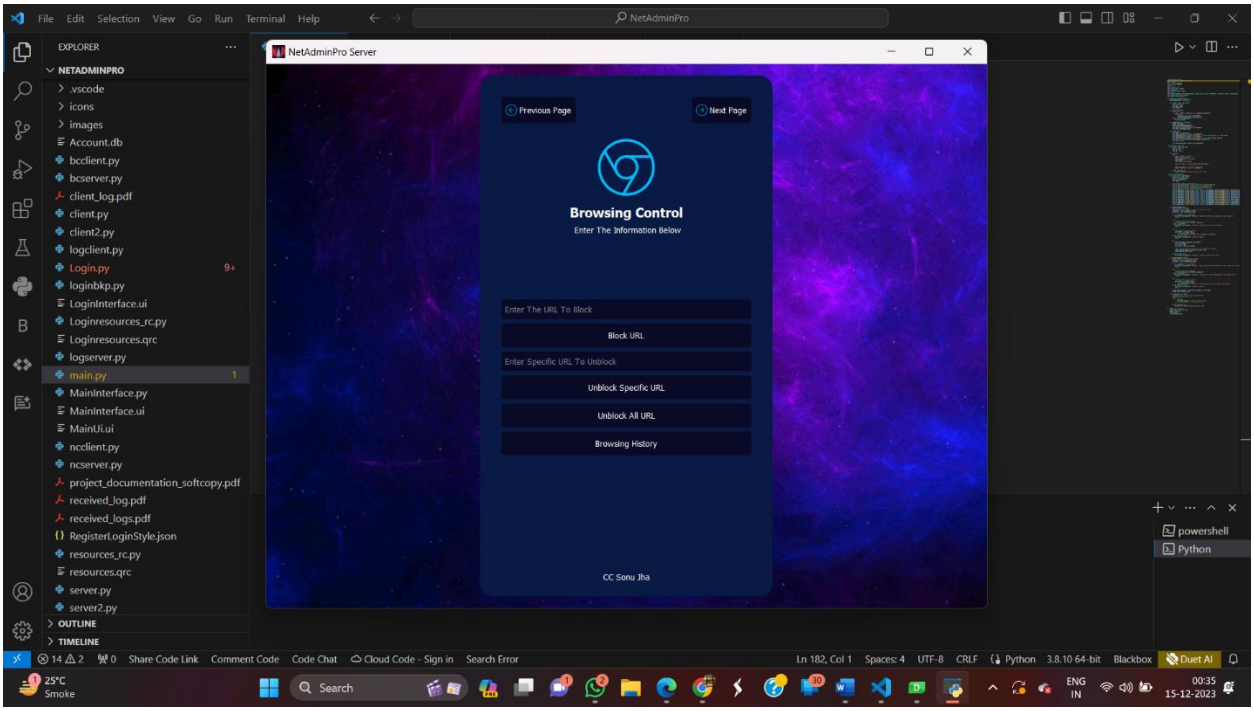


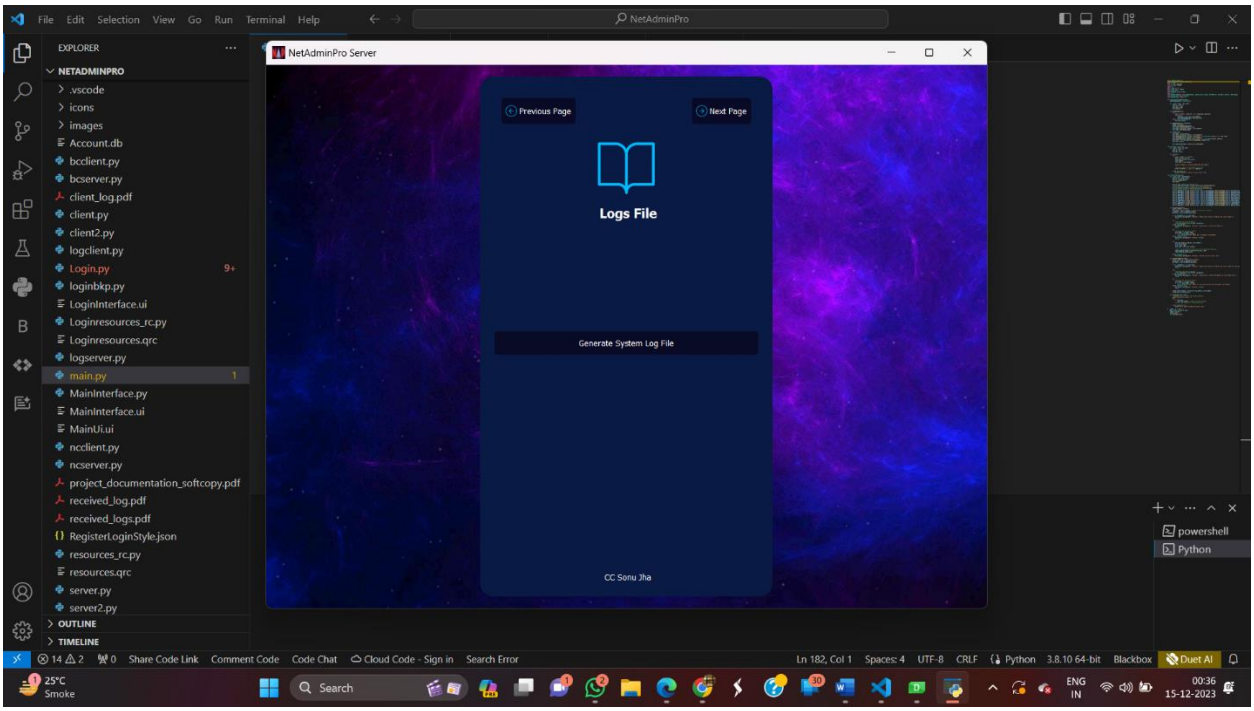


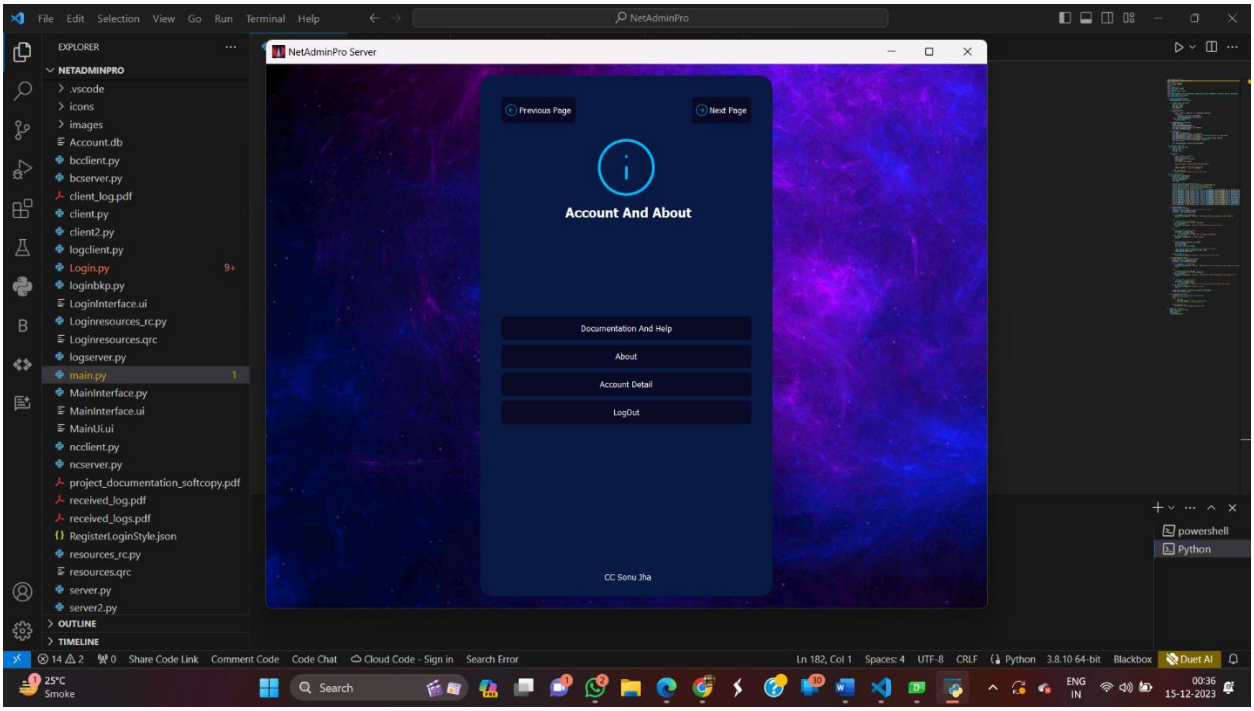

NetAdminPro Server
Previous Page
Next Page
Remote Access
Enter the Information Below
Enter I.P Address
Enter Port Number
Start Server 1
Enter I.P Address
Enter Another Port Number
Start Server 2
Enter the Command
Send Command
CC Sonu Jha

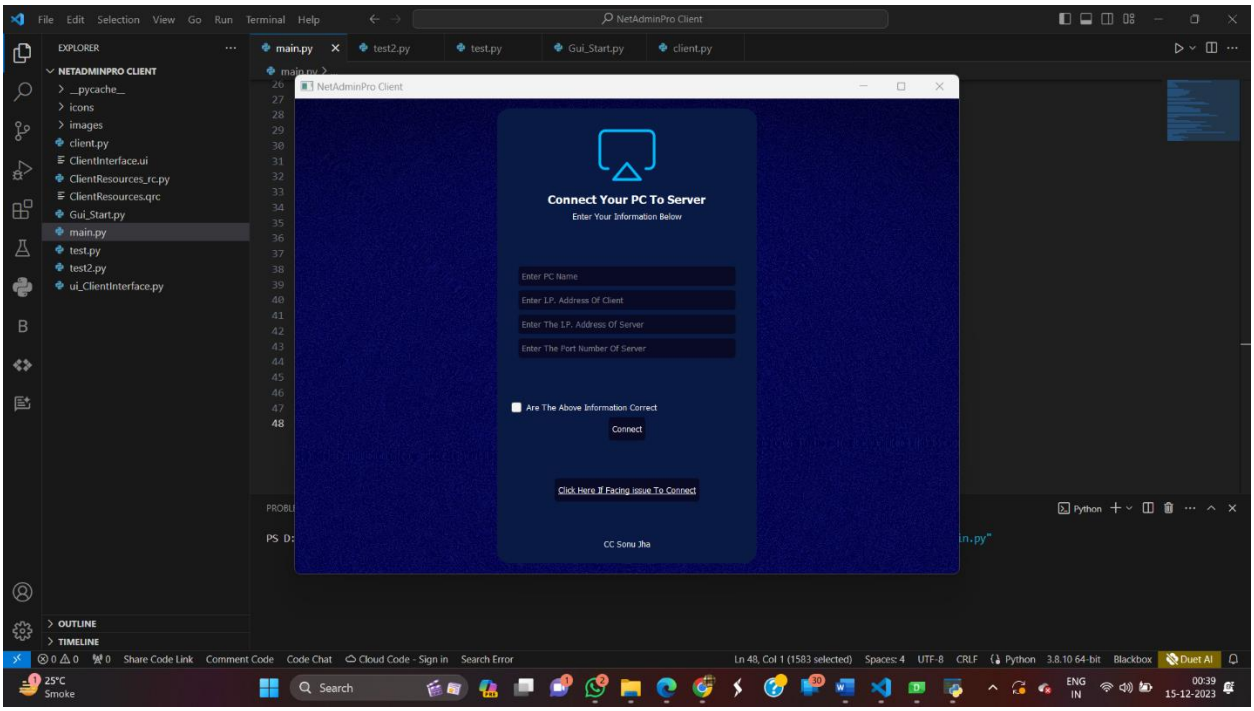

NetAdminPro Server
Previous Page
Next Page
Shared File Transfer
Enter The Information Below
Select File
The file You Have Selected
Send File
CC Sonu Jha


NetAdminPro Server
Previous Page
Next Page
Network Configuration
Enter the Information Below
Change I.P. Address Of Client
Change DNS Of CLient
Enter The Information Which You Have Selected
Change
Block Internet
Allow Internet
CC Sonu Jha


NetAdminPro Server
Previous Page
Next Page
Browsing Control
Enter The Information Below
Enter The URL To Block
Block URL
Enter Specific URL To Unblock
Unblock Specific URL
Unblock All URL
Browsing History
CC Sonu Jha


NetAdminPro Server
Previous Page
Next Page
Logs File
Generate System Log File
CC Sonu Jha


NetAdminPro Server
Previous Page
Next Page
Account And About
Documentation And Help
