



**CHANDIGARH**  
**UNIVERSITY**

Discover. Learn. Empower.

## **Mini Project Report**

on

### **Secure Login Page**

#### **SUBMITTED BY:-**

Avinesh Kumar

UID:- 25MCI10375

Branch:- MCA[AI&ML]

Section:- 25MAM6-A

#### **SUBMITTED TO:-**

Ms. Sweta

Assistant Professor

## Acknowledgement

I am deeply grateful to all those who have contributed to the successful completion of my mini project entitled

### **“Secure Login Page”**

I express my sincere thanks to **Ms. Sweta**, my project guide, for her valuable guidance, encouragement, and constant support throughout the development of this project. Her insightful suggestions and motivation helped me to understand and implement the concepts effectively.

My heartfelt gratitude goes to **University Institute of Computing** and **Chandigarh University** for including this project as part of the curriculum, giving me an opportunity to apply theoretical knowledge to a practical system.

Lastly, I am thankful to my friends and family members for their constant encouragement and help during this project.

### **Signature of Student**

**Name:** Avinesh Kumar  
**Course:** MCA[AI&ML]  
**UID:-** 25MCI10375

### **Signature of Guide**

*Ms. Sweta  
Professor*

## **Table of Content:-**

Acknowledgement.....	2
Contents.....	3
Introduction .....	4
Background .....	4
Objectives.....	4-5
Technologies Used.....	5
Requirements.....	5-6
System Design.....	6-7
Source Code.....	7-13
Result.....	14-15
Functionalities.....	16
Future Problem.....	16-17
Learning Outcome.....	17-18
GitHub Screenshot & Link.....	18
Bibliography.....	19

## **Introduction**

The Secure Login Page using Python and MySQL is a desktop-based application that provides a user authentication system with database connectivity. It is designed to securely manage user credentials, allowing users to sign up, log in, and perform basic account management tasks such as password updates and user deletions.

The project integrates Tkinter (Python's standard GUI library) for building an interactive and user-friendly interface, and MySQL as the backend database for storing user credentials securely. Upon successful login, the system provides access to a dashboard that displays all user records and allows administrative operations such as adding new users, deleting users, and changing passwords.

This project demonstrates the essential concept of database-driven authentication and ensures data integrity and security through controlled access and unique user management. It is an ideal example of how front-end and back-end technologies can work together in a simple yet effective system for secure login management.

## **Background**

In today's digital era, securing user data and verifying identity have become essential in all software systems. Most applications, from banking to social media, rely on secure login mechanisms to protect user information.

The Secure Login Page using Python and MySQL project is based on this concept of authentication and data protection. It uses Python (Tkinter) for the graphical user interface and MySQL for backend database management. This integration allows safe storage, retrieval, and management of user credentials.

The project demonstrates how a simple and efficient login system can be built to perform user registration, login validation, password updates, and account management. It highlights the importance of database connectivity, data security, and user authentication in modern applications.

## **Objectives**

The main objectives of the Secure Login Page using Python and MySQL project are:

1. To develop a secure authentication system that allows users to log in and sign up safely using unique credentials.
2. To integrate Python with MySQL for efficient storage, retrieval, and management of user data.

3. To design a user-friendly graphical interface using Tkinter for easy interaction and smooth operation.
4. To implement basic account management features such as adding, deleting, and updating user information.
5. To ensure data security and prevent unauthorized access by validating user credentials before granting access.
6. To demonstrate the practical use of database connectivity in real-world applications using Python programming.

## **Technologies Used**

The Secure Login Page using Python and MySQL project utilizes several technologies and tools that work together to create a secure, interactive, and efficient login system. The major technologies used are:

1. Python – The core programming language used for developing the application's logic and functionality. Its simplicity and readability make it ideal for GUI and database applications.
2. Tkinter – A built-in Python library used for creating the Graphical User Interface (GUI). It provides widgets like buttons, labels, and entry fields that make the interface interactive and user-friendly.
3. MySQL – A Relational Database Management System (RDBMS) used to store and manage user data such as usernames and passwords. It ensures data security, integrity, and easy access through structured queries.
4. mysql.connector – A Python module that allows database connectivity between Python and MySQL. It enables operations such as inserting, updating, deleting, and retrieving data.
5. Tkinter Messagebox & Treeview Widgets – Used for displaying messages (e.g., success, error, or warnings) and for showing tabular data like user records in an organized format.
6. Operating System (Windows/Linux) – The platform on which the application runs. The program is compatible with most operating systems that support Python and MySQL.
7. Visual Studio Code / IDLE – The development environment used to write, test, and debug Python code efficiently.

