FINAL PROJECT

LAB MATERIALS MANAGEMENT (GUI)	
Course Code: CPE 010	**Program:** Computer Engineering
Course Title: Object Oriented Programming B	**Date Performed:** December 2, 2024
Section: CPE21S4	**Date Submitted:** December 2, 2024
Name(s): - Adia, James Russel E. - Carag, Carl Jervie B. - Gob, Mark Jeonel Kenn E. - Guariño, Danica T. - Rio, Aries C.	**Instructor:** Engr. Maria Rizette H. Sayo

1. Objectives(s)

General Objectives:
- The program should have a Graphical User Interface (GUI)
- The program should utilize OOP Concepts (Class, Object, Polymorphism, Inheritance, Encapsulation)
- The program must be related to Computer Engineering

Specific Objectives:
- The program will be done using the Python programming language
- The program will utilize lists to hold instances of the materials, and dictionaries to store the materials with their respective quantities

2. Intended Learning Outcomes (ILOs)

- Students will demonstrate their ability to apply object-oriented programming concepts
- Students will acquire proficiency in creating user-friendly graphical user interfaces (GUIs) using Python and relevant libraries
- Students will apply fundamental data structures (such as lists and dictionaries) to effectively manage and manipulate the materials database within the system.

3. Discussion

I. Background and Concepts

This program was created to manage laboratory materials more easily. It tracks available materials and allows students to borrow items quickly. It helps organize material quantities, reducing errors and making sure materials are always up-to-date. Each of the students are given equal access to the laboratory equipment making every student have their turn to borrow materials in the tool room.

The program uses object-oriented programming (OOP) to organize the code. A class is a blueprint for creating objects, such as materials with name and quantity. Polymorphism lets one function handle different materials. Inheritance allows one class to take features from another, and encapsulation keeps data safe by controlling how it's accessed. Encapsulation is a way to prevent access or modification to the particular components of the code. Abstractions hides the unnecessary details of the code from the developer and instead shows the important functions to the users.

PyQt5 is used to create the graphical user interface (GUI). It makes the program easy to use by adding buttons, text fields, and windows, making it simple for students and admins to depict and choose which available materials to borrow, the buttons enables the proper selection of the actions made in the program, and text fields provide the students the items inside the tool room.

The program includes several libraries: sys for system tasks, csv for reading and writing material data, PyQt5 for the GUI, datetime for showing the current time, and pytz to make sure time-related tasks use the correct time zone. The import sys is also defined as the System-specific parameter and functions, this module provides access to some variables used and to other included functions that interact with the interpreter. The import csv is one of the best ways to import and export spreadsheets and databases. Import QtWidgets is the common base of all user interface objects(*QWidget — QT for Python*, n.d.)

II. Scope and Limitations

This program helps manage lab materials by allowing students and administrators to track available materials and borrow items easily. It provides a simple interface to register users, log borrowings, and update the number of materials. The goal is to make sure that materials are available when needed and that the borrowing process runs smoothly.

This program is limited to PC users and does not require internet connectivity. While this ensures that the program runs smoothly without relying on an external network, it also means that the program cannot be accessed or updated remotely. Updates to the data will only be reflected locally, and if multiple users interact with the system simultaneously, material availability may not always be accurate. Additionally, without internet access, the program cannot integrate with cloud-based systems or provide real-time synchronization across different devices.

4. Materials and Equipment

- Personal Computer or Laptop with Python installed (Python 3.x)
    - Use any IDE that you are comfortable using to develop, run, and debug the program such as:
        - PyCharm
        - Visual Studio Code (VS Code)
        - Spyder
- Make sure that PyQt5 and pytz libraries are installed using pip

5. Procedure

ILO A: The students will demonstrate their ability to apply object-oriented programming concepts

ILO B: Students will acquire proficiency in creating user-friendly graphical user interfaces (GUIs) using Python and relevant libraries

ILO C: Students will apply fundamental data structures (such as lists and dictionaries) to effectively manage and manipulate the materials database within the system.

We would first need to include the appropriate modules/libraries to create the GUI application

```
import sys
import csv
from PyQt5 import QtWidgets, QtGui, QtCore
from datetime import datetime
import pytz
```

Set the timezone that the program will follow and implement the current date and time

```
tz = pytz.timezone('Asia/Manila')
current_time = datetime.now(tz).strftime("%Y-%m-%d %H:%M")
```

Create the class for the Material based from the following pseudocode:

- This is used to represent the laboratory materials

```python
Class Material
    Method __init__(name, quantity)
        Set self.name = name
        Set self.quantity = quantity
End Class
```

Create the class for the Account Manager based from the following pseudocode:

- Initialize with a filename (account.txt) to store account information

```python
Class AccountManager
    Method __init__(filename='accounts.txt')
        Set self.filename = filename
    End Method
```

- Add a register method to check if a user already exists and, if not, adds the new user to the file

```python
    Method register(name, student_number)
        Open self.filename in append mode
        For each line in file
            Split line into existing_name, existing_number
            If existing_name == name OR existing_number == student_number
                Return False
        Write name, student_number to file
        Return True
    End Method
```

- Add a login method that verifies user credentials by checking the inputted name and student number against the stored accounts

```python
Method login(name, student_number)
```

```
        Open self.filename in read mode
        For each line in file
            Split line into existing_name, existing_number
            If existing_name == name AND existing_number == student_number
                Return True
        Return False
    End Method
End Class
```

Create the class for the Database Manager based from the following pseudocode:

- Initialize with a filename (database.txt) to manage the materials database

```
Python
Method __init__(filename='database.txt')
        Set self.filename = filename
        Set self.materials = load_materials()
    End Method
```

- Add the load_materials method to read the materials from the .txt file and store them in a dictionary with the material name as the key and quantity as the value

```
Python
    Method load_materials()
        Set materials = empty dictionary
        Try
            Open self.filename in read mode
            For each line in file
                Split line into name, quantity
                Add name, quantity to materials
        Except FileNotFoundError
            Pass
        Return materials
    End Method
```

- Add a method to manage the materials for adding, removing, and saving the materials which ensures that the database is kept up-to-date with the current inventory

```
Python
    Method save_materials()
```

```
        Open self.filename in write mode
        For each name, quantity in self.materials
            Write name, quantity to file
    End Method

    Method add_material(name, quantity)
        Add name, quantity to self.materials
        Call save_materials()
    End Method

    Method remove_material(name)
        If name in self.materials
            Delete name from self.materials
            Call save_materials()
    End Method
End Class
```

Create the class for the Borrowing App / Window based from the following pseudocode:

- Initialize with the student's name and number, which sets up the user interface for borrowing materials

Python
```
  Method __init__(student_name, student_number)
        Call super().__init__()
        Set self.student_name = student_name
        Set self.student_number = student_number
        Set self.materials = empty list
        Set self.db_manager = DatabaseManager()
        Call init_ui()
    End Method
```

- Add the init_ui method that sets up the layout, including the buttons and input fields for selecting and managing materials.

Python
```
    Method init_ui()
        Set window title and size
        Create layout
        Add title label to layout
        Add date_time_label to layout
        Add material_label to layout
        Create material_combo
        Populate material_combo with materials from DatabaseManager
        Add material_combo to layout
        Create available_label
        Add available_label to layout
        Connect material_combo.currentIndexChanged to update_available_quantity
        Create quantity_input
        Add quantity_input to layout
        Create add_button
        Connect add_button.clicked to add_material
        Add add_button to layout
        Create remove_button
        Connect remove_button.clicked to remove_material
        Add remove_button to layout
        Create finish_button
        Connect finish_button.clicked to finish_borrowing
        Add finish_button to layout
        Create materials_list
        Add materials_list to layout
        Create back_button
        Connect back_button.clicked to go_back
        Add back_button to layout
    End Method
```

- Add the Borrowing Process methods below that will handle the borrowing process, updating the materials list and database accordingly

Python
```
 Method update_available_quantity()
        Get current index from material_combo
        Get available quantity from material_combo data
        Update available_label text with available quantity
    End Method

    Method add_material()
        If number of materials in list >= 12
            Display "Limit Reached" error message
            Return
        Get material name from material_combo
        Get quantity from quantity_input
        If material name is valid and in DatabaseManager
            Get available quantity from DatabaseManager
            If quantity > available quantity
                Display "Quantity Error" message
                Return
            Add material to materials list
            Update materials_list display
            Decrease available quantity in DatabaseManager
            Refresh material_combo
            Update available quantity label
            Reset quantity_input to 1
    End Method

    Method remove_material()
        Get selected item from materials_list
        If selected item
            Get material name from selected item
            Remove material from materials list
            Remove selected item from materials_list display
        Else
            Display "Selection Error" message
    End Method

    Method finish_borrowing()
        If materials list is empty
            Display "Input Error" message
            Return
        Get current time in Philippine Standard Time
        Prepare borrowing information string
        Display borrowing information in message box
        Call log_borrowing(current_time)
        Call update_database()
        Call clear_inputs()
    End Method

    Method log_borrowing(current_time)
        Open 'log.csv' in append mode
        If log file is empty
            Write header to log file
        Prepare materials_borrowed string
        Write student name, student number, current time, materials_borrowed to log
file
    End Method

    Method update_database()
        For each material in materials list
            Decrease quantity in DatabaseManager
            If quantity < 0
                Set quantity to 0
        Save updated materials to DatabaseManager
    End Method

    Method clear_inputs()
        Clear materials list
        Clear materials_list display
        Clear material_combo
        Populate material_combo with materials from DatabaseManager
        Reset quantity_input to 1
    End Method

    Method go_back()
        Close BorrowingApp
        Initialize LoginWindow
        Show LoginWindow
    End Method
End Class
```

Create the class for the Admin App / Window based from the following pseudocode:

- Initialize to provide an interface for managing the materials database

```
Python
Class AdminApp Inherits QWidget
    Method __init__()
        Call super().__init__()
        Set self.db_manager = DatabaseManager()
        Call init_ui()
    End Method
```

- Add the init_ui method that sets up the layout, including the buttons and input fields for adding, updating, and removing materials

```
Python
  Method init_ui()
        Set window title and size
        Create layout
        Add title label to layout
        Create material_name_input
        Add material_name_input to layout
        Create quantity_input
        Add quantity_input to layout
        Create add_button
        Connect add_button.clicked to add_update_material
        Add add_button to layout
        Create remove_button
        Connect remove_button.clicked to remove_material
        Add remove_button to layout
        Create materials_list
        Call update_materials_list()
        Add materials_list to layout
        Create back_button
        Connect back_button.clicked to go_back
        Add back_button to layout
    End Method
```

- Add the Materials Management methods that will allow the admin to manage the materials inventory

```python
   Method add_update_material()
        Get material name from material_name_input
        Get quantity from quantity_input
        If material name is valid
            Add or update material in DatabaseManager
            Call update_materials_list()
            Clear material_name_input
            Reset quantity_input to 1
    End Method

    Method remove_material()
        Get selected item from materials_list
        If selected item
            Get material name from selected item
            Remove material from DatabaseManager
            Call update_materials_list()
    End Method

    Method update_materials_list()
        Clear materials_list
        For each material in DatabaseManager
            Add material to materials_list
    End Method

    Method go_back()
        Close AdminApp
        Initialize LoginWindow
        Show LoginWindow
    End Method
End Class
```

Create the class for the Login Window based from the following pseudocode:

- Initialize to handle user login and registration

Python

```
Class LoginWindow Inherits QWidget
    Method __init__()
        Call super().__init__()
        Set self.account_manager = AccountManager()
        Call init_ui()
    End Method
```

- Add the init_ui method that sets up the layout, including input fields for the student's name and number, and buttons for login and registration

Python

```
    Method init_ui()
        Set window title and size
        Create layout
        Add title_frame to layout
        Add title_label to title_frame
        Add title_frame to layout
        Create name_input
        Add name_input to layout
        Create number_input
        Add number_input to layout
        Create login_button
        Connect login_button.clicked to login
        Add login_button to layout
        Create register_button
        Connect register_button.clicked to register
        Add register_button to layout
    End Method
```

- Add the authentication methods that will handle user authentication which opens the appropriate app / window (Borrowing App or Admin App) based on the user's inputted credentials

Python

```
  Method login()
        Get student_name from name_input
        Get student_number from number_input
        If student_name == "Admin" AND student_number == "admin"
            Close self
            Call open_admin_app()
        Else if student_name == "" AND student_number == ""
            Display "No Input" error message
        Else if student_name == "" OR student_number == ""
```

```
            Display "Missing Input" error message
        Else if Call login(student_name, student_number) on self.account_manager
            Close self
            Call open_borrowing_app(student_name, student_number)
        Else
            Display "Incorrect credentials" error message
    End Method

    Method register()
        Get student_name from name_input
        Get student_number from number_input
        Else if student_name is empty OR student_number is empty
            Display warning message "Name and student number cannot be blank."
        Else if student_number is not a digit
            Display warning message "Student number must consist of integers only."
        Else if Call register(student_name, student_number) on self.account_manager
            Display "Registration Successful" message
        Else if student_name is empty AND student_number is empty
            Display warning message "No Input. Please try again."
        Else if student_name is empty OR student_number is empty
            Display warning message "Missing input. Please try again."
        Else
            Display warning message "Name or student number already exists."
    End Method

    Method open_borrowing_app(student_name, student_number)
        Initialize BorrowingApp with student_name and student_number
        Show BorrowingApp
    End Method

    Method open_admin_app()
        Initialize AdminApp
        Show AdminApp
    End Method
End Class
```

Create the Execution of the Main program based from the following pseudocode:

- The main program execution initializes the application, creates an instance of the Login Window, and starts the application event loop

