

DYNAMIC SEATING ALLOCATION SYSTEM

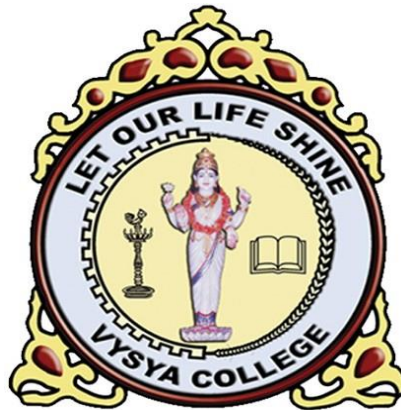
A Project Work submitted in partial fulfilment of the
requirements for the degree of
BACHELOR OF COMPUTER APPLICATIONS
to the

PERIYAR UNIVERSITY, SALEM - 11

Submitted By

G. SURESH

(Reg. No. C21UG131CAP041)



DEPARTMENT OF COMPUTER APPLICATIONS

VYSYA COLLEGE

(AFFILIATED TO PERIYAR UNIVERSITY)

SALEM – 636 103

MARCH – 2024

Date:

CERTIFICATE

This is to certify that the Project Work entitled "**DYNAMIC SEATING ALLOCATION SYSTEM**" submitted in partial fulfillment of the requirements of the degree of Bachelor of Computer Applications to the Periyar University, Salem is a record of bonafide work carried out by **G. SURESH** Reg. No.**C21UG131CAP041** under my supervision and guidance.

Head of the Department

Internal Guide

Date of Viva-voice:

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I am immensely grateful to **Thiru J. Rajendra Prasad**, Correspondent of Vysya College, Salem, for his unwavering support and invaluable guidance throughout the project. His encouragement and provision of essential information were instrumental in its successful completion.

I extend my heartfelt appreciation to **Dr. P. Venkatesan M.Sc, Ph.D.** Principal of Vysya College, Salem, for granting me the opportunity to undertake this project and for his continuous encouragement.

I am indebted to the Head of the Department of Computer Applications at Vysya College, Salem, for his insightful suggestions, innovative ideas, and constructive feedback, which greatly contributed to the project's development.

Special thanks to my guide and the dedicated faculty members of our department for their constant support and assistance during the project's execution.

I also express my gratitude to all the other faculty members and individuals who generously shared their expertise and resources, making significant contributions to the project's success.

I would like to express my special gratitude and thanks to all above mentioned people for giving us such attention and time. My thanks and appreciations also go to our other faculties in developing the project and people who have willingly helped us out with their abilities.

I sincerely appreciate the time and attention provided by everyone involved, without which this project would not have been possible.

CONTENTS

CHAPTER	PAGE NO
COLLEGE BONAFIDE CERTIFICATE	
ACKNOWLEDGEMENT	
SYNOPSIS	
CHAPTER 1	
1. INTRODUCTION	1
1.1 SYSTEM SPECIFICATION	2
1.1.1 HARDWARE CONFIGURATION	2
1.1.2 SOFTWARE SPECIFICATION	2
CHAPTER 2	
2. SYSTEM STUDY	
2.1 EXISTING SYSTEM	3
2.1.1 DESCRIPTION	3
2.1.2 DRAWBACKS	4
2.2 PROPOSED SYSTEM	5
2.2.1 DESCRIPTION	5
2.2.2 FEATURES	6
CHAPTER 3	
3. SYSTEM DESIGN AND DEVELOPMENT	
3.1 FILE DESIGN	7
3.2 INPUT DESIGN	7
3.3 OUTPUT DESIGN	8
3.4 CODE DESIGN	8
3.5 SYSTEM DEVELOPMENT	9
3.5.1 DESCRIPTION OF MODULES	9
CHAPTER 4	
4. SYSTEM DESIGN AND IMPLEMENTATION	12
CHAPTER 5	
5. CONCLUSION	15
5.1 FUTURE ENHANCEMENT	15
CHAPTER 6	
6. BIBLIOGRAPHY	16
APPENDICES	
A. DATA FLOW DIAGRAM	17
B. USE CASE DIAGRAM	18
C. SAMPLE CODING	19
D. SAMPLE INPUT	37
E. SAMPLE OUTPUT	40

SYNOPSIS

SYNOPSIS

The Dynamic Seating Allocation System is a software-based solution designed to streamline and optimize the process of seating arrangement for exams in college. This system leverages advanced algorithms and data analysis techniques to automate the tedious task of assigning seats to students based on various criteria. Traditionally, exam seating arrangements are manually handled, consuming significant time and effort from administrators. In our system aims to alleviate this burden by providing an efficient and accurate seating allocation process. By utilizing the power of technology, it ensures fairness, security, and convenience for both students and exam administrators. The system incorporates several key features. First, it allows administrators to define and input relevant parameters, such as the number of students, seating a capacity, and any special requirements. It then processes this information to generate optimal seating plans that satisfy constraints such as seating capacity, student preferences, and separation between certain individuals. It intelligently assigns seats to ensure fairness and prevent cheating while respecting any specific by the institution. The system also offers flexibility and adaptability. It can handle last-minute changes, such and student absences or new enrollments, and quickly generate revised seating arrangements.

INTRODUCTION

CHAPTER 1

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 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.

1.1 SYSTEM SPECIFICATION

1.1.1 Hardware Configuration

Hardware requirements for this system are listed below:

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

1.1.2 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

CHAPTER 2

SYSTEM STUDY

2.1 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.

2.1.1 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.

2.1.2 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.

2.2 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 .

2.2.1 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.

2.2.2 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. 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.

4. Error Reduction:

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

5. Quick Generation of Seating Arrangements:

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

6. Flexibility for Modifications:

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

7. 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

SYSTEM DESIGN AND DEVELOPMENT

CHAPTER 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

The file design of a student seat allocation system is essential for effectively managing student data, room information, and the seating arrangement process. It involves organizing data into structured files that facilitate efficient storage, retrieval, and manipulation of information. A well-designed file structure ensures the integrity, security, and accessibility of data throughout the seat allocation process.

