EXAM HALL SEATING ARRANGEMENT SYSTEM

SYNOPSIS

The Exam Hall Seating Arrangement System is a well-defined solution designed specifically for Vysya College. It's created to simplify the process of organizing seating for exams, making it easier and more efficient for college staff.

With this system, Vysya College administrators can input exam details effortlessly, assign seats to students, and generate seating plans quickly. It's user-friendly and equipped with advanced features to handle exams of all sizes and complexities, ensuring it meets the college's diverse needs.

Data security is crucial, and the system prioritizes safeguarding student information at every step. Administrators can trust that sensitive data will be protected throughout the seating allocation process.

Flexibility is also essential. The system allows for real-time adjustments to seating arrangements, accommodating any changes or unexpected situations that may arise during the exam period.

Tailored to align with Vysya College's unique exam policies and room configurations, the system provides customized seating arrangements that meet the college's specific requirements.

Moreover, it enhances the overall exam experience for students by providing clear and organized seating information, thereby reducing confusion.

As Vysya College evolves, so too does the Exam Hall Seating Arrangement System. Regular updates and refinements based on user feedback and emerging technologies ensure that the system remains at the forefront of exam administration practices, delivering long-term value and innovation to Vysya College and its stakeholders.

INTRODUCTION

In today's rapidly changing world driven by advanced data management and automation, educational institutions face the challenge of efficiently organizing and allocating resources. Among these resources, seating arrangements for students in classrooms pose a significant logistical challenge. This project addresses these challenges by providing a simple yet effective solution designed to streamline the process for teachers and Vysya college administrators.

By using modern technology and innovative approaches, this project aims to transform the way seating arrangements are managed in learning environments. With a focus on user-friendly interfaces and intuitive workflows, the system empowers educators and administrators to create optimal seating arrangements with ease and efficiency.

In the past, assigning seats manually and dealing with paperwork was the norm. However, this project introduces a new modern age for educational institutions, using automation and digital tools to streamline the process of allocating seats. This advancement saves time and effort for everyone involved. While Excel and VBA macros have progressed to address certain needs, this project goes beyond by providing a user-friendly graphical interface. This interface improves accessibility and usability for administrators, teachers, and students, making the whole process smoother and more efficient.

Objective:

The primary objective is to optimize and streamline the allocation of seats to students, with the overarching goal of enhancing the overall efficiency and effectiveness of the seating arrangement process. By providing a user-friendly interface and implementing effective seating arrangements, the project aims to simplify seat allocation, making it more accessible and manageable for administrators, teachers, and students.

Benefits:

- Saves time and effort for educators.
- Ensures a fair and balanced distribution of students in classrooms.
- Provides an easy-to-use tool for effective seat management.

SYSTEM SPECIFICATION

Hardware Configuration

Minimum hardware requirements for this system are listed below:

- System : Intel Core i5 or AMD Ryzen 5
- Hard Disk : 250 GB HDD
- Monitor : 1280 x 720 pixels
- RAM : 4GB
- Keyboard : Standard Keyboard

Software Specification

Software requirements for this system are as listed

- Operating System : Mostly Preferable (Any Windows)
- Platform : Python Technology
- Tool : Python 3.12/Visual Studio Code
- Frontend : PyQt5,Tkinter,DataFrames
- Backend : Python, Excel Sheet

SYSTEM STUDY

EXISTING SYSTEM

- The existing system in our college relies on Excel coupled with VBA macros for seating arrangements. This system typically involves manually inputting student data, room specifications, and seating preferences into Excel spreadsheets, with VBA macros automating certain aspects of the allocation process.
- The system allows for highly customizable seating arrangements tailored to specific exam formats, room layouts, and scheduling requirements
- VBA macros automate repetitive tasks, such as assigning students to seats based on predefined criteria and generating seating charts or reports. This automation enhances efficiency and reduces the time and effort required for manual seat allocation.
- The system's functionality may be dependent on individual expertise in Excel and VBA programming. Staff turnover or changes in personnel could disrupt the continuity and maintenance of the system, leading to potential inefficiencies or errors.

Description:

- While the existing system requires users to manually input student data and room specifications into Excel sheets, my project provides a graphical user interface (GUI) that simplifies this process. Instead of navigating complex spreadsheets and writing VBA macros, users can input data effortlessly using intuitive forms and menus.
- Furthermore, the existing system's reliance on VBA macros for automation poses challenges for users who lack programming skills or familiarity with Excel. In contrast, my project eliminates the need for VBA programming by offering built-in algorithms and automation features accessible through the GUI. This approach enhances accessibility and usability for all users, regardless of their technical expertise.
- Overall, my current project represents a significant improvement over the existing system by offering a modern, user-friendly, and automated solution for seating arrangement management in our college.

Drawbacks:

While the existing system with Excel and Macros/VBA may have served its purpose to some extent, it is not without its drawbacks:

1. **Limited Scalability:**

   The existing system's reliance on Excel and VBA macros may limit its scalability, especially when dealing with large datasets or complex seating arrangements. As the volume of data increases, the performance of Excel may degrade, resulting in slower processing times and reduced efficiency.

2. **Dependency on Technical Expertise:**

   Operating and maintaining the existing system requires a certain level of technical expertise, particularly in VBA programming.

3. Complexity:

Managing large datasets and seating configurations in Excel can be complex and cumbersome, particularly for administrators without advanced Excel skills.

PROPOSED SYSTEM

The proposed system is a modern, user-friendly software application designed to revolutionize the process of seating arrangement management in educational institutions. Unlike the existing system, which relies on manual data entry and limited automation through Excel and VBA macros, the proposed system offers a comprehensive solution with advanced features and enhanced usability.

Description

- Imagine a system where you can easily input exam details, such as the number of students and available rooms, using a familiar and intuitive interface. With just a few clicks, you can define seating preferences and let the system handle the rest.
- This system is all about automation. It takes care of assigning seats to students based on various factors like room capacity and special requirements, so you don't have to worry about manual calculations or errors keeping in.
- One of the best parts is its flexibility. Whether you're organizing a small classroom test or a large-scale exam, the system adapts seamlessly to your needs, making it suitable for any educational institution.
- Finally, this system offers a simple and user-friendly solution to the challenges faced in managing seating arrangements for exams in educational institutions. It's like having a smart assistant that handles all the complexities behind the scenes, leaving educators and administrators with a hassle-free experience.

Features

1. **Excel Data Processing:**

   The system allows users to import student data from Excel files, ensuring compatibility with existing data sources.

2. **Efficient Data Allocation:**

   Utilizing Python and pandas, the system processes and allocates student seats in a more efficient and optimized manner.

3. **Dynamic Scalability:**

   The proposed system is designed to handle varying student numbers and classroom configurations, providing scalability for growing educational institutions.

4. **User-Friendly Interface:**

   A Tkinter-based graphical user interface (GUI) makes the system accessible to users with varying technical expertise, reducing dependency on specialized skills.

5. **Error Reduction:**

   Automation minimizes the risk of human errors, enhancing the accuracy of student seat allocation.

6. **Quick Generation of Seating Arrangements:**

   The system generates seating arrangements swiftly, saving time for educators and administrators.

