# Laboratory Exercise 3: Flet User Login Application

## I. Objective

To create a simple user login application using the Flet framework. This exercise will cover creating a user interface, handling user input, and connecting to a MySQL database for user authentication.

## II. Prerequisites

Before you begin, ensure you have the following installed:

- Python 3.x
- Flet library
- MySQL Connector for Python
- A running MySQL server instance.

## III. Instructions

### Part 1: Project Setup

1. **Create a project directory and a new Flet project.** Open your terminal or command prompt and run the following commands:

   ```
   mkdir week3_labs
   cd week3_labs
   flet create --project-name userlogin
   ```

   This will create a new directory named `week3_labs`, navigate into it, and then create a new directory named `userlogin` with the basic Flet project structure inside `week3_labs`.

2. **Install required packages.** We need `mysql-connector-python` to interact with our MySQL database. Install it using pip:

   ```
   pip install mysql-connector-python
   ```

### Part 2: Database Setup

1. **Create a database.** Access your MySQL server and create a new database named `fletapp`.

   ```sql
   CREATE DATABASE fletapp;
   ```

2. **Create a `users` table.** Use the `fletapp` database and create a table to store user credentials.

```sql
USE fletapp;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

3. **Insert sample data.** Let's add a sample user to our table for testing purposes.

```sql
INSERT INTO users (username, password) VALUES ('testuser', 'password123');
```

### Part 3: Database Connection

Inside your project, navigate to the `src` directory. Create a new file named `db_connection.py`. This file will handle the connection to your MySQL database.

**`src/db_connection.py`** Define a function `connect_db` that returns a `mysql.connector.connect` object. The connection should use the following parameters:

- `host`: "localhost"
- `user`: "root"
- `password`: "admin123" (IMPORTANT: Replace with your MySQL root password)
- `database`: "fletapp"

### Part 4: Building the User Interface (`main.py`)

Now, let's modify the `src/main.py` file to create our login interface. Replace the existing content of `src/main.py` with the following, step-by-step.

1. **Import necessary libraries.** Import `flet` as `ft`, `mysql.connector`, and the `connect_db` function from `db_connection`.

2. **Define the `main` function and configure the page.** Define a `main` function that takes `page: ft.Page` as an argument. Inside this function, configure the page with the following properties:

   - The window should load in the center.
   - The window should be frameless (no title bar, minimize/maximize buttons).