3.2 INPUT DESIGN

The input file for the student seat allocation system is an Excel spreadsheet (.xlsx). It's like a table where each column represents a different department, like "BCA" or "MATHS". In each department's column, you'll see a list of students, with each student listed on a separate row. Every column shows the student's unique register number, which helps to identify them. This organized setup makes it easy to find and manage student information, which is important for assigning seats during the allocation process. During the allocation process, the input file will be read by alternative columns, with odd columns read alone and even columns read alone. It's essential to ensure that the input file remains consistent throughout this process.

3.3 OUTPUT DESIGN

The student seat allocation system provides seating arrangements in two formats: Excel and PDF.

In the Excel format, seating arrangements are organized like tables in a spreadsheet (.xlsx). Each sheet represents a different class or room in the exam venue. Students are assigned specific seats, displayed in rows and columns with students' register numbers are listed in the cells. This layout helps manage student information efficiently and ensures clear seat assignments. Room numbers are also included for easy identification of exam rooms. It's important to keep the data consistent throughout the process.

The PDF format presents seating arrangements in a visually appealing way. Each page of the PDF corresponds to a specific class or room in the exam hall. Seats are arranged in rows and columns, with students' register numbers listed next to each seat. The PDF includes headers with essential information such as the college name, exam details, and signatures of hall supervisors and principals at the bottom of each page. This format offers a straightforward and comprehensive overview of the seating arrangement, making exam management easier.

3.4 CODE DESIGN

At the code implies, the student seat allocation system is like a smoothly running code, made up of different parts that work together seamlessly. First, it gathers information about students and the available rooms from files, like Excel sheets. Then, it uses a smart algorithm to figure out where each student should sit. This decision is based on things like how big the room is and how many students there are. Once the seats are assigned, the system creates a clear seating plan. This could be in the form of Excel sheets or PDFs, making it easy to see where each student should go. If something goes wrong along the way, the system handles it gracefully. It gives clear messages to help understand and fix any issues that arise.

3.5 SYSTEM DEVELOPMENT

System development encompasses the entire process of creating, implementing, and maintaining software systems to fulfill specific requirements. It begins by comprehending the users' needs and strategizing how to fulfill them, which involves setting objectives, scheduling tasks, and determining resource requirements.

It is a structured approach to creating and improving software systems to meet specific needs. It starts with understanding what the college administrators needed. It involved planning out each step carefully to make sure everything would work well. As the only developer, I had to think through everything and turn ideas into reality. Testing was important to make sure everything worked smoothly. Finally, the system was ready for use by the administrators, meeting their needs effectively.

3.5.1 DESCRIPTION OF MODULES

The modules of the project are:

- Login Module
- Signup Module
- Room Specification
- Generate Excel Output
- Generate PDF Output

Login Module:

The login module is the gateway for administrators to access the system securely. Here's how it works:

- ❖ Administrators need to enter their username and password in the designated fields provided on the screen. Once they've filled in their details, they can click the "Sign in" button.
- ❖ To make it easier, there's a checkbox that allows administrators to show or hide their password while typing.
- ❖ If administrators make a mistake and enter the wrong username or password, the system will show them an error message to let them know and ask them to try again with the correct details.

Signup Module:

The sign-up module serves as a platform for new users to register and create an account within the system.

- ❖ It provides a user-friendly interface where individuals can input their desired username and password.
- ❖ Upon submission, the system ensures that the entered passwords match correctly before proceeding further. In case of any discrepancies, such as mismatched passwords, custom error dialogs are displayed to notify the user about the issue.
- ❖ Valid user credentials are securely stored in a text file for future authentication. Once the registration process is successfully completed, users receive confirmation through a custom information dialog.

Room Specification:

The room specification module facilitates the configuration of exam rooms within the system.

- ❖ It offers a user-friendly interface where administrators can define various parameters related to each exam room, such as room number, capacity, and seating arrangement.
- ❖ Administrators can input details such as the room number and the maximum capacity of students allowed in each room.
- ❖ Additionally, they can specify the layout of the seating arrangement, including the number of rows and columns in the room.

Generate Excel Output

This module is responsible for creating structured Excel spreadsheets that contain the seating arrangements for the exam halls.

- ❖ This module takes the allocated seats data and organizes it into an Excel file format (.xlsx), making it easy to manage and distribute.
- ❖ The process begins by gathering the allocated seating information from the system's Excel Sheet.
- ❖ Then, using the openpyxl library and pandas in Python, the module generates an Excel workbook with multiple sheets, each representing a different exam hall or room.
- ❖ Within each sheet, the seating arrangement is laid out in a tabular format, with rows and columns corresponding to the seats in the room
- ❖ The student's registration number is placed in each cell to denote their allocated seat.

Generate PDF Output

- ❖ This module transforms the allocated seating data into PDF documents, which are easy to distribute and view across different devices.
- ❖ The process begins by gathering the allocated seating information from the system's Excel Sheet , similar to the Excel output module.
- ❖ Using a PDF generation library such as ReportLab and PyFPDF in Python, the module creates PDF files containing the seating arrangements.
- ❖ The seating arrangement is presented in a structured layout, with rows and columns representing individual seats
- ❖ Each seat is labeled with the student's unique identifier or registration number to indicate their allocated position.
- ❖ The PDF output may include other relevant details such as the room number, exam date, and supervisor instructions.

SYSTEM DESIGN AND IMPLEMENTATION

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

TESTING

Testing is the ultimate step of software development. Testing software extend towards coding phase and it represents the ultimate review of configuration, design, coding. A series of test cases are created that are intended to demolish the component that has been built. It is essential that all components of an application be tested, as the failure to do so many results in a series of bugs after the software id put to use.