7. **Flexibility for Modifications:**

   Changes in student enrollment or classroom setups can be easily accommodated through the system's adaptable design.

8. **Output in Excel Format:**

   The final seating arrangement is exported to Excel, providing a familiar and widely used format for easy integration into existing workflows.

MODULES

- LOGIN PAGE
- SINUP PAGE
- HOME PAGE
- ROOM SPECIFICATION
- GENERATE REPORT

SYSTEM DESIGN AND DEVELOPMENT

File Design

The file design of the proposed system outlines the organization and structure of data files used during the student seat allocation process. This section provides an overview of the key files and their purposes.

Input File

Excel File (Student Data)

Purpose: The primary input file containing student data.

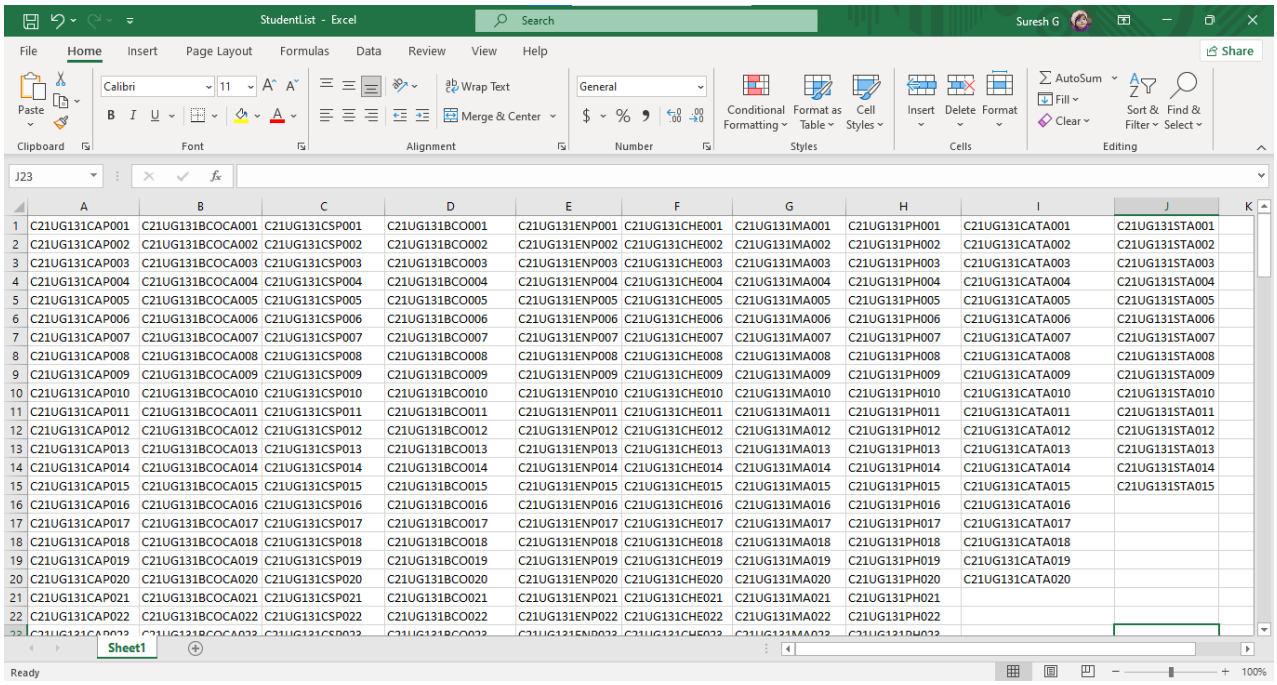
Format: Microsoft Excel (.xlsx)

Structure:

Each column represents the department of students.

Columns may include Students Register Number with decreasing ordered format.

Design:



	A	B	C	D	E	F	G	H	I	J
1	C21UG131CAP001	C21UG131BCOCA001	C21UG131CSP001	C21UG131BCO001	C21UG131ENP001	C21UG131CHE001	C21UG131MA001	C21UG131PH001	C21UG131CATA001	C21UG131STA001
2	C21UG131CAP002	C21UG131BCOCA002	C21UG131CSP002	C21UG131BCO002	C21UG131ENP002	C21UG131CHE002	C21UG131MA002	C21UG131PH002	C21UG131CATA002	C21UG131STA002
3	C21UG131CAP003	C21UG131BCOCA003	C21UG131CSP003	C21UG131BCO003	C21UG131ENP003	C21UG131CHE003	C21UG131MA003	C21UG131PH003	C21UG131CATA003	C21UG131STA003
4	C21UG131CAP004	C21UG131BCOCA004	C21UG131CSP004	C21UG131BCO004	C21UG131ENP004	C21UG131CHE004	C21UG131MA004	C21UG131PH004	C21UG131CATA004	C21UG131STA004
5	C21UG131CAP005	C21UG131BCOCA005	C21UG131CSP005	C21UG131BCO005	C21UG131ENP005	C21UG131CHE005	C21UG131MA005	C21UG131PH005	C21UG131CATA005	C21UG131STA005
6	C21UG131CAP006	C21UG131BCOCA006	C21UG131CSP006	C21UG131BCO006	C21UG131ENP006	C21UG131CHE006	C21UG131MA006	C21UG131PH006	C21UG131CATA006	C21UG131STA006
7	C21UG131CAP007	C21UG131BCOCA007	C21UG131CSP007	C21UG131BCO007	C21UG131ENP007	C21UG131CHE007	C21UG131MA007	C21UG131PH007	C21UG131CATA007	C21UG131STA007
8	C21UG131CAP008	C21UG131BCOCA008	C21UG131CSP008	C21UG131BCO008	C21UG131ENP008	C21UG131CHE008	C21UG131MA008	C21UG131PH008	C21UG131CATA008	C21UG131STA008
9	C21UG131CAP009	C21UG131BCOCA009	C21UG131CSP009	C21UG131BCO009	C21UG131ENP009	C21UG131CHE009	C21UG131MA009	C21UG131PH009	C21UG131CATA009	C21UG131STA009
10	C21UG131CAP010	C21UG131BCOCA010	C21UG131CSP010	C21UG131BCO010	C21UG131ENP010	C21UG131CHE010	C21UG131MA010	C21UG131PH010	C21UG131CATA010	C21UG131STA010
11	C21UG131CAP011	C21UG131BCOCA011	C21UG131CSP011	C21UG131BCO011	C21UG131ENP011	C21UG131CHE011	C21UG131MA011	C21UG131PH011	C21UG131CATA011	C21UG131STA011
12	C21UG131CAP012	C21UG131BCOCA012	C21UG131CSP012	C21UG131BCO012	C21UG131ENP012	C21UG131CHE012	C21UG131MA012	C21UG131PH012	C21UG131CATA012	C21UG131STA012
13	C21UG131CAP013	C21UG131BCOCA013	C21UG131CSP013	C21UG131BCO013	C21UG131ENP013	C21UG131CHE013	C21UG131MA013	C21UG131PH013	C21UG131CATA013	C21UG131STA013
14	C21UG131CAP014	C21UG131BCOCA014	C21UG131CSP014	C21UG131BCO014	C21UG131ENP014	C21UG131CHE014	C21UG131MA014	C21UG131PH014	C21UG131CATA014	C21UG131STA014
15	C21UG131CAP015	C21UG131BCOCA015	C21UG131CSP015	C21UG131BCO015	C21UG131ENP015	C21UG131CHE015	C21UG131MA015	C21UG131PH015	C21UG131CATA015	C21UG131STA015
16	C21UG131CAP016	C21UG131BCOCA016	C21UG131CSP016	C21UG131BCO016	C21UG131ENP016	C21UG131CHE016	C21UG131MA016	C21UG131PH016	C21UG131CATA016	
17	C21UG131CAP017	C21UG131BCOCA017	C21UG131CSP017	C21UG131BCO017	C21UG131ENP017	C21UG131CHE017	C21UG131MA017	C21UG131PH017	C21UG131CATA017	
18	C21UG131CAP018	C21UG131BCOCA018	C21UG131CSP018	C21UG131BCO018	C21UG131ENP018	C21UG131CHE018	C21UG131MA018	C21UG131PH018	C21UG131CATA018	
19	C21UG131CAP019	C21UG131BCOCA019	C21UG131CSP019	C21UG131BCO019	C21UG131ENP019	C21UG131CHE019	C21UG131MA019	C21UG131PH019	C21UG131CATA019	
20	C21UG131CAP020	C21UG131BCOCA020	C21UG131CSP020	C21UG131BCO020	C21UG131ENP020	C21UG131CHE020	C21UG131MA020	C21UG131PH020	C21UG131CATA020	
21	C21UG131CAP021	C21UG131BCOCA021	C21UG131CSP021	C21UG131BCO021	C21UG131ENP021	C21UG131CHE021	C21UG131MA021	C21UG131PH021		
22	C21UG131CAP022	C21UG131BCOCA022	C21UG131CSP022	C21UG131BCO022	C21UG131ENP022	C21UG131CHE022	C21UG131MA022	C21UG131PH022		