Python

```
Main Program Execution
    If this script is the main module being run:
        Initialize QApplication with command line arguments
        Create an instance of LoginWindow
        Show the LoginWindow
        Start the application event loop
        Exit the program when the event loop ends
End Main Program Execution
```

6. Output

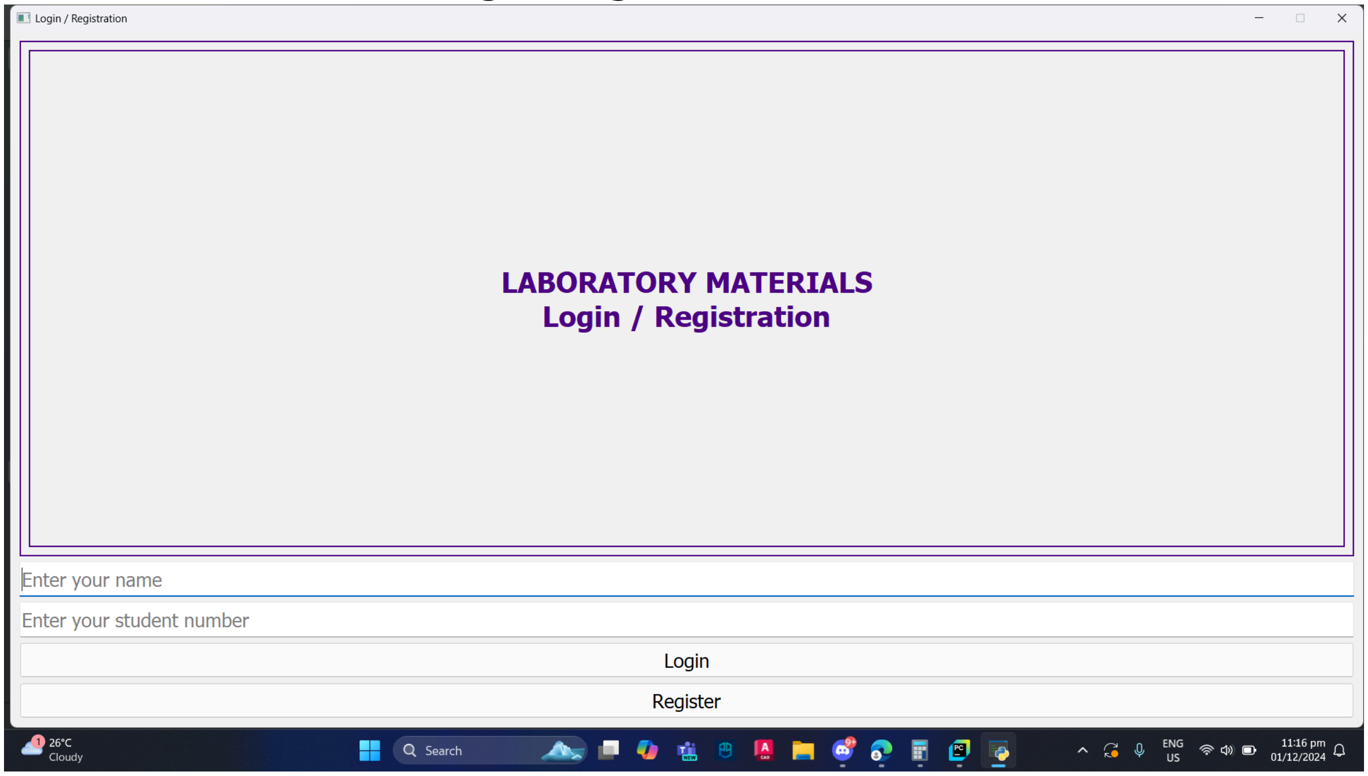
I. Source Code

```python
import sys
import csv
from PyQt5 import QtWidgets, QtGui, QtCore
from datetime import datetime
import pytz

tz = pytz.timezone('Asia/Manila')
current_time = datetime.now(tz).strftime("%Y-%m-%d %H:%M")

class Material:
    """Class to represent a laboratory material with a name and quantity."""
    def __init__(self, name, quantity):
        self.name = name
        self.quantity = quantity

class AccountManager:
    """Class to manage user accounts."""
    def __init__(self, filename='accounts.txt'):
        self.filename = filename

    def register(self, name, student_number):
        """Register a new user if the name or student number does not already exist."""
        with open(self.filename, 'a+') as file:
            file.seek(0)
            for line in file:
                existing_name, existing_number = line.strip().split(',')
                if existing_name == name or existing_number == student_number:
                    return False  # User already exists
            file.write(f"{name},{student_number}\n")  # Save new user
        return True  # Registration successful

    def login(self, name, student_number):
        """Check if the provided credentials match an existing account."""
        with open(self.filename, 'r') as file:
            for line in file:
                existing_name, existing_number = line.strip().split(',')
                if existing_name == name and existing_number == student_number:
                    return True  # Login successful
        return False  # Credentials incorrect

class DatabaseManager:
    """Class to manage materials in the database."""
    def __init__(self, filename='database.txt'):
        self.filename = filename
        self.materials = self.load_materials()

    def load_materials(self):
        """Load materials from the database file."""
        materials = {}
        try:
            with open(self.filename, 'r') as file:
                for line in file:
                    name, quantity = line.strip().split(',')
                    materials[name] = int(quantity)
        except FileNotFoundError:
            pass  # If the file doesn't exist, return an empty dictionary
        return materials

    def save_materials(self):
        """Save materials to the database file."""
        with open(self.filename, 'w') as file:
            for name, quantity in self.materials.items():
                file.write(f"{name},{quantity}\n")

    def add_material(self, name, quantity):
        """Add a new material or update the quantity."""
        self.materials[name] = quantity
        self.save_materials()

    def remove_material(self, name):
        """Remove a material from the database."""
        if name in self.materials:
            del self.materials[name]
            self.save_materials()

class BorrowingApp(QtWidgets.QWidget):
    """Main application for borrowing laboratory materials."""
    def __init__(self, student_name, student_number):
        super().__init__()
        self.student_name = student_name
        self.student_number = student_number
        self.materials = []  # List to hold borrowed materials
        self.db_manager = DatabaseManager()  # Initialize database manager
        self.init_ui()  # Initialize the user interface
    def init_ui(self):
        """Set up the user interface for borrowing materials."""
        self.setWindowTitle('Laboratory Materials Borrowing')
        self.setFixedSize(1910, 980)

        self.layout = QtWidgets.QVBoxLayout()
        self.setLayout(self.layout)

        # Label for title
        title_label = QtWidgets.QLabel("Laboratory Materials Borrowing", self)
        title_label.setAlignment(QtCore.Qt.AlignCenter)  # Center the title

        # Set the font
        font = QtGui.QFont("Helvetica", 24, QtGui.QFont.Bold)
        title_label.setFont(font)

        # Set the text color to Maroon
        title_label.setStyleSheet("color: Maroon;")

        # Add the title label to the layout
        self.layout.addWidget(title_label)

        # Label for borrowing date and time
        self.date_time_label = QtWidgets.QLabel(f"Borrowing Date and Time: 
{current_time}")
        self.date_time_label.setFixedHeight(50)  # Set a fixed height
        self.date_time_label.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.date_time_label)

        # Label for materials input
        self.material_label = QtWidgets.QLabel("Select laboratory materials and their 
quantities:")
        self.material_label.setFixedHeight(50)  # Set a fixed height
        self.material_label.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.material_label)

        # Dropdown for materials
        self.material_combo = QtWidgets.QComboBox(self)
        self.material_combo.setEditable(True)  # Allow searching
        for material_name, quantity in self.db_manager.materials.items():
            self.material_combo.addItem(material_name)  # Only show the name
            self.material_combo.setItemData(self.material_combo.count() - 1, quantity)  
# Set availability as data
        self.material_combo.setFixedHeight(50)  # Set a fixed height
        self.material_combo.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.material_combo)

        # QLabel to display available quantity
        self.available_label = QtWidgets.QLabel("Available: 0", self)
        self.available_label.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.available_label)

        # Connect the currentIndexChanged signal to update the available quantity
        self.material_combo.currentIndexChanged.connect(self.update_available_quantity)

        # Input field for material quantity
        self.quantity_input = QtWidgets.QSpinBox(self)
        self.quantity_input.setRange(1, 100)  # Range for quantity
        self.quantity_input.setFixedHeight(50)  # Set a fixed height
        self.quantity_input.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.quantity_input)

        # Button to add material
        self.add_button = QtWidgets.QPushButton("Add Material")
        self.add_button.setFixedHeight(50)  # Set a fixed height
        self.add_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.add_button.clicked.connect(self.add_material)  # Connect button to
add_material method
        self.layout.addWidget(self.add_button)

        # Button to remove material
        self.remove_button = QtWidgets.QPushButton("Remove Selected Material")
        self.remove_button.setFixedHeight(50)  # Set a fixed height
        self.remove_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.remove_button.clicked.connect(self.remove_material)  # Connect button to
remove_material method
        self.layout.addWidget(self.remove_button)

        # Button to finish borrowing
        self.finish_button = QtWidgets.QPushButton("Finish Borrowing")
        self.finish_button.setFixedHeight(50)  # Set a fixed height
        self.finish_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.finish_button.clicked.connect(self.finish_borrowing)  # Connect button to
finish_borrowing method
        self.layout.addWidget(self.finish_button)

        # List widget to display added materials
        self.materials_list = QtWidgets.QListWidget(self)
        self.layout.addWidget(self.materials_list)

        # Adding the back button
        self.back_button = QtWidgets.QPushButton("Back")
        self.back_button.setFixedHeight(50)  # Set a fixed height
        self.back_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.back_button.clicked.connect(self.go_back)
        self.layout.addWidget(self.back_button)

    # Method to update the available quantity label
    def update_available_quantity(self):
        current_index = self.material_combo.currentIndex()  # Get the current index
        available_quantity = self.material_combo.itemData(current_index)  # Get the
available quantity
        self.available_label.setText(f"Available: {available_quantity}")  # Update the
label text

    def add_material(self):
        """Add a material to the borrowing list."""
        if len(self.materials) >= 12:
            QtWidgets.QMessageBox.warning(self, "Limit Reached",
                                          "You can only borrow up to 12 different types
of materials.")
            return

        material_name = self.material_combo.currentText().strip()  # Get material name
from dropdown
        quantity = self.quantity_input.value()  # Get material quantity input

        # If the material name is valid, add it to the list
        if material_name and material_name in self.db_manager.materials:
            available_quantity = self.db_manager.materials[material_name]
            if quantity > available_quantity:
                QtWidgets.QMessageBox.warning(self, "Quantity Error",
                                              f"Only {available_quantity} available for
{material_name}.")
                return

            # Append material to the list
            self.materials.append(Material(material_name, quantity))
            self.materials_list.addItem(f"{material_name}: {quantity}")  # Update the
displayed list

            # Update the database or the source of materials
            self.db_manager.materials[material_name] -= quantity  # Decrease available
quantity

            # Refresh the ComboBox
            self.material_combo.clear()  # Clear the dropdown for new input
            for name, qty in self.db_manager.materials.items():
                self.material_combo.addItem(name)  # Add material name
                self.material_combo.setItemData(self.material_combo.count() - 1, qty)
# Set available quantity as data

            # Reset the available quantity label
            self.update_available_quantity()  # Update the label to reflect the current
selected material's availability

            self.quantity_input.setValue(1)  # Reset quantity to default

    def remove_material(self):
        """Remove a selected material from the borrowing list."""
        selected_item = self.materials_list.currentItem()  # Get the selected item
        if selected_item:
            material_name = selected_item.text().split(':')[0]  # Extract material name
            # Find the material in the materials list and remove it
            for material in self.materials:
                if material.name == material_name:
                    self.materials.remove(material)  # Remove material from the list
                    break
            self.materials_list.takeItem(self.materials_list.row(selected_item))  #
Remove from the displayed list
        else:
            QtWidgets.QMessageBox.warning(self, "Selection Error", "Please select a
material to remove.")

    def finish_borrowing(self):
        """Finalize the borrowing process and log the information."""
        if not self.materials:
            QtWidgets.QMessageBox.warning(self, "Input Error", "Please add at least one
material.")
            return

        # Get the current date and time in Philippine Standard Time
        tz = pytz.timezone('Asia/Manila')
        current_time = datetime.now(tz).strftime("%Y-%m-%d %H:%M")  # Format date and
time

        # Prepare borrowing information for display
        borrowing_info = f"Borrowing Information:\nDate and Time:
{current_time}\nBorrower: {self.student_name} ({self.student_number})\nMaterials
Borrowed:\n"
        for material in self.materials:
            borrowing_info += f"- {material.name}: {material.quantity}\n"

        # Show the borrowing information in a message box
        QtWidgets.QMessageBox.information(self, "Borrowing Information",
borrowing_info)
        self.log_borrowing(current_time)  # Log the borrowing details
        self.update_database()  # Update the database with borrowed quantities
        self.clear_inputs()  # Clear inputs for the next borrowing session.

    def log_borrowing(self, current_time):
        """Log the borrowing information to a CSV file, including the borrower's