About
Account Detail
LogOut
CC Sonu Jha


NetAdminPro Client
Connect Your PC To Server
Enter Your Information Below
Enter PC Name
Enter I.P. Address Of Client
Enter The I.P. Address Of Server
Enter The Port Number Of Server
Are The Above Information Correct
Connect
Click Here If Facing Issue To Connect
CC Sonu Jha

❖CODE

1. UI_LOGININTERFACE.PY

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'LoginInterface.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide2.QtCore import *
from PySide2.QtGui import *
from PySide2.QtWidgets import *

from Custom_Widgets.Widgets import QCustomStackedWidget

import Loginresources_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(1040, 722)
        MainWindow.setMinimumSize(QSize(0, 0))
        MainWindow.setMaximumSize(QSize(1040, 722))
        MainWindow.setStyleSheet(u"*{\n"
"   border: none;\n"
"   background-color: transparent;\n"
"   background: transparent;\n"
"   padding: 0;\n"
"   margin: 0;\n"
"   color: #fff;\n"
"}\n"
"#centralwidget{\n"
"   background-color: ;\n"
"   background-image: url(:/image/images/bg.jpg);\n"
"}\n"
"#widget{\n"
"   background-color: rgb(9, 27, 68);\n"
"   border-radius:20px;\n"
"}\n"
```