Output File:

Excel File (Seating Arrangement)

Purpose: The output file storing the final seating arrangement.

Format: Microsoft Excel (.xlsx)

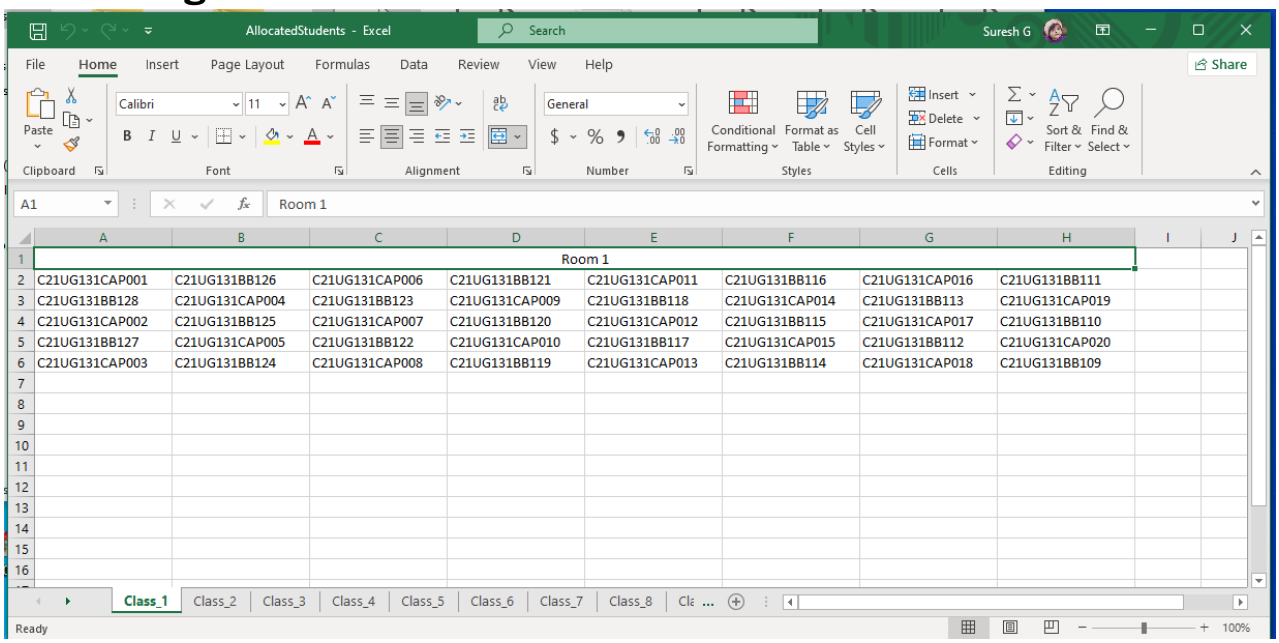
Structure:

Each sheet corresponds to a classroom.

Columns represent seat positions, and rows represent students.

Constraints: Follows the specified format for easy integration into existing workflows.

Design:



AllocatedStudents - Excel

Search

Suresh G

File Home Insert Page Layout Formulas Data Review View Help Share

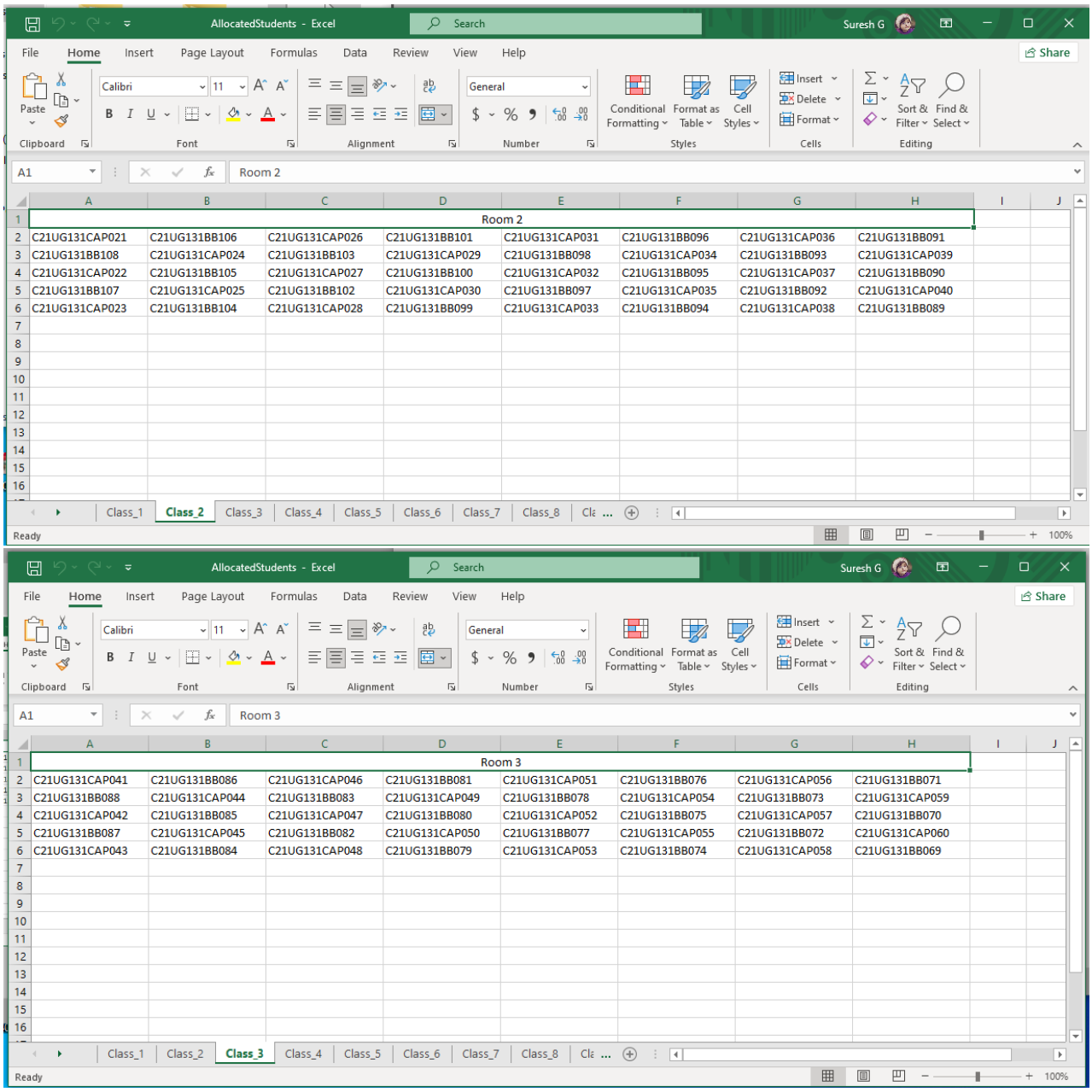
Paste Clipboard Calibri 11 Font Alignment General Number Conditional Formatting Format as Table Cell Styles Styles Insert Delete Format Cells Sort & Filter Find & Select Editing

