LHC_2 Exam

Name: PAYAL MAKWANA

Roll no.: 92301733057

Task: App Design Using Kivy with Database Integration and Login/Sign-Up Logic

Part 1: Basic UI Design with Kivy

Develop a simple app application using the Kivy framework that includes:

- A login screen with fields for entering a username and password.
- A sign-up screen with fields for username, email, and password.
- A navigation option to toggle between the login and sign-up screens.

Tasks:

- Implement the layout for both login and sign-up screens using appropriate Kivy widgets such as TextInput, Button, and Label.
- Ensure there are "Login" and "Sign Up" buttons to trigger the respective actions.

Solution:

To accomplish this task, I used **KivyMD**, so we can take advantage of the material design components that come with the framework. KivyMD provides better styling and animations.

Kivy is an open-source software library for developing applications, and KivyMD is a collection of widgets for use with Kivy.

```
PROBLEMS  PORTS  DEBUG CONSOLE  TERMINAL  OUTPUT
PS D:\SEM 3 Subjects\Python\LHC2>
pip install kivymd
```

UI Design

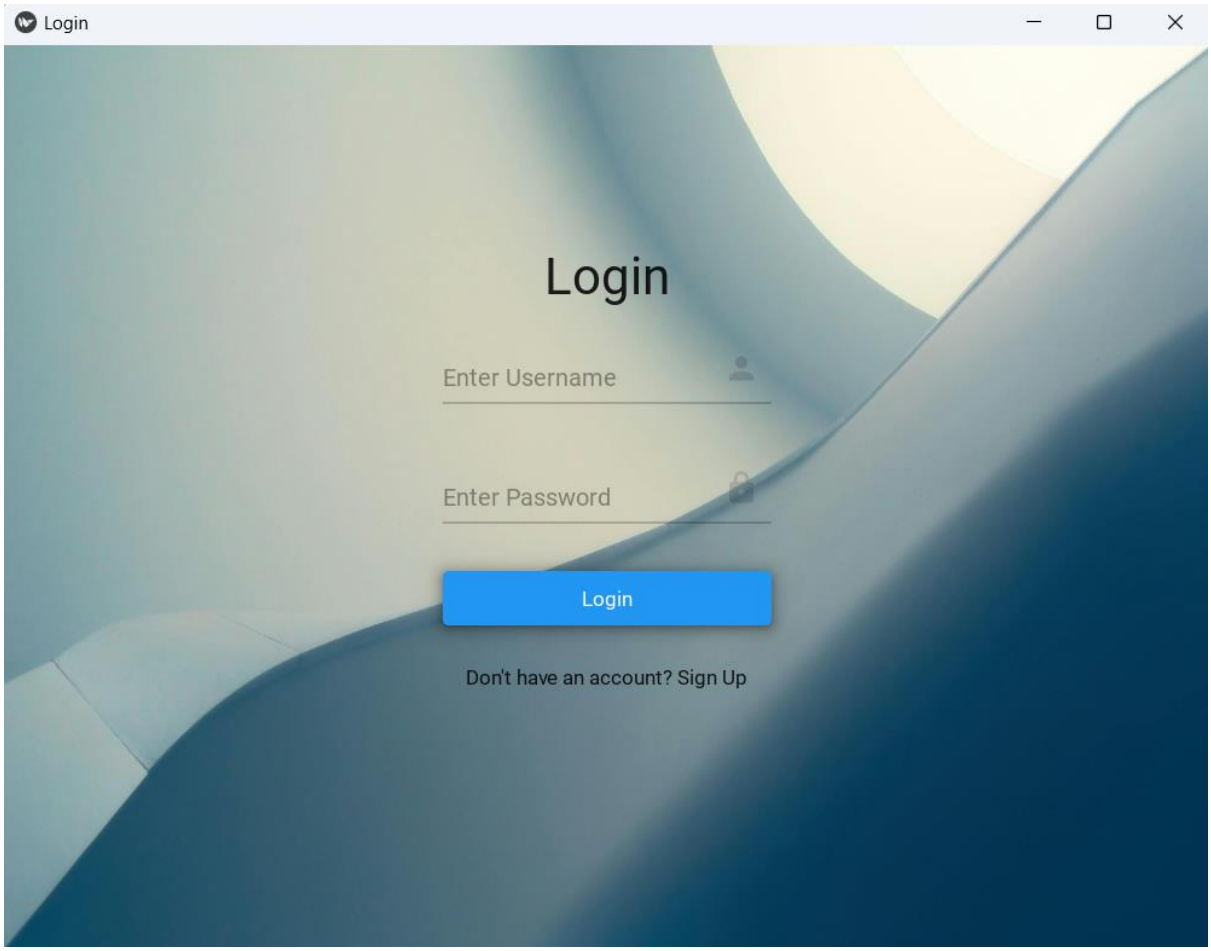
KivyMD Components:

- **MDTextField**: Provides a more better look with built-in icons and animations.
- **MDRaisedButton**: For primary actions like "Login" and "Sign Up" with a material design effect.
- **MDFlatButton**: For secondary actions like switching between "Login" and "Sign Up" screens.
- **MDDialog**: For displaying pop-up messages with a sleek material design.

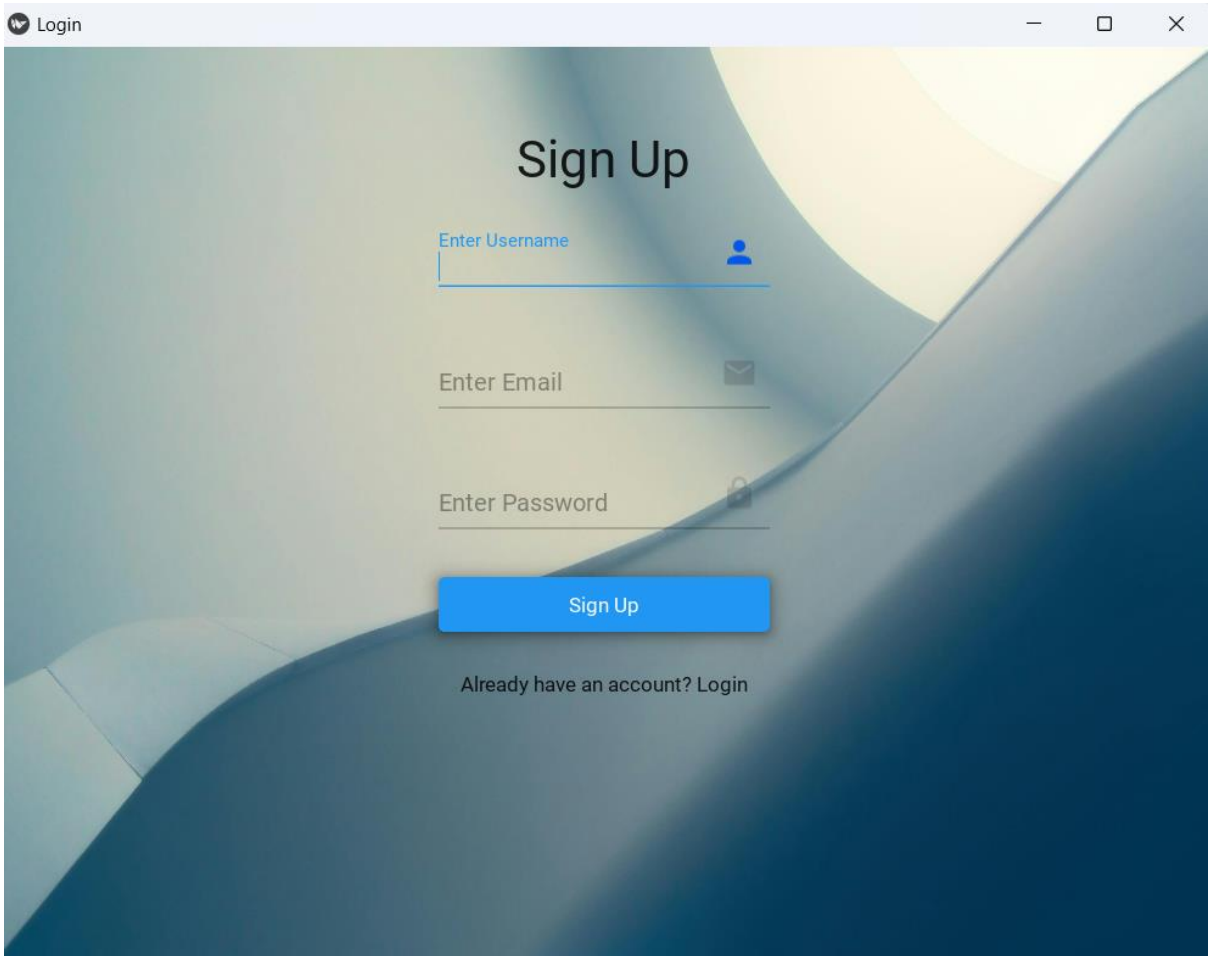
I have set the app color theme to blue. And set one background image.