```
"QLineEdit{\n"
"   background-color: rgb(9, 10, 37);\n"
"   padding: 5px 3px;\n"
"   border-radius:5px;\n"
"   \n"
"}\n"
"QPushButton{\n"
"   background-color:  rgb(9, 10, 37);\n"
"   padding: 10px 5px;\n"
"   border-radius:5px;\n"
"}\n"
"#To_Login , #To_Register{\n"
"   background-color: transparent;\n"
"}\n"
"")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.verticalLayout = QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName(u"verticalLayout")
        self.widget = QWidget(self.centralwidget)
        self.widget.setObjectName(u"widget")
        self.widget.setMinimumSize(QSize(400, 700))
        self.widget.setMaximumSize(QSize(400, 700))
        self.verticalLayout_2 = QVBoxLayout(self.widget)
        self.verticalLayout_2.setObjectName(u"verticalLayout_2")
        self.LoginWidget = QCustomStackedWidget(self.widget)
        self.LoginWidget.setObjectName(u"LoginWidget")
        self.RegisterPage = QWidget()
        self.RegisterPage.setObjectName(u"RegisterPage")
        self.verticalLayout_3 = QVBoxLayout(self.RegisterPage)
        self.verticalLayout_3.setObjectName(u"verticalLayout_3")
        self.label = QLabel(self.RegisterPage)
        self.label.setObjectName(u"label")
        self.label.setMinimumSize(QSize(95, 95))
        self.label.setMaximumSize(QSize(95, 95))
        self.label.setPixmap(QPixmap(u":/icons/Icons/user-plus.png"))

        self.verticalLayout_3.addWidget(self.label, 0, Qt.AlignHCenter)

        self.label_2 = QLabel(self.RegisterPage)
        self.label_2.setObjectName(u"label_2")
        font = QFont()
        font.setPointSize(15)
        font.setBold(True)
        font.setWeight(75)
```

```
        self.label_2.setFont(font)

        self.verticalLayout_3.addWidget(self.label_2, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.label_3 = QLabel(self.RegisterPage)
        self.label_3.setObjectName(u"label_3")

        self.verticalLayout_3.addWidget(self.label_3, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.verticalSpacer = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer)

        self.frame = QFrame(self.RegisterPage)
        self.frame.setObjectName(u"frame")
        self.frame.setFrameShape(QFrame.StyledPanel)
        self.frame.setFrameShadow(QFrame.Raised)
        self.verticalLayout_4 = QVBoxLayout(self.frame)
        self.verticalLayout_4.setObjectName(u"verticalLayout_4")
        self.SignUpUserNameInput = QLineEdit(self.frame)
        self.SignUpUserNameInput.setObjectName(u"SignUpUserNameInput")

        self.verticalLayout_4.addWidget(self.SignUpUserNameInput)

        self.SignUpEmailInput = QLineEdit(self.frame)
        self.SignUpEmailInput.setObjectName(u"SignUpEmailInput")

        self.verticalLayout_4.addWidget(self.SignUpEmailInput)

        self.SignUpMobileNumInput = QLineEdit(self.frame)
        self.SignUpMobileNumInput.setObjectName(u"SignUpMobileNumInput")

        self.verticalLayout_4.addWidget(self.SignUpMobileNumInput)

        self.SignUpPasswordInput = QLineEdit(self.frame)
        self.SignUpPasswordInput.setObjectName(u"SignUpPasswordInput")
        self.SignUpPasswordInput.setEchoMode(QLineEdit.Password)

        self.verticalLayout_4.addWidget(self.SignUpPasswordInput)

        self.SignUpConPasswordInput = QLineEdit(self.frame)
        self.SignUpConPasswordInput.setObjectName(u"SignUpConPasswordInput")
```

```
        self.SignUpConPasswordInput.setEchoMode(QLineEdit.Password)

        self.verticalLayout_4.addWidget(self.SignUpConPasswordInput)


        self.verticalLayout_3.addWidget(self.frame)

        self.verticalSpacer_2 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_2)

        self.checkBox = QCheckBox(self.RegisterPage)
        self.checkBox.setObjectName(u"checkBox")

        self.verticalLayout_3.addWidget(self.checkBox)

        self.RegisterBtn = QPushButton(self.RegisterPage)
        self.RegisterBtn.setObjectName(u"RegisterBtn")
        icon = QIcon()
        icon.addFile(u":/icons/Icons/log-in.png", QSize(), QIcon.Normal,
QIcon.Off)
        self.RegisterBtn.setIcon(icon)

        self.verticalLayout_3.addWidget(self.RegisterBtn, 0, Qt.AlignHCenter)

        self.verticalSpacer_3 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_3)

        self.To_Login = QPushButton(self.RegisterPage)
        self.To_Login.setObjectName(u"To_Login")
        font1 = QFont()
        font1.setUnderline(True)
        self.To_Login.setFont(font1)

        self.verticalLayout_3.addWidget(self.To_Login, 0, Qt.AlignHCenter)

        self.verticalSpacer_4 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_4)

        self.label_4 = QLabel(self.RegisterPage)
```

```
        self.label_4.setObjectName(u"label_4")

        self.verticalLayout_3.addWidget(self.label_4, 0, Qt.AlignHCenter)

        self.LoginWidget.addWidget(self.RegisterPage)
        self.LoginPage = QWidget()
        self.LoginPage.setObjectName(u"LoginPage")
        self.verticalLayout_6 = QVBoxLayout(self.LoginPage)
        self.verticalLayout_6.setObjectName(u"verticalLayout_6")
        self.label_5 = QLabel(self.LoginPage)
        self.label_5.setObjectName(u"label_5")
        self.label_5.setMinimumSize(QSize(100, 100))
        self.label_5.setMaximumSize(QSize(100, 100))
        self.label_5.setPixmap(QPixmap(u":/icons/Icons/user-check.png"))

        self.verticalLayout_6.addWidget(self.label_5, 0, Qt.AlignHCenter)

        self.label_8 = QLabel(self.LoginPage)
        self.label_8.setObjectName(u"label_8")
        self.label_8.setFont(font)

        self.verticalLayout_6.addWidget(self.label_8, 0, Qt.AlignHCenter)

        self.label_7 = QLabel(self.LoginPage)
        self.label_7.setObjectName(u"label_7")

        self.verticalLayout_6.addWidget(self.label_7, 0, Qt.AlignHCenter)

        self.verticalSpacer_6 = QSpacerItem(20, 87, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_6.addItem(self.verticalSpacer_6)

        self.frame_2 = QFrame(self.LoginPage)
        self.frame_2.setObjectName(u"frame_2")
        self.frame_2.setFrameShape(QFrame.StyledPanel)
        self.frame_2.setFrameShadow(QFrame.Raised)
        self.verticalLayout_5 = QVBoxLayout(self.frame_2)
        self.verticalLayout_5.setObjectName(u"verticalLayout_5")
        self.LoginUserNameInput = QLineEdit(self.frame_2)
        self.LoginUserNameInput.setObjectName(u"LoginUserNameInput")

        self.verticalLayout_5.addWidget(self.LoginUserNameInput)

        self.LoginPasswordInput = QLineEdit(self.frame_2)
```

```
        self.LoginPasswordInput.setObjectName(u"LoginPasswordInput")
        self.LoginPasswordInput.setEchoMode(QLineEdit.Password)

        self.verticalLayout_5.addWidget(self.LoginPasswordInput)


        self.verticalLayout_6.addWidget(self.frame_2)

        self.verticalSpacer_7 = QSpacerItem(20, 87, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_6.addItem(self.verticalSpacer_7)

        self.LoginBtn = QPushButton(self.LoginPage)
        self.LoginBtn.setObjectName(u"LoginBtn")
        self.LoginBtn.setIcon(icon)

        self.verticalLayout_6.addWidget(self.LoginBtn, 0, Qt.AlignHCenter)

        self.To_Register = QPushButton(self.LoginPage)
        self.To_Register.setObjectName(u"To_Register")
        self.To_Register.setFont(font1)

        self.verticalLayout_6.addWidget(self.To_Register, 0, Qt.AlignHCenter)

        self.verticalSpacer_5 = QSpacerItem(20, 87, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_6.addItem(self.verticalSpacer_5)

        self.label_6 = QLabel(self.LoginPage)
        self.label_6.setObjectName(u"label_6")

        self.verticalLayout_6.addWidget(self.label_6, 0, Qt.AlignHCenter)

        self.LoginWidget.addWidget(self.LoginPage)

        self.verticalLayout_2.addWidget(self.LoginWidget)


        self.verticalLayout.addWidget(self.widget, 0, Qt.AlignHCenter)

        self.widget_2 = QWidget(self.centralwidget)
        self.widget_2.setObjectName(u"widget_2")
        self.horizontalLayout = QHBoxLayout(self.widget_2)
```