SOFTWARE TESTING

Software testing is the process used to measure the quality of developed computer software. Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to revel quality-related information about product with respect to the context in which it is intended to operate.

TESTING SOFTWARE MODULES

It characterized as a process of investigating a software and identifying the difference and among the actual and obliged conditions and also to validate the characteristics of the system or component. It can be described as an action of eliminating the software errors from an application. The process of testing is nothing but determining the errors in an application. Activity time the software advance. list is being performed next to using several test cases and using these exams by checking how it works is evaluated headed prove if applications by functioning according in the direction of user obligation or not. Software tough be the most important activity that exhibits the final analysis of design and code generation.

Following testing types are performed after coding.

- Unit Testing
- Integration Testing
- Functionality Testing

UNIT TESTING

Unit testing is implemented to evaluate individual units or components of the project, such as functions, methods, or classes, in isolation. Each unit is tested independently to verify its correctness and functionality, ensuring that it behaves as intended and produces the expected output for a given set of inputs. In unit testing, the focus is on testing small, atomic units of code, typically at the function or method level, to ensure they perform as expected and meet the specified requirements. This approach allows for the early detection and resolution of defects, promoting code reliability and maintainability.

INTEGRATION TESTING

Integration testing focuses on testing how different units or components of the project interact and work together when integrated into the system. This testing ensures that the integration of various modules does not introduce any unexpected behavior or errors and that the system functions seamlessly as a whole.

Integration tests are designed to validate the flow of data and control between different modules, ensuring that data is passed correctly between components and that the overall system behavior meets the specified requirements.

During integration testing, various modules or components are integrated incrementally, and their interactions are examined to detect any inconsistencies, interface mismatches, or communication issues.

FUNCTIONALITY TESTING

Functionality testing assesses the overall functionality of the project by validating its features and capabilities against the specified requirements. This testing ensures that all functionalities work as intended, perform their intended tasks accurately, and meet the user's expectations.

Verify that administrators can securely log in to the system using their credentials and access the necessary functionalities.

Test the system's ability to allocate seats to students based on predefined criteria, such as exam schedules, seating capacity, and special accommodations.

Ensure that seats are assigned accurately and efficiently, considering factors like student preferences, accessibility requirements, and exam regulations.

SYSTEM DEVELOPMENT

Implementation of modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system. Each program is tests individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environments is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so, the system is going to be implemented very soon

CONCLUSION

CHAPTER 5

CONCLUSION

The DYNAMIC SEATING ALLOCATION SYSTEM project was a journey of understanding, planning, and implementation to meet the needs of college administrators effectively. Through meticulous analysis, thoughtful design, and thorough testing, we ensured that the system met its objectives of simplifying exam hall management.

5.1 FUTURE ENHANCEMENT

- For future improvements, we can make the DYNAMIC SEATING ALLOCATION SYSTEM better in a few ways. One idea is exploring features such as dynamic seating allocation algorithms to optimize space utilization and accommodate varying exam schedules efficiently.
- Moreover, integrating a notification system to alert students about their allocated seats and exam details in real-time could enhance user experience and reduce administrative workload.

BIBLIOGRAPHY

CHAPTER 6

BIBILOGRAPHY

For Python

<https://www.w3schools.com/python/>

For PyQt5

<https://doc.qt.io/qtforpython-6/quickstart.html>

For Tkinter

<https://www.geeksforgeeks.org/python-gui-tkinter/>

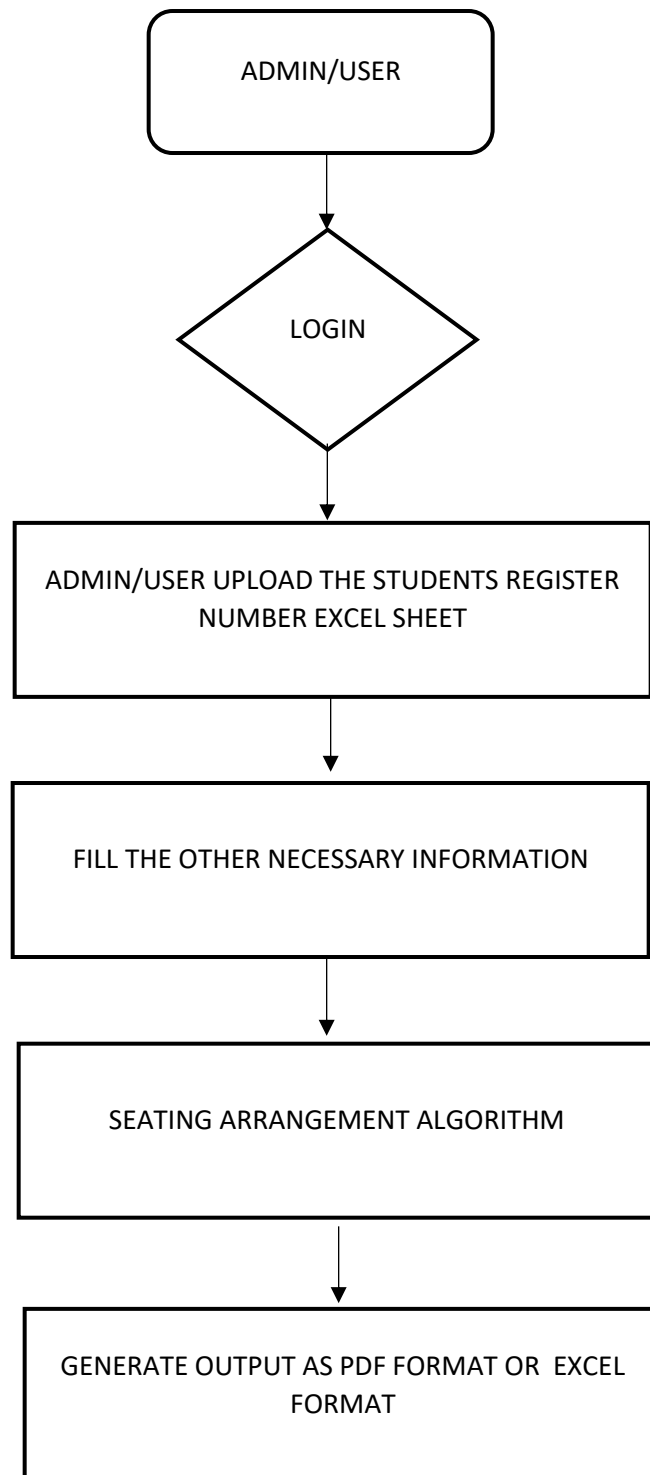
Book:

1. Skiena, S. S. (2008). Algorithm Design Manual. Springer.

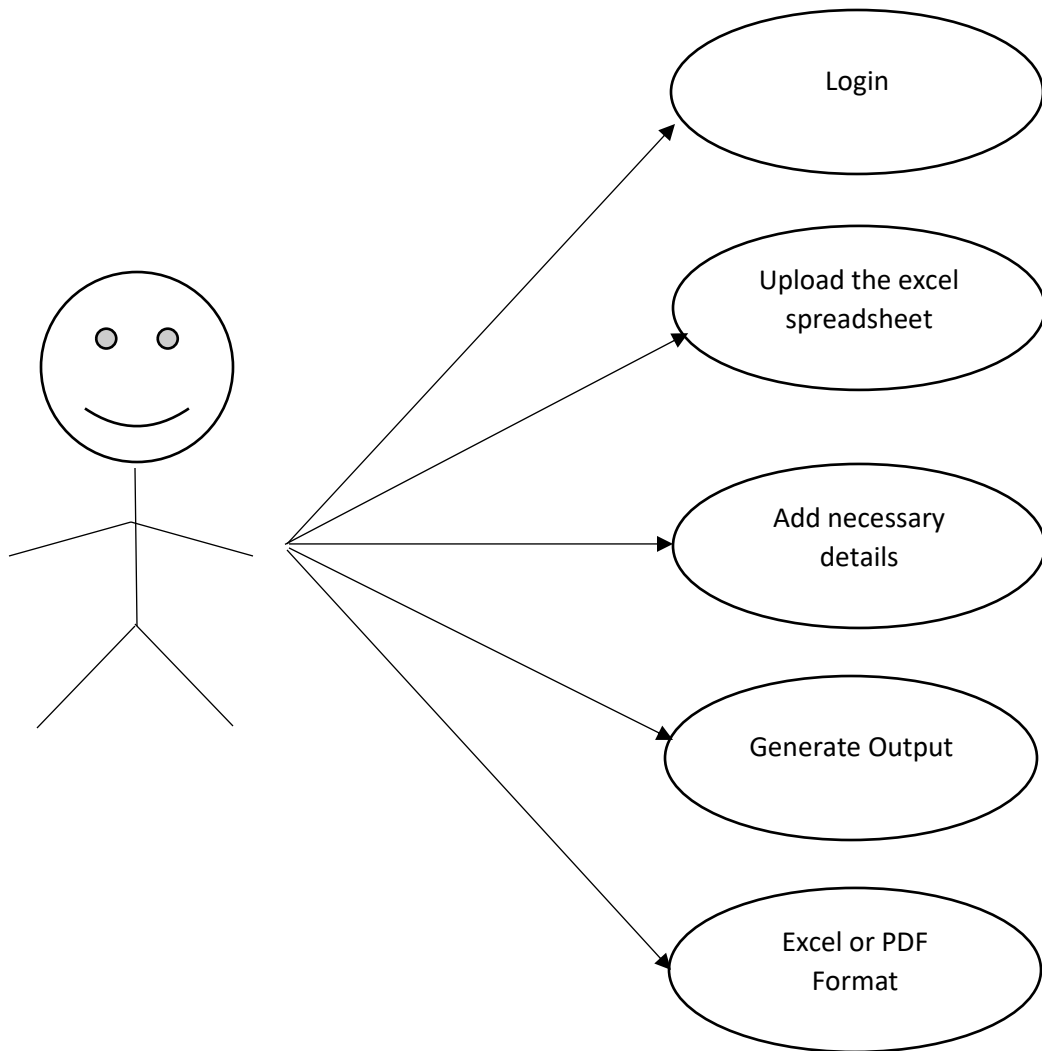
APPENDICES

A. DATA FLOW DIAGRAM

ADMIN FLOW DIAGRAM



A. USE CASE DIAGRAM



C. SAMPLE CODING

Main.py

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QVBoxLayout, QFileDialog, QMessageBox

from PyQt5.QtCore import Qt

from design_module import DesignModule

from allocation_module import AllocationModule

from pdf_module import PdfModule

import sys


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):

        self.setWindowTitle('Classroom Generator')

        self.setGeometry(100, 100, 400, 300)

        layout = QVBoxLayout(self)

        layout.addWidget(self.design_module)


        self.generate_excel_button = QPushButton('Generate Excel Output', self)
        self.generate_excel_button.clicked.connect(self.generate_excel_output)
        layout.addWidget(self.generate_excel_button)


        self.generate_pdf_button = QPushButton('Generate PDF Output', self)
        self.generate_pdf_button.clicked.connect(self.generate_pdf_output)
        layout.addWidget(self.generate_pdf_button)
```

```

def generate_excel_output(self):
    file_path = self.design_module.file_entry.text()
    students_per_class_text = self.design_module.class_entry.text()
    rows_text = self.design_module.rows_entry.text()
    columns_text = self.design_module.columns_entry.text()

    if not all([file_path, students_per_class_text, rows_text, columns_text]):
        QMessageBox.critical(self, 'Error', 'Please fill in all the input fields.')
        return

    try:
        students_per_class = int(students_per_class_text)
        rows = int(rows_text)
        columns = int(columns_text)
    except ValueError:
        QMessageBox.critical(self, 'Error', 'Please enter valid numerical values
for students per class, rows, and columns.')
        return