name."""
        with open('log.csv', 'a', newline='') as log_file:
            log_writer = csv.writer(log_file)

            # Write the header only if the file is empty
            if log_file.tell() == 0:
                log_writer.writerow(['Borrower', 'Student Number', 'Date',
'Materials'])  # Write header

            # Write the borrowing details to the log file
            materials_borrowed = "; ".join([f"{material.name}:{material.quantity}" for
material in self.materials])
            log_writer.writerow([self.student_name, self.student_number, current_time,
materials_borrowed])

    def update_database(self):
        """Update the database to reflect the quantities of borrowed materials."""
        for material in self.materials:
            if material.name in self.db_manager.materials:
                self.db_manager.materials[material.name] -= material.quantity  #
Subtract borrowed quantity
                if self.db_manager.materials[material.name] < 0:
                    self.db_manager.materials[material.name] = 0  # Prevent negative
stock
        self.db_manager.save_materials()  # Save updated materials to the database file

    def clear_inputs(self):
        """Clear the input fields and materials list for a new borrowing session."""
        self.materials.clear()  # Clear the materials list
        self.materials_list.clear()  # Clear the displayed materials
        self.material_combo.clear()  # Clear the material dropdown
        self.material_combo.addItems(self.db_manager.materials.keys())  # Refresh
dropdown items
        self.quantity_input.setValue(1)  # Reset quantity to default

    def go_back(self):
        self.close()
        self.login_window = LoginWindow()
        self.login_window.show()

class AdminApp(QtWidgets.QWidget):
    """ Admin application for managing materials."""
    def __init__(self):
        super().__init__()
        self.db_manager = DatabaseManager()  # Initialize database manager
        self.init_ui()  # Initialize the user interface

    def init_ui(self):
        """Set up the user interface for the admin application."""
        self.setWindowTitle('Admin Materials Management')
        self.setFixedSize(1910, 980)

        self.layout = QtWidgets.QVBoxLayout()
        self.setLayout(self.layout)

        # Label for materials management
        title_label = QtWidgets.QLabel("Materials Manager", self)
        title_label.setAlignment(QtCore.Qt.AlignCenter)  # Center the title

        # Set the font
        font = QtGui.QFont("Helvetica", 24, QtGui.QFont.Bold)
        title_label.setFont(font)

        # Set the text color to blue
        title_label.setStyleSheet("color: blue;")

        self.layout.addWidget(title_label)  # Add the title label to the layout

        # Input field for material name
        self.material_name_input = QtWidgets.QLineEdit(self)
        self.material_name_input.setPlaceholderText("Material name")
        self.material_name_input.setFixedHeight(50)  # Set a fixed height
        self.material_name_input.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.material_name_input)

        # Input field for material quantity
        self.quantity_input = QtWidgets.QSpinBox(self)
        self.quantity_input.setRange(1, 100)  # Range for quantity
        self.quantity_input.setFixedHeight(50)  # Set a fixed height
        self.quantity_input.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.layout.addWidget(self.quantity_input)

        # Button to add/update material
        self.add_button = QtWidgets.QPushButton("Add/Update Material")
        self.add_button.setFixedHeight(50)  # Set a fixed height
        self.add_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.add_button.clicked.connect(self.add_update_material)  # Connect button to
add_update_material method
        self.layout.addWidget(self.add_button)

        # Button to remove material
        self.remove_button = QtWidgets.QPushButton("Remove Material")
        self.remove_button.setFixedHeight(50)  # Set a fixed height
        self.remove_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.remove_button.clicked.connect(self.remove_material)  # Connect button to
remove_material method
        self.layout.addWidget(self.remove_button)

        # List widget to display materials
        self.materials_list = QtWidgets.QListWidget(self)
        self.update_materials_list()  # Populate the list with current materials
        self.layout.addWidget(self.materials_list)

        # Adding the back button
        self.back_button = QtWidgets.QPushButton("Back")
        self.back_button.setFixedHeight(50)  # Set a fixed height
        self.back_button.setFont(QtGui.QFont("Helvetica", 16))  # Set font size
        self.back_button.clicked.connect(self.go_back)
        self.layout.addWidget(self.back_button)

    def add_update_material(self):
        """Add a new material or update an existing one."""
        material_name = self.material_name_input.text().strip()  # Get material name
input
        quantity = self.quantity_input.value()  # Get material quantity input

        if material_name:
            self.db_manager.add_material(material_name, quantity)  # Add or update
material in the database
            self.update_materials_list()  # Refresh the materials list
            self.material_name_input.clear()  # Clear the input field
            self.quantity_input.setValue(1)  # Reset quantity to default

    def remove_material(self):
        """Remove a selected material from the database."""
        selected_item = self.materials_list.currentItem()  # Get the selected item
        if selected_item:
            material_name = selected_item.text().split(':')[0]  # Extract material name
            self.db_manager.remove_material(material_name)  # Remove material from the
database
            self.update_materials_list()  # Refresh the materials list

    def update_materials_list(self):
        """Update the displayed list of materials."""
        self.materials_list.clear()  # Clear the current list
        for name, quantity in self.db_manager.materials.items():
            self.materials_list.addItem(f"{name}: {quantity}")  # Add materials to the
list

    def go_back(self):
        self.close()
        self.login_window = LoginWindow()
        self.login_window.show()

class LoginWindow(QtWidgets.QWidget):
    """Login window for entering student information."""
    def __init__(self):
        super().__init__()
        self.account_manager = AccountManager()  # Initialize account manager
        self.init_ui()  # Initialize the user interface

    def init_ui(self):
        """Set up the user interface for the login window."""
        self.setWindowTitle('Login / Registration')
        self.setFixedSize(1910, 980)
        self.layout = QtWidgets.QVBoxLayout()
        self.setLayout(self.layout)

        # Create a frame for the title label
        title_frame = QtWidgets.QFrame(self)
        title_frame.setFrameShape(QtWidgets.QFrame.Box)  # Set the frame shape to Box
        title_frame.setStyleSheet("border: 2px solid indigo;")  # Set the border color
and style

        # Label for title
        title_label = QtWidgets.QLabel("LABORATORY MATERIALS\nLogin / Registration",
self)
        title_label.setAlignment(QtCore.Qt.AlignCenter)  # Center the title

        # Set the font
        font = QtGui.QFont("Helvetica", 24, QtGui.QFont.Bold)
        title_label.setFont(font)

        # Set the text color to indigo
        title_label.setStyleSheet("color: indigo;")

        # Create a layout for the title frame and add the title label to it
        title_layout = QtWidgets.QVBoxLayout(title_frame)
        title_layout.addWidget(title_label)  # Add the title label to the frame's
layout

        # Set the layout for the title frame
        title_frame.setLayout(title_layout)

        # Add the title frame to the main layout
        self.layout.addWidget(title_frame)

        # Input field for student name
        self.name_input = QtWidgets.QLineEdit(self)
        self.name_input.setPlaceholderText("Enter your name")
        self.name_input.setFixedHeight(50)  # Set a fixed height
        font = QtGui.QFont("Helvetica", 16)  # Increase font size
        self.name_input.setFont(font)
        self.layout.addWidget(self.name_input)

        # Input field for student number
        self.number_input = QtWidgets.QLineEdit(self)
        self.number_input.setPlaceholderText("Enter your student number")
        self.number_input.setFixedHeight(50)  # Set a fixed height
        self.number_input.setFont(font)  # Use the same font size
        self.layout.addWidget(self.number_input)

        # Button to log in
        self.login_button = QtWidgets.QPushButton("Login")
        self.login_button.setFixedHeight(50)  # Set a fixed height
        self.login_button.setFont(QtGui.QFont("Helvetica", 16))  # Increase font size
        self.login_button.clicked.connect(self.login)  # Connect button to login method
        self.layout.addWidget(self.login_button)

        # Button to register
        self.register_button = QtWidgets.QPushButton("Register")
        self.register_button.setFixedHeight(50)  # Set a fixed height
        self.register_button.setFont(QtGui.QFont("Helvetica", 16))  # Increase font
size
        self.register_button.clicked.connect(self.register)  # Connect button to
register method
        self.layout.addWidget(self.register_button)

    def login(self):
        """Handle the login process."""
        student_name = self.name_input.text().strip()  # Get student name input
        student_number = self.number_input.text().strip()  # Get student number input

        if student_name == "Admin" and student_number == "admin":
            self.close()  # Close the login window
            self.open_admin_app()  # Open the admin application
        elif student_name == "" and student_number == "":
            QtWidgets.QMessageBox.warning(self, "Login Error", "No Input. Please try
again.")
        elif student_name == "" or student_number == "":
            QtWidgets.QMessageBox.warning(self, "Login Error", "Missing Input. Please
try again.")
        elif self.account_manager.login(student_name, student_number):
            self.close()  # Close the login window
            self.open_borrowing_app(student_name, student_number)  # Open the borrowing
application
        else:
            QtWidgets.QMessageBox.warning(self, "Login Error", "Incorrect credentials.
Please try again.")

    def register(self):
        """Handle the registration process."""
        student_name = self.name_input.text().strip()  # Get student name input
        student_number = self.number_input.text().strip()  # Get student number input

        if not student_name or not student_number:
            QtWidgets.QMessageBox.warning(self, "Registration Error", "Name and student
number cannot be blank.")
        elif not student_number.isdigit():
            QtWidgets.QMessageBox.warning(self, "Registration Error", "Student number
must consist of integers only.")
        elif self.account_manager.register(student_name, student_number):
            QtWidgets.QMessageBox.information(self, "Registration Successful", "You
have been registered!")
        elif student_name == "" and student_number == "":
```

```python
            QtWidgets.QMessageBox.warning(self, "Registration Error", "No Input. Please
try again.")
        elif student_name == "" or student_number == "":
            QtWidgets.QMessageBox.warning(self, "Registration Error", "Missing Input.
Please try again.")
        else:
            QtWidgets.QMessageBox.warning(self, "Registration Error", "Name or student
number already exists.")

    def open_borrowing_app(self, student_name, student_number):
        """Open the borrowing application with the provided student information."""
        self.borrowing_app = BorrowingApp(student_name, student_number)
        self.borrowing_app.show()

    def open_admin_app(self):
        """Open the admin application."""
        self.admin_app = AdminApp()
        self.admin_app.show()

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    login_window = LoginWindow()  # Create the login window
    login_window.show()  # Show the login window
    sys.exit(app.exec_())  # Start the application event loop
```