```
        self.horizontalLayout.setObjectName(u"horizontalLayout")

        self.verticalLayout.addWidget(self.widget_2)

        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)

        self.LoginWidget.setCurrentIndex(0)


        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"NetAdminPro", None))
        self.label.setText("")
        self.label_2.setText(QCoreApplication.translate("MainWindow", u"Sign Up",
None))
        self.label_3.setText(QCoreApplication.translate("MainWindow", u"Enter
Your Information Below", None))
        self.SignUpUserNameInput.setPlaceholderText(QCoreApplication.translate("M
ainWindow", u"UserName", None))
        self.SignUpEmailInput.setPlaceholderText(QCoreApplication.translate("Main
Window", u"E-Mail", None))
        self.SignUpMobileNumInput.setPlaceholderText(QCoreApplication.translate("
MainWindow", u"Mobile Number With ISD Code", None))
        self.SignUpPasswordInput.setPlaceholderText(QCoreApplication.translate("M
ainWindow", u"Password", None))
        self.SignUpConPasswordInput.setPlaceholderText(QCoreApplication.translate
("MainWindow", u"Confirm Password", None))
        self.checkBox.setText(QCoreApplication.translate("MainWindow", u"Agree
With Our Terms", None))
        self.RegisterBtn.setText(QCoreApplication.translate("MainWindow",
u"Register", None))
        self.To_Login.setText(QCoreApplication.translate("MainWindow", u"Already
Registered ? Login", None))
        self.label_4.setText(QCoreApplication.translate("MainWindow", u"CC Sonu
Jha", None))
        self.label_5.setText("")
        self.label_8.setText(QCoreApplication.translate("MainWindow", u"Log In",
None))
        self.label_7.setText(QCoreApplication.translate("MainWindow", u"Enter
Your Information Below", None))
```

```
        self.LoginUserNameInput.setPlaceholderText(QCoreApplication.translate("Ma
inWindow", u"UserName", None))
        self.LoginPasswordInput.setPlaceholderText(QCoreApplication.translate("Ma
inWindow", u"Password", None))
        self.LoginBtn.setText(QCoreApplication.translate("MainWindow", u"Login",
None))
        self.To_Register.setText(QCoreApplication.translate("MainWindow", u"Not
Registered ? Register", None))
        self.label_6.setText(QCoreApplication.translate("MainWindow", u"CC Sonu
Jha", None))
    # retranslateUi
```

UI_MAIN.PY

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'ClientInterfaceyEybaH.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide2.QtCore import *
from PySide2.QtGui import *
from PySide2.QtWidgets import *

import ClientResources_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(1022, 731)
        MainWindow.setStyleSheet(u"*{\n"
"   border: none;\n"
"   background-color: transparent;\n"
"   background: transparent;\n"
"   padding: 0;\n"
"   margin: 0;\n"
```

```
"   color: #fff;\n"
"}\n"
"\n"
"#centralwidget{\n"
"   background-image: url(:/images/images/BgClient.jpg);\n"
"}\n"
"\n"
"#widget{\n"
"   background-color: rgb(9, 27, 68);\n"
"   border-radius:20px;\n"
"}\n"
"\n"
"QLineEdit{\n"
"   background-color: rgb(9, 10, 37);\n"
"   padding: 5px 3px;\n"
"   border-radius:5px;\n"
"   \n"
"}\n"
"QPushButton{\n"
"   background-color:  rgb(9, 10, 37);\n"
"   padding: 10px 5px;\n"
"   border-radius:5px;\n"
"}")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.verticalLayout = QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName(u"verticalLayout")
        self.widget = QWidget(self.centralwidget)
        self.widget.setObjectName(u"widget")
        self.widget.setMinimumSize(QSize(400, 700))
        self.widget.setMaximumSize(QSize(400, 700))
        self.verticalLayout_2 = QVBoxLayout(self.widget)
        self.verticalLayout_2.setObjectName(u"verticalLayout_2")
        self.stackedWidget = QStackedWidget(self.widget)
        self.stackedWidget.setObjectName(u"stackedWidget")
        self.InformationPage = QWidget()
        self.InformationPage.setObjectName(u"InformationPage")
        self.verticalLayout_3 = QVBoxLayout(self.InformationPage)
        self.verticalLayout_3.setObjectName(u"verticalLayout_3")
        self.InfoLogo = QLabel(self.InformationPage)
        self.InfoLogo.setObjectName(u"InfoLogo")
        self.InfoLogo.setPixmap(QPixmap(u":/icons/icons/airplay.png"))

        self.verticalLayout_3.addWidget(self.InfoLogo, 0, Qt.AlignHCenter)
```

```
        self.InfoText2 = QLabel(self.InformationPage)
        self.InfoText2.setObjectName(u"InfoText2")
        font = QFont()
        font.setPointSize(11)
        font.setBold(True)
        font.setWeight(75)
        self.InfoText2.setFont(font)

        self.verticalLayout_3.addWidget(self.InfoText2, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.InfoText = QLabel(self.InformationPage)
        self.InfoText.setObjectName(u"InfoText")

        self.verticalLayout_3.addWidget(self.InfoText, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.verticalSpacer = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer)

        self.UserInputInfo = QFrame(self.InformationPage)
        self.UserInputInfo.setObjectName(u"UserInputInfo")
        self.UserInputInfo.setFrameShape(QFrame.StyledPanel)
        self.UserInputInfo.setFrameShadow(QFrame.Raised)
        self.verticalLayout_4 = QVBoxLayout(self.UserInputInfo)
        self.verticalLayout_4.setObjectName(u"verticalLayout_4")
        self.PcName = QLineEdit(self.UserInputInfo)
        self.PcName.setObjectName(u"PcName")

        self.verticalLayout_4.addWidget(self.PcName)

        self.InfoClientIpAddr = QLineEdit(self.UserInputInfo)
        self.InfoClientIpAddr.setObjectName(u"InfoClientIpAddr")

        self.verticalLayout_4.addWidget(self.InfoClientIpAddr)

        self.InfoServerIpAddr = QLineEdit(self.UserInputInfo)
        self.InfoServerIpAddr.setObjectName(u"InfoServerIpAddr")

        self.verticalLayout_4.addWidget(self.InfoServerIpAddr)

        self.InfoServerPort = QLineEdit(self.UserInputInfo)
        self.InfoServerPort.setObjectName(u"InfoServerPort")
```

```
        self.verticalLayout_4.addWidget(self.InfoServerPort)


        self.verticalLayout_3.addWidget(self.UserInputInfo)

        self.verticalSpacer_2 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_2)

        self.InfoCheckBox = QCheckBox(self.InformationPage)
        self.InfoCheckBox.setObjectName(u"InfoCheckBox")

        self.verticalLayout_3.addWidget(self.InfoCheckBox)

        self.InfoConnectBTn = QPushButton(self.InformationPage)
        self.InfoConnectBTn.setObjectName(u"InfoConnectBTn")

        self.verticalLayout_3.addWidget(self.InfoConnectBTn, 0, Qt.AlignHCenter)

        self.verticalSpacer_4 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_4)

        self.InfoConnectIssueBtn = QPushButton(self.InformationPage)
        self.InfoConnectIssueBtn.setObjectName(u"InfoConnectIssueBtn")
        font1 = QFont()
        font1.setUnderline(True)
        self.InfoConnectIssueBtn.setFont(font1)

        self.verticalLayout_3.addWidget(self.InfoConnectIssueBtn, 0,
Qt.AlignHCenter)

        self.verticalSpacer_3 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_3)

        self.InfoCC = QLabel(self.InformationPage)
        self.InfoCC.setObjectName(u"InfoCC")

        self.verticalLayout_3.addWidget(self.InfoCC, 0, Qt.AlignHCenter)
```

```
        self.stackedWidget.addWidget(self.InformationPage)

        self.verticalLayout_2.addWidget(self.stackedWidget)


        self.verticalLayout.addWidget(self.widget, 0, Qt.AlignHCenter)

        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)

        self.stackedWidget.setCurrentIndex(0)


        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"NetAdminPro Client", None))
        self.InfoLogo.setText("")
        self.InfoText2.setText(QCoreApplication.translate("MainWindow", u"Connect
Your PC To Server", None))
        self.InfoText.setText(QCoreApplication.translate("MainWindow", u"Enter
Your Information Below", None))
        self.PcName.setPlaceholderText(QCoreApplication.translate("MainWindow",
u"Enter PC Name", None))
        self.InfoClientIpAddr.setPlaceholderText(QCoreApplication.translate("Main
Window", u"Enter I.P. Address Of Client", None))
        self.InfoServerIpAddr.setPlaceholderText(QCoreApplication.translate("Main
Window", u"Enter The I.P. Address Of Server", None))
        self.InfoServerPort.setPlaceholderText(QCoreApplication.translate("MainWi
ndow", u"Enter The Port Number Of Server", None))
        self.InfoCheckBox.setText(QCoreApplication.translate("MainWindow", u"Are
The Above Information Correct", None))
        self.InfoConnectBTn.setText(QCoreApplication.translate("MainWindow",
u"Connect", None))
        self.InfoConnectIssueBtn.setText(QCoreApplication.translate("MainWindow",
u"Click Here If Facing issue To Connect", None))
        self.InfoCC.setText(QCoreApplication.translate("MainWindow", u"CC Sonu
Jha", None))
    # retranslateUi
```