    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 generate_pdf_output(self):
    try:
        file_path = self.design_module.file_entry.text()
        students_per_class_text = self.design_module.class_entry.text()
        rows_text = self.design_module.rows_entry.text()
        columns_text = self.design_module.columns_entry.text()

```

```

        exam_name = self.design_module.exam_name_combobox.currentText()

        exam_date =
self.design_module.date_edit.date().toString(Qt.DefaultLocaleLongDate)

        session = self.design_module.session_combobox.currentText()


        college_name = self.design_module.get_college_name() #College Name


        if not all([file_path, students_per_class_text, rows_text,
columns_text]):

            QMessageBox.critical(self, 'Error', 'Please fill in all the input fields.')

            return


        try:

            students_per_class = int(students_per_class_text)

            rows = int(rows_text)

            columns = int(columns_text)

        except ValueError:

            QMessageBox.critical(self, 'Error', 'Please enter valid numerical
values for students per class, rows, and columns.')

            return


        if rows * columns != students_per_class:

            QMessageBox.critical(self, 'Error', 'The specified number of rows
and columns is not sufficient to accommodate the students per class.')

            return


        13, _ =
self.allocation_module.read_excel_and_generate_output(file_path,
students_per_class)

```

```

        pdf = PdfModule.generate_pdf_output(l3, students_per_class, rows,
columns, exam_name, exam_date,session,college_name)

        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog

        filename, _ = QFileDialog.getSaveFileName(self, "Save PDF file",
"AllocatedStudents.pdf", "PDF Files (*.pdf);;All Files (*)", options=options)

        if filename:
            pdf.output(filename)

            QMessageBox.information(self, 'Success', f'PDF file "{filename}" has
been created successfully.')

        except Exception as e:
            QMessageBox.critical(self, 'Error', f'An error occurred: {str(e)}')
            print(f'An error occurred: {str(e)}')

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

if __name__ == '__main__':
    main()

```

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')
```

Allocation_module.py

```
# allocation_module.py
```

```
import pandas as pd
```

```
from PyQt5.QtWidgets import QMessageBox, QFileDialog
```

```
from openpyxl.styles import Alignment
```

```
from openpyxl.utils import get_column_letter
```

```
from openpyxl import Workbook
```

```
from openpyxl.utils.dataframe import dataframe_to_rows
```

```
class AllocationModule:
```

```
    def __init__(self, parent):
```

```
        self.parent = parent
```

```

def read_excel_and_generate_output(self, file_path, students_per_class):
    df = pd.read_excel(file_path, header=None)

    l1 = []
    l2 = []

    for i in range(0, df.shape[1], 2):
        l1.extend(df.iloc[:, i].dropna().tolist())
        l2.extend(df.iloc[:, i + 1].dropna().tolist())

    len_diff = len(l1) - len(l2)

    if len_diff > 0:
        num_to_append = len_diff // 2
        l2.extend(l1[-num_to_append:])
        l1 = l1[:-num_to_append]
    elif len_diff < 0:
        num_to_append = abs(len_diff) // 2
        l1.extend(l2[-num_to_append:])
        l2 = l2[:-num_to_append]

    if len_diff % 2 == 1:
        last_element_l2 = l2.pop()

    l2.reverse()

    l3 = [item for pair in zip(l1, l2) for item in pair]

    if len_diff % 2 == 1:

```

```

        l3.append(last_element_12)

    return l3, students_per_class

def generate_excel_output(self, l3, students_per_class, rows, columns):
    data = {'Student_ID': l3}
    df_output = pd.DataFrame(data)

    students_total = len(l3)
    classes = -(students_total // students_per_class)

    if rows * columns < students_per_class:
        QMessageBox.critical(self.parent, 'Error', 'The specified number of
        rows and columns is not sufficient to accommodate the students per
        class.')
        return

    remaining_slots = students_per_class * classes - students_total
    l3 += ['NONE'] * remaining_slots

    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    filename, _ = QFileDialog.getSaveFileName(self.parent, "Save Excel file",
    "AllocatedStudents.xlsx", "Excel Files (*.xlsx);;All Files (*)",
    options=options)

    if filename:
        if not filename.endswith(".xlsx"):
            filename += ".xlsx"

    wb = Workbook()

```

```

for i in range(classes):
    start_row = i * students_per_class
    end_row = min((i + 1) * students_per_class, students_total)
    class_data = df_output.iloc[start_row:end_row]

    if not class_data.empty:
        none_values = pd.DataFrame({'Student_ID': ['NONE'] *
(students_per_class - len(class_data))})
        class_data = pd.concat([class_data, none_values],
ignore_index=True)

        reshaped_data = class_data.values.reshape((rows,
columns), order='F')
        reshaped_df = pd.DataFrame(reshaped_data)

        reshaped_df.columns = [f'Room {j + 1}' for j in
range(reshaped_df.shape[1])]

    ws = wb.create_sheet(title=f'Class_{i + 1}')

    for col in range(1, reshaped_df.shape[1] + 1):
        column_letter = get_column_letter(col)
        ws.column_dimensions[column_letter].width = 19

        for r in dataframe_to_rows(reshaped_df, index=False,
header=True):
            ws.append(r)

        ws.merge_cells(start_row=1, start_column=1, end_row=1,
end_column=columns)
        cell = ws.cell(row=1, column=1)

```

```

        cell.value = f'Room {i + 1}'
        cell.alignment = Alignment(horizontal='center')

    default_sheet = wb.active
    wb.remove(default_sheet)

    wb.save(filename)

    QMessageBox.information(self.parent, 'Success', f'Excel file
    "{filename}" has been created successfully.')

