# AI-Based Automated Exam Scheduling System - Complete Guide

## Step 1: Install Tkinter and Dependencies

1. Install Python (if not installed):  
 - Download Python from https://www.python.org/downloads/  
 - Add Python to PATH during installation.  
   
2. Install required dependencies using pip:  
 ```  
 pip install tkinter pandas pymysql matplotlib seaborn openpyxl  
 ```  
3. Verify installation:  
 ```  
 python -m tkinter  
 ```  
 If a blank window appears, Tkinter is installed correctly.

## Step 2: Create the Main Window

import tkinter as tk  
from tkinter import ttk, messagebox  
import pymysql  
import pandas as pd  
  
# MySQL connection details  
db\_config = {  
 'host': 'localhost',  
 'user': 'root',  
 'password': '',  
 'database': 'exam\_scheduler'  
}  
  
def load\_schedule():  
 try:  
 conn = pymysql.connect(\*\*db\_config)  
 query = "SELECT \* FROM exam\_schedule"  
 schedule = pd.read\_sql(query, conn)  
 conn.close()  
 return schedule  
 except Exception as e:  
 messagebox.showerror("Error", f"Failed to load schedule: {e}")  
 return None  
  
def display\_schedule():  
 for row in tree.get\_children():  
 tree.delete(row)  
  
 schedule = load\_schedule()  
 if schedule is not None:  
 for \_, row in schedule.iterrows():  
 tree.insert("", "end", values=(  
 row['day'], row['date'], row['time\_slot'],  
 row['course\_name'], row['classroom\_name'], row['students\_registered']  
 ))  
  
def generate\_schedule():  
 try:  
 conn = pymysql.connect(\*\*db\_config)  
 with conn.cursor() as cursor:  
 cursor.execute("CALL generate\_exam\_schedule()")  
 conn.commit()  
 conn.close()  
 display\_schedule()  
 messagebox.showinfo("Success", "Exam schedule generated successfully!")  
 except pymysql.MySQLError as e:  
 if e.args[0] == 1305:  
 messagebox.showwarning("Missing Procedure", "Stored procedure not found. Using Python-based scheduling.")  
 generate\_schedule\_fallback()  
 else:  
 messagebox.showerror("Error", f"Failed to generate schedule: {e}")  
  
# More code follows...

## Step 3: Create the Data Entry Window

import tkinter as tk  
from tkinter import ttk, messagebox  
import pymysql  
  
def open\_data\_entry\_window():  
 data\_window = tk.Toplevel(root)  
 data\_window.title("Data Entry Window")  
 data\_window.geometry("1200x700")  
  
 tabs = ttk.Notebook(data\_window)  
 tabs.pack(expand=1, fill="both")  
  
 # Tab 1: Classrooms  
 classroom\_tab = ttk.Frame(tabs)  
 tabs.add(classroom\_tab, text="Classrooms")  
  
 tk.Label(classroom\_tab, text="Classroom ID:").grid(row=0, column=0, padx=5, pady=5)  
 classroom\_id\_entry = tk.Entry(classroom\_tab)  
 classroom\_id\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 def insert\_classroom():  
 id = classroom\_id\_entry.get()  
 if not id:  
 messagebox.showwarning("Invalid Input", "Please fill all fields.")  
 return  
  
 tk.Button(classroom\_tab, text="Add Classroom", command=insert\_classroom, bg="green", fg="white").grid(row=3, column=0, columnspan=2, pady=5)  
  
# More code follows...

## Step 4: Create Stored Procedure in MySQL

DELIMITER $$  
CREATE PROCEDURE generate\_exam\_schedule()  
BEGIN  
 DECLARE done INT DEFAULT FALSE;  
 DECLARE c\_id INT;  
 DECLARE c\_name VARCHAR(255);  
 DECLARE c\_students INT;  
 DECLARE c\_duration INT;  
 DECLARE cur CURSOR FOR SELECT course\_id, course\_name, students\_registered, duration FROM courses;  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  
  
 OPEN cur;  
 read\_loop: LOOP  
 FETCH cur INTO c\_id, c\_name, c\_students, c\_duration;  
 IF done THEN  
 LEAVE read\_loop;  
 END IF;  
  
 INSERT INTO exam\_schedule (course\_id, course\_name, students\_registered, time\_slot, classroom\_name)  
 SELECT c\_id, c\_name, c\_students, '09:00-11:00', classroom\_name  
 FROM classrooms  
 WHERE capacity >= c\_students  
 LIMIT 1;  
 END LOOP;  
 CLOSE cur;  
END$$  
DELIMITER ;

## Step 5: Clean and Load Data

import pymysql  
import pandas as pd  
  
conn = pymysql.connect(\*\*db\_config)  
classrooms = pd.read\_sql('SELECT \* FROM classrooms', conn)  
courses = pd.read\_sql('SELECT \* FROM courses', conn)  
students = pd.read\_sql('SELECT \* FROM students', conn)  
preferences = pd.read\_sql('SELECT \* FROM preferences', conn)  
conn.close()  
  
# Clean data  
classrooms['capacity'] = pd.to\_numeric(classrooms['capacity'], errors='coerce').fillna(0).astype(int)  
courses['students\_registered'] = pd.to\_numeric(courses['students\_registered'], errors='coerce').fillna(0).astype(int)  
students['student\_id'] = pd.to\_numeric(students['student\_id'], errors='coerce').fillna(0).astype(int)  
preferences['student\_id'] = pd.to\_numeric(preferences['student\_id'], errors='coerce').fillna(0).astype(int)

## Step 6: Generate Reports

import os  
  
output\_dir = "Exam\_Reports"  
os.makedirs(output\_dir, exist\_ok=True)  
  
exam\_schedule.to\_excel(f"{output\_dir}/exam\_schedule.xlsx", index=False)  
student\_schedule.to\_excel(f"{output\_dir}/student\_schedule.xlsx", index=False)  
  
print("✅ Reports saved in 'Exam\_Reports' folder!")