UI_CLIENTINTERFACE.PY

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'ClientInterfaceyEybaH.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide2.QtCore import *
from PySide2.QtGui import *
from PySide2.QtWidgets import *

import ClientResources_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(1022, 731)
        MainWindow.setStyleSheet(u"*{\n"
"   border: none;\n"
"   background-color: transparent;\n"
"   background: transparent;\n"
"   padding: 0;\n"
"   margin: 0;\n"
"   color: #fff;\n"
"}\n"
"\n"
"#centralwidget{\n"
"   background-image: url(:/images/images/BgClient.jpg);\n"
"}\n"
"\n"
"#widget{\n"
"   background-color: rgb(9, 27, 68);\n"
"   border-radius:20px;\n"
"}\n"
"\n"
"QLineEdit{\n"
"   background-color: rgb(9, 10, 37);\n"
```

```
"   padding: 5px 3px;\n"
"   border-radius:5px;\n"
"   \n"
"}\n"
"QPushButton{\n"
"   background-color:  rgb(9, 10, 37);\n"
"   padding: 10px 5px;\n"
"   border-radius:5px;\n"
"}")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.verticalLayout = QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName(u"verticalLayout")
        self.widget = QWidget(self.centralwidget)
        self.widget.setObjectName(u"widget")
        self.widget.setMinimumSize(QSize(400, 700))
        self.widget.setMaximumSize(QSize(400, 700))
        self.verticalLayout_2 = QVBoxLayout(self.widget)
        self.verticalLayout_2.setObjectName(u"verticalLayout_2")
        self.stackedWidget = QStackedWidget(self.widget)
        self.stackedWidget.setObjectName(u"stackedWidget")
        self.InformationPage = QWidget()
        self.InformationPage.setObjectName(u"InformationPage")
        self.verticalLayout_3 = QVBoxLayout(self.InformationPage)
        self.verticalLayout_3.setObjectName(u"verticalLayout_3")
        self.InfoLogo = QLabel(self.InformationPage)
        self.InfoLogo.setObjectName(u"InfoLogo")
        self.InfoLogo.setPixmap(QPixmap(u":/icons/icons/airplay.png"))

        self.verticalLayout_3.addWidget(self.InfoLogo, 0, Qt.AlignHCenter)

        self.InfoText2 = QLabel(self.InformationPage)
        self.InfoText2.setObjectName(u"InfoText2")
        font = QFont()
        font.setPointSize(11)
        font.setBold(True)
        font.setWeight(75)
        self.InfoText2.setFont(font)

        self.verticalLayout_3.addWidget(self.InfoText2, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.InfoText = QLabel(self.InformationPage)
        self.InfoText.setObjectName(u"InfoText")
```

```
        self.verticalLayout_3.addWidget(self.InfoText, 0,
Qt.AlignHCenter|Qt.AlignTop)

        self.verticalSpacer = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer)

        self.UserInputInfo = QFrame(self.InformationPage)
        self.UserInputInfo.setObjectName(u"UserInputInfo")
        self.UserInputInfo.setFrameShape(QFrame.StyledPanel)
        self.UserInputInfo.setFrameShadow(QFrame.Raised)
        self.verticalLayout_4 = QVBoxLayout(self.UserInputInfo)
        self.verticalLayout_4.setObjectName(u"verticalLayout_4")
        self.PcName = QLineEdit(self.UserInputInfo)
        self.PcName.setObjectName(u"PcName")

        self.verticalLayout_4.addWidget(self.PcName)

        self.InfoClientIpAddr = QLineEdit(self.UserInputInfo)
        self.InfoClientIpAddr.setObjectName(u"InfoClientIpAddr")

        self.verticalLayout_4.addWidget(self.InfoClientIpAddr)

        self.InfoServerIpAddr = QLineEdit(self.UserInputInfo)
        self.InfoServerIpAddr.setObjectName(u"InfoServerIpAddr")

        self.verticalLayout_4.addWidget(self.InfoServerIpAddr)

        self.InfoServerPort = QLineEdit(self.UserInputInfo)
        self.InfoServerPort.setObjectName(u"InfoServerPort")

        self.verticalLayout_4.addWidget(self.InfoServerPort)


        self.verticalLayout_3.addWidget(self.UserInputInfo)

        self.verticalSpacer_2 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_2)

        self.InfoCheckBox = QCheckBox(self.InformationPage)
        self.InfoCheckBox.setObjectName(u"InfoCheckBox")
```