## **Requirements**

The Secure Login Page using Python and MySQL project requires both hardware and software components to ensure proper functioning and smooth execution.

### **1. Hardware Requirements**

- Processor: Intel Core i3 or higher

- RAM: Minimum 4 GB (8 GB recommended)

- Storage: At least 200 MB of free disk space
- Display: 1024x768 resolution or higher

- Input Devices: Keyboard and Mouse

## **2. Software Requirements**

- Operating System: Windows 10 / 11, Linux, or macOS
- Programming Language: Python 3.8 or above
- Database: MySQL Server (8.0 or higher recommended)
- Python Libraries:
  - mysql.connector (for database connectivity)
  - tkinter (for GUI design)
  - ttk and messagebox (for tables and user notifications)
- Code Editor/IDE: Visual Studio Code, PyCharm, or IDLE
- XAMPP / MySQL Workbench (optional): For managing and visualizing the database

## **3. Functional Requirements**

- User should be able to sign up with a new username and password.
- The system must allow users to log in securely using valid credentials.
- The admin should be able to add, delete, or update user details.
- The system must validate inputs and display appropriate success or error messages.
- Passwords should be modifiable only by the corresponding user.

## **System Design**

The **Secure Login Page using Python and MySQL** is designed using a simple **client-server architecture**. The **frontend** is built with **Python's Tkinter**, which provides the graphical user interface for user login, signup, and dashboard management. The **backend** uses **MySQL** to store and manage user data securely.

When a user logs in or signs up, the system connects to the MySQL database using the **mysql.connector** module. The entered credentials are verified, and based on the result, access is granted or denied. After successful login, a **dashboard** appears where the user or admin can add, delete, or update user details.

The main components include:

- **Login & Signup Module** – Handles authentication.

- **Dashboard Module** – Displays and manages user records.
- **Database Module** – Stores usernames and passwords in a secure table.

This simple design ensures **data security**, **easy access**, and **smooth interaction** between the GUI and database.

## Source Code

```
import mysql.connector
import tkinter as tk
from tkinter import messagebox, ttk

# ----- BACKEND SETUP -----
def init_db():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="3804"
    )
    cur = conn.cursor()
    cur.execute("CREATE DATABASE IF NOT EXISTS LoginDemo")
    cur.execute("USE LoginDemo")
    cur.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INT AUTO_INCREMENT PRIMARY KEY,
            username VARCHAR(100) UNIQUE,
            password VARCHAR(100)
        )
    """)
    cur.execute("INSERT IGNORE INTO users (username, password) VALUES ('admin', 'admin123')")
```

```

conn.commit()

cur.close()

conn.close()

init_db()

# ----- DASHBOARD AFTER LOGIN -----

def show_dashboard(loggedin_user):
    dash = tk.Toplevel(root)
    dash.title("SQL User Dashboard")
    dash.geometry("475x420")
    dash.configure(bg="#f3f8fa")

    tk.Label(dash, text="User Database", font=("Arial Bold", 17), fg="#138d75",
    bg="#f3f8fa").pack(pady=12)

    tree = ttk.Treeview(dash, columns=("ID", "Username", "Password"), show="headings", height=7)
    tree.heading("ID", text="ID")
    tree.heading("Username", text="Username")
    tree.heading("Password", text="Password")
    for col in ("ID", "Username", "Password"): tree.column(col, width=115)
    tree.pack(padx=18, pady=6)

def refresh_table():
    for row in tree.get_children(): tree.delete(row)

    conn = mysql.connector.connect(host="localhost", user="root", password="10382",
    database="LoginDemo")

    cur = conn.cursor()
    cur.execute("SELECT id, username, password FROM users")
    for row in cur.fetchall():

        tree.insert("", "end", values=row)

    cur.close(); conn.close()

refresh_table()

```

```

# Add User Form

add_frame = tk.Frame(dash, bg="#f3f8fa")

tk.Label(add_frame, text="New Username:", font=("Arial", 10),
bg="#f3f8fa").grid(row=0,column=0,padx=6,pady=4)

enu = tk.Entry(add_frame, width=15)

enu.grid(row=0,column=1)

tk.Label(add_frame, text="New Password:", font=("Arial", 10),
bg="#f3f8fa").grid(row=0,column=2,padx=6)

epw = tk.Entry(add_frame, width=15)

epw.grid(row=0,column=3)

add_frame.pack(pady=3)

def add_user():

    u = enu.get()

    p = epw.get()

    if not u or not p:

        messagebox.showwarning("Input", "Both fields required.", parent=dash)

        return

    conn = mysql.connector.connect(host="localhost", user="root", password="10382",
database="LoginDemo")

    cur = conn.cursor()

    try:

        cur.execute("INSERT INTO users(username,password) VALUES(%s,%s)", (u,p))

        conn.commit()

        messagebox.showinfo("User Added", "User created successfully.", parent=dash)

        refresh_table()

    except:

        messagebox.showerror("Error", "Username already exists!", parent=dash)

        cur.close(); conn.close()

    tk.Button(add_frame, text="Add User", font=("Arial", 10), command=add_user, bg="#aed6f1",
relief="ridge").grid(row=0,column=4,padx=8)

# Delete User form

del_frame = tk.Frame(dash, bg="#f3f8fa")