```

Pdf_module.py

```

from fpdf import FPDF
from PyQt5.QtWidgets import QMessageBox

class PdfModule:
    @staticmethod
    def generate_pdf_output(l3, students_per_class, rows, columns,
exam_name, exam_date, session, college_name):
        hall_supervisor_signature = "Hall Supervisor Signature"
        principal_signature = "Principal Signature"
        class CustomPDF(FPDF):
            def __init__(self, hall_supervisor_signature="Hall Supervisor
Signature", principal_signature="Principal Signature"):
                super().__init__()
                self.hall_supervisor_signature = hall_supervisor_signature
                self.principal_signature = principal_signature

        def footer(self):
            self.set_y(-12)

```

```

        self.set_font("Arial", style='I', size=8)
        self.set_text_color(128, 128, 127) # Gray color
        self.cell(0, 10, "Developed by @Suresh G", 0, 0, 'C')

pdf = CustomPDF(hall_supervisor_signature, principal_signature)
pdf.set_auto_page_break(auto=True, margin=15)
pdf.set_font("Arial", size=7)

students_total = len(l3)
classes = -(students_total // students_per_class)

for i in range(classes):
    start_row = i * students_per_class
    end_row = min((i + 1) * students_per_class, students_total)
    class_data = l3[start_row:end_row]

pdf.add_page(orientation='L') # Set orientation to landscape

    college_name = college_name.upper() # Convert college name to
uppercase

# Add Title: COLLEGE NAME
pdf.set_font("Arial", style='B', size=26)
pdf.cell(270, 10, txt=college_name, ln=True, align='C') # Adjust width
for landscape
pdf.set_font("Arial", size=7)

# Add Title: Exam HALL Seating Arrangement
pdf.set_font("Arial", style='B', size=18)
pdf.cell(270, 10, txt='EXAM HALL SEATING ARRANGEMENT', ln=True,
align='C') # Adjust width for landscape

```

```

pdf.set_font("Arial", size=7)

# Add Examination Name
pdf.set_font("Arial", style='B', size=16)
pdf.cell(270, 10, txt=exam_name, ln=True, align='C') # Adjust width
for landscape

# Add ROOM Title
pdf.set_font("Arial", style='B', size=12)
room_title = f'ROOM {i + 1}'
pdf.cell(270, 10, txt=room_title, ln=True, align='C') # Adjust width for
landscape

# Add Date
x_left = 10 # Set left margin
y_top = 50 # Set top margin
pdf.set_font("Arial", style='B', size=10)
pdf.text(x_left, y_top, f'Date: {exam_date}') # Draw the date

x_left = 255 # Set left margin
y_top = 50 # Set top margin
pdf.set_font("Arial", style='B', size=10)
pdf.text(x_left, y_top, f'Session : {session}') # Add Session

total_tables = -(-len(class_data) // (rows * columns))

for table_index in range(total_tables):
    pdf.ln()

# Add column headers with bold font

```



```

x_center = (pdf.w - 275) / 2 # Calculate the x-coordinate to center
the table horizontally

pdf.set_x(x_center)

pdf.set_font("Arial", style='B', size=8)

for col_header in range(1, columns + 1):

    pdf.cell(33.75, 10, txt=f'Column {col_header}', border=1, ln=0,
align='C') # Adjust width for landscape

    if col_header % 2 == 0 and col_header != columns: # Add space
after every second column except the last one

        pdf.cell(2.5, 10, '', ln=0) # Add a small gap between columns

pdf.set_font("Arial", style='B',size=7)

pdf.ln()


cell_height = 15
cell_width = 28.75

# Move to the center of the page before drawing the table
pdf.set_x(x_center)


serial_number = 1 # Initialize serial number


for row in range(rows): # Iterate over rows first
    pdf.ln()
    for col in range(columns): # Then iterate over columns
        index = table_index * (rows * columns) + row + col * rows
        if index < len(class_data):
            student_id = class_data[index]
            # Add serial number for each cell
            serial_number = col * rows + row + 1
            pdf.cell(5, cell_height, txt=str(serial_number), border=1,
ln=0, align='C')