```
        self.verticalLayout_3.addWidget(self.InfoCheckBox)

        self.InfoConnectBTn = QPushButton(self.InformationPage)
        self.InfoConnectBTn.setObjectName(u"InfoConnectBTn")

        self.verticalLayout_3.addWidget(self.InfoConnectBTn, 0, Qt.AlignHCenter)

        self.verticalSpacer_4 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_4)

        self.InfoConnectIssueBtn = QPushButton(self.InformationPage)
        self.InfoConnectIssueBtn.setObjectName(u"InfoConnectIssueBtn")
        font1 = QFont()
        font1.setUnderline(True)
        self.InfoConnectIssueBtn.setFont(font1)

        self.verticalLayout_3.addWidget(self.InfoConnectIssueBtn, 0,
Qt.AlignHCenter)

        self.verticalSpacer_3 = QSpacerItem(20, 40, QSizePolicy.Minimum,
QSizePolicy.Expanding)

        self.verticalLayout_3.addItem(self.verticalSpacer_3)

        self.InfoCC = QLabel(self.InformationPage)
        self.InfoCC.setObjectName(u"InfoCC")

        self.verticalLayout_3.addWidget(self.InfoCC, 0, Qt.AlignHCenter)

        self.stackedWidget.addWidget(self.InformationPage)

        self.verticalLayout_2.addWidget(self.stackedWidget)


        self.verticalLayout.addWidget(self.widget, 0, Qt.AlignHCenter)

        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)

        self.stackedWidget.setCurrentIndex(0)
```

```
        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"NetAdminPro Client", None))
        self.InfoLogo.setText("")
        self.InfoText2.setText(QCoreApplication.translate("MainWindow", u"Connect
Your PC To Server", None))
        self.InfoText.setText(QCoreApplication.translate("MainWindow", u"Enter
Your Information Below", None))
        self.PcName.setPlaceholderText(QCoreApplication.translate("MainWindow",
u"Enter PC Name", None))
        self.InfoClientIpAddr.setPlaceholderText(QCoreApplication.translate("Main
Window", u"Enter I.P. Address Of Client", None))
        self.InfoServerIpAddr.setPlaceholderText(QCoreApplication.translate("Main
Window", u"Enter The I.P. Address Of Server", None))
        self.InfoServerPort.setPlaceholderText(QCoreApplication.translate("MainWi
ndow", u"Enter The Port Number Of Server", None))
        self.InfoCheckBox.setText(QCoreApplication.translate("MainWindow", u"Are
The Above Information Correct", None))
        self.InfoConnectBTn.setText(QCoreApplication.translate("MainWindow",
u"Connect", None))
        self.InfoConnectIssueBtn.setText(QCoreApplication.translate("MainWindow",
u"Click Here If Facing issue To Connect", None))
        self.InfoCC.setText(QCoreApplication.translate("MainWindow", u"CC Sonu
Jha", None))
    # retranslateUi
```

LOGIN.PY

```
import os
import sys
import sqlite3
import subprocess
import threading
from twilio.rest import Client
import random


# Create a SQLite database connection
```

```
conn = sqlite3.connect("Account.db")
cursor = conn.cursor()

# Create a table to store user information
cursor.execute("""
CREATE TABLE IF NOT EXISTS UsersInfo (
    UserName TEXT PRIMARY KEY,
    Email TEXT,
    MobileNum INT,
    Password TEXT
)
""")
conn.commit()
conn.close()


account_sid = 'AC1afce35a727c6ca1d69d2034c0a62355'
auth_token = 'ae4f9af03bb0651cc74fd7815098c1fd'

# Create a Twilio client
client = Client(account_sid, auth_token)

# Your Twilio phone number
twilio_phone_number = '+13346506140'


def send_verification_code(phone_number):
        # Generate a random six-digit verification code
        verification_code = ''.join(random.choices('0123456789', k=6))

        # Send the verification code via SMS
        message = client.messages.create(
            body=f'Your verification code for NetAdminPro software is:
{verification_code}',
            from_=twilio_phone_number,
            to=phone_number
        )

        return verification_code




from ui_LoginInterface import *
```

```
from Custom_Widgets.Widgets import *



class CodeInputDialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)

        self.setWindowTitle("Verification Code")

        layout = QVBoxLayout()
        self.setLayout(layout)

        self.code_input_box = QLineEdit()
        self.code_input_box.setPlaceholderText("Enter the verification code")

        submit_button = QPushButton("Submit")
        submit_button.clicked.connect(self.accept)

        layout.addWidget(self.code_input_box)
        layout.addWidget(submit_button)







class Login(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        loadJsonStyle(self, self.ui, jsonFiles = {
            "D:\\NetAdminPro\\RegisterLoginStyle.json"
            })

        self.show()

        self.ui.RegisterBtn.clicked.connect(self.register)
        self.ui.LoginBtn.clicked.connect(self.authenticate)
```

```
    def register(self):
        SignUp_UserName_Input = self.ui.SignUpUserNameInput.text()
        SignUp_Email_Input = self.ui.SignUpEmailInput.text()
        SignUp_Password_Input = self.ui.SignUpPasswordInput.text()
        SignUp_ConPassword_Input = self.ui.SignUpConPasswordInput.text()
        SignUp_MobileNum_Input = self.ui.SignUpMobileNumInput.text()
        self.ui.LoginWidget.setCurrentWidget(self.ui.LoginPage)

        if not SignUp_UserName_Input or not SignUp_Email_Input or not
SignUp_Password_Input or not SignUp_ConPassword_Input or not
SignUp_MobileNum_Input:
            QMessageBox.warning(self, "Registration Failed", "Please enter all
the credentials.")
        else:
            # Define a list of allowed email domains
            allowed_domains = ["gmail.com", "outlook.com", "aol.com",
"protonmail.com", "proton.me", "zohomail.in", "icloud.com", "yahoo.com",
"myyahoo.com"]

            # Extract the domain from the provided email address
            email_domain = SignUp_Email_Input.split('@')[-1]

            # Check if the email domain is in the allowed list
            if email_domain not in allowed_domains:
                QMessageBox.warning(self, "Registration Failed", "Email domain is
not allowed.")
                return

            try:
                conn = sqlite3.connect("Account.db")
                cursor = conn.cursor()

                # Check if the user already exists
                cursor.execute("SELECT UserName FROM UsersInfo WHERE UserName =
?", (SignUp_UserName_Input,))
                existing_user = cursor.fetchone()

                if existing_user:
                    QMessageBox.warning(self, "Registration Failed", "Username
already exists.")
                elif SignUp_Password_Input != SignUp_ConPassword_Input:
```

```
                    QMessageBox.warning(self, "Registration Failed", "Password
and Confirm Password do not match.")
                else:
                    # Send the verification code via SMS using Twilio
                    verification_code =
send_verification_code(SignUp_MobileNum_Input)

                    code_input_dialog = CodeInputDialog(self)
                    result = code_input_dialog.exec_()

                    if result == QDialog.Accepted:
                        # Get the user's input from the QLineEdit
                        user_input = code_input_dialog.code_input_box.text()