   - The title of the window should be "User Login".
   - Content within the page should be vertically centered.
   - Content within the page should be horizontally centered.
   - The window height should be 350 pixels.
   - The window width should be 400 pixels.
   - The background color of the page should be `ft.Colors.AMBER_ACCENT`

3. **Create the UI controls.** Add a title, two text fields for username and password, and a login button.

- **Login Title:** Create a text control that displays "User Login" centered, with a size of 20, bold weight, and Arial font family.
- **Username Input Field:** Create a text field with the label "User name", hint text "Enter your user name", and helper text "This is your unique identifier". It should have a width of 300, autofocus enabled, be initially enabled, display a person icon, and have a `LIGHT_BLUE_ACCENT` background color.
- **Password Input Field:** Create a text field with the label "Password", hint text "Enter your password", and helper text "This is your secret key". It should have a width of 300, be initially enabled, obscure text (password mode), allow revealing the password, display a password icon, and have a `LIGHT_BLUE_ACCENT` background color.

## Part 5: Implementing the Login Logic

1. **Create the `login_click` function.** Define an asynchronous function `login_click` that takes `e` as an argument. This function will contain the logic for validating input and authenticating against the database.
2. **Create Dialogs for Feedback.** Define the following alert dialog instances inside the `login_click` function:
   - **Success Dialog (`success_dialog`):** This dialog should have a title "Login Successful", content displaying "Welcome, [username]!" centered, an "OK" button to close it, and a green check circle icon.
   - **Failure Dialog (`failure_dialog`):** This dialog should have a title "Login Failed", content displaying "Invalid username or password" centered, an "OK" button to close it, and a red error icon.
   - **Invalid Input Dialog (`invalid_input_dialog`):** This dialog should have a title "Input Error", content displaying "Please enter username and password" centered, an "OK" button to close it, and a blue info icon.
   - **Database Error Dialog (`database_error_dialog`):** This dialog should have a title "Database Error", content displaying "An error occurred while connecting to the database", and an "OK" button to close it.
3. **Add Validation and Database Logic.** Inside the `login_click` function, after defining the dialogs, implement the following logic:
   - Check if `username` or `password` are empty. If so, open `invalid_input_dialog` and return.
   - Use a `try-except` block to handle `mysql.connector.Error`.