```

```

tk.Label(del_frame, text="User ID to Delete:", font=("Arial", 10),
bg="#f3f8fa").grid(row=0,column=0,padx=6,pady=5)

eid = tk.Entry(del_frame, width=8)

eid.grid(row=0,column=1)

def delete_user():

    idv = eid.get()

    if not idv:

        messagebox.showwarning("Input", "Enter User ID.", parent=dash)

        return

    conn = mysql.connector.connect(host="localhost", user="root", password="10382",
database="LoginDemo")

    cur = conn.cursor()

    cur.execute("DELETE FROM users WHERE id=%s", (idv,))

    if cur.rowcount>0:

        messagebox.showinfo("Deleted", "User deleted.", parent=dash)

    else:

        messagebox.showerror("Error", "No user with that ID.", parent=dash)

    conn.commit(); cur.close(); conn.close(); refresh_table()

    tk.Button(del_frame, text="Delete User", font=("Arial", 10), command=delete_user, bg="#f7c8c8",
relief="ridge").grid(row=0,column=2,padx=10)

del_frame.pack(pady=3)

tk.Button(dash, text="Refresh Table", font=("Arial", 10), command=refresh_table, bg="#b7e8b8",
relief="ridge").pack(pady=7)

# ----- Change Password feature -----

pchange_frame = tk.Frame(dash, bg="#f3f8fa")

tk.Label(pchange_frame, text=f"Change password for '{loggedin_user}':", font=("Arial", 10),
bg="#f3f8fa").grid(row=0,column=0,padx=4,pady=7)

newpass = tk.Entry(pchange_frame, width=15, show="*")

newpass.grid(row=0,column=1)

def chpw():

    npw = newpass.get()

    if not npw:

```

```
messagebox.showwarning("Input", "Enter new password.", parent=dash)

return

conn = mysql.connector.connect(host="localhost", user="root", password="10382",
database="LoginDemo")

cur = conn.cursor()

cur.execute("UPDATE users SET password=%s WHERE username=%s", (npw,loggedin_user))

if cur.rowcount>0:

    messagebox.showinfo("Password Updated", "Password changed successfully!", parent=dash)

    refresh_table()

else:

    messagebox.showerror("Error", "Could not update password.", parent=dash)

conn.commit(); cur.close(); conn.close()

newpass.delete(0,"end")

tk.Button(pchange_frame, text="Update Password", font=("Arial", 10), command=chpw, bg="#fed97f",
relief="ridge").grid(row=0,column=2,padx=8)

pchange_frame.pack(pady=5)
```

# ----- LOGIN PAGE -----

```
def login():

uname = entry_user.get()

pwd = entry_pass.get()

conn = mysql.connector.connect(host="localhost", user="root", password="10382",
database="LoginDemo")

cur = conn.cursor()

cur.execute("SELECT * FROM users WHERE username=%s AND password=%s", (uname, pwd))

res = cur.fetchone()

cur.close(); conn.close()

if res:

    messagebox.showinfo("Login Success", f"Welcome, {uname}!", parent=root)

    root.configure(bg="#d4f7d4")

    title_label.config(text=f"Hello, {uname}!", fg="#297d43", bg="#d4f7d4")

    show_dashboard(uname)

else:
```

```

messagebox.showerror("Login Failed", "Invalid username or password.", parent=root)

root.configure(bg="#f7d4d4")

title_label.config(text="Login Failed", fg="#a52a2a", bg="#f7d4d4")


def signup():

    uname = entry_user.get()

    pwd = entry_pass.get()

    if not uname or not pwd:

        messagebox.showwarning("Input", "Enter username and password.", parent=root)

        return

    conn = mysql.connector.connect(host="localhost", user="root", password="10382",
database="LoginDemo")

    cur = conn.cursor()

    try:

        cur.execute("INSERT INTO users (username, password) VALUES (%s, %s)", (uname, pwd))

        conn.commit()

        messagebox.showinfo("Signup Success", "Signup successful! You can now login.", parent=root)

        root.configure(bg="#f7f4d4")

        title_label.config(text="Signup Successful", fg="#297d43", bg="#f7f4d4")

    except:

        messagebox.showerror("Signup Failed", "Username already taken!", parent=root)

        root.configure(bg="#f7d4d4")

        title_label.config(text="Signup Failed", fg="#a52a2a", bg="#f7d4d4")

    cur.close(); conn.close()


root = tk.Tk()

root.title("★ LOGIN PAGE | PYTHON + SQL PROJECT ★")
root.geometry("420x380")
root.configure(bg="#e7f0fd")


title_label = tk.Label(root, text="Welcome to Secure Login Portal", font=("Arial Bold", 18), bg="#e7f0fd",
fg="#297d43")

title_label.pack(pady=18)