                        # Check if the user-entered code matches the generated
code
                        if user_input == verification_code:
                            # Insert the new user
                            cursor.execute("INSERT INTO UsersInfo (UserName,
Email, Password, MobileNum) VALUES (?, ?, ?, ?)",
                                           (SignUp_UserName_Input,
SignUp_Email_Input, SignUp_Password_Input, SignUp_MobileNum_Input))
                            conn.commit()
                            conn.close()
                            QMessageBox.information(self, "Registration
Successful", "Registration successful!")
                        else:
                            QMessageBox.warning(self, "Registration Failed",
"Invalid verification code. Registration failed.")

            except sqlite3.Error as e:
                print("SQLite error:", e)
                QMessageBox.warning(self, "Registration Failed", "Registration
failed. Please try again.")


    def authenticate(self):
        Login_UserName_Input = self.ui.LoginUserNameInput.text()
        Login_Password_Input = self.ui.LoginPasswordInput.text()
        if not Login_UserName_Input or not Login_Password_Input:
            QMessageBox.warning(self, "Login Failed", "Please enter both username
and password.")


        try:
```

```
            conn = sqlite3.connect("Account.db")
            cursor = conn.cursor()

            # Check if the user exists and the password matches
            cursor.execute("SELECT UserName, Password FROM UsersInfo WHERE
UserName = ?", (Login_UserName_Input,))
            user_data = cursor.fetchone()

            if user_data:
                stored_password = user_data[1]
                if Login_Password_Input == stored_password:
                    QMessageBox.information(self, "Login Successful", "Welcome, "
+ Login_UserName_Input)
                    self.open_second_window()
                    # subprocess.Popen(["python",
"D:\\NetAdminPro\\MainInterface.py"])
                    # self.close()
                    # Add code here to open your main interface window
                else:
                    QMessageBox.warning(self, "Login Failed", "Invalid
password.")
            else:
                QMessageBox.warning(self, "Login Failed", "User not found.")

            conn.close()
        except sqlite3.Error as e:
            print("SQLite error:", e)
            QMessageBox.warning(self, "Login Failed", "Login failed. Please try
again.")


    def open_second_window(self):
        try:
            # Replace 'second_script.py' with the actual name of your second
script
            subprocess.Popen(['python', 'D:\\NetAdminPro\\main.py'])
            self.close()
        except Exception as e:
            print("Error running second script:", str(e))

    def run_second_script(self):
        try:
            # Replace 'second_script.py' with the actual name of your second
script
            subprocess.run(['python', 'D:\\NetAdminPro\\main.py'], check=True)
```

```
        except subprocess.CalledProcessError:
            print("Error running second script")




if __name__ == "__main__":
    app = QApplication(sys.argv)

    window = Login()
    window.show()
    sys.exit(app.exec_())
```

SERVER.PY

```
from ui_MainUi import *
from Custom_Widgets.Widgets import *
import socket
from os import getlogin
from PIL import Image
import io
import numpy as np
from random import randint
import pyautogui
from threading import Thread
import sys
from PySide2.QtWidgets import QMainWindow, QApplication, QLabel, QPushButton,
QLineEdit, QAction, QMessageBox
from PySide2.QtGui import QPixmap
from PySide2.QtCore import Qt

class RemoteDesktop(QMainWindow):
    updateImageSignal = Signal(bytes)

    def __init__(self, conn, addr):
        super().__init__()
        self.conn = conn
        self.addr = addr
        self.initUI()
```

```
    def changeImage(self):
        try:
            print("[SERVER]: CONNECTED: {0}!".format(self.addr[0]))
            while True:
                img_bytes = self.conn.recv(9999999)
                self.updateImageSignal.emit(img_bytes)
        except ConnectionResetError:
            self.conn.close()

    def updateImage(self, img_bytes):
        pixmap = QPixmap()
        pixmap.loadFromData(img_bytes)
        self.label.setScaledContents(True)
        self.label.resize(self.width(), self.height())
        self.label.setPixmap(pixmap)

    def initUI(self):
        self.label = QLabel(self)
        self.label.resize(self.width(), self.height())
        self.setGeometry(QRect(pyautogui.size()[0] // 4, pyautogui.size()[1] //
4, 800, 450))
        self.setFixedSize(self.width(), self.height())
        self.setWindowTitle("[SERVER] Remote Desktop: " + str(randint(99999,
999999)))
        self.start = Thread(target=self.changeImage, daemon=True)
        self.start.start()

        self.updateImageSignal.connect(self.updateImage)

class SecondServer(Thread):
    def __init__(self, ip, port):
        super().__init__()
        self.ip = ip
        self.port = port

    def run(self):
        try:
            print("[SERVER 2]: STARTED")
            sock = socket.socket()
            sock.bind((self.ip, self.port))
            sock.listen()
            conn, addr = sock.accept()

            print(f"[SERVER 2]: ACCEPTED CONNECTION FROM {addr}")
```

```
            # Add your logic for the second server here
            print(f"[SERVER 2]: Client IP: {addr[0]}")

        except Exception as e:
            print(f"[SERVER 2]: Failed to start server: {e}")

class MainUi(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.show()

        # Button and LineEdit for Second Server
        self.ui.RAStartServer2Bttn.clicked.connect(self.StartSecondServer)
        # Connect the first server's button to its functionality
        self.ui.RAStartserverBttn.clicked.connect(self.ServerStarted)
        # Connect other buttons to their functionality
        self.ui.RANextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.SharedFileTransferPage))
        self.ui.RAPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.OtherModulePage))
        self.ui.SFPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.RemoteAccessPage))
        self.ui.SFNextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.NetworkConfigurationPage))
        self.ui.NCPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.SharedFileTransferPage))
        self.ui.NCNextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.BrowsingControlPage))
        self.ui.BCPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.NetworkConfigurationPage))
        self.ui.BCNextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.LogFilePage))
        self.ui.LogPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.BrowsingControlPage))
        self.ui.LogNextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.OtherModulePage))
        self.ui.OMPrevBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.LogFilePage))
        self.ui.OMNextBttn.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.RemoteAccessPage))

    def ServerStarted(self):
```

```
        print("[SERVER]: STARTED")
        self.sock = socket.socket()  # Assign to the class attribute
        ip_address = self.ui.RAIPLine.text()
        port_text = self.ui.RAPortLine.text()

        if not ip_address or not port_text:
            QMessageBox.warning(self, "Warning", "Please enter both an IP address
and a port number.")
            return

        try:
            # Validate and parse IP address
            socket.inet_pton(socket.AF_INET, ip_address)
        except socket.error:
            QMessageBox.warning(self, "Warning", "Please enter a valid IPv4
address.")
            return

        try:
            # Validate and parse port number
            port_number = int(port_text)
            if not 0 < port_number < 65536:
                raise ValueError("Port number must be between 1 and 65535")
        except ValueError as e:
            QMessageBox.warning(self, "Warning", str(e))
            return

        try:
            self.sock.bind((ip_address, port_number))
            self.sock.listen()
            global conn, addr
            conn, addr = self.sock.accept()

            # Start the RemoteDesktop window with the established connection
            remote_desktop_window = RemoteDesktop(conn, addr)
            remote_desktop_window.show()

        except Exception as e:
            QMessageBox.warning(self, "Warning", f"Failed to start server: {e}")

    def StartSecondServer(self):
        print("[SERVER 2]: START BUTTON CLICKED")
        ip_address = self.ui.RAIP2Line.text()
        port_text = self.ui.RA2PortLine.text()
```

