EDGE CSE CUET DIGITAL SKILLS TRAINING

Project Report on

Student Voting System

Batch Name: CBF-024 Basic Python Programming

This Project report (CBF-024 Basic Python Programming) is submitted to the Department of CSE, Chittagong University of Engineering and Technology (CUET) to fulfill the partial requirement of the Degree of Basic Python Programming Course.

Submitted by:
Name: Sazzad Hossain

Batch: CBF-024 Basic Python Programming

Supervised by:
Shaikat Sharma
Trainer, EDGE-CSE-CUET

DECLARATION

This project report is submitted to the department of Computer Science & Engineering, Chittagong University of Engineering and Technology (CUET) in partial fulfillment of the requirements for the degree of Basic Python Programming. So, we hereby declare that this report is based on the surveys found by us and our original work, which has not been submitted anywhere for any award. Materials of work found by other researchers are mentioned with due reference. All the contents provided here are totally based on our own effort dedicated to the completion of the project. The work is done under the guidance of Mr. Shaikat Sharma, Trainer at EDGE-CSE-CUET program.

Sazzad Hossain
CBF-024 Basic Python Programming

_____
Name
Batch

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project Supervisor, Mr. Shaikat Sharma, for his valuable input, guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project. His useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

We deeply express our sincere thanks to him for encouraging and allowing us to present the project on the topic " **Student Voting System**" at our department premises for the partial fulfillment of the requirements. We take this opportunity to thank all our trainers who have directly or indirectly helped with our project.

We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Finally, we express our thanks to our friends for their cooperation and support.

Sazzad Hossain
CBF-024 Basic Python Programming
Name
Batch

Abstract

This project involves the design and implementation of a student voting system using Python's Tkinter library for the graphical user interface (GUI) and MySQL as the backend database. The system allows users to register students, add candidates, cast votes, and display voting results dynamically. The primary goal is to demonstrate the practical application of Python with Tkinter for building user interfaces integrated with a relational database.

Introduction

Voting systems play a crucial role in organizing fair and transparent elections, especially within educational institutions where student elections are common. The goal of this project is to build a student voting system that mimics real-world voting principles, ensuring secure, fair, and efficient participation. By integrating Python for the frontend interface and MySQL for the backend, this system provides a comprehensive learning experience in building interactive applications with a database.

The voting system offers the following key features:

- Student Registration: Students are registered with unique IDs.
- Candidate Management: Candidates can be added for elections, preventing duplication.
- Secure Voting Process: Ensures one vote per student with real-time checks.
- Live Result Display: Displays candidate-wise vote counts and total votes.

The development of this system focuses on data integrity, usability, and performance, providing a practical framework for understanding database-driven applications.

Methodology

3.1 Tools and Technologies :

- **Programming Language**: Python
- **Graphical Interface**: Tkinter
- **Database Management**: MySQL
- **Database Connector**: PyMySQL

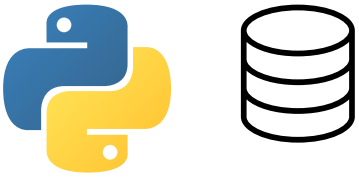

figure: 3.1

3.2 System Architecture :

The architecture of the Student Voting System follows a **three-tiered** structure:
Presentation Layer (Frontend): Built using Tkinter in Python, this layer provides the graphical user interface for students to interact with. It includes windows for registration, **candidate** addition, **voting** and **results** display. Each function (like voting or registration) has a separate Tkinter interface for user interaction.
Application Layer (Logic): This layer contains the core functionality of the system. It includes Python functions to connect to the database, process user actions (such as adding **candidates**, **voting** or **viewing results**), and handle errors. The logic layer ensures data validation and business rules are enforced (e.g., students cannot vote more than once).

Data Layer (Database): Managed with MySQL, this layer stores all data in structured tables: The users table contains student information, including their unique ID and voting status. The candidates table records each candidate's name and vote count.