```

```
frame = tk.Frame(root, bd=3, relief="groove", bg="#fafeff")
frame.place(relx=0.5, rely=0.52, anchor="center")

tk.Label(frame, text="Username:", font=("Arial", 12), bg="#fafeff").grid(row=0, column=0, padx=14,
pady=10)
entry_user = tk.Entry(frame, font=("Arial", 12), width=20)
entry_user.grid(row=0, column=1, padx=8, pady=10)

tk.Label(frame, text="Password:", font=("Arial", 12), bg="#fafeff").grid(row=1, column=0, padx=14,
pady=10)
entry_pass = tk.Entry(frame, show='*', font=("Arial", 12), width=20)
entry_pass.grid(row=1, column=1, padx=8, pady=10)

btn_login = tk.Button(frame, text="Login", font=("Arial", 12), bg="#99cff", fg="white",
width=17, command=login)
btn_login.grid(row=2, column=0, pady=18, padx=8)

btn_signup = tk.Button(frame, text="Signup", font=("Arial", 12), bg="#91e495", fg="white",
width=17, command=signup)
btn_signup.grid(row=2, column=1, pady=18, padx=8)

hint_label = tk.Label(root, text='Demo Login: admin / admin123', font=("Arial", 11), bg="#e7f0fd",
fg="#2b2b2b")
hint_label.pack(pady=7)

root.mainloop()
```

# Result

## 1. Login Page Interface:-

This is the **main entry screen** of the application titled “*Welcome to Secure Login Portal*.”

- It allows users to enter their **Username** and **Password**.
- There are two main buttons —
  - **Login:** To authenticate existing users.
  - **Signup:** To register new users into the database.
- A **demo credential** (admin / admin123) is shown for testing purposes.

This interface is designed using **Tkinter** in Python with a **clean, light-blue theme**.



## 2. Login Success Message:-

Once the user enters valid credentials:

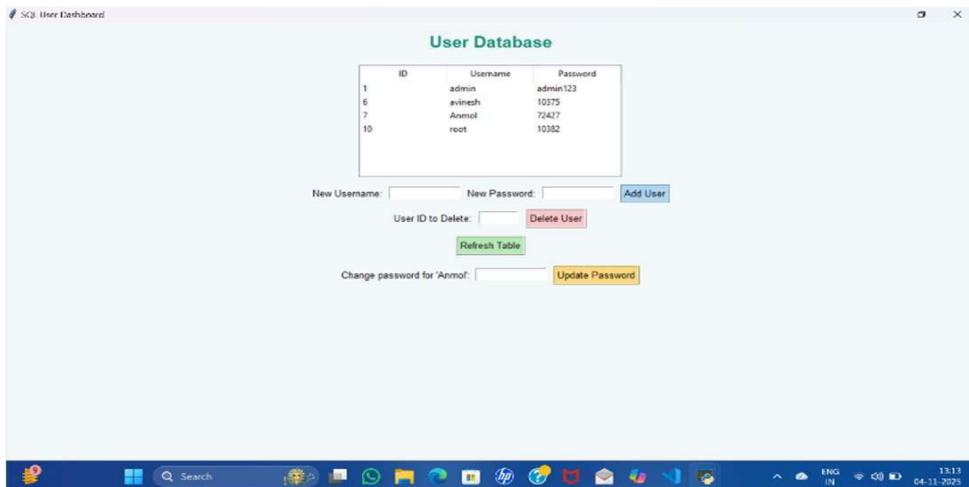
- A **popup message box** appears displaying “**Login Success**” and a personalized greeting — for example, “*Welcome, Avinesh Kumar!*”
- This confirms that the authentication process has been successfully verified from the SQL database.



### 3. User Dashboard Interface:-

After successful login, the user is redirected to the **User Dashboard**, which displays the **User Database Table**. Features include:

- View all registered users (ID, Username, and Password).
- Add a new user with the *Add User* button.
- Delete existing users by entering their ID.
- Update the password for a selected username.
- Refresh the table to view recent changes.