A1 Room 1

	A	B	C	D	E	F	G	H	I	J
1	Room 1									
2	C21UG131CAP001	C21UG131BB126	C21UG131CAP006	C21UG131BB121	C21UG131CAP011	C21UG131BB116	C21UG131CAP016	C21UG131BB111		
3	C21UG131BB128	C21UG131CAP004	C21UG131BB123	C21UG131CAP009	C21UG131BB118	C21UG131CAP014	C21UG131BB113	C21UG131CAP019		
4	C21UG131CAP002	C21UG131BB125	C21UG131CAP007	C21UG131BB120	C21UG131CAP012	C21UG131BB115	C21UG131CAP017	C21UG131BB110		
5	C21UG131BB127	C21UG131CAP005	C21UG131BB122	C21UG131CAP010	C21UG131BB117	C21UG131CAP015	C21UG131BB112	C21UG131CAP020		
6	C21UG131CAP003	C21UG131BB124	C21UG131CAP008	C21UG131BB119	C21UG131CAP013	C21UG131BB114	C21UG131CAP018	C21UG131BB109		
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										

Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Cla ...

Ready 100%



AllocatedStudents - Excel

Room 2

	A	B	C	D	E	F	G	H
1	Room 2							
2	C21UG131CAP021	C21UG131BB106	C21UG131CAP026	C21UG131BB101	C21UG131CAP031	C21UG131BB096	C21UG131CAP036	C21UG131BB091
3	C21UG131BB108	C21UG131CAP024	C21UG131BB103	C21UG131CAP029	C21UG131BB098	C21UG131CAP034	C21UG131BB093	C21UG131CAP039
4	C21UG131CAP022	C21UG131BB105	C21UG131CAP027	C21UG131BB100	C21UG131CAP032	C21UG131BB095	C21UG131CAP037	C21UG131BB090
5	C21UG131BB107	C21UG131CAP025	C21UG131BB102	C21UG131CAP030	C21UG131BB097	C21UG131CAP035	C21UG131BB092	C21UG131CAP040
6	C21UG131CAP023	C21UG131BB104	C21UG131CAP028	C21UG131BB099	C21UG131CAP033	C21UG131BB094	C21UG131CAP038	C21UG131BB089

AllocatedStudents - Excel

Room 3

	A	B	C	D	E	F	G	H
1	Room 3							
2	C21UG131CAP041	C21UG131BB086	C21UG131CAP046	C21UG131BB081	C21UG131CAP051	C21UG131BB076	C21UG131CAP056	C21UG131BB071
3	C21UG131BB088	C21UG131CAP044	C21UG131BB083	C21UG131CAP049	C21UG131BB078	C21UG131CAP054	C21UG131BB073	C21UG131CAP059
4	C21UG131CAP042	C21UG131BB085	C21UG131CAP047	C21UG131BB080	C21UG131CAP052	C21UG131BB075	C21UG131CAP057	C21UG131BB070
5	C21UG131BB087	C21UG131CAP045	C21UG131BB082	C21UG131CAP050	C21UG131BB077	C21UG131CAP055	C21UG131BB072	C21UG131CAP060
6	C21UG131CAP043	C21UG131BB084	C21UG131CAP048	C21UG131BB079	C21UG131CAP053	C21UG131BB074	C21UG131CAP058	C21UG131BB069

Internal Data Structures:

Lists and DataFrames (Intermediate Data)

Purpose: Temporary data structures for efficient processing.

Format: Python lists and pandas DataFrames.

Structure: Dynamic storage of student data during allocation.

SAMPLE CODING:

Main.py

```
# main_program.py
import sys
from PyQt5.QtWidgets import QApplication, QWidget
from design_module import DesignModule
from allocation_module import AllocationModule

class MainProgram(QWidget):
    def __init__(self):
        super().__init__()

        self.design_module = DesignModule(self)
        self.allocation_module = AllocationModule(self)

        self.init_ui()

    def init_ui(self):
        # Additional initialization if needed
        pass

    def generate_excel_output(self):
        # Retrieve necessary information from the design module
        file_path = self.design_module.file_entry.text()
        students_per_class = int(self.design_module.class_entry.text())
        rows = int(self.design_module.rows_entry.text())
        columns = int(self.design_module.columns_entry.text())
```

```
        # Use the allocation module to perform the logic
        l3, _ = self.allocation_module.read_excel_and_generate_output(file_path, students_per_class)
        self.allocation_module.generate_excel_output(l3, students_per_class, rows, columns)


def main():
    app = QApplication(sys.argv)
    window = MainProgram()
    window.show()
    sys.exit(app.exec_())


if __name__ == '__main__':
    main()
```