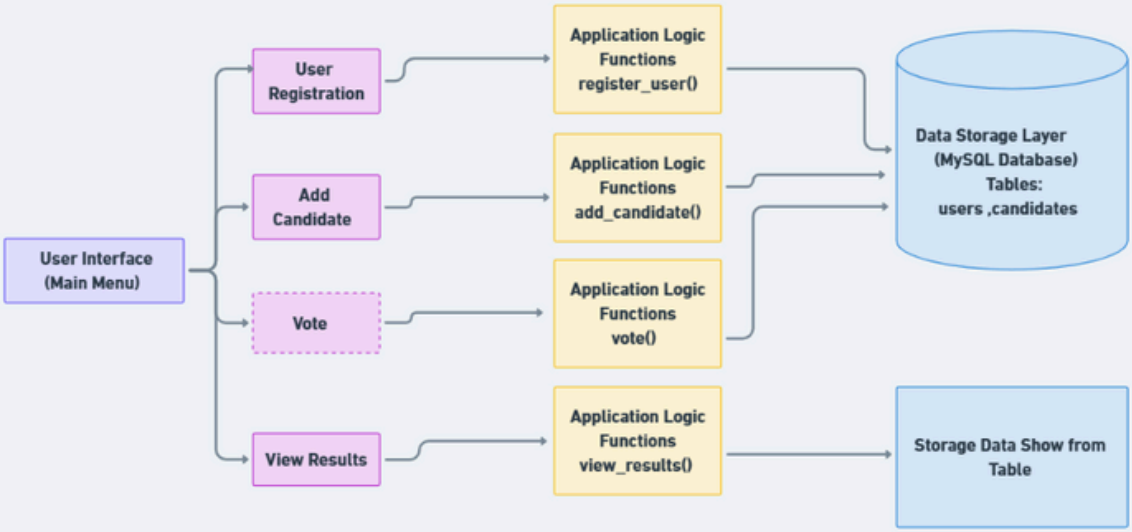

figure: 3.2

3.3 Workflow and Design :

1. Database Setup :

The database vote_db contains two tables:

- **users**: Stores student_id, student_name, and has_voted (to track whether a student has voted).
- **candidates**: Stores name and vote_count (to record each candidate's votes).

Code Section :

```
CREATE DATABASE vote_db;

CREATE TABLE users (
   Id int PRIMARY KEY,
   student_id VARCHAR(20) ,
   student_name VARCHAR(100) NOT NULL,
   has_voted BOOLEAN DEFAULT FALSE );

CREATE TABLE candidates (
   Id int PRIMARY KEY,
   name VARCHAR(100) ,
   vote_count INT DEFAULT 0 );
```

2. Database Connection:

The function connect_db establishes a connection to the MySQL database, handling exceptions for any connection errors.

Code Section :

```
def connect_db():
    try:
      return pymysql.connect(host=DB_HOST, user=DB_USER,password=DB_PASSWORD,
                             database=DB_NAME)
    except pymysql.MySQLError as e:
        messagebox.showerror("Database Error", f"Error connecting to database: {e}")
        return None
```

3. User Registration :

- A student provides their ID and name.
- The system checks if the ID already exists to prevent duplication.
- If the ID is unique, the user is added to the database.

This function inserts a new user if the student_id does not already exist.

Code Section :

```
def register_user(student_id, student_name):
    conn = connect_db()
    if conn is None:
        return
    try:
        with conn.cursor() as cursor:
            cursor.execute("SELECT * FROM users WHERE student_id=%s", (student_id,))
            if cursor.fetchone():
                messagebox.showerror("Error", "Student ID already exists.")
                return
            cursor.execute("INSERT INTO users (student_id, student_name, has_voted) V
                           VALUES (%s, %s, %s)",
                           (student_id, student_name, False))
            conn.commit()
            messagebox.showinfo("Success", f"User with ID '{student_id}' registered
successfully.")
    finally:
        conn.close()
```

4. Add Candidate:

- Candidates can be added using the GUI.
- The system checks for duplicate candidate names.

Adds a candidate to the candidates table if they do not already exist.

Code Section :

```
def add_candidate(name):
    conn = connect_db()
    if conn is None:
        return
    try:
        with conn.cursor() as cursor:
            cursor.execute("SELECT * FROM candidates WHERE name=%s", (name,))
            if cursor.fetchone():
                messagebox.showerror("Error", "Candidate already exists.")
                return
            cursor.execute("INSERT INTO candidates (name, vote_count)
                                VALUES (%s, %s)", (name, 0))
            conn.commit()
            messagebox.showinfo("Success", f"Candidate '{name}' added successfully.")
    finally:
        conn.close()
```

5. Vote:

- A student logs in with their ID and selects a candidate to vote for.
- The system ensures the student can vote only once.

Checks if the user has already voted, updates the vote_count for the chosen candidate, and marks has_voted as True for the user.