Input fields and buttons are centered with padding and spacing.

On running the app, there will be two fields for username and password for Login page.



And if the user don't have an account, Sign up page will appear which will have three fields for username, email and password.



Code:

```
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton

class LoginApp(MDApp):
    dialog = None

    def build(self):
        create_database()  # Create the user database
        self.theme_cls.primary_palette = "Blue"
        self.theme_cls.theme_style = "Light"
        return Builder.load_string(KV)

    def switch_to_signup(self):
        self.root.current = 'signup'

    def switch_to_login(self):
        self.root.current = 'login'
```

```
    def login(self, username, password):
        # Ensure that fields are not empty
        if not username or not password:
            self.show_dialog("Error", "Please fill in all fields.")
            return

        # Connect to the database to verify credentials
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM users WHERE username = ?', (username,))
        user = cursor.fetchone()
        conn.close()

        # Check if user exists and password matches
        if user is None:
            self.show_dialog("Error", "Username not found.")
        elif user[3] == password:  # Password stored as plain text, so simple
comparison
            self.show_dialog("Login", "Login successful")
        else:
            self.show_dialog("Error", "Incorrect password.")

    def signup(self, username, email, password):
        # Ensure that all fields are filled in
        if not username or not email or not password:
            self.show_dialog("Error", "Please fill in all fields.")
            return

        # Insert user data into the database
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        try:
            cursor.execute('INSERT INTO users (username, email, password)
VALUES (?, ?, ?)',
                           (username, email, password))  # Store password as
plain text
            conn.commit()
            self.show_dialog("Sign Up", "Sign up successful")
        except sqlite3.IntegrityError:
            self.show_dialog("Error", "Username already exists.")
        finally:
            conn.close()

    def show_dialog(self, title, message):
        # Display dialog messages
        if not self.dialog:
            self.dialog = MDDialog(
```

```
                title=title,
                text=message,
                buttons=[
                    MDFlatButton(
                        text="CLOSE",
                        on_release=lambda x: self.dialog.dismiss()
                    ),
                ],
            )
        self.dialog.title = title
        self.dialog.text = message
        self.dialog.open()

if __name__ == "__main__":
    LoginApp().run()
```

Part 2: Database Integration

Connect your app to a database (SQLite) to store user credentials. Implement the following functionalities:

- When a user signs up, store their details (username, email, password) in the database.
- During login, verify the entered credentials by checking them against the stored data.

Tasks:

- Design the database schema to handle user information.
- Write the Python code to establish a connection between the app and the database.
- Implement both sign-up and login functionalities using the database.

Database Connectivity:

We have to create a **SQLite table** called users to store user credentials (username, email, password).

A users table is created with fields: id, username, email, and password. The username field is marked as **UNIQUE** to prevent duplicate entries.

A connection to the SQLite database is established in the create_database method. If the database file does not exist, it will be created.