Login.py

```
import tkinter as tk
from tkinter import Toplevel, Label, Button, Entry, Checkbutton, BooleanVar
from PIL import Image, ImageTk
import ast


def show_custom_error(title, message):
    error_dialog = Toplevel()
    error_dialog.title(title)
    error_dialog.iconbitmap(r"C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\logoico.ico")
    error_dialog.geometry('400x150+500+300')
    error_dialog.configure(bg='#c0392b')  # Set background color to light red
    error_dialog.resizable(False, False)
```

```
    Label(error_dialog, text=message, fg='#f5f6fa', bg='#c0392b', font=('Arial', 14, 'bold')).pack(pady=5)


    def close_dialog():
        error_dialog.destroy()
        # Resume background tasks here


    Button(error_dialog, text='OK', command=close_dialog, bg='#0652DD', fg='white', font=('Arial', 14, 'bold')).pack(pady=30)


    # Make the error dialog modal
    error_dialog.grab_set()
    # Wait until the error dialog is closed before continuing with the background tasks
    error_dialog.wait_window()


def toggle_password_visibility():
    global show_password
    show_password = not show_password
    if show_password:
        code.config(show="")
    else:
        code.config(show="*")


def signin():
    username = user.get()
    password = code.get()
```

```
    if username == 'admin' and password == '1234':
        screen = Toplevel()
        screen.title("App")
        screen.geometry('925x500+300+200')
        screen.config(bg='white')

        Label(screen, text='Hello Everyone', bg="#fff", font=('Calibri(Body)', 50,
'bold')).pack(expand=True)

        screen.mainloop()
    elif username != 'admin' and password != '1234':
        show_custom_error("Incorrect", "Incorrect Username and Password")

    elif password != '1234':
        show_custom_error("Incorrect", "Incorrect Password")

    elif username != 'admin':
        show_custom_error("Incorrect", "Incorrect Password")

root = tk.Tk()
root.title('EXAM HALL SEATING ARRANGEMENT')
root.iconbitmap(r'C:\Users\sures\OneDrive\Desktop\Exam
Seat\Assests\img\logoico.ico')
root.geometry('925x500+300+200')
root.configure(bg='#e84118')  # Set background color to #e84118
root.resizable(False, False)

# Load and display the background image
try:
```

```
    pil_image = Image.open(r'C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\bg1.jpg')
    img = ImageTk.PhotoImage(pil_image)
    label = Label(root, image=img)
    label.place(x=0, y=0, relwidth=1, relheight=1)
except Exception as e:
    print(f"An error occurred: {e}")


# Create a custom title bar
title_bar = tk.Frame(root, bg='#1abc9c', height=30)
title_bar.pack(fill='x')


# Create a label for the title
title_label = tk.Label(title_bar, text='Login Page', bg='#1abc9c', fg='white', font=('Helvetica', 12,'bold'))
title_label.pack(side='left', padx=10)


# Create the frame
frame = tk.Frame(root, width=350, height=350, bg='white')
frame.place(x=480, y=70)


# Add heading label to the frame
heading = tk.Label(frame, text='Sign In', fg='#1abc9c', bg='white', font=('Microsoft Vahei UI L', 23, 'bold'))
heading.place(x=120, y=5)


###=================================================
# Create an Username Entry widget
```

```
def on_enter_user(e):
    user.delete(0,'end')

def on_leave_user(e):
    name=user.get()
    if name=='':
        user.insert(0,'Username')



user = tk.Entry(frame, width=25, fg='black', border=0, bg='white', font=('Microsoft Vahei UI Light', 11))
user.place(x=30, y=80)  # Adjust the position as needed
user.insert(0,'Username')
user.bind('<FocusIn>',on_enter_user)
user.bind('<FocusOut>',on_leave_user)

tk.Frame(frame,width=295,height=2,bg='black').place(x=25,y=107)

###==================================================
# Create a Password Entry widget
def on_enter_code(e):
    code.delete(0,'end')

def on_leave_code(e):
    name=code.get()
    if name=='':
        code.insert(0,'Password')
```

```
show_password = False

code = tk.Entry(frame, width=20, fg='black', border=0, bg='white',
font=('Microsoft Vahei UI Light', 11), show="*")

code.place(x=30, y=150)  # Adjust the position as needed

code.insert(0,'Password')

code.bind('<FocusIn>',on_enter_code)

code.bind('<FocusOut>',on_leave_code)


tk.Frame(frame,width=295,height=2,bg='black').place(x=25,y=177)


##============================================================

##============================================================

tk.Button(frame, width=39, pady=7, text='Sign in', bg='#1abc9c', fg='white',
border=0, command=signin).place(x=35, y=240)

label = tk.Label(frame, text="Don't have an account?", fg='black', bg='white',
font=('Microsoft Vahei UI Light', 9))

label.place(x=75, y=290)


sign_up = tk.Button(frame, width=6, text='Sign up', border=0, bg='white',
cursor='hand2', fg='#1abc9c')

sign_up.place(x=210, y=290)



# Checkbox for hiding and showing password

show_password_var = BooleanVar()

show_password_check = Checkbutton(frame, text="Show Password",
variable=show_password_var, command=toggle_password_visibility, bg='white',
fg='#1abc9c')

show_password_check.place(x=30, y=200)
```

```
root.mainloop()
```