# Functionalities

The **Secure Login Page** system provides several key functions to ensure safe and efficient user authentication and account management.

## **1. User Registration (Signup)**

- Allows new users to create an account by entering a username and password.
- Checks for duplicate usernames to prevent repetition.

## **2. User Login**

- Authenticates users by verifying credentials stored in the MySQL database.
- Displays a success message upon valid login or an error message for invalid details.

## **3. Admin Dashboard**

- Provides an interface for the admin to manage all user records.
- Admin can **add new users, delete users, or update passwords**.

## **4. Password Update**

- Users can securely change their password through the dashboard.
- Ensures updated data is reflected instantly in the database.

## **5. Data Storage and Validation**

- All user data is stored in the **MySQL database**.
- Includes input validation to prevent empty or incorrect entries.

## **6. User-Friendly GUI**

- Developed using **Tkinter** for a simple and interactive interface.
- Provides easy navigation between login, signup, and dashboard windows.

# Future Work

The current version of the **Secure Login Page** provides basic authentication and user management. However, it can be enhanced further in the future through the following improvements:

### **1. Password Encryption:**

- Implement password hashing using libraries like bcrypt or hashlib for better security.

### **2. Email or OTP Verification:**

- Add multi-factor authentication (MFA) for verifying user identity via email or mobile OTP.

### **3. Forgot Password Feature:**

- Allow users to reset their password securely through a verification link or code.

### **4. Role-Based Access:**

- Introduce user roles such as *Admin*, *User*, and *Guest* with different permissions.

### **5. Activity Logging:**

- Maintain a log of user login attempts and database changes for security monitoring.

### **6. Web or Cloud Integration:**

- Convert the system into a web-based or cloud-hosted platform for remote access.

### **7. Improved UI Design:**

- Enhance the graphical interface using frameworks like **CustomTkinter** or **PyQt** for a modern look.

## **Learning Outcome**

Through the development of this project, several important technical and conceptual skills were gained:

### **1. Integration of Python with MySQL:**

- Learned how to connect Python programs with MySQL databases for data storage and retrieval.

### **2. Database Management:**

- Gained practical experience in creating, updating, and managing user data securely in a relational database.

### **3. GUI Development:**

- Understood how to design user-friendly interfaces using **Tkinter** in Python.

### **4. Authentication Process:**

- Learned the core logic behind secure login systems, including input validation and access control.

### **5. Problem-Solving Skills:**

- Developed logical thinking and debugging skills while handling login errors and database connections.

## 6. Project Implementation:

- Learned how to integrate multiple components (frontend, backend, and database) into a single working application.

## 7. Teamwork and Documentation:

- Improved collaboration, documentation, and presentation skills essential for real-world software projects.

## GitHub Screenshot

The screenshot shows a GitHub repository page for 'sql-project/new3 at main · ar2882421-svg/sql-project'. The main.py file is displayed in the code editor. The code sets up a MySQL connection and creates a 'LoginDemo' database, then creates a 'users' table with columns id, username, and password.

```
1 import mysql.connector
2 import tkinter as tk
3 from tkinter import messagebox, ttk
4
5 # ----- BACKEND SETUP -----
6 def init_db():
7     conn = mysql.connector.connect(
8         host="localhost",
9         user="root",
10        password="10382"
11    )
12    cur = conn.cursor()
13    cur.execute("CREATE DATABASE IF NOT EXISTS LoginDemo")
14    cur.execute("USE LoginDemo")
15    cur.execute("""
16        CREATE TABLE IF NOT EXISTS users (
17            id INT AUTO_INCREMENT PRIMARY KEY,
18            username VARCHAR(100) UNIQUE,
19            password VARCHAR(100)
20        )
21    """)
```

Link:- <https://github.com/ar2882421-svg/sql-project>

## **Bibliography**

1. Python Documentation – <https://docs.python.org/3/>
2. MySQL Official Documentation – <https://dev.mysql.com/doc/>
3. Tkinter GUI Reference – <https://docs.python.org/3/library/tkinter.html>
4. Reema Thareja, “*Programming in Python*,” Oxford University Press, 2021.
5. Korth, Henry F., and Silberschatz, Abraham, “*Database System Concepts*,” McGraw-Hill Education, 7th Edition, 2019.
6. TutorialsPoint, *Python MySQL Tutorial* – [https://www.tutorialspoint.com/python\\_mysql/](https://www.tutorialspoint.com/python_mysql/)
7. GeeksforGeeks, *Python and MySQL Connectivity Guide* – <https://www.geeksforgeeks.org/>