II. Outputs

Login / Registration Window:



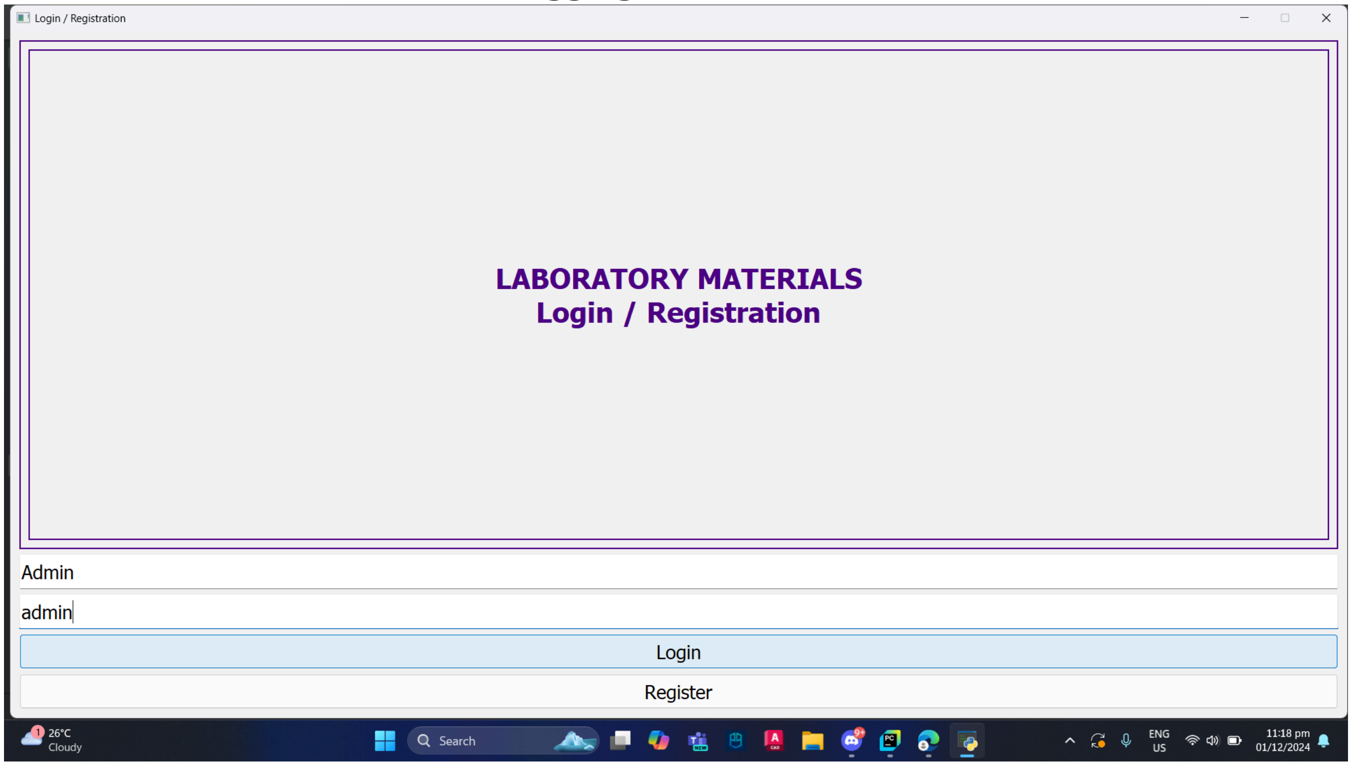
Login / Registration

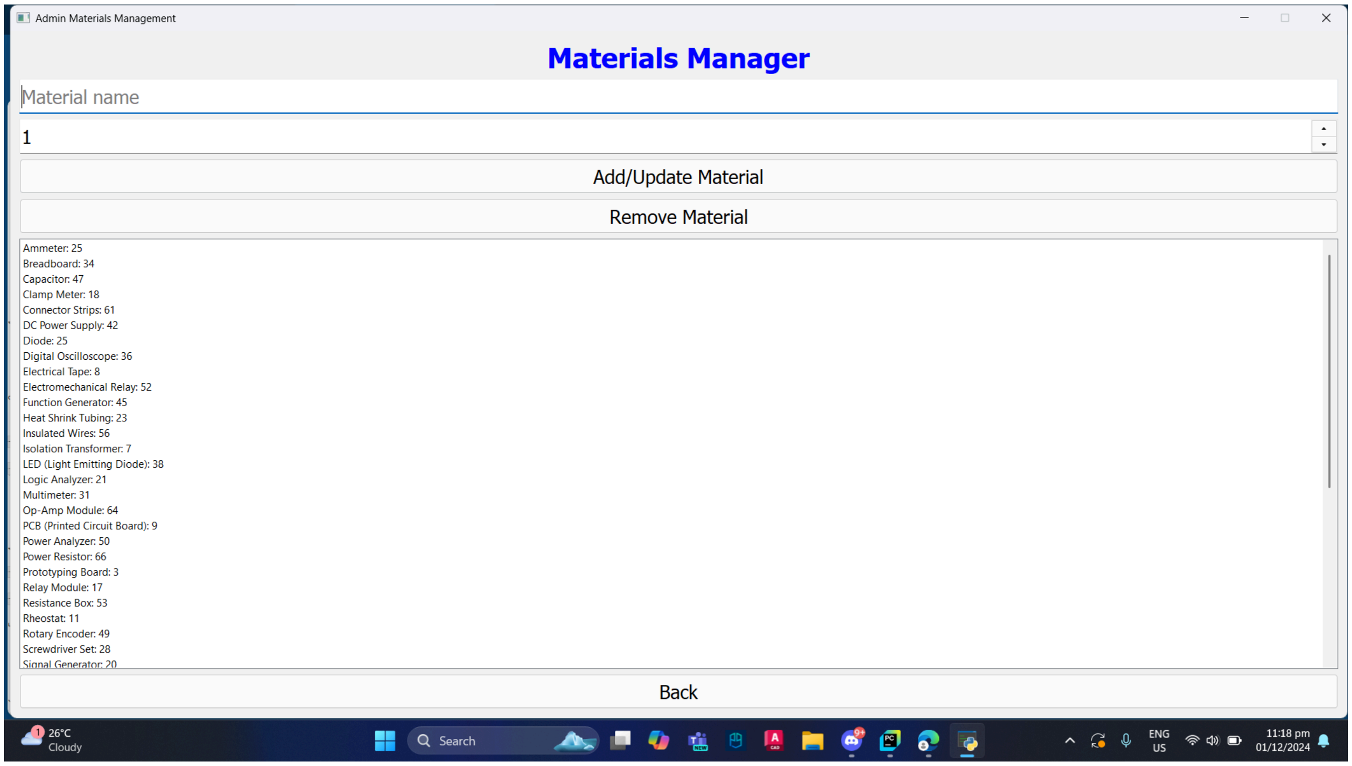
LABORATORY MATERIALS
Login / Registration

Enter your name

Enter your student number

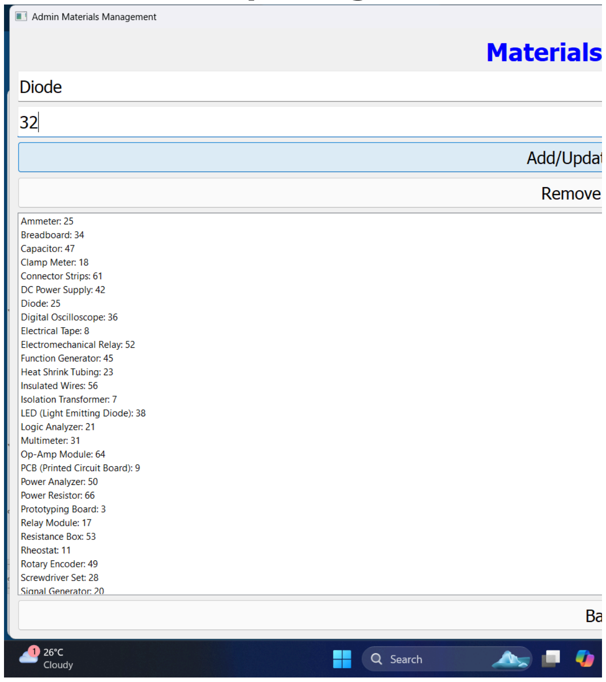
Logging in as Admin:





```
Ammeter: 25
Breadboard: 34
Capacitor: 47
Clamp Meter: 18
Connector Strips: 61
DC Power Supply: 42
Diode: 25
Digital Oscilloscope: 36
Electrical Tape: 8
Electromechanical Relay: 52
Function Generator: 45
Heat Shrink Tubing: 23
Insulated Wires: 56
Isolation Transformer: 7
LED (Light Emitting Diode): 38
Logic Analyzer: 21
Multimeter: 31
Op-Amp Module: 64
PCB (Printed Circuit Board): 9
Power Analyzer: 50
Power Resistor: 66
Prototyping Board: 3
Relay Module: 17
Resistance Box: 53
Rheostat: 11
Rotary Encoder: 49
Screwdriver Set: 28
Signal Generator: 20
```

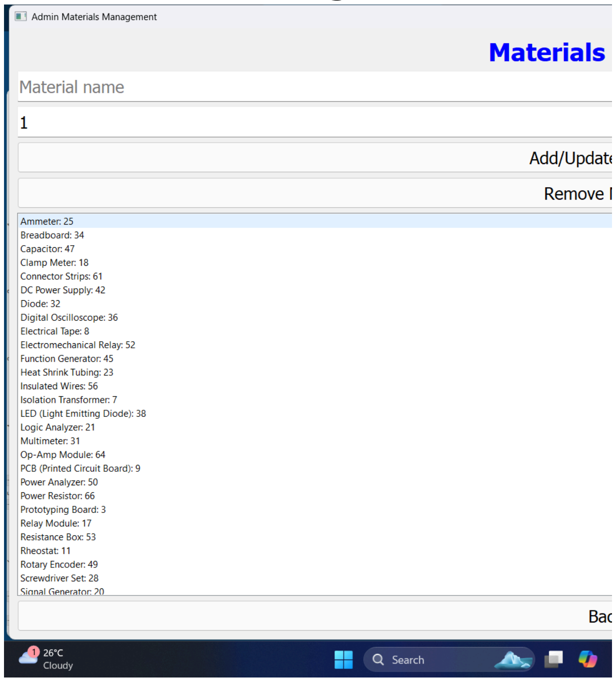
Updating Materials in Admin Materials Manager:



Diode

32

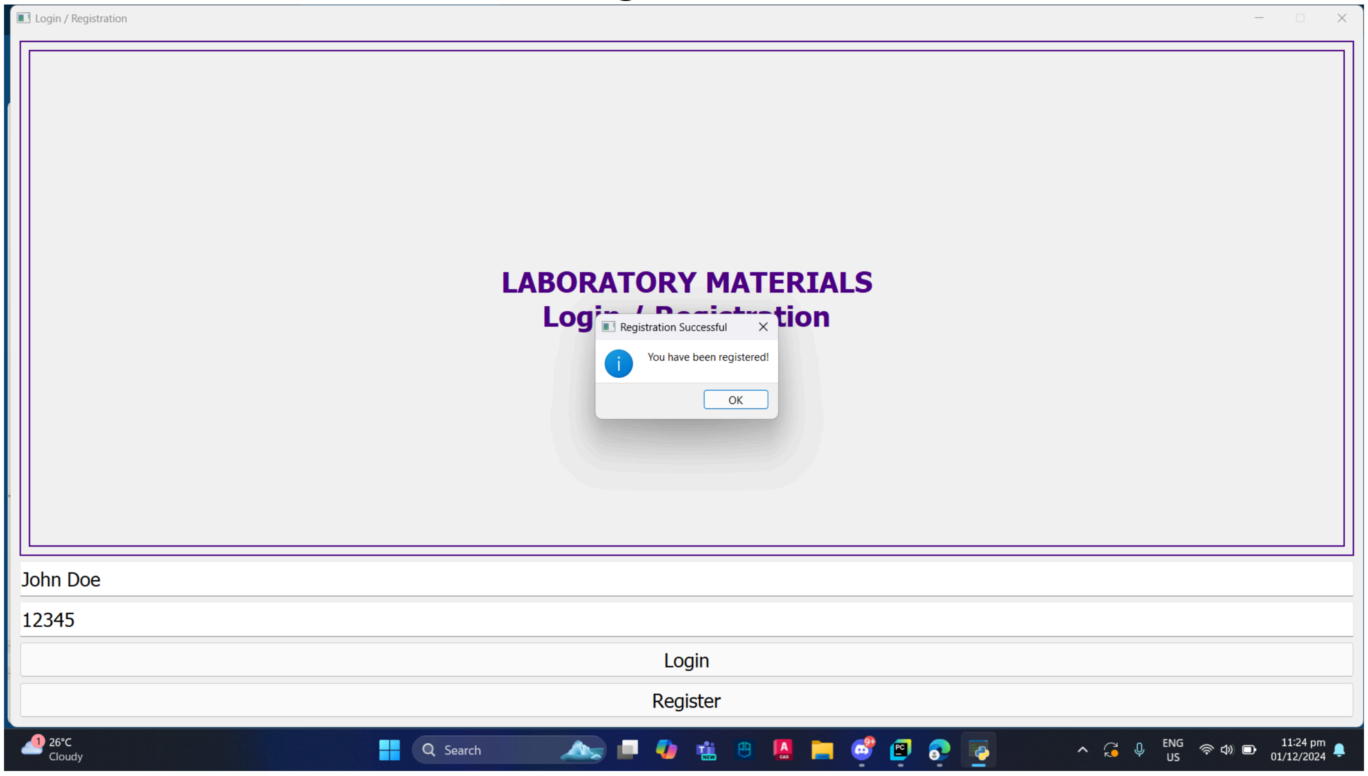
```
Ammeter: 25
Breadboard: 34
Capacitor: 47
Clamp Meter: 18
Connector Strips: 61
DC Power Supply: 42
Diode: 25
Digital Oscilloscope: 36
Electrical Tape: 8
Electromechanical Relay: 52
Function Generator: 45
Heat Shrink Tubing: 23
Insulated Wires: 56
Isolation Transformer: 7
LED (Light Emitting Diode): 38
Logic Analyzer: 21
Multimeter: 31
Op-Amp Module: 64
PCB (Printed Circuit Board): 9
Power Analyzer: 50
Power Resistor: 66
Prototyping Board: 3
Relay Module: 17
Resistance Box: 53
Rheostat: 11
Rotary Encoder: 49
Screwdriver Set: 28
Signal Generator: 20
```



1

```
Ammeter: 25
Breadboard: 34
Capacitor: 47
Clamp Meter: 18
Connector Strips: 61
DC Power Supply: 42
Diode: 32
Digital Oscilloscope: 36
Electrical Tape: 8
Electromechanical Relay: 52
Function Generator: 45
Heat Shrink Tubing: 23
Insulated Wires: 56
Isolation Transformer: 7
LED (Light Emitting Diode): 38
Logic Analyzer: 21
Multimeter: 31
Op-Amp Module: 64
PCB (Printed Circuit Board): 9
Power Analyzer: 50
Power Resistor: 66
Prototyping Board: 3
Relay Module: 17
Resistance Box: 53
Rheostat: 11
Rotary Encoder: 49
Screwdriver Set: 28
Signal Generator: 20
```

Registration:

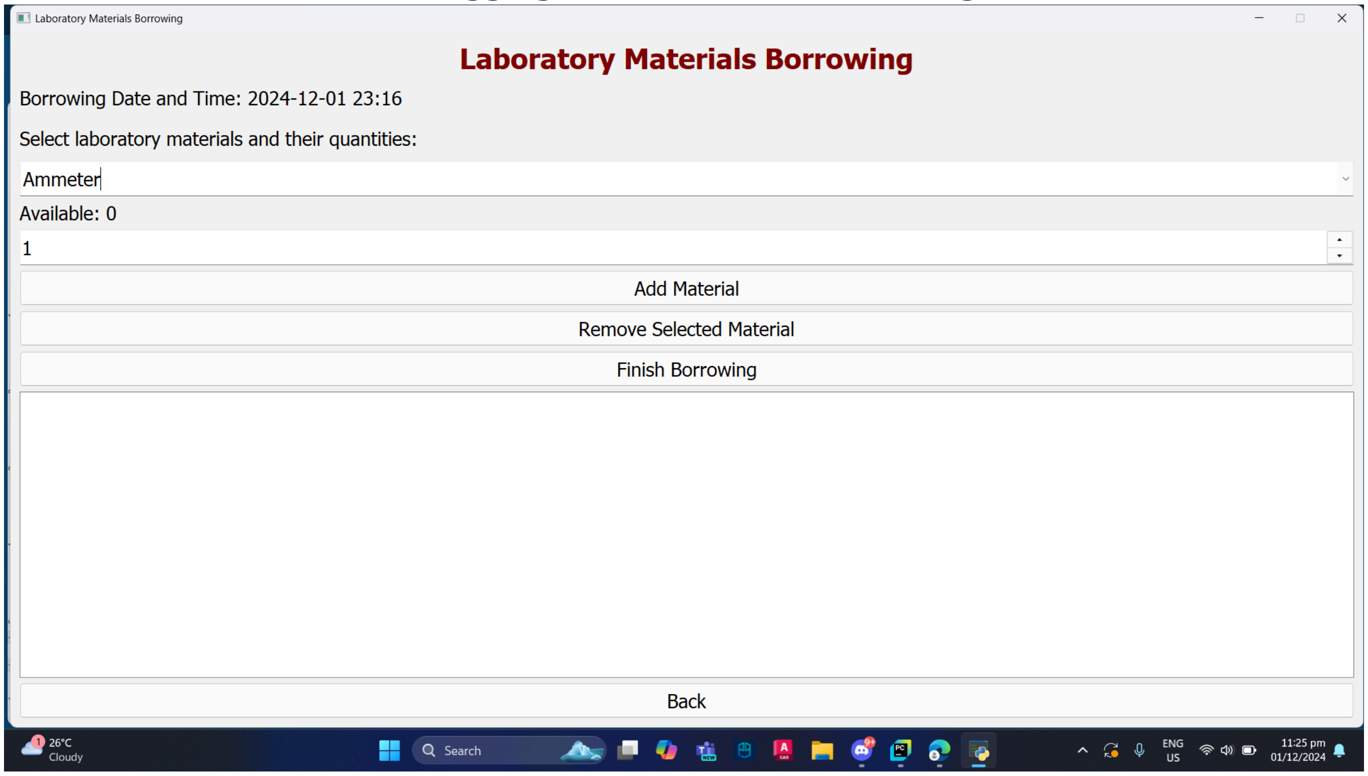


You have been registered!