SignUp.py

```
from tkinter import *
from tkinter import messagebox
import ast
import tkinter as tk
from tkinter import Toplevel, Label, Button, Entry
from PIL import Image, ImageTk



window=Tk()
window.title("EXAM HALL SEATING ARRANGEMENT")
window.geometry('925x500+300+200')
window.configure(bg='#fff')
window.resizable(False,False)
window.iconbitmap(r'C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\logoico.ico')


def show_custom_error(title, message):
    error_dialog = Toplevel()
    error_dialog.title(title)
    error_dialog.iconbitmap(r"C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\logoico.ico")
    error_dialog.geometry('400x150+500+300')
    error_dialog.configure(bg='#c0392b')  # Set background color to light red
    error_dialog.resizable(False, False)


    Label(error_dialog, text=message, fg='#f5f6fa', bg='#c0392b', font=('Arial', 14, 'bold')).pack(pady=5)
```

```
    def close_dialog():
        error_dialog.destroy()
        # Resume background tasks here

    Button(error_dialog, text='OK', command=close_dialog, bg='#0652DD', fg='white', font=('Arial', 14, 'bold')).pack(pady=30)

    # Make the error dialog modal
    error_dialog.grab_set()
    # Wait until the error dialog is closed before continuing with the background tasks
    error_dialog.wait_window()

def show_custom_info(title, message):
    info_dialog = Toplevel()
    info_dialog.title(title)
    info_dialog.iconbitmap(r"C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\logoico.ico")
    info_dialog.geometry('400x150+500+300')
    info_dialog.configure(bg='#1abc9c')  # Set background color to light green
    info_dialog.resizable(False, False)

    Label(info_dialog, text=message, fg='#f5f6fa', bg='#1abc9c', font=('Arial', 14, 'bold')).pack(pady=5)

    def close_dialog():
        info_dialog.destroy()
        # Resume background tasks here

    Button(info_dialog, text='OK', command=close_dialog, bg='#0652DD', fg='white', font=('Arial', 14, 'bold')).pack(pady=30)
```

```
    # Make the info dialog modal
    info_dialog.grab_set()
    # Wait until the info dialog is closed before continuing with the background tasks
    info_dialog.wait_window()

def signup():
    username = user.get()
    password = code.get()
    confirm_password = confirm_code.get()

    if password == confirm_password:
        try:
            file = open('datasheet.txt', 'r+')
            d = file.read()
            r = ast.literal_eval(d)

            dict2 = {username: password}
            r.update(dict2)
            file.truncate(0)
            file.close()

            file = open('datasheet.txt', 'w')
            w = file.write(str(r))

            show_custom_info('SignUp', 'Successfully Signed Up')
        except:
            file = open('datasheet.txt', 'w')
            pp = str({'Username': 'Password'})
```

```
            file.write(pp)
            file.close()
    else:
        show_custom_error('Incorrect', 'Password should not match')




def sign():
    window.destroy()


# Load and display the background image
try:
    pil_image = Image.open(r'C:\Users\sures\OneDrive\Desktop\Exam Seat\Assests\img\bg1.jpg')
    img = ImageTk.PhotoImage(pil_image)
    label = Label(window, image=img)
    label.place(x=0, y=0, relwidth=1, relheight=1)
except Exception as e:
    print(f"An error occurred: {e}")


# Create a custom title bar
title_bar = tk.Frame(window, bg='#1abc9c', height=30)
title_bar.pack(fill='x')


# Create a label for the title
title_label = tk.Label(title_bar, text='SignUp Page', bg='#1abc9c', fg='white', font=('Helvetica', 12,'bold'))
title_label.pack(side='left', padx=10)


#Create Frame
```

```
frame=Frame(window,width=350,height=390,bg='white')
frame.place(x=480,y=50)


heading = tk.Label(frame, text='Sign Up', fg='#1abc9c', bg='white', font=('Microsoft Vahei UI L', 23, 'bold'))
heading.place(x=120, y=5)


###------------------------------------------------------------------------------------------------------------
def on_enter(e):
    user.delete(0,'end')
def on_leave(e):
    if user.get()='':
        user.insert(0,'Username')


user=Entry(frame,width=25,fg='black',border=0,bg='white',font=('Microsoft Vahei UI L', 11))
user.place(x=30,y=80)
user.insert(0,'Username')
user.bind('<FocusIn>',on_enter)
user.bind('<FocusOut>',on_leave)



Frame(frame,width=295,height=2,bg='black').place(x=25,y=107)


###------------------------------------------------------------------------------------------------------------
def on_enter(e):
    code.delete(0,'end')
def on_leave(e):
    if code.get()='':
```

```
        code.insert(0,'Password')


code=Entry(frame,width=25,fg='black',border=0,bg='white',font=('Microsoft Vahei UI
Light', 11))
code.place(x=30,y=150)
code.insert(0,'Password')
code.bind('<FocusIn>',on_enter)
code.bind('<FocusOut>',on_leave)


Frame(frame,width=295,height=2,bg='black').place(x=25,y=177)
###-----------------------------------------------------------------------------------------------------
------
def on_enter(e):
    confirm_code.delete(0,'end')
def on_leave(e):
    if confirm_code.get()=='':
        confirm_code.insert(0,'Confirm Password')


confirm_code=Entry(frame,width=25,fg='black',border=0,bg='white',font=('Microsoft
Vahei UI Light', 11))
confirm_code.place(x=30,y=220)
confirm_code.insert(0,'Confirm Password')
confirm_code.bind('<FocusIn>',on_enter)
confirm_code.bind('<FocusOut>',on_leave)


Frame(frame,width=295,height=2,bg='black').place(x=25,y=247)
#-------------------------------------------------------------------------------


Button(frame,width=39,pady=7,text='Sign
Up',bg='#1abc9c',fg='white',border=0,command=signup).place(x=35,y=280)
```

```
label=Label(frame,text='I have an account',fg='black',bg='white',font=('Microsoft Vahei UI Light', 9))

label.place(x=90,y=340)


signin=Button(frame,width=6,text='Sign in',border=0,bg='white',cursor='hand2',fg='#1abc9c',command=sign)

signin.place(x=190,y=340)



window.mainloop()
```

Design_module.py

```
# design_module.py

from PyQt5.QtWidgets import QLabel, QLineEdit, QPushButton, QVBoxLayout, QFileDialog


class DesignModule:
    def __init__(self, parent):
        self.parent = parent
        self.init_ui()

    def init_ui(self):
        self.parent.setWindowTitle('Classroom Generator')
        self.parent.setGeometry(100, 100, 400, 200)

        self.file_label = QLabel('Select Excel file:')
        self.file_entry = QLineEdit()
        self.browse_button = QPushButton('Browse', self.parent)
        self.browse_button.clicked.connect(self.browse_file)
```

```
self.class_label = QLabel('Students per Class:')
self.class_entry = QLineEdit()

self.rows_label = QLabel('Rows:')
self.rows_entry = QLineEdit()

self.columns_label = QLabel('Columns:')
self.columns_entry = QLineEdit()

self.generate_button = QPushButton('Generate Output', self.parent)
self.generate_button.clicked.connect(self.parent.generate_excel_output)



layout = QVBoxLayout()
layout.addWidget(self.file_label)
layout.addWidget(self.file_entry)
layout.addWidget(self.browse_button)

layout.addWidget(self.class_label)
layout.addWidget(self.class_entry)

layout.addWidget(self.rows_label)
layout.addWidget(self.rows_entry)

layout.addWidget(self.columns_label)
layout.addWidget(self.columns_entry)

layout.addWidget(self.generate_button)
```

```
        self.parent.setLayout(layout)


    def browse_file(self):
        filename, _ = QFileDialog.getOpenFileName(self.parent, 'Select Excel file', '/', 'Excel files (*.xlsx);;All files (*.*)')
        self.file_entry.setText(filename)
```

Allocation_module.py

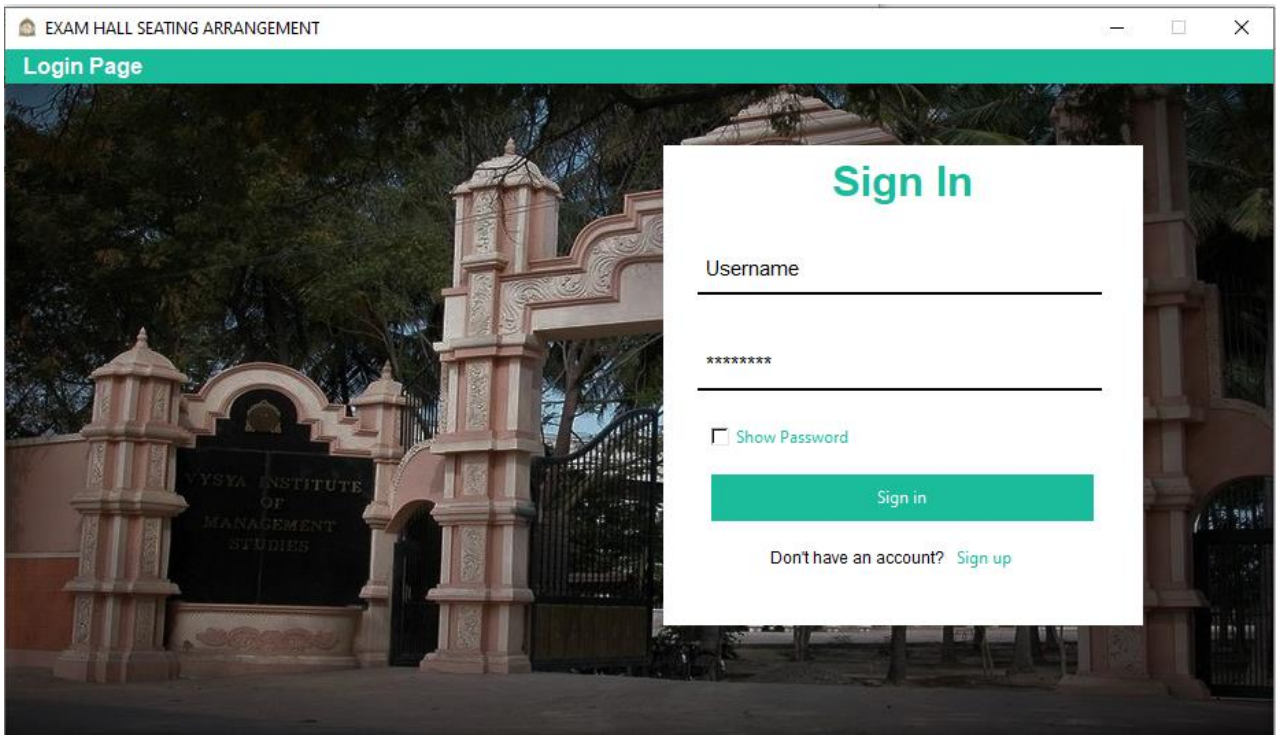
```
# design_module.py

from PyQt5.QtWidgets import QLabel, QLineEdit, QPushButton, QVBoxLayout, QFileDialog


class DesignModule:
    def __init__(self, parent):
        self.parent = parent
        self.init_ui()


    def init_ui(self):
        self.parent.setWindowTitle('Classroom Generator')
        self.parent.setGeometry(100, 100, 400, 200)


        self.file_label = QLabel('Select Excel file:')
        self.file_entry = QLineEdit()
        self.browse_button = QPushButton('Browse', self.parent)
        self.browse_button.clicked.connect(self.browse_file)


        self.class_label = QLabel('Students per Class:')
        self.class_entry = QLineEdit()
```