Code Section :

```
def vote(student_id, candidate_name):
    conn = connect_db()
    if conn is None:
        return
    try:
        with conn.cursor() as cursor:
            cursor.execute("SELECT has_voted FROM users WHERE student_id=%s",
                           (student_id,))
            user = cursor.fetchone()
            if user is None or user[0]:
                messagebox.showerror("Error", "You have already voted!")
                return
            cursor.execute("UPDATE candidates SET vote_count = vote_count + 1
                           WHERE name=%s", (candidate_name,))
            cursor.execute("UPDATE users SET has_voted = %s WHERE student_id=%s",
                           (True, student_id))
            conn.commit()
            messagebox.showinfo("Success", f"Vote cast for '{candidate_name}'!")
    finally:
        conn.close()
```

5. View Results :

- A tree view displays the candidates along with their respective vote counts.

Fetches and displays the vote counts of each candidate.

Code Section :

```
def view_results():
    conn = connect_db()
    if conn is None:
        return
    try:
        with conn.cursor() as cursor:
            cursor.execute("SELECT name, vote_count FROM candidates")
            results = cursor.fetchall()
            for candidate in results:
                print(f"{candidate[0]}: {candidate[1]} votes")
    finally:
        conn.close()
```

3.4 Student Voting System Process Flowchart :

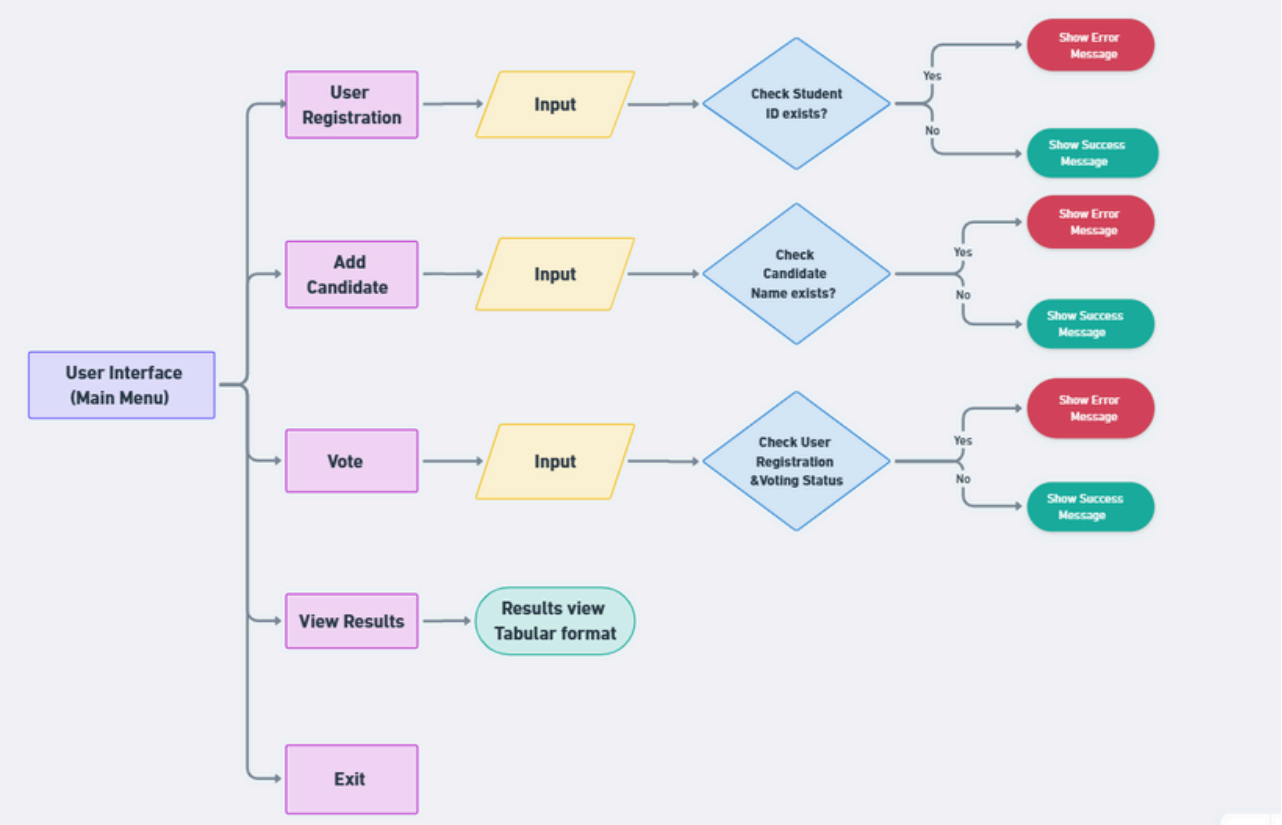

figure: 3.4

Results

The system was tested with multiple scenarios to ensure functionality, accuracy, and robustness

1. **Registration:**
    - The system correctly detected and prevented duplicate student registrations.
    - Successfully registered new students with unique IDs.