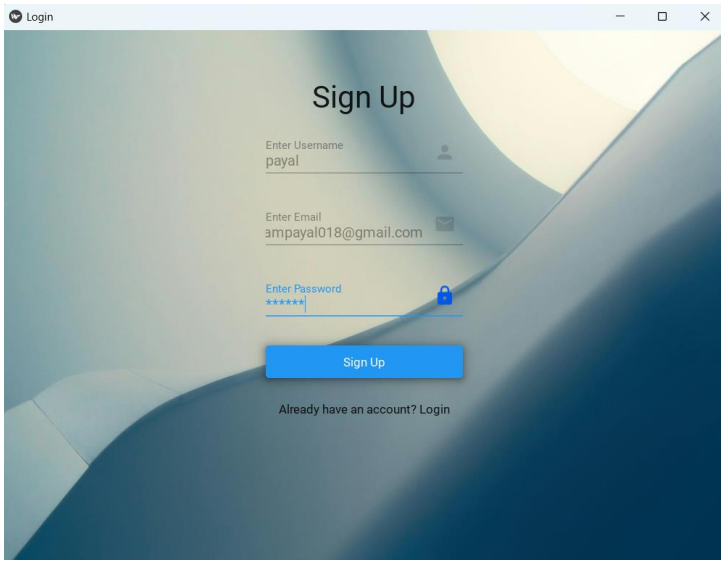
Sql query:

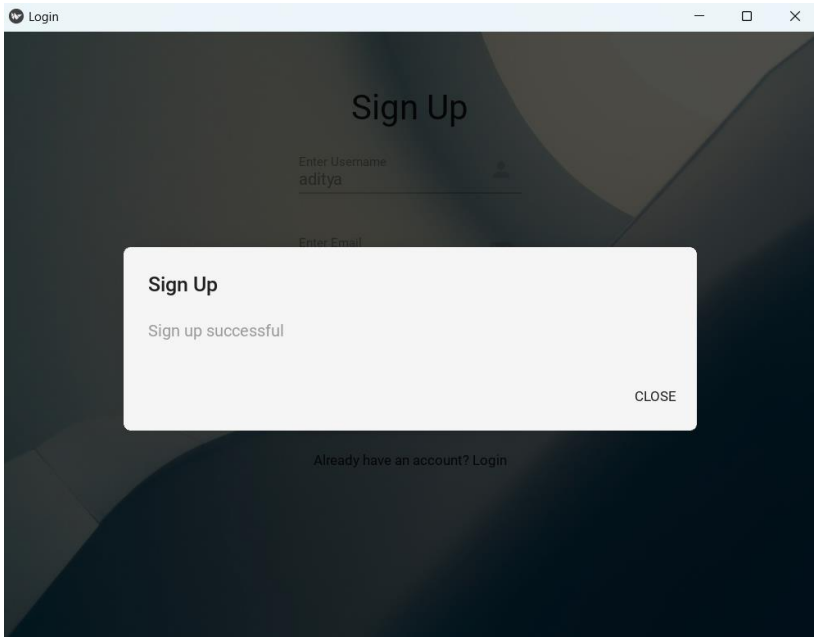
```
DB_NAME = 'users.db'

def create_database():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE,
            email TEXT,
            password TEXT
        )
    ''')
    conn.commit()
    conn.close()
```

Code:

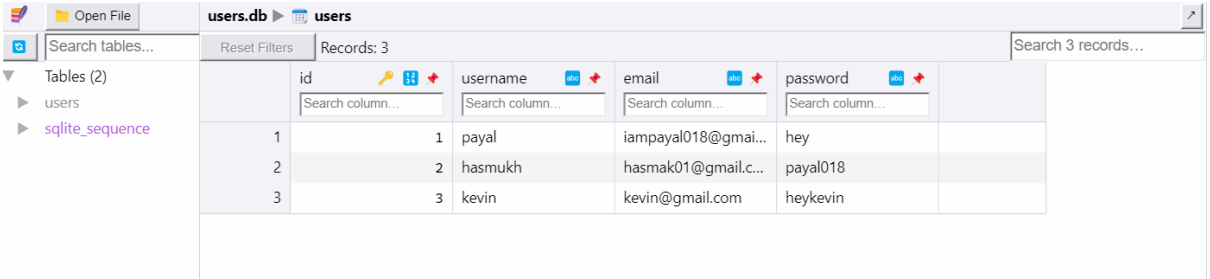
```
import sqlite3

DB_NAME = 'users.db'

# Create a database for storing user credentials
def create_database():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE,
            email TEXT,
            password TEXT
        )
    ''')
    conn.commit()
    conn.close()
```

UI DESIGN:





users.db is created



Part 3: User Authentication Logic

Implement the logic to handle user authentication:

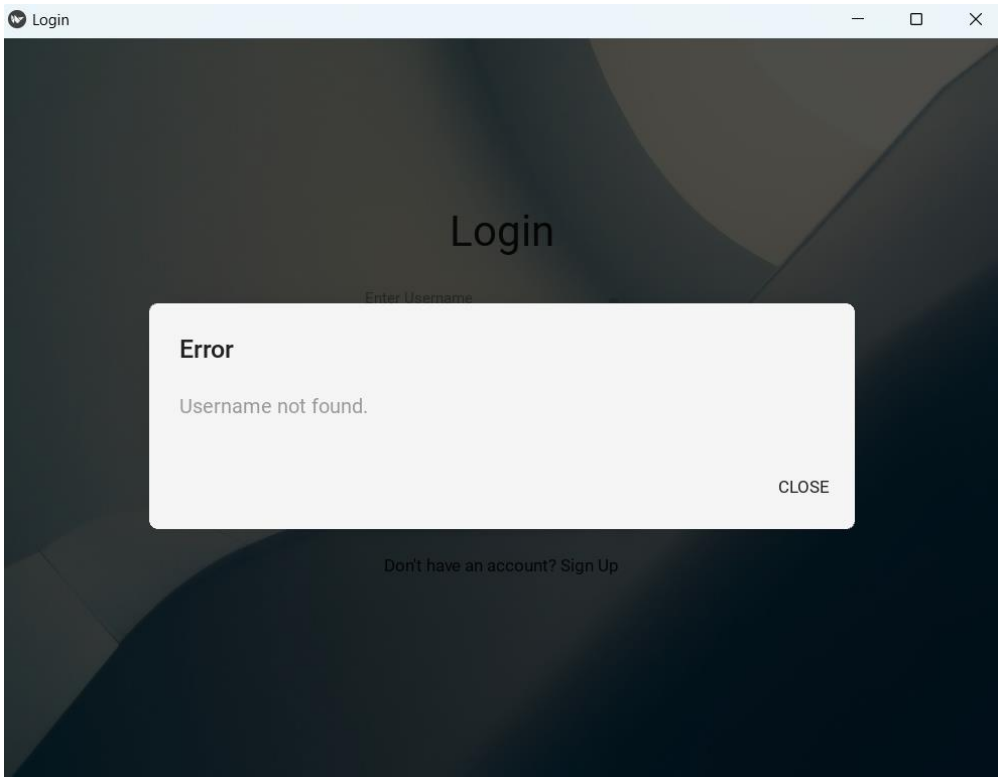
- When a user logs in, verify their username and password against the database records.
- Display appropriate error messages if the username is not found or the password is incorrect.
- Ensure that duplicate usernames cannot be created during the sign-up process.

Tasks:

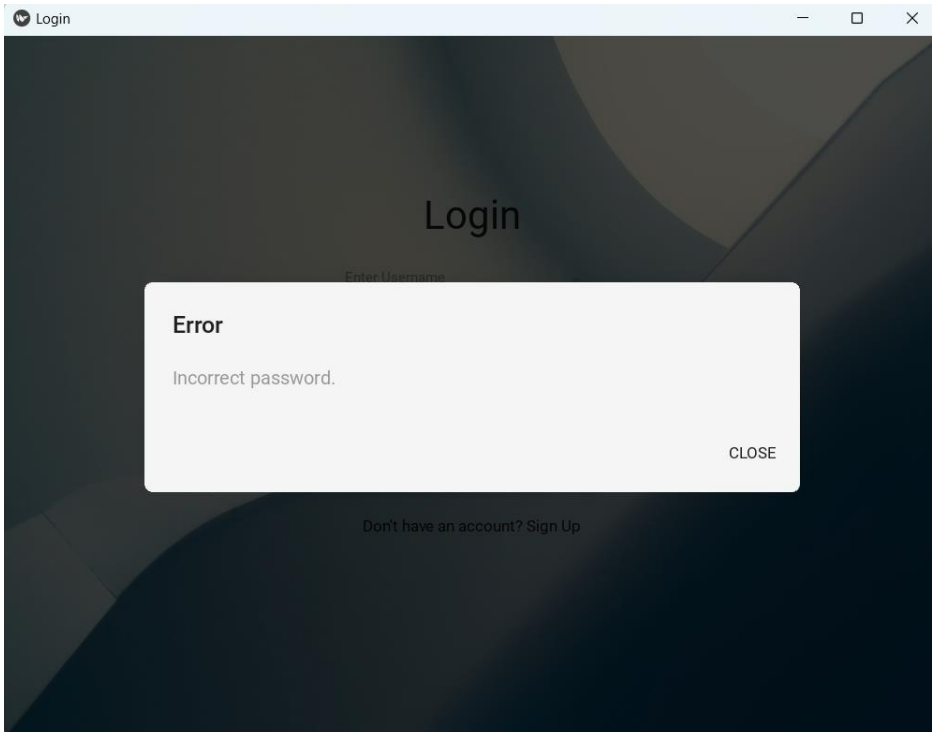
- Write Python logic to authenticate users.
- Include error handling for incorrect login details.
- Prevent the creation of duplicate usernames during sign-up.