```
        if not ip_address or not port_text:
            QMessageBox.warning(self, "Warning", "Please enter both an IP address
and a port number for the second server.")
            return

        try:
            # Validate and parse IP address
            socket.inet_pton(socket.AF_INET, ip_address)
        except socket.error:
            QMessageBox.warning(self, "Warning", "Please enter a valid IPv4
address for the second server.")
            return

        try:
            # Validate and parse port number
            port_number = int(port_text)
            if not 0 < port_number < 65536:
                raise ValueError("Port number for the second server must be
between 1 and 65535")
        except ValueError as e:
            QMessageBox.warning(self, "Warning", str(e))
            return

        second_server_thread = SecondServer(ip_address, port_number)
        second_server_thread.start()

    def closeEvent(self, event):
        # Implement logic to close the server gracefully
        print("Closing the server...")
        try:
            if self.sock:
                self.sock.close()  # Close the server socket
            # Perform any additional cleanup operations

        except Exception as e:
            print(f"Error while closing the server: {e}")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainUi()
    window.show()
    sys.exit(app.exec())
```

CLIENT.PY

```
from ui_ClientInterface import *
from PySide2.QtWidgets import QMessageBox, QMainWindow, QApplication
import sys
import socket
import threading
import subprocess

class Client(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.show()

        # Existing socket for the first code
        self.sock = socket.socket()

        # Connect existing button to the existing function
        self.ui.InfoConnectBTn.clicked.connect(self.CommonClient)

    def CommonClient(self):
        pc_name = self.ui.PcName.text()
        client_ip = self.ui.InfoClientIpAddr.text()
        server_ip = self.ui.InfoServerIpAddr.text()
        server_port = self.ui.InfoServerPort.text()

        if not all([pc_name, client_ip, server_ip, server_port]):
            QMessageBox.warning(self, "Warning", "Please fill in all fields.")
            return

        try:
            # Connect to the server
            self.sock.connect((server_ip, int(server_port)))

            # Send client information to the server
            info_str = f"PC Name: {pc_name}\nClient IP: {client_ip}"
            self.sock.send(info_str.encode())

            # Start receiving commands in a new thread
        except Exception as e:
```

```
            QMessageBox.warning(self, "Connection Error", f"Failed to connect:
{str(e)}")
            return

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = Client()
    sys.exit(app.exec_())
```

❖FUTURE ENHANCEMENT

1. **Security Features:**
   - Implement encryption for communication between the server and client to ensure data privacy.
   - Add user authentication to restrict access to authorized users.

2. **User Interface Improvements:**
   - Enhance the user interface with a modern and intuitive design.
   - Provide customization options for users to personalize the appearance.

3. **File Transfer Functionality:**
   - Enable file transfer capabilities between the server and client.
   - Implement drag-and-drop functionality for easy file sharing.

4. **Multi-Platform Support:**
   - Ensure compatibility across different operating systems.
   - Develop mobile applications for remote access and control.

5. **Remote Control Features:**
   - Include remote control functionalities to allow the server to control the client's desktop.
   - Implement support for keyboard and mouse input from the server to the client.

6. **Logging and Monitoring:**
   - Implement logging of activities for auditing and troubleshooting.
   - Provide real-time monitoring of connections and activities.

7. **Multi-User Support:**
   - Allow multiple clients to connect to the server simultaneously.
   - Implement user management to handle multiple connections securely.

8. **Improved Command Execution:**

- Enhance the command execution mechanism to support a broader range of commands.
- Implement a command history feature.

9. **Performance Optimization:**
   - Optimize image transmission for better performance.
   - Implement compression algorithms to reduce data transfer size.

10. **Network Configuration Options:**
    - Provide advanced network configuration options, such as proxy support.
    - Implement dynamic IP detection and handling.

11. **Remote Desktop Recording:**
    - Add the ability to record the remote desktop session for later review.
    - Implement screenshot capture for specific intervals.

12. **Notification System:**
    - Implement a notification system to alert users of important events or connection status changes.
    - Include sound alerts for critical events.

13. **Localization and Internationalization:**
    - Support multiple languages to make the application accessible to a broader audience.
    - Implement internationalization features for date, time, and numeric formats.

14. **Automated Testing:**
    - Develop a comprehensive set of automated tests to ensure the stability and reliability of the application.

15. **Documentation and Help System:**
    - Create detailed documentation for users and developers.
    - Implement an interactive help system within the application.

❖UNDERTAKING

I, **Sonu Jha**, the sole developer of the **NetAdmin Pro: Centralized Network Management System for Educational Institutions**, hereby declare that the project is entirely my independent work. I take full responsibility for the conception, design, and implementation of the system described in this documentation.

I confirm that:

1. The NetAdmin Pro system is developed to facilitate centralized network management within educational institutions, promoting ethical use of college resources and restricting access to non-academic content.

2. The features outlined in the documentation, including user management, activity logging, device management, network access policies, and alert systems, have been designed and implemented based on the requirements discussed.

3. The ERD diagram provided is a representation of the data model designed for the NetAdmin Pro system, and I am accountable for ensuring the consistency and accuracy of the database structure.

4. The system adheres to the specified stakeholders' needs, including technical, user, and client requirements, as discussed throughout our collaboration.

5. I have followed the industry-standard practices in software development and adhered to ethical guidelines in creating a secure and efficient network management solution.

6. The project documentation includes accurate representations of the system components, including UML diagrams, flowcharts, and runtime output screenshots.

7. Testing procedures have been conducted to verify the functionality, security, and performance of the NetAdmin Pro system.

8. I have compiled a comprehensive bibliography acknowledging the sources of information, tools, and frameworks utilized during the development of the project.

I acknowledge that any breach of academic integrity or misrepresentation in this undertaking may result in the appropriate actions by the educational institution.

❖REFERENCES

1. https://youtu.be/p3tSLatmGvU?si=kOVDj8I4r-mNUirE
2. https://youtu.be/YXPyB4XeYLA?si=UUDX36R3FS6ZUitH
3. https://youtu.be/Vde5SH8e1OQ?si=wlujyOKVoUJzW8Tv
4. https://youtu.be/Z1N9JzNax2k?si=MffH23i2L5rUdx4_
5. https://www.youtube.com/watch?v=adC48qZ8p5Y
6. https://youtu.be/Y_iOLhwtfVA?si=bVM3XoVSxAeWXrJW
7. https://youtu.be/mkBwInKhBsA
8. https://youtu.be/jj8L51oiB4c
9. https://www.qt.io/
10. https://doc.qt.io/
11. https://www.python.org/doc/
12. https://www.pythonguis.com/pyside2-tutorial/
13. https://www.geeksforgeeks.org/python-introduction-to-pyqt5/
14. https://github.com/KhamisiKibet/QT-PyQt-PySide-Custom-Widgets/blob/main/Custom_Widgets/Widgets.py
15. https://khamisikibet.github.io/Docs-QT-PyQt-PySide-Custom-Widgets/
16. https://chat.openai.com/share/84de8baf-094f-4640-b587-e955db6941b2
17. https://chat.openai.com/share/c90f3356-b127-42df-83b1-9b2883f7ee45
18. https://chat.openai.com/share/4c83dc6d-2df5-4cab-a3ad-2d28b5939f7c