   - Inside the `try` block:
     - Establish a database connection using `connect_db()`.
     - Create a cursor.
     - Execute a parameterized SQL query to select a user where `username` and `password` match the input values. **Crucially, emphasize the use of parameterized queries to prevent SQL injection.**
     - Fetch the result.

        - Close the database connection.
        - If a result is found, open `success_dialog`; otherwise, open `failure_dialog`.
        - Call `page.update()`.
    - Inside the `except` block, open `database_error_dialog` and call `page.update()`.

4. **Create the Login Button.** Create an elevated button with the text "Login", an `on_click` handler set to the `login_click` function, a width of 100, and a login icon.

### Part 6: Arranging Controls and Running the App

1. **Add all controls to the page.** Add the login title, a container holding a column with the username and password fields (with 20 pixels spacing), and another container holding the login button (aligned to the top right with a margin of 0, 20, 40, 0) to the page.

2. **Start the Flet app.** Add `ft.app(target=main)` at the very end of the file, outside the `main` function.

3. **Run the application.** Open your terminal in the project's root directory (the `userlogin` directory) and run:

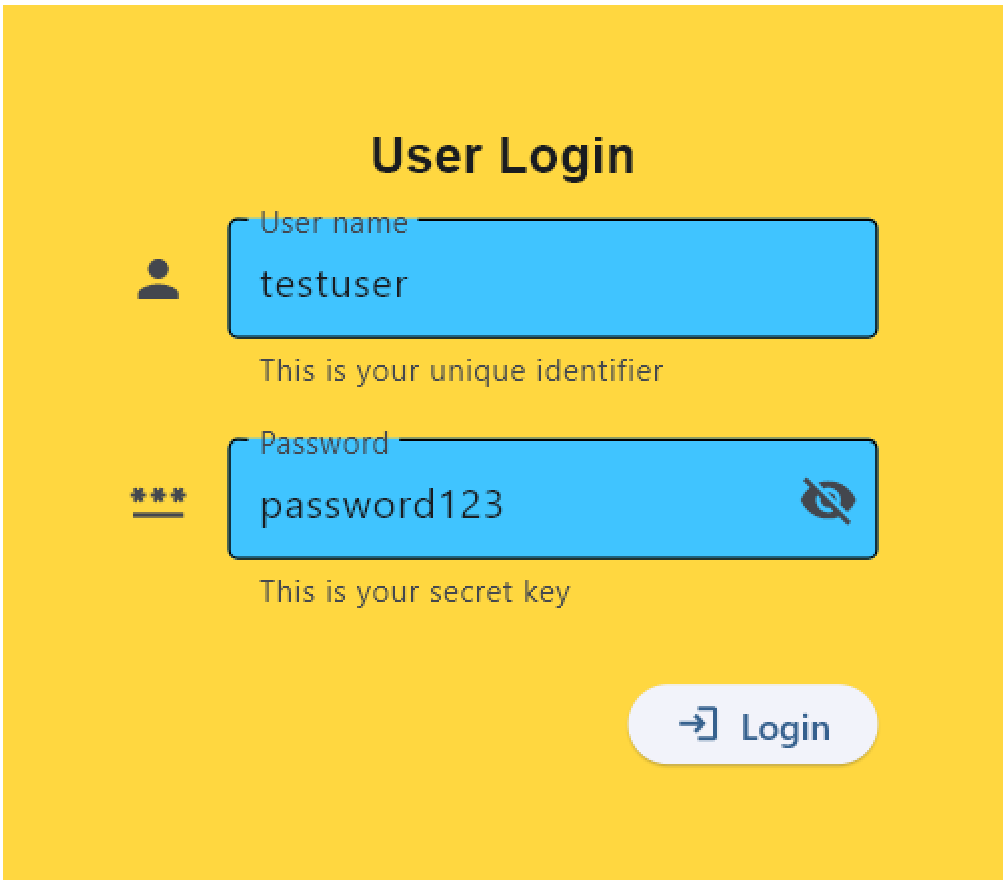
   ```
   flet run
   ```

