Building a CSV-Powered Desktop CRUD App with Python, Tkinter & MySQL

```
import tkinter as tk
from tkinter import messagebox, filedialog
import mysql.connector
import csv
import logging

logging.basicConfig(
    filename="app.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

try:
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",#password
        database="tkinter",
        autocommit=False
    )
    logging.info("Connected to MySQL database successfully.")
except mysql.connector.Error as err:
    logging.error(f"MySQL Connection Error: {err}")
    raise

cursor = conn.cursor()


def csv_file_upload():
    try:
        filename = filedialog.askopenfilename(filetypes=[("CSV FILES",
"*.csv")])
        if not filename:
            logging.info("CSV upload canceled by user.")
            return

        with open(filename, "r") as f:
            data = csv.reader(f)
```

```
            next(data)  # skip header

            for row in data:
                name = row[0].strip()
                age = int(row[1].strip())
                grade = row[2].strip()

                cursor.execute(
                    "INSERT INTO STUDENT (NAME, AGE, GRADE) VALUES (%s,
%s, %s)",
                    (name, age, grade)
                )
        conn.commit()
        messagebox.showinfo("Success", "CSV Data inserted into
Database!")
        logging.info(f"CSV file '{filename}' uploaded successfully.")

    except Exception as e:
        conn.rollback()
        messagebox.showerror("Error", str(e))
        logging.error(f"Error uploading CSV: {e}")

def show_grid():
    try:
        grid.configure(state=tk.NORMAL)
        grid.delete("1.0", tk.END)
        grid.insert(tk.END, "ID\tName\tAge\tGrade\n")

        cursor.execute("SELECT * FROM STUDENT")
        for row in cursor.fetchall():
            grid.insert(tk.END,
f"{row[0]}\t{row[1]}\t{row[2]}\t{row[3]}\n")

        grid.configure(state=tk.DISABLED)
        logging.info("Grid displayed successfully.")
    except Exception as e:
        logging.error(f"Error displaying grid: {e}")

def add_record():
    name = name_entry.get().strip()
    age = age_entry.get().strip()
    grade = grade_entry.get().strip()
```

```
    if name and age and grade:
        try:
            cursor.execute(
                "INSERT INTO STUDENT (NAME, AGE, GRADE) VALUES (%s, %s,
%s)",
                (name, age, grade)
            )
            conn.commit()
            messagebox.showinfo("INFO", "Record Added")
            logging.info(f"Record added: Name={name}, Age={age},
Grade={grade}")

            grid.configure(state=tk.NORMAL)
            grid.delete("1.0", tk.END)
            show_grid()
        except Exception as e:
            conn.rollback()
            messagebox.showerror("Error", str(e))
            logging.error(f"Error adding record: {e}")
    else:
        messagebox.showwarning("Warning", "Enter Valid Data")
        logging.warning("Add record failed: Incomplete data entered.")

def edit_record():
    try:
        grid.configure(state=tk.NORMAL)
        messagebox.showinfo("INFO", "Grid Enabled — You can edit
directly.")
        logging.info("Grid enabled for editing.")
    except Exception as e:
        messagebox.showerror("Error", str(e))
        logging.error(f"Error enabling grid for edit: {e}")

def save_edit():
    try:
        lines = grid.get("2.0", tk.END).strip().split("\n")
        for line in lines:
            parts = line.split("\t")
            if len(parts) == 4:
                id_, name, age, grade = parts
                cursor.execute(
                    "UPDATE STUDENT SET NAME=%s, AGE=%s, GRADE=%s WHERE
ID=%s",
```

```
                    (name.strip(), int(age.strip()), grade.strip(),
int(id_.strip()))
                )
        conn.commit()
        messagebox.showinfo("Success", "Records updated successfully!")
        grid.configure(state=tk.DISABLED)
        show_grid()
        logging.info("Records updated successfully via grid edit.")
    except Exception as e:
        messagebox.showerror("Error", str(e))
        logging.error(f"Error saving edited records: {e}")

def delete_record():
    id_ = id_entry.get()
    try:
        if not id_.isdigit():
            raise ValueError("ID must be a number")

        cursor.execute("SELECT * FROM STUDENT WHERE ID = %s", (id_,))
        row = cursor.fetchone()
        if row is None:
            raise LookupError("ID does not exist in the table")

        cursor.execute("DELETE FROM STUDENT WHERE ID = %s", (id_,))
        conn.commit()
        messagebox.showinfo("INFO", "Record Deleted Successfully")
        logging.info(f"Record with ID={id_} deleted successfully.")

        grid.configure(state=tk.NORMAL)
        grid.delete("1.0", tk.END)
        show_grid()
    except ValueError as ve:
        messagebox.showwarning("Warning", str(ve))
        logging.warning(f"Delete record failed: {ve}")
    except LookupError as le:
        messagebox.showwarning("Warning", str(le))
        logging.warning(f"Delete record failed: {le}")
    except Exception as e:
        conn.rollback()
        messagebox.showerror("Error", f"Something went wrong:
{str(e)}")
        logging.error(f"Error deleting record: {e}")
```

```
root = tk.Tk()
root.title("Upload CSV Data")
root.geometry('400x300')

tk.Button(root, text="Upload CSV",
command=csv_file_upload).pack(pady=40)
grid = tk.Text(root, height=15, width=60)
grid.pack(pady=10)

input_frame = tk.Frame(root)
input_frame.pack()

id_label = tk.Label(input_frame, text="ID:").grid(row=0, column=0)
id_entry = tk.Entry(input_frame, width=5)
id_entry.grid(row=0, column=1, padx=10, pady=10)

name_label = tk.Label(input_frame, text="Name:").grid(row=0, column=2)
name_entry = tk.Entry(input_frame, width=15)
name_entry.grid(row=0, column=3, padx=10, pady=10)

age_label = tk.Label(input_frame, text="Age:").grid(row=0, column=4)
age_entry = tk.Entry(input_frame, width=7)
age_entry.grid(row=0, column=5, padx=10, pady=10)

grade_label = tk.Label(input_frame, text="Grade:").grid(row=0,
column=6)
grade_entry = tk.Entry(input_frame, width=5)
grade_entry.grid(row=0, column=7, padx=10, pady=10)

tk.Button(input_frame, text="Add", command=add_record).grid(row=1,
column=0, padx=10, pady=10)
tk.Button(input_frame, text="Edit", command=edit_record).grid(row=1,
column=1, padx=10, pady=10)
tk.Button(input_frame, text="Save", command=save_edit).grid(row=1,
column=2, padx=10, pady=10)
tk.Button(input_frame, text="Delete",
command=delete_record).grid(row=1, column=3, padx=10, pady=10)

show_grid()
root.mainloop()
```