User Authentication:

Before checking the password, we first verify if the username exists in the database. If it doesn't, we display an error message: **"Username not found".**

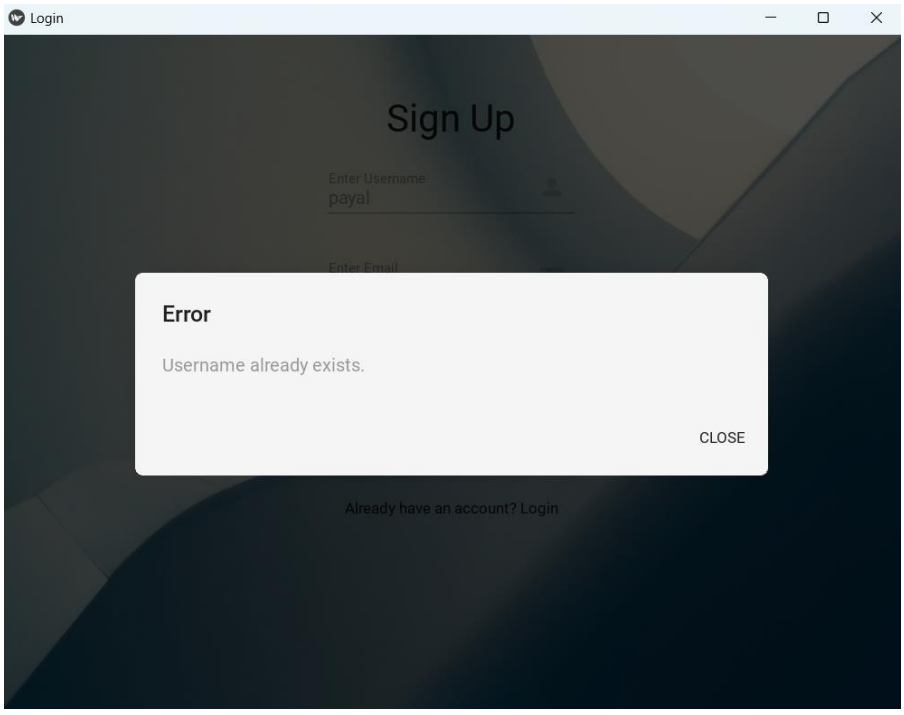


If the username exists, we then compare the entered password with the stored password. If it doesn't match, we display **"Incorrect password"**.