## IV. Expected Output

A small, frameless window should appear in the center of your screen with an amber background. It will have a "User Login" title, fields for username and password, and a "Login" button.

- **Successful Login:** If you enter the correct credentials (`testuser`, `password123`), a "Login Successful" dialog will appear.
- **Failed Login:** If you enter incorrect credentials, a "Login Failed" dialog will appear.
- **Empty Fields:** If you click login without entering a username or password, an "Input Error" dialog will appear.
- **Database Error:** If the application cannot connect to the database, a "Database Error" dialog will appear.

**User Login Entry**



User Login

User name

testuser

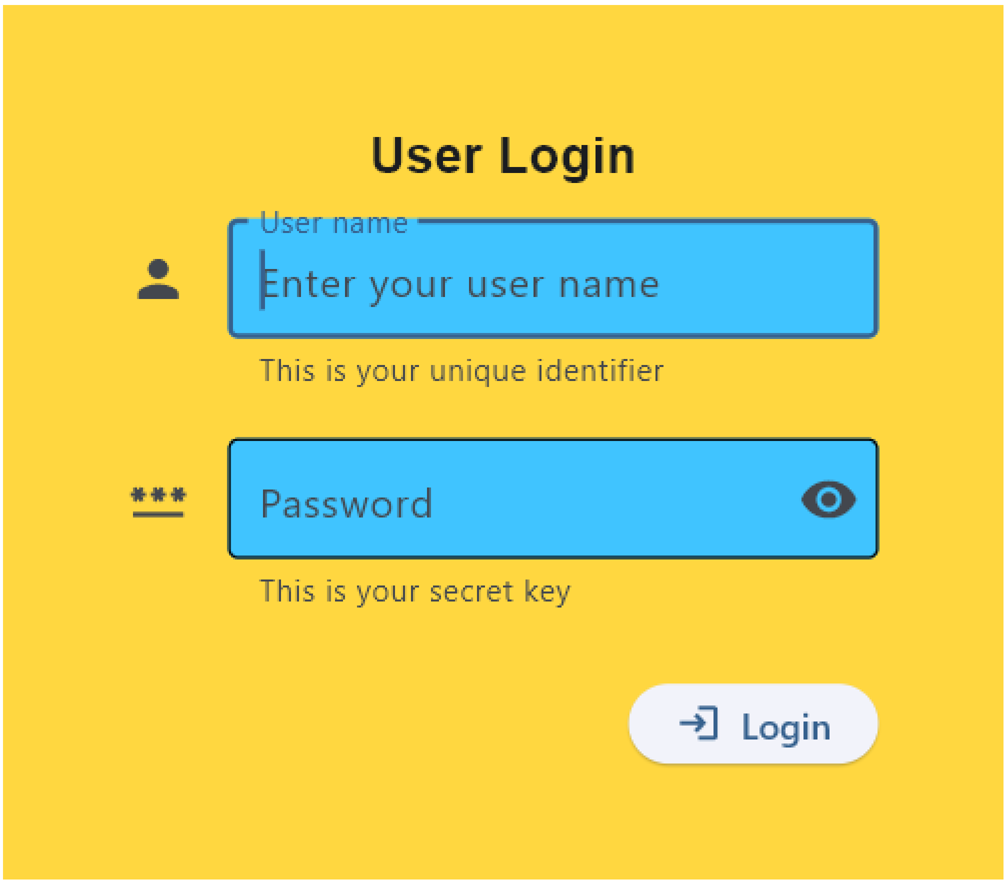
This is your unique identifier

Password

password123

This is your secret key

**User Login Blank**



User Login

User name

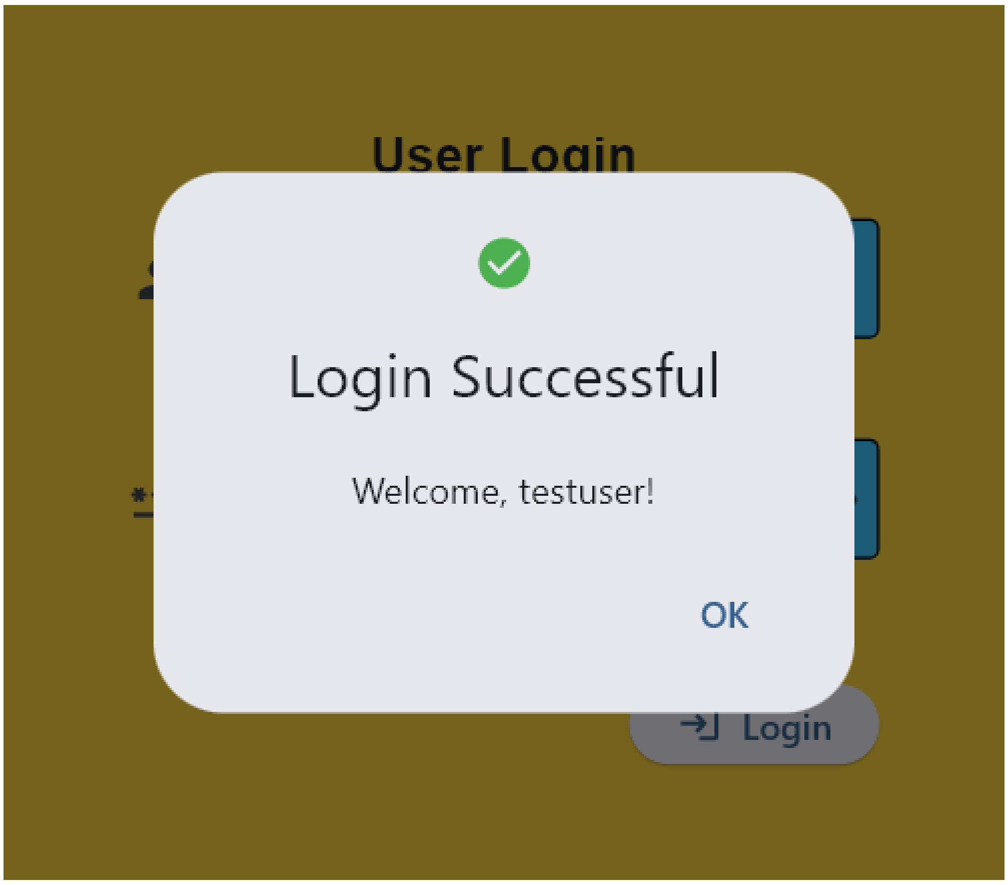
Enter your user name

This is your unique identifier

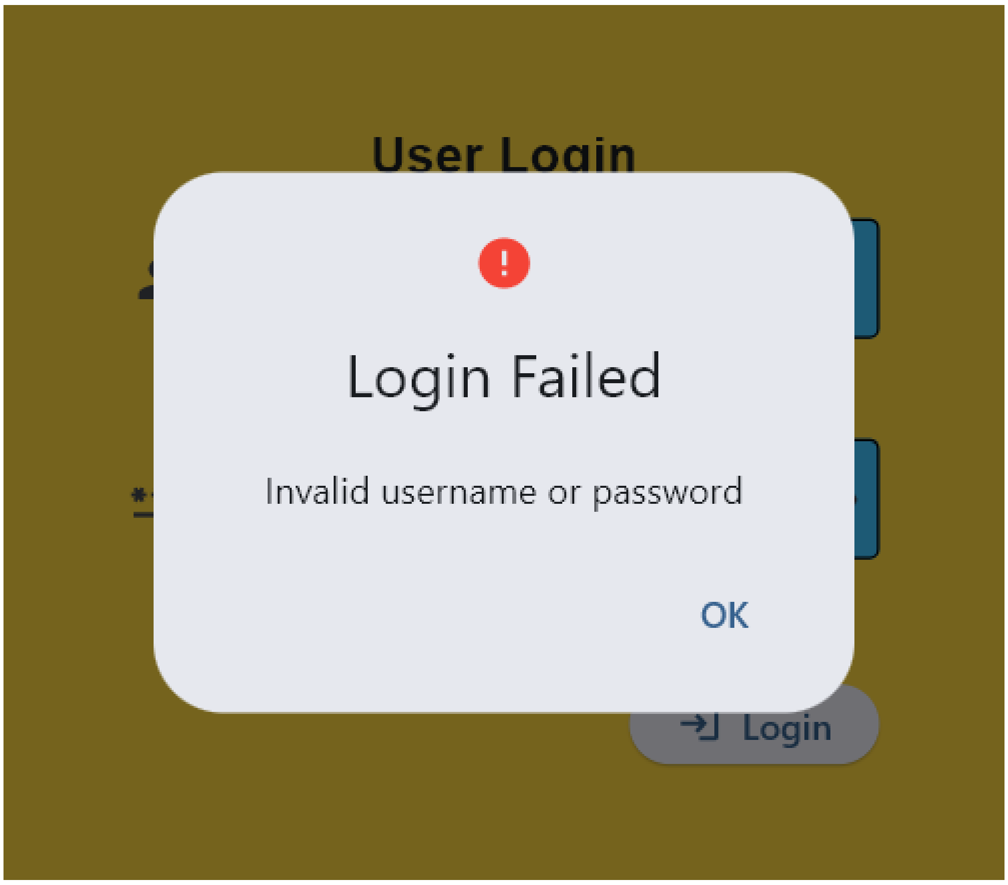
Password

This is your secret key

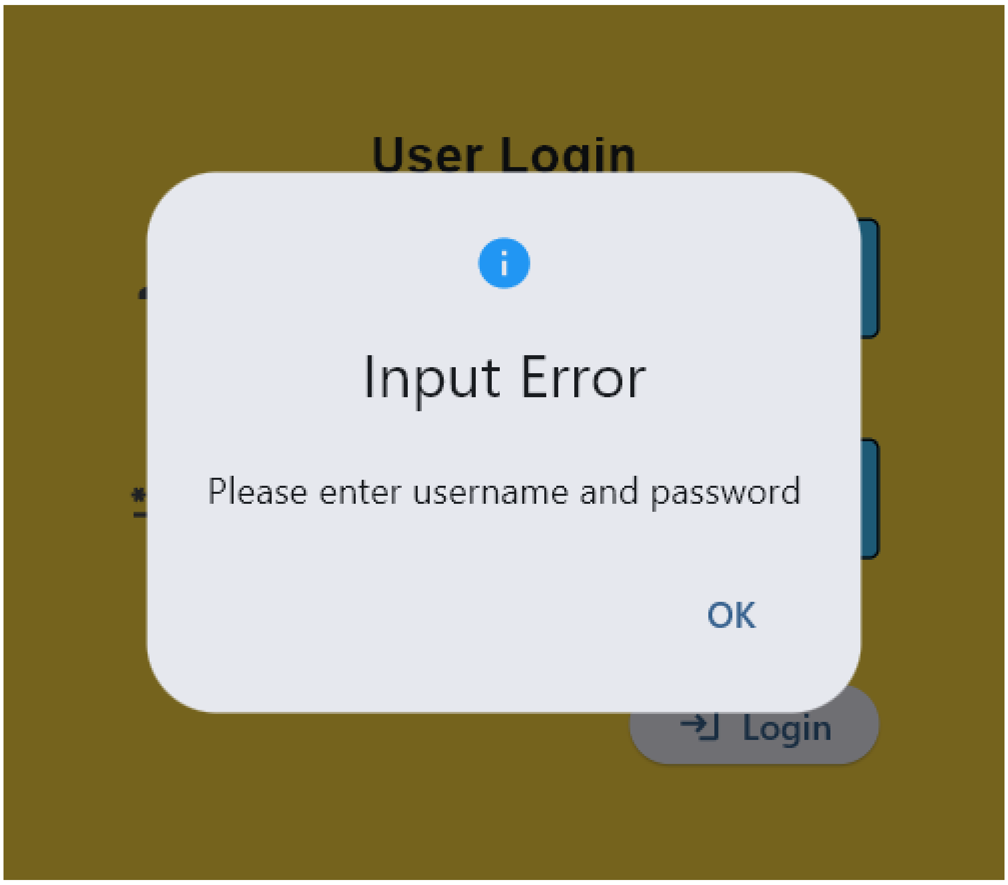
**Login Successful**



**Login Failed**



**Input Error**

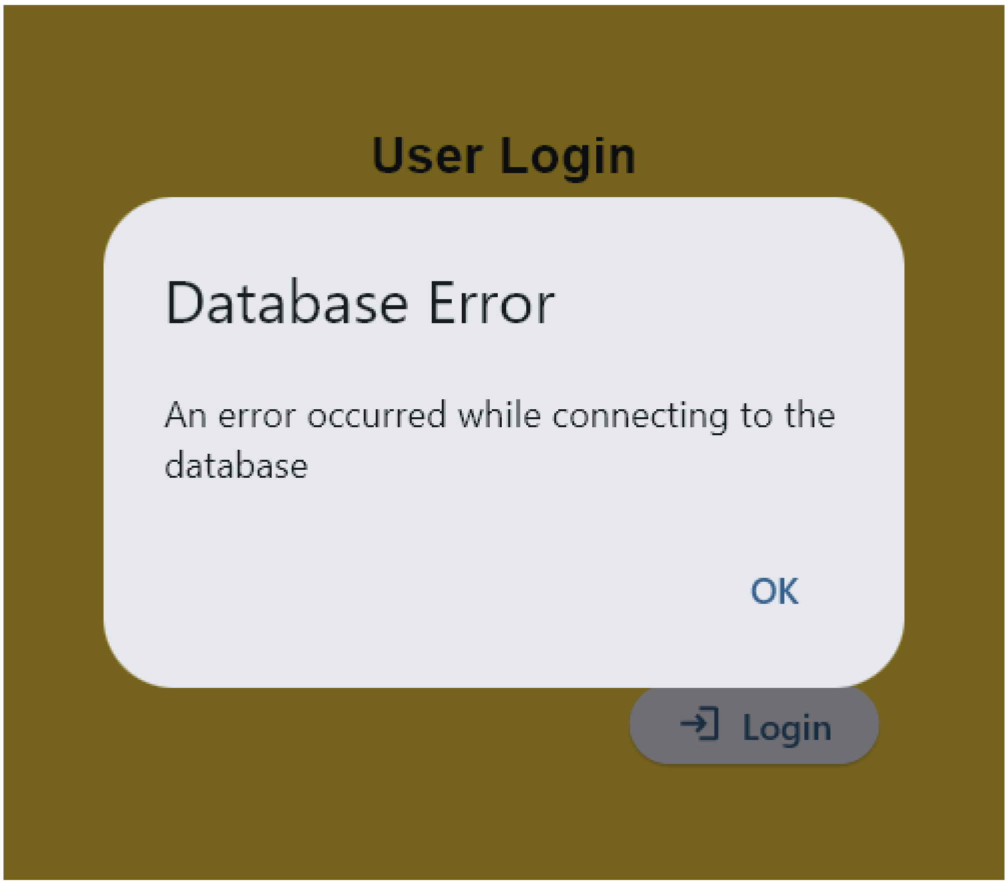

User Login

Input Error

Please enter username and password

OK

**Database Error**


User Login

Database Error

An error occurred while connecting to the database

OK