```

```

        pdf.cell(cell_width, cell_height, student_id, 1, 0, 'C')
        # Add extra space after every two columns
        if col % 2 == 1 and col != columns - 1: # Add space after
every second column except the last one
            pdf.cell(2.5, 0, "", ln=0) # Add a small gap
between columns
        else:
            serial_number = col * rows + row + 1
            pdf.cell(5, cell_height, str(serial_number), 1, 0, 'C') # Empty
cell for serial number
            pdf.cell(cell_width, cell_height, "", 1, 0, 'C') # Empty cell
            if col % 2 == 1 and col != columns - 1: # Add space after
every second column except the last one
                pdf.cell(2.5, 0, "", ln=0) # Add a small gap between
columns

    pdf.ln()

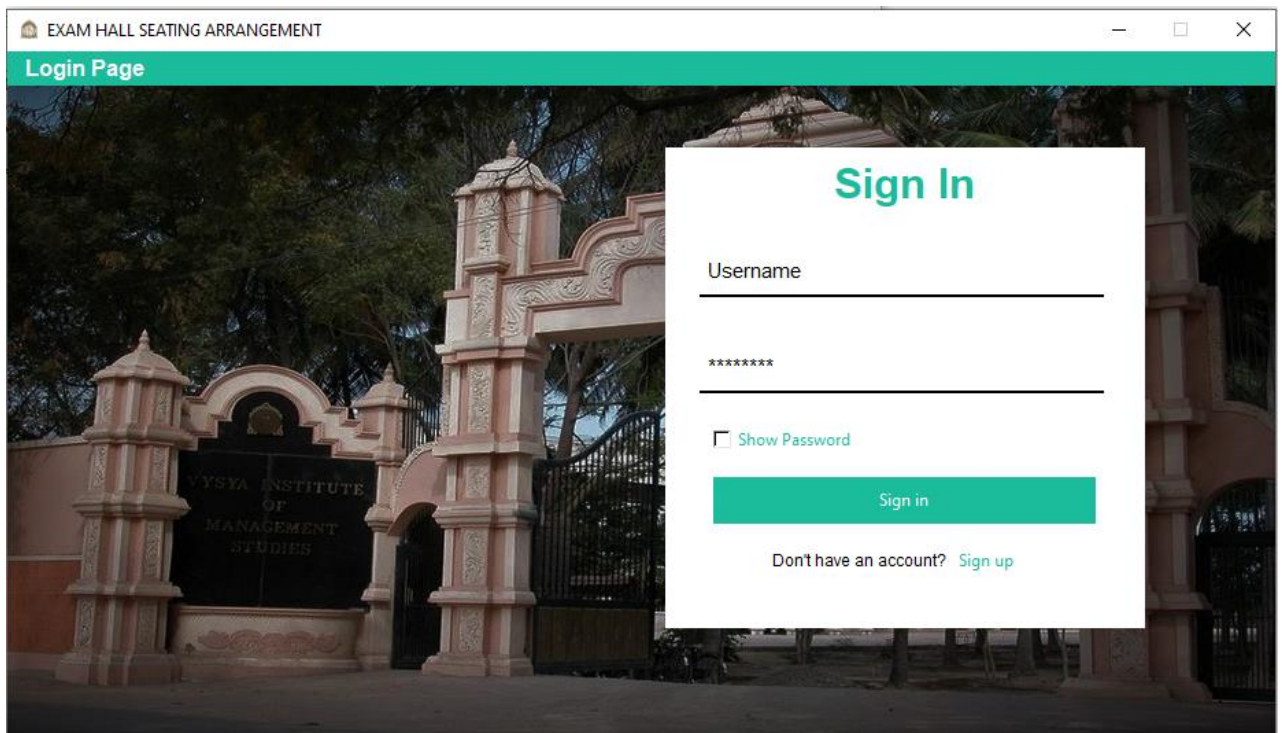
# Add signatures after drawing the tables
x_signature = 10
y_signature = pdf.h - 45 # Ensure y-coordinate is within the page
height
pdf.set_font("Arial", style='B', size=10)
pdf.text(x_signature, y_signature, pdf.hall_supervisor_signature) #
Access class attribute using instance
        x_principal_signature = pdf.w -
pdf.get_string_width(pdf.principal_signature) - 10
        pdf.text(x_principal_signature, y_signature,
pdf.principal_signature) # Access class attribute using instance

return pdf

```

D. SAMPLE INPUT

Login Page



The screenshot shows a web browser window titled "EXAM HALL SEATING ARRANGEMENT". The page has a green header bar with the text "Login Page". The background is a photograph of the entrance gate of Vysya Institute of Management Studies. Overlaid on the right side is a white "Sign In" form. The form contains a "Username" input field, a password input field with masked characters "*****", a checkbox labeled "Show Password", a green "Sign in" button, and a link "Don't have an account? Sign up".

EXAM HALL SEATING ARRANGEMENT

Login Page

Sign In

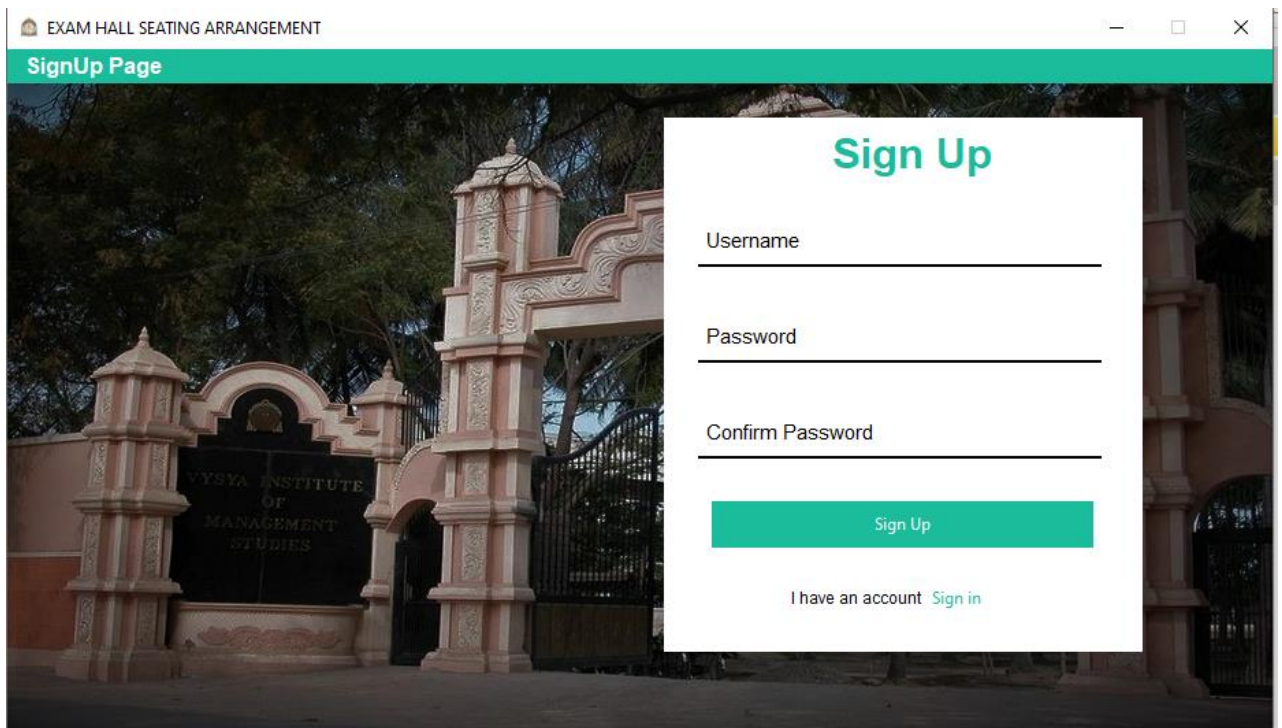
Username

☐ Show Password

Sign in

Don't have an account? [Sign up](#)

SignUp Page



The screenshot shows a web browser window titled "EXAM HALL SEATING ARRANGEMENT". The page has a green header bar with the text "SignUp Page". The background is the same photograph of the entrance gate of Vysya Institute of Management Studies. Overlaid on the right side is a white "Sign Up" form. The form contains "Username", "Password", and "Confirm Password" input fields, a green "Sign Up" button, and a link "I have an account Sign in".