John Doe

12345

After Logging in to Lab Materials Manager:


Laboratory Materials Borrowing

Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Available: 0

1

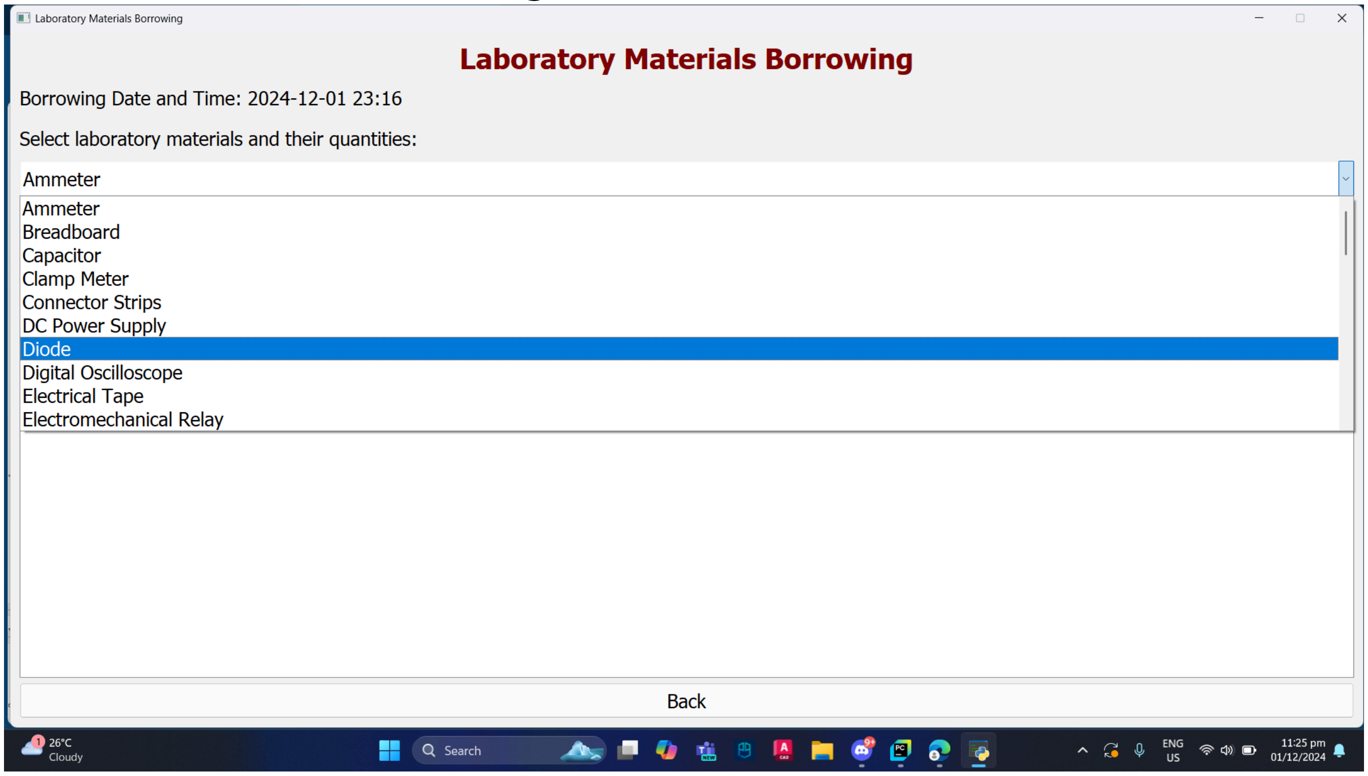
Add Material

Remove Selected Material

Finish Borrowing

Back

Borrowing Materials from Combobox:


Laboratory Materials Borrowing

Laboratory Materials Borrowing

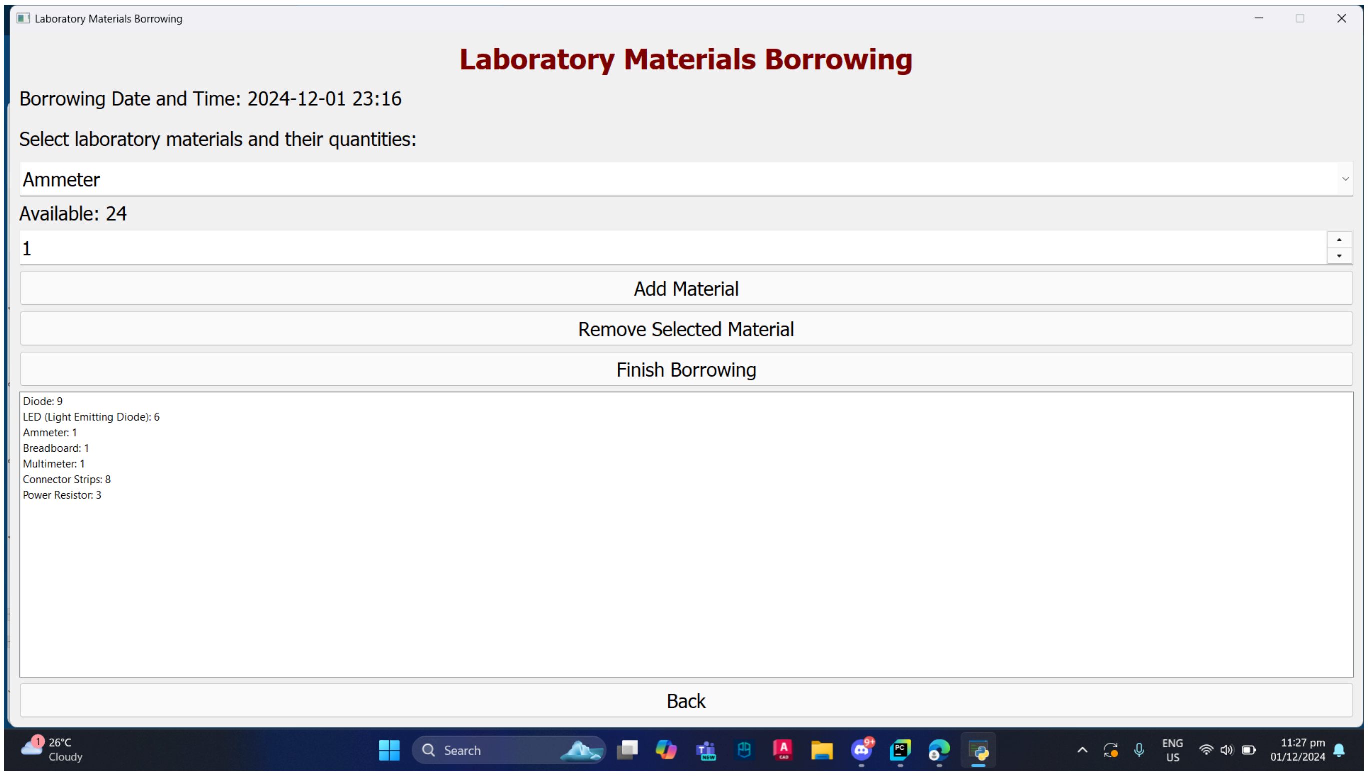
Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Ammeter
Breadboard
Capacitor
Clamp Meter
Connector Strips
DC Power Supply
Diode
Digital Oscilloscope
Electrical Tape
Electromechanical Relay

Back



Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

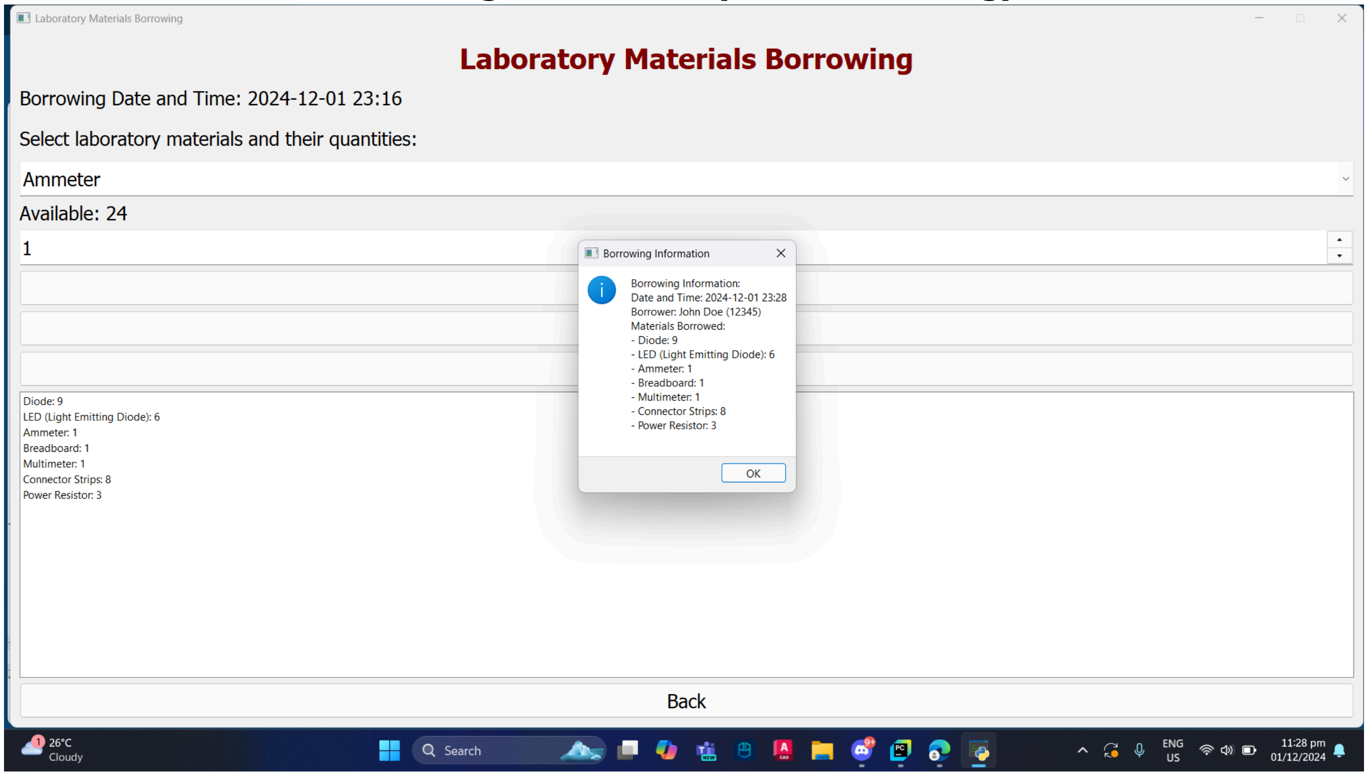
Ammeter

Available: 24

1

```
Diode: 9
LED (Light Emitting Diode): 6
Ammeter: 1
Breadboard: 1
Multimeter: 1
Connector Strips: 8
Power Resistor: 3
```

Borrowing Information (Finish Borrowing):



Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Available: 24

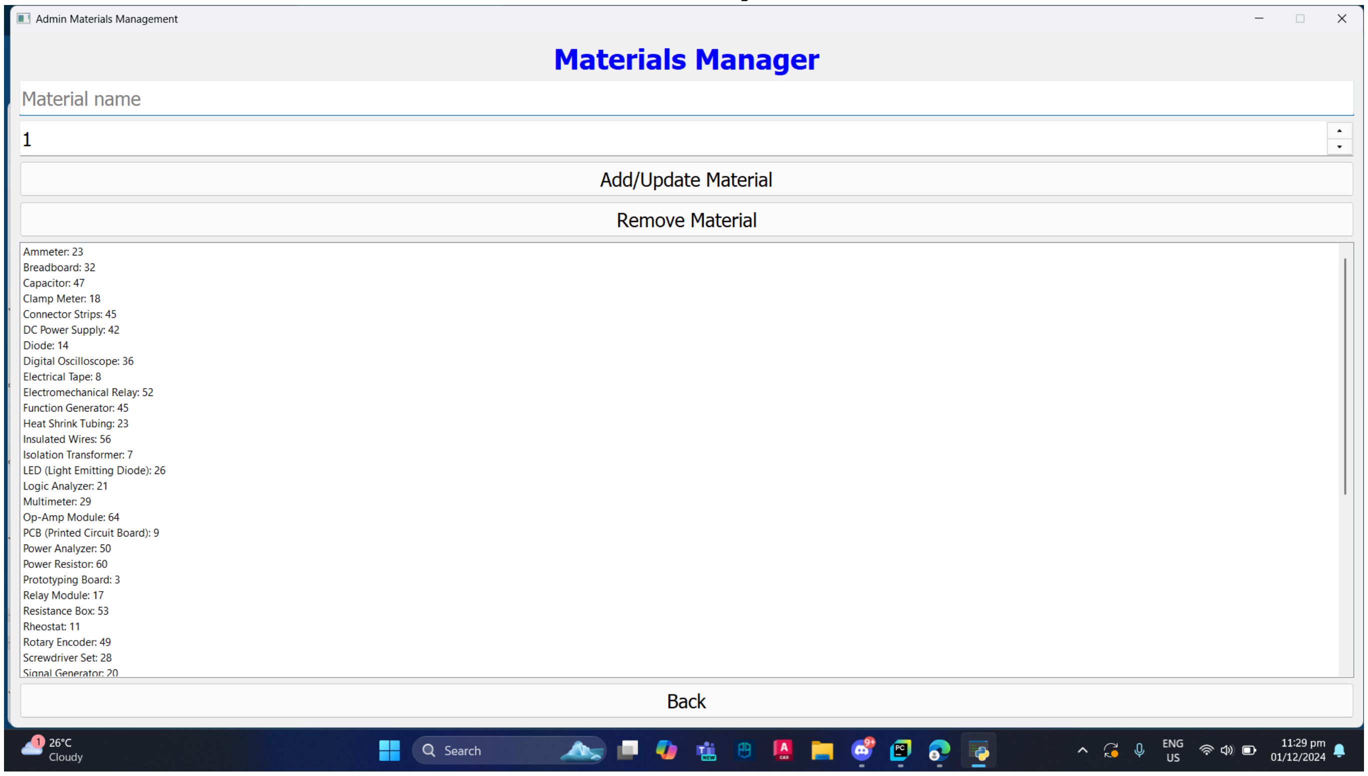
1

```
Diode: 9
LED (Light Emitting Diode): 6
Ammeter: 1
Breadboard: 1
Multimeter: 1
Connector Strips: 8
Power Resistor: 3
```