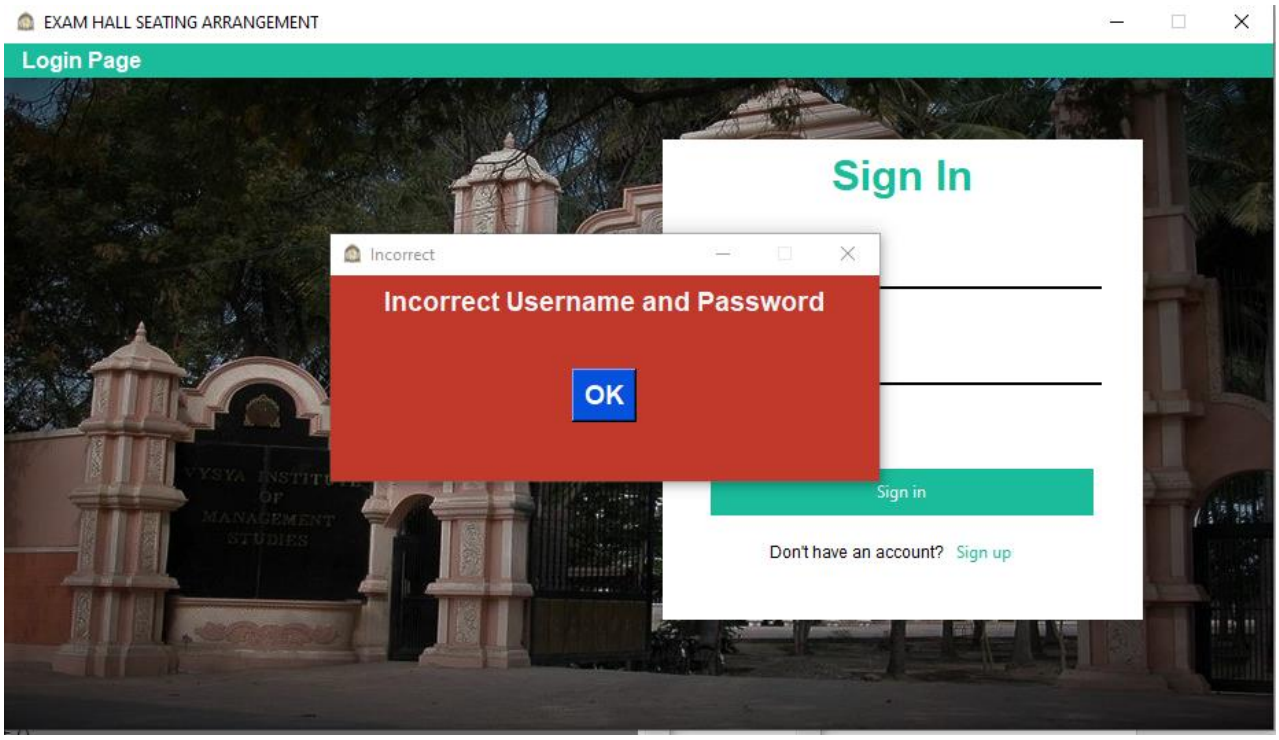
```
self.rows_label = QLabel('Rows:')
self.rows_entry = QLineEdit()

self.columns_label = QLabel('Columns:')
self.columns_entry = QLineEdit()

self.generate_button = QPushButton('Generate Output', self.parent)
self.generate_button.clicked.connect(self.parent.generate_excel_output)



layout = QVBoxLayout()
layout.addWidget(self.file_label)
layout.addWidget(self.file_entry)
layout.addWidget(self.browse_button)

layout.addWidget(self.class_label)
layout.addWidget(self.class_entry)

layout.addWidget(self.rows_label)
layout.addWidget(self.rows_entry)

layout.addWidget(self.columns_label)
layout.addWidget(self.columns_entry)

layout.addWidget(self.generate_button)

self.parent.setLayout(layout)
```

```
    def browse_file(self):
        filename, _ = QFileDialog.getOpenFileName(self.parent, 'Select Excel file', '/', 'Excel files (*.xlsx);;All files (*.*)')
        self.file_entry.setText(filename)
```