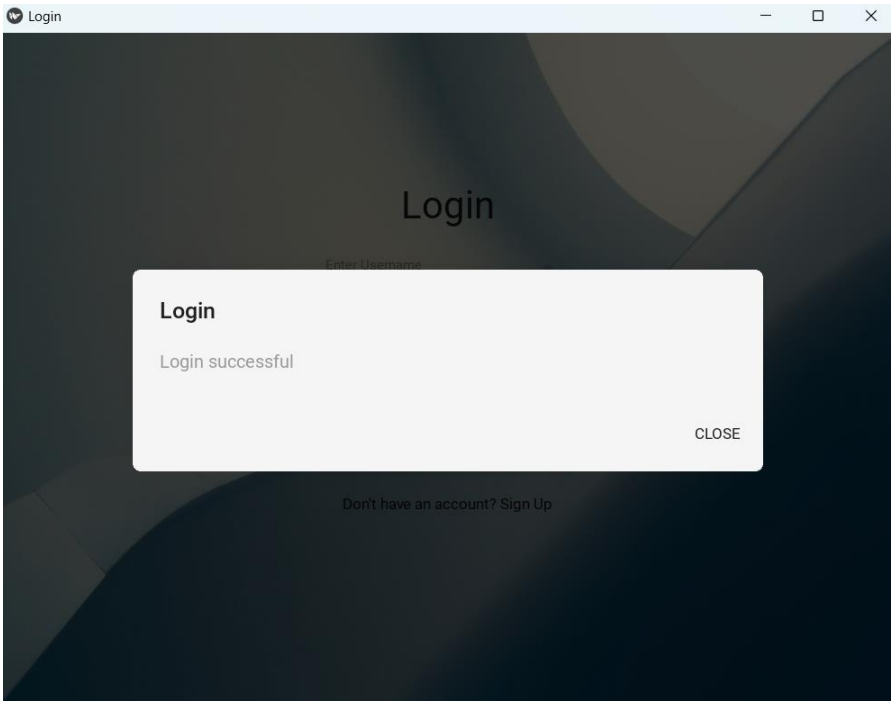


We catch **sqlite3.IntegrityError** exceptions when trying to insert a new user. This handles the scenario where a duplicate username is being used.

During sign up, If the username already exists, an error message is displayed: **"Username already exists"**. Please choose a different username.



After successfully login, the popup message will be displayed as **"Login successful".**



Code:

```
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton

class LoginApp(MDApp):
    dialog = None

    def build(self):
        create_database()  # Create the user database
        self.theme_cls.primary_palette = "Blue"
        self.theme_cls.theme_style = "Light"
        return Builder.load_string(KV)

    def switch_to_signup(self):
        self.root.current = 'signup'

    def switch_to_login(self):
        self.root.current = 'login'

    def login(self, username, password):
        # Ensure that fields are not empty
        if not username or not password:
            self.show_dialog("Error", "Please fill in all fields.")
            return

        # Connect to the database to verify credentials
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
```

```
        cursor.execute('SELECT * FROM users WHERE username = ?', (username,))
        user = cursor.fetchone()
        conn.close()

        # Check if user exists and password matches
        if user is None:
            self.show_dialog("Error", "Username not found.")
        elif user[3] == password:  # Password stored as plain text, so simple
comparison
            self.show_dialog("Login", "Login successful")
        else:
            self.show_dialog("Error", "Incorrect password.")

    def signup(self, username, email, password):
        # Ensure that all fields are filled in
        if not username or not email or not password:
            self.show_dialog("Error", "Please fill in all fields.")
            return

        # Insert user data into the database
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        try:
            cursor.execute('INSERT INTO users (username, email, password)
VALUES (?, ?, ?)',
                           (username, email, password))  # Store password as
plain text
            conn.commit()
            self.show_dialog("Sign Up", "Sign up successful")
        except sqlite3.IntegrityError:
            self.show_dialog("Error", "Username already exists.")
        finally:
            conn.close()

    def show_dialog(self, title, message):
        # Display dialog messages
        if not self.dialog:
            self.dialog = MDDialog(
                title=title,
                text=message,
                buttons=[
                    MDFlatButton(
                        text="CLOSE",
                        on_release=lambda x: self.dialog.dismiss()
                    ),
                ],
            )
        self.dialog.title = title
```

```
        self.dialog.text = message
        self.dialog.open()

if __name__ == "__main__":
    LoginApp().run()
```

For the submission, students are required to submit a comprehensive report that includes the following:

1. **Report (PDF Format):**
    - Include the Python code for each task, clearly separated by the respective part (UI Design, Database Connectivity, User Authentication, etc.).
    - Add relevant screenshots demonstrating the working of each functionality, such as login, sign-up, database integration, and error handling.
2. **Python Script (.py files):**
    - Submit all Python .py files associated with the project, ensuring they are well-organized and functional.

Both the **PDF report** and **Python scripts** should be submitted on **Canvas** before the due time.

Best of luck