Borrowing Information

```
Borrowing Information:
Date and Time: 2024-12-01 23:28
Borrower: John Doe (12345)
Materials Borrowed:
- Diode: 9
- LED (Light Emitting Diode): 6
- Ammeter: 1
- Breadboard: 1
- Multimeter: 1
- Connector Strips: 8
- Power Resistor: 3
```

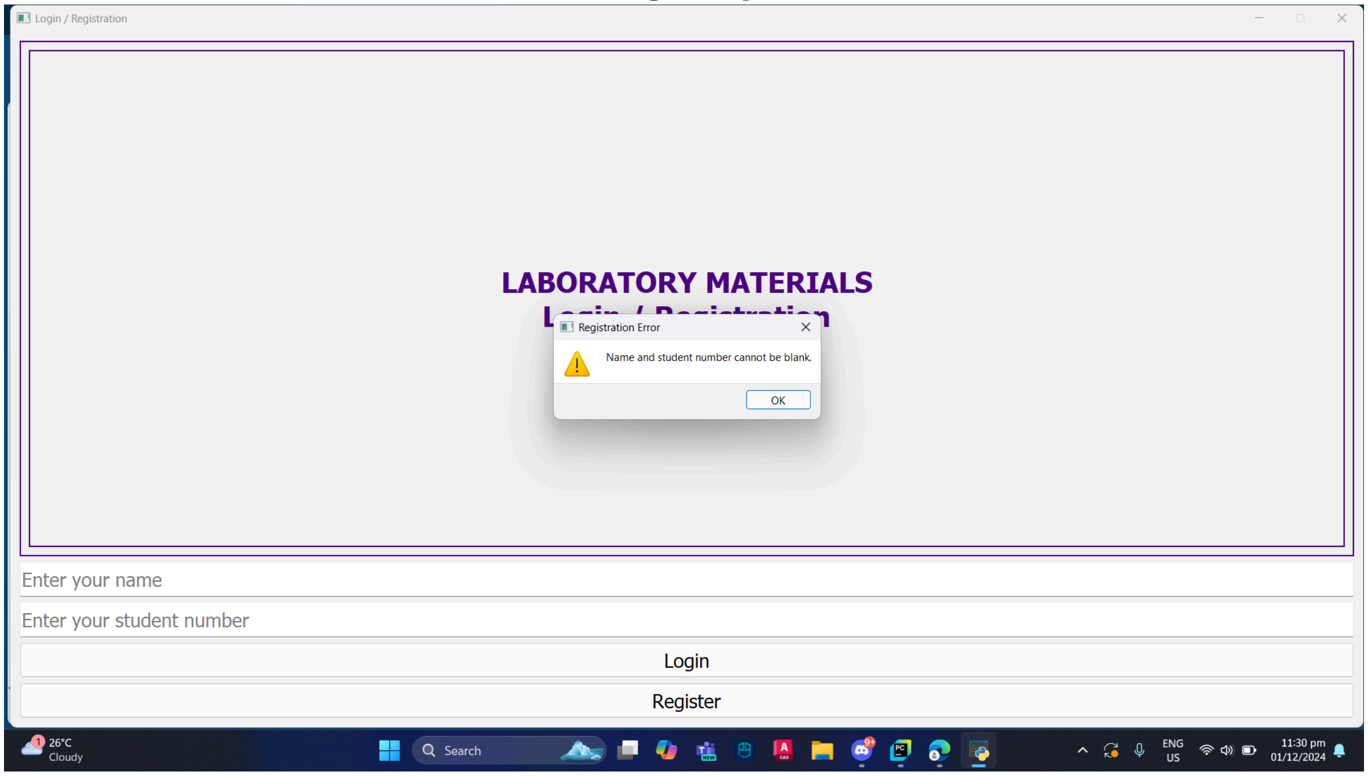
Admin Materials Manager after Borrowing (Materials borrowed deducted from database):



```
Ammeter: 23
Breadboard: 32
Capacitor: 47
Clamp Meter: 18
Connector Strips: 45
DC Power Supply: 42
Diode: 14
Digital Oscilloscope: 36
Electrical Tape: 8
Electromechanical Relay: 52
Function Generator: 45
Heat Shrink Tubing: 23
Insulated Wires: 56
Isolation Transformer: 7
LED (Light Emitting Diode): 26
Logic Analyzer: 21
Multimeter: 29
Op-Amp Module: 64
PCB (Printed Circuit Board): 9
Power Analyzer: 50
Power Resistor: 60
Prototyping Board: 3
Relay Module: 17
Resistance Box: 53
Rheostat: 11
Rotary Encoder: 49
Screwdriver Set: 28
Signal Generator: 20
```

Error Message Box Conditions:

No Login Input:


Login / Registration

LABORATORY MATERIALS
Login / Registration

Registration Error

Name and student number cannot be blank.

OK

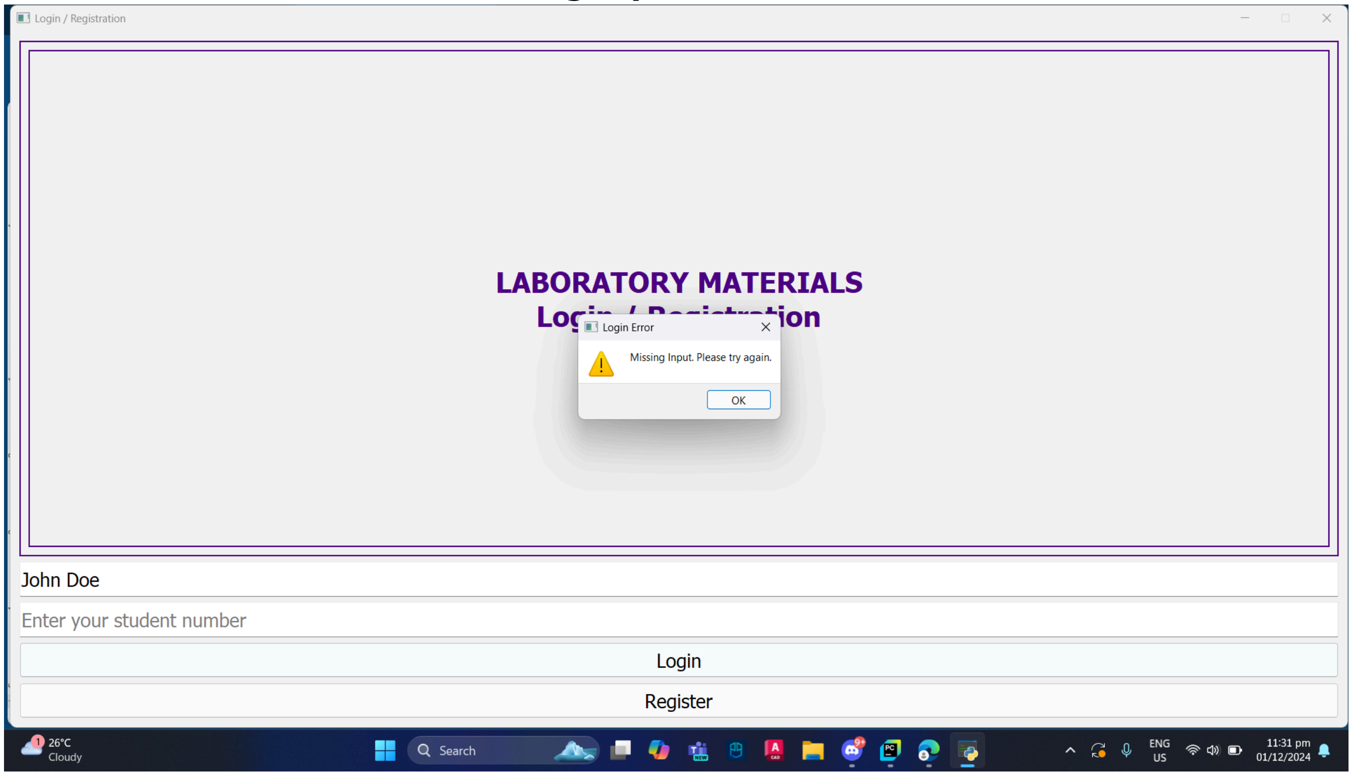
Enter your name

Enter your student number

Login

Register

Missing Input Credentials:


Login / Registration

LABORATORY MATERIALS
Login / Registration

Login Error

Missing Input. Please try again.

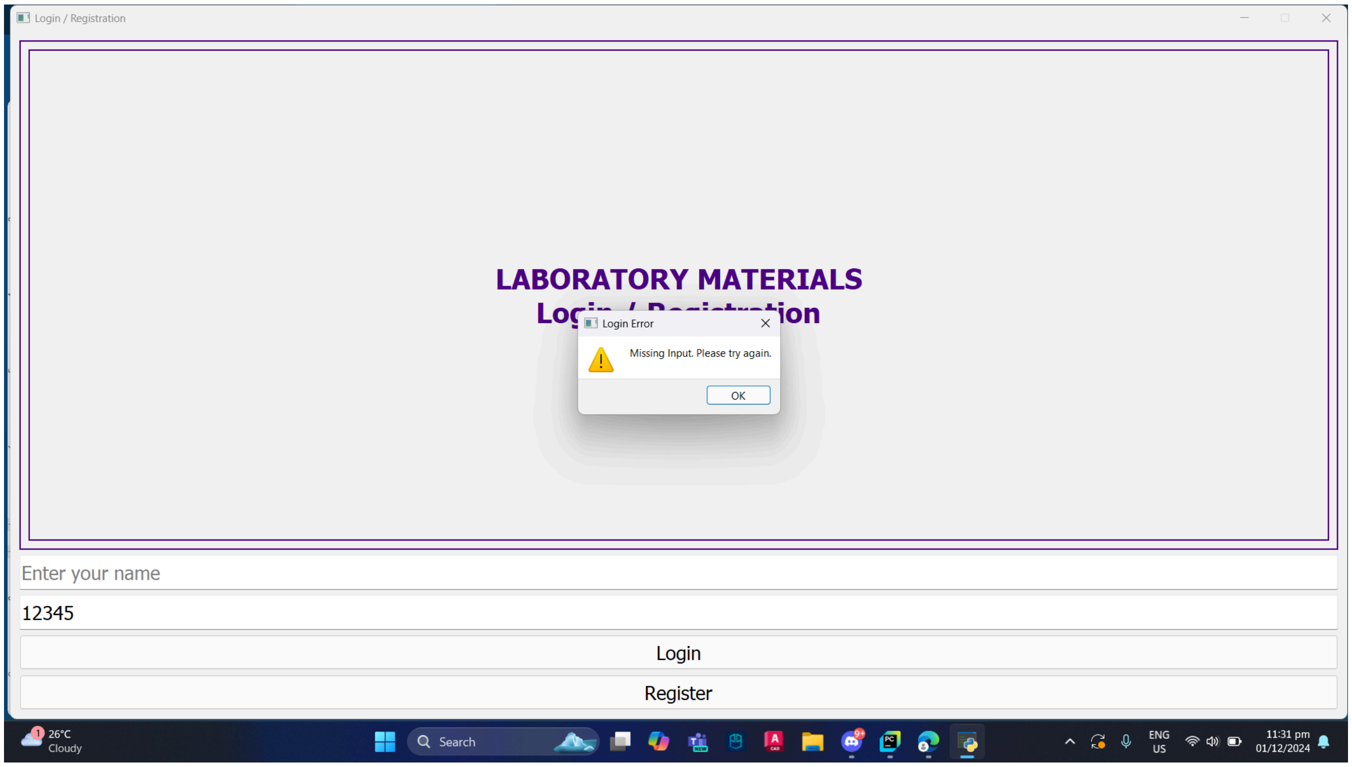
OK

John Doe

Enter your student number

Login

Register


Login / Registration

LABORATORY MATERIALS
Login / Registration

Login Error

Missing Input. Please try again.