2. **Candidate Management:**
    - The system allowed the addition of candidates and blocked duplicate entries.

3. **Voting Process:**
    - Voting was restricted to one vote per student as intended.
    - Attempting to vote a second time triggered a notification indicating the student had already voted.

4. **Results View:**
    - The final results were accurate and updated dynamically based on the number of votes.
    - The Tkinter table provided a clear, user-friendly visualization of election outcomes.

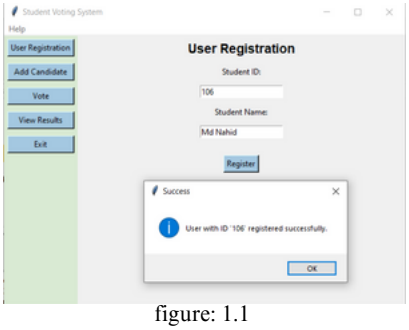

figure: 1.1

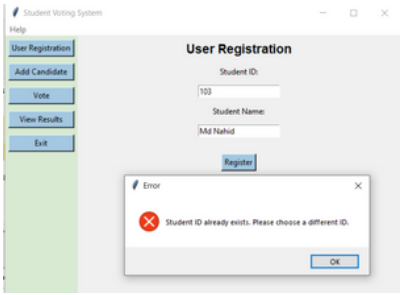

figure: 1.2

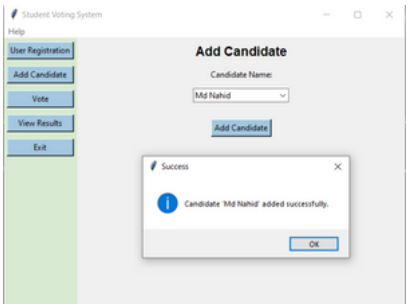

figure: 2.2

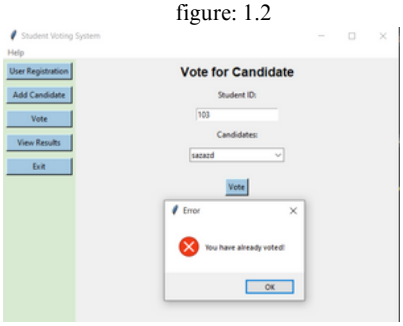

figure: 2.2

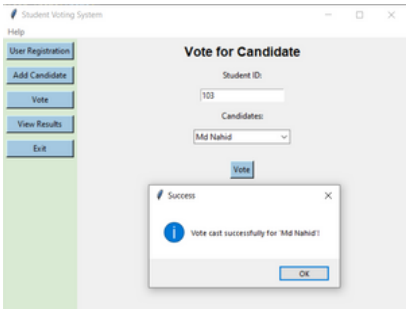

figure: 3.1

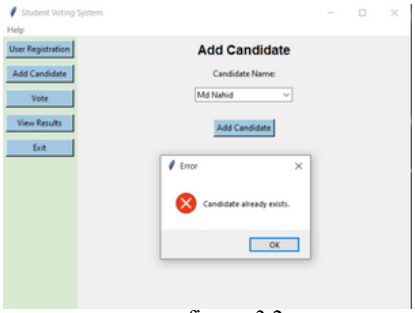

figure: 3.2

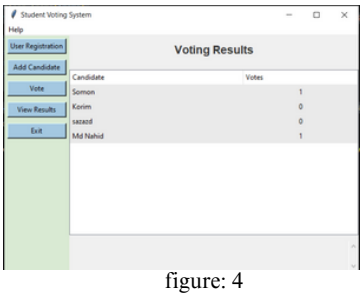

figure: 4

Conclusion

The student voting system successfully integrates a Tkinter-based GUI with a MySQL database, providing a functional platform for student voting. This project demonstrates how Python and MySQL can be used to build reliable applications with a user-friendly interface. The system's ability to prevent duplicate registrations and restrict voting ensures data integrity. Future improvements could include enhancing security features, adding user authentication, and deploying the system on a cloud-based server for remote access.

References

1. **Tkinter Documentation:** https://docs.python.org/3/library/tkinter.html
2. **PyMySQL Documentation**: https://pymysql.readthedocs.io/en/latest/
3. **MySQL Reference Manual:** https://dev.mysql.com/doc/
4. **Python Official Documentation:** https://www.python.org/doc/