EXAM HALL SEATING ARRANGEMENT

SignUp Page

Sign Up

Username

Password

Confirm Password

Sign Up

I have an account [Sign in](#)

Allocation Module

Classroom Generator

Select Excel file:

Browse

Name of the College:

Vysya College of arts and science

Students per Class:

Rows:

Columns:

Date of Examination:

3/2/2024

Name of Examination:

Periyar University Examination

Session:

FN

Generate Excel Output

Generate PDF Output

Classroom Generator

Select Excel file:

C:/Users/sures/OneDrive/Desktop/StudentList.xlsx

Name of the College:

Vysya College of arts and science

Students per Class:

40

Rows:

5

Columns:

8

Date of Examination:

3/2/2024

Name of Examination:

Periyar University Examination

Session:

FN

Generate Excel Output

Generate PDF Output

Save PDF file

Look in: C:/Users/sures/OneDrive/Desktop/Exam Hall Seating Arrangement

Name	Size	Type	Date Modified
__pycache__		File folder	3/2/2024 3:54 PM
Assessts		File folder	2/12/2023 2:23 AM
AllocatedStudents.pdf	71...KIB	pdf File	3/2/2024 3:15 PM

File name: AllocatedStudents1.pdf

Files of type: PDF Files (*.pdf)

Save

Cancel

EXCEL INPUT

StudentList - Excel										
Suresh G										
File Home Insert Page Layout Formulas Data Review View Help										
Clipboard Font Alignment Number Styles Cell Styles Cells Editing										
J23										
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		
23	C21UG131CAP023	C21UG131BCOCA023	C21UG131CSP023	C21UG131BCO023	C21UG131ENP023	C21UG131CHE023	C21UG131MA023	C21UG131PH023		

E. SAMPLE OUTPUT:

PDF OUTPUT

VYSYA COLLEGE OF ARTS AND SCIENCE

EXAM HALL SEATING ARRANGEMENT

Periyar University Examination

ROOM 1

Date: Saturday, February 24, 2024

Session : FN

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
----------	----------	----------	----------	----------	----------	----------	----------

1	C21UG131CAP001	6	C21UG131BB126	11	C21UG131CAP006	16	C21UG131BB121	21	C21UG131CAP011	26	C21UG131BB116	31	C21UG131CAP016	36	C21UG131BB111
2	C21UG131BB128	7	C21UG131CAP004	12	C21UG131BB123	17	C21UG131CAP009	22	C21UG131BB118	27	C21UG131CAP014	32	C21UG131BB113	37	C21UG131CAP019
3	C21UG131CAP002	8	C21UG131BB125	13	C21UG131CAP007	18	C21UG131BB120	23	C21UG131CAP012	28	C21UG131BB115	33	C21UG131CAP017	38	C21UG131BB110
4	C21UG131BB127	9	C21UG131CAP005	14	C21UG131BB122	19	C21UG131CAP010	24	C21UG131BB117	29	C21UG131CAP015	34	C21UG131BB112	39	C21UG131CAP020
5	C21UG131CAP003	10	C21UG131BB124	15	C21UG131CAP008	20	C21UG131BB119	25	C21UG131CAP013	30	C21UG131BB114	35	C21UG131CAP018	40	C21UG131BB109

Hall Supervisor Signature

Principal Signature

Developed by @Suresh G

VYSYA COLLEGE OF ARTS AND SCIENCE

EXAM HALL SEATING ARRANGEMENT

Periyar University Examination

ROOM 2

Date: Saturday, February 24, 2024

Session : FN

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
----------	----------	----------	----------	----------	----------	----------	----------

1	C21UG131CAP021	6	C21UG131BB106	11	C21UG131CAP026	16	C21UG131BB101	21	C21UG131CAP031	26	C21UG131BB096	31	C21UG131CAP036	36	C21UG131BB091
2	C21UG131BB108	7	C21UG131CAP024	12	C21UG131BB103	17	C21UG131CAP029	22	C21UG131BB098	27	C21UG131CAP034	32	C21UG131BB093	37	C21UG131CAP039
3	C21UG131CAP022	8	C21UG131BB105	13	C21UG131CAP027	18	C21UG131BB100	23	C21UG131CAP032	28	C21UG131BB095	33	C21UG131CAP037	38	C21UG131BB090
4	C21UG131BB107	9	C21UG131CAP025	14	C21UG131BB102	19	C21UG131CAP030	24	C21UG131BB097	29	C21UG131CAP035	34	C21UG131BB092	39	C21UG131CAP040
5	C21UG131CAP023	10	C21UG131BB104	15	C21UG131CAP028	20	C21UG131BB099	25	C21UG131CAP033	30	C21UG131BB094	35	C21UG131CAP038	40	C21UG131BB089

Hall Supervisor Signature

Principal Signature

Developed by @Suresh G

EXCEL OUTPUT

AllocatedStudents - Excel

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 Clc ...

AllocatedStudents - Excel

Room 2

	A	B	C	D	E	F	G	H	I	J
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		
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 Clc ...