OK

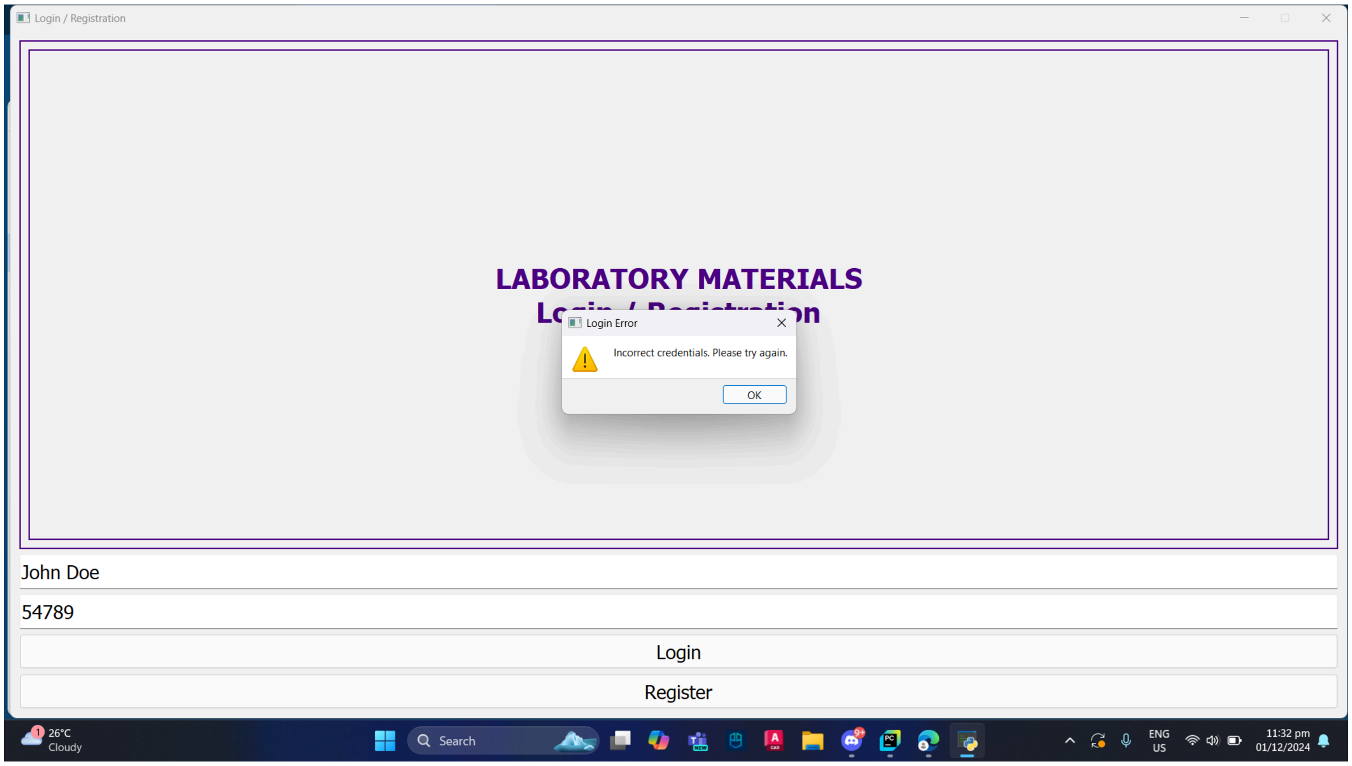
Enter your name

12345

Login

Register

Incorrect Credentials:


Login / Registration

LABORATORY MATERIALS
Login / Registration

Login Error

Incorrect credentials. Please try again.

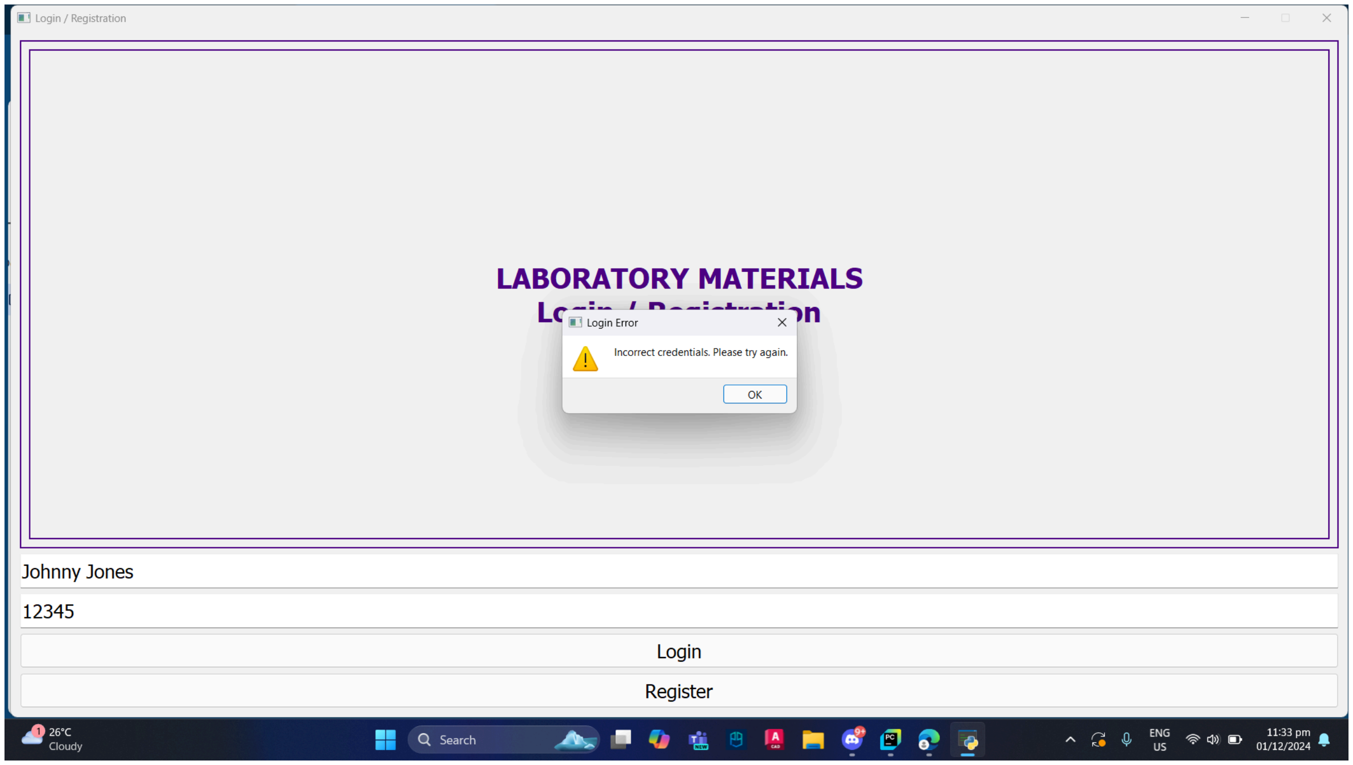
OK

John Doe

54789

Login

Register



Login / Registration

LABORATORY MATERIALS
Login / Registration

Login Error

Incorrect credentials. Please try again.

OK

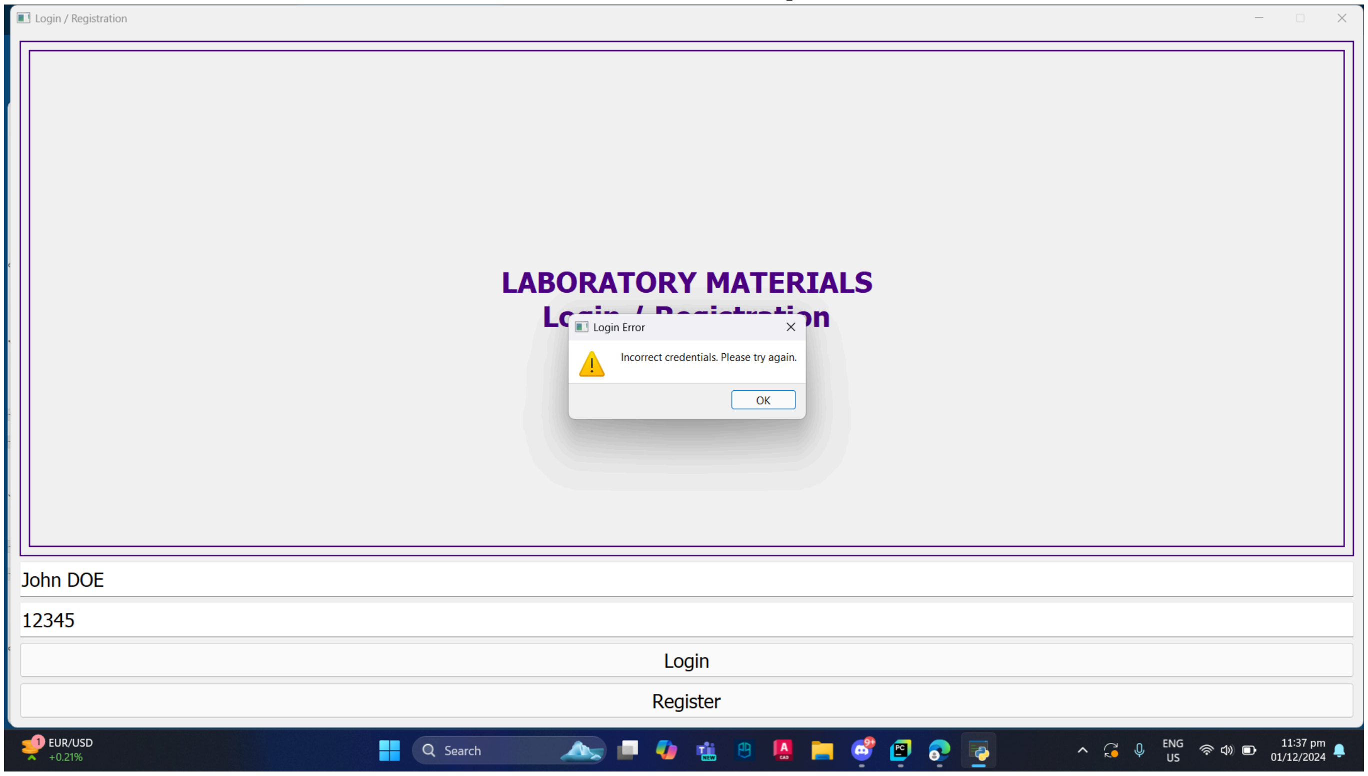
Johnny Jones

12345

Login

Register

Case Sensitivity:



Login / Registration

LABORATORY MATERIALS
Login / Registration

Login Error

Incorrect credentials. Please try again.

OK

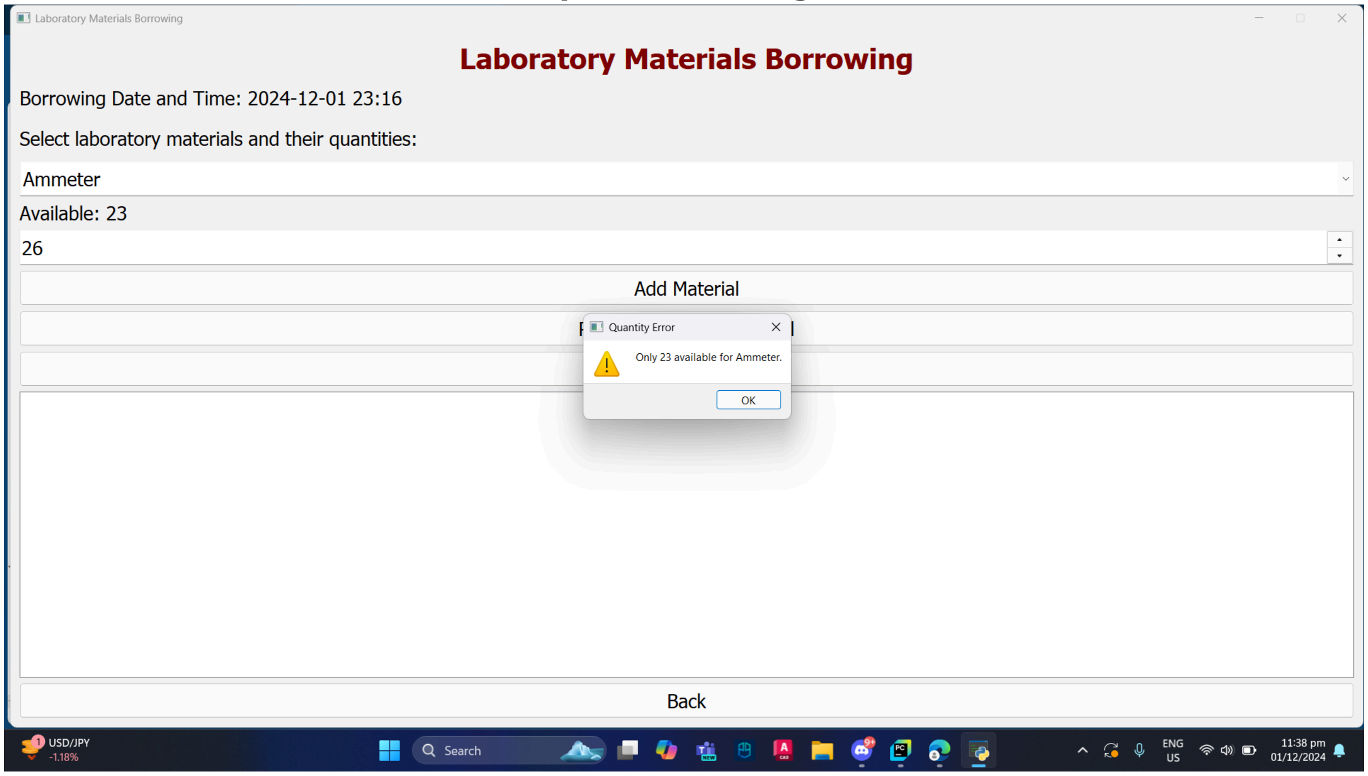
John DOE

12345

Login

Register

Borrow Request Exceeding Available:


Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Available: 23

26

Add Material

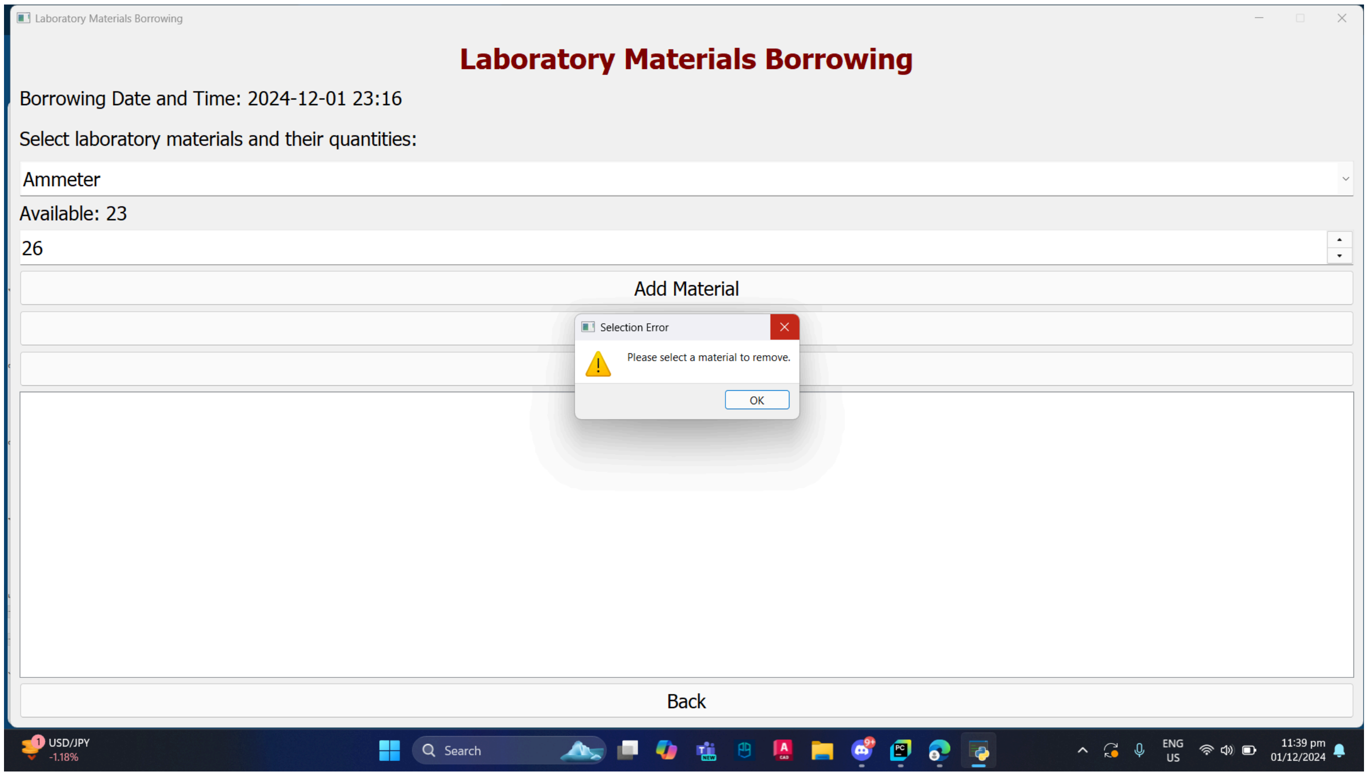
Quantity Error

Only 23 available for Ammeter.

OK

Back

No Material Selected for Removal:


Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Available: 23

26

Add Material

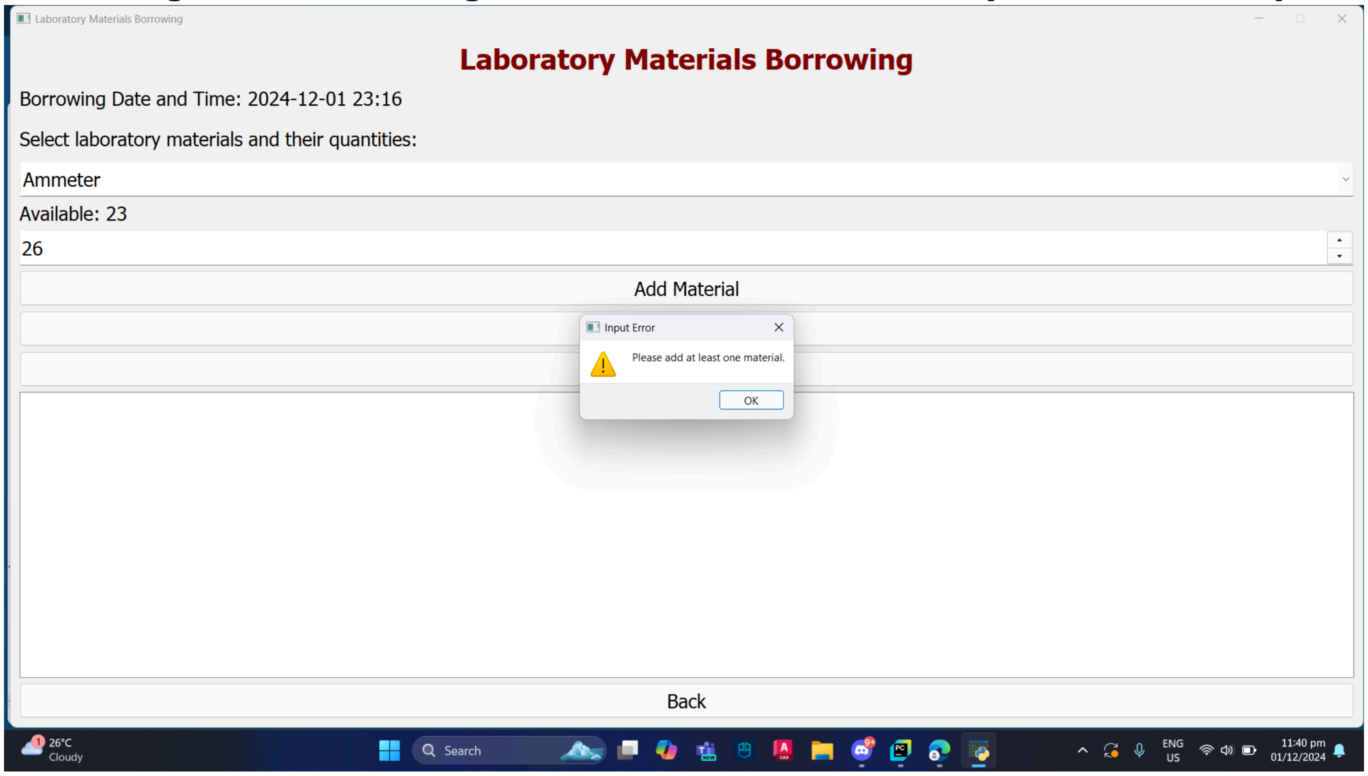
Selection Error

Please select a material to remove.

OK

Back

Clicking "Finish Borrowing" when no Materials Borrowed (Hit 'Back' instead):


Laboratory Materials Borrowing

Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-01 23:16

Select laboratory materials and their quantities:

Ammeter

Available: 23

26

Add Material

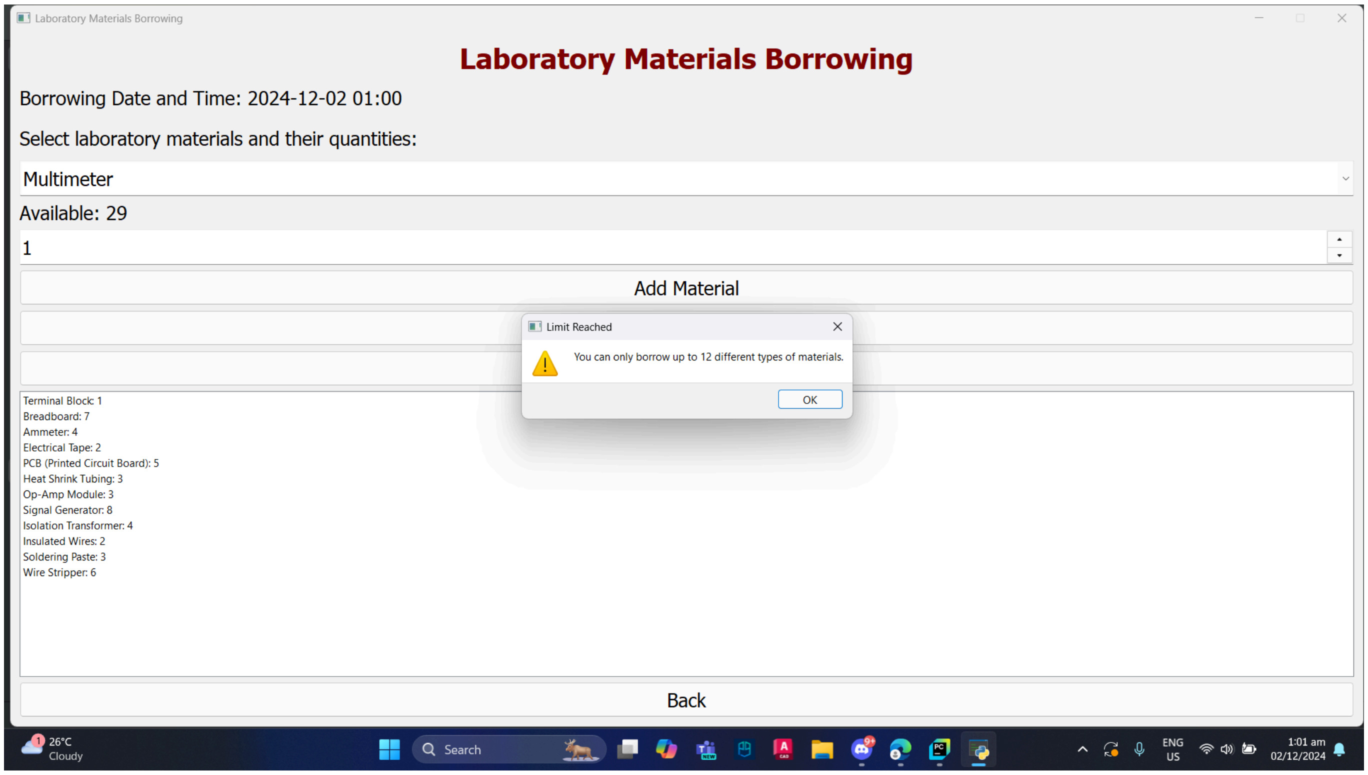
Input Error

Please add at least one material.

OK

Back

Materials Borrowed Exceeding 12 Different Items:


Laboratory Materials Borrowing

Borrowing Date and Time: 2024-12-02 01:00

Select laboratory materials and their quantities:

Multimeter

Available: 29

1

Add Material

Limit Reached

You can only borrow up to 12 different types of materials.

OK

Terminal Block: 1
Breadboard: 7
Ammeter: 4
Electrical Tape: 2
PCB (Printed Circuit Board): 5
Heat Shrink Tubing: 3
Op-Amp Module: 3
Signal Generator: 8
Isolation Transformer: 4
Insulated Wires: 2
Soldering Paste: 3
Wire Stripper: 6

Back

7. Conclusion

I. Conclusion

The implementation of the Laboratory Materials Management System provides an efficient and user-friendly interface for students and administrators to manage laboratory materials. The system includes a login and registration feature, allowing students to access their accounts securely. The administrative functionalities enable the updating and management of materials, ensuring that the inventory is accurate and up-to-date. Additionally, the borrowing feature allows students to request materials easily, with appropriate error handling to address common issues such as incorrect credentials or insufficient inventory. Overall, the system improves the process of managing lab materials, making it more efficient and user-friendly.

II. Recommendations

- User Feedback Mechanism: Establish a feedback system within the application to allow users to report issues and suggest improvements. This feedback will be invaluable for ongoing development, helping to prioritize future updates and enhancements.

- Regular Updates and Maintenance: Implement a schedule for regular system updates to fix bugs, add new features, and enhance security. This will ensure that the application remains reliable and secure over time.

- Mobile Application Development: Develop a mobile-friendly version of the system to enhance accessibility. This will allow users to check material availability and manage their borrowing requests from their smartphones, increasing convenience and engagement.

- Enhanced Security Features: Incorporate additional security measures, such as two-factor authentication, to protect user accounts and sensitive data.

- Real-time Inventory Updates: Implement real-time inventory updates using a local network or cloud-based solution. This will ensure that all users have access to the most current data regarding material availability, preventing conflicts when multiple users attempt to borrow the same materials.

- Database Backup and Recovery: Establish a regular data backup and recovery system to prevent data loss in case of system failures. Automated backups of the material database and user accounts will ensure quick restoration when needed.

- Integration with Academic Systems: Explore the integration of the materials management system with existing academic systems (such as student information systems) to streamline processes. This will allow for automatic updates of student information and borrowing history, improving efficiency for both students and administrators.

By addressing these recommendations, the Laboratory Materials Management System can continue to improve, effectively serve its users, and enhance the educational experience for students while improving operational efficiency for administrators.

8. References

csv — CSV File Reading and Writing — Python 3.8.1 documentation. (2020). Python.org.

https://docs.python.org/3/library/csv.html

Data Abstraction in Python. (2023, October 17). GeeksforGeeks.

https://www.geeksforgeeks.org/data-abstraction-in-python/

QWidget — Qt for Python. (2024). Doc.qt.io.

https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QWidget.html#more *sys — System-specific parameters and functions — Python 3.7.3 documentation*. (2010). Python.org. https://docs.python.org/3/library/sys.html