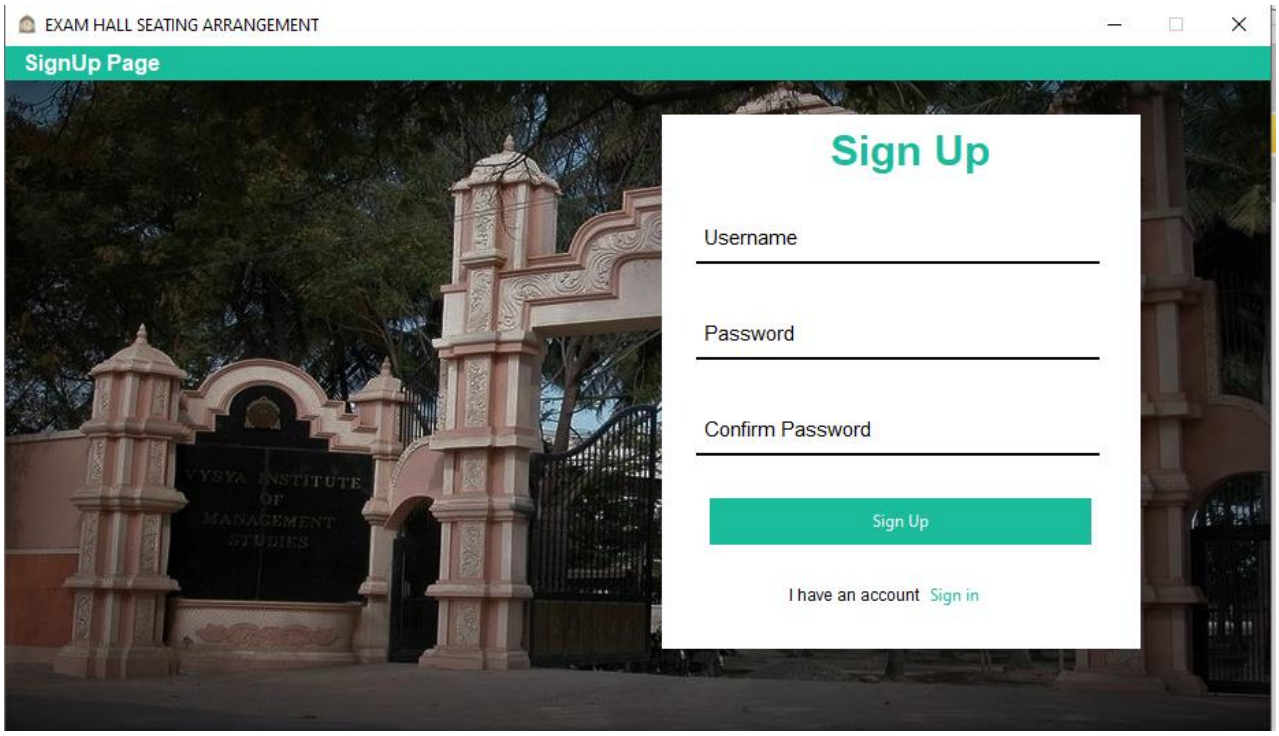
SAMPLE INPUT AND OUTPUT

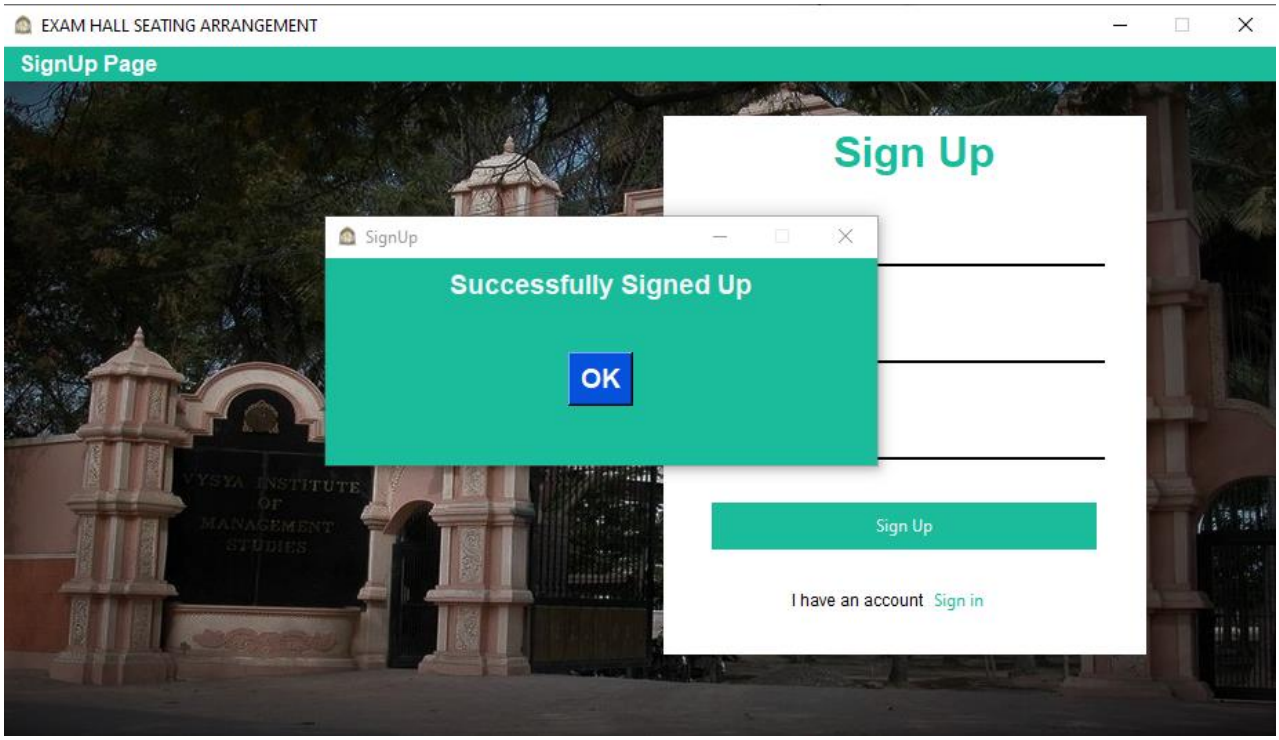
Login Page



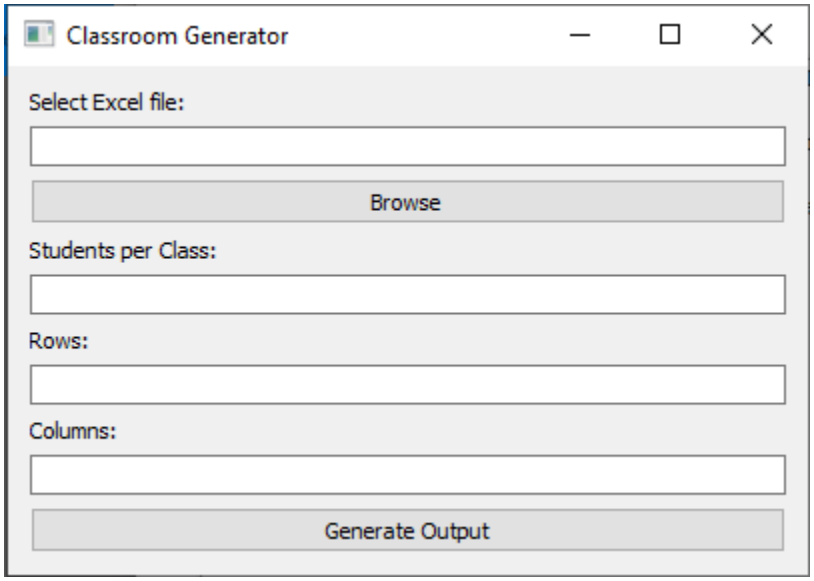

EXAM HALL SEATING ARRANGEMENT
Login Page
Sign In
Incorrect
Incorrect Username and Password
OK
Sign in
Don't have an account? Sign up

SignUp Page


EXAM HALL SEATING ARRANGEMENT
SignUp Page
Sign Up
Username
Password
Confirm Password
Sign Up
I have an account Sign in


EXAM HALL SEATING ARRANGEMENT
SignUp Page
Sign Up
SignUp
Successfully Signed Up
OK
Sign Up
I have an account Sign in

Home Page


Classroom Generator
Select Excel file:
Browse
Students per Class:
Rows:
Columns:
Generate Output