Online Exam System

A Python Project

AGENDA

- Project Introduction
- Structure and Feature
- How to Use
- OOP Concepts
- Error Handle
- Conclusion

Introduction to the Project

This project incorporates:

- **Object-Oriented Programming (OOP)**
- **Abstract Methods**
- **File Handling**
- **Basic Python functionalities.**

The **Goal** is to create a user-friendly system for managing exams online.

Online Exam System

Creating and Taking exams, while providing robust data analysis and reporting features.

Problem Statement

Traditional examination systems require manual effort for exam creation, question management, and result tracking. This project aims to:

- Automate the **exam management process**.
- Provide a **secure and efficient** platform for students to take exams.
- Enable **admins** to easily create, manage, and track exams and results.

Structure and Feature

Admin Functionalities

1

User Authentication

Secure login system

Exam Management

Add, delete, and view exams
Manage exam questions

2

3

Student Performance Tracking

Record and display student scores

Structure and Feature

Student Functionalities

1 User Authentication
Secure login system

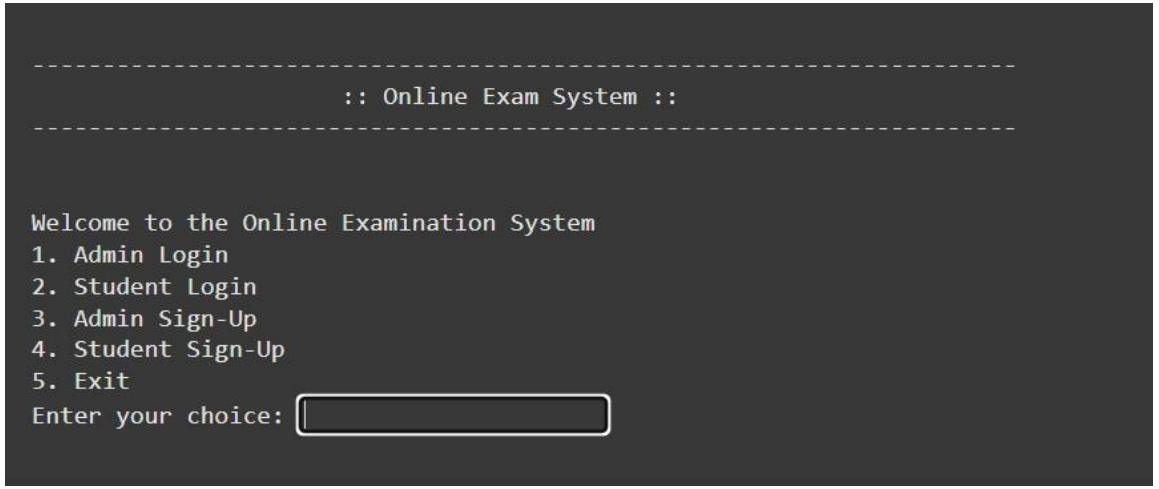
2 Exam Participation
View available exams
Attempt exams interactively

3 Result Viewing
Check exam scores

How to Use

- Terminal based Program
- Run Code Using IDE (Visual Studio Code, Jupyter Notebook etc.)
- Run Code Using Online Platform (Google Colab etc.)

After running the code the user may **sign in** or **signup** as a **student** or as a **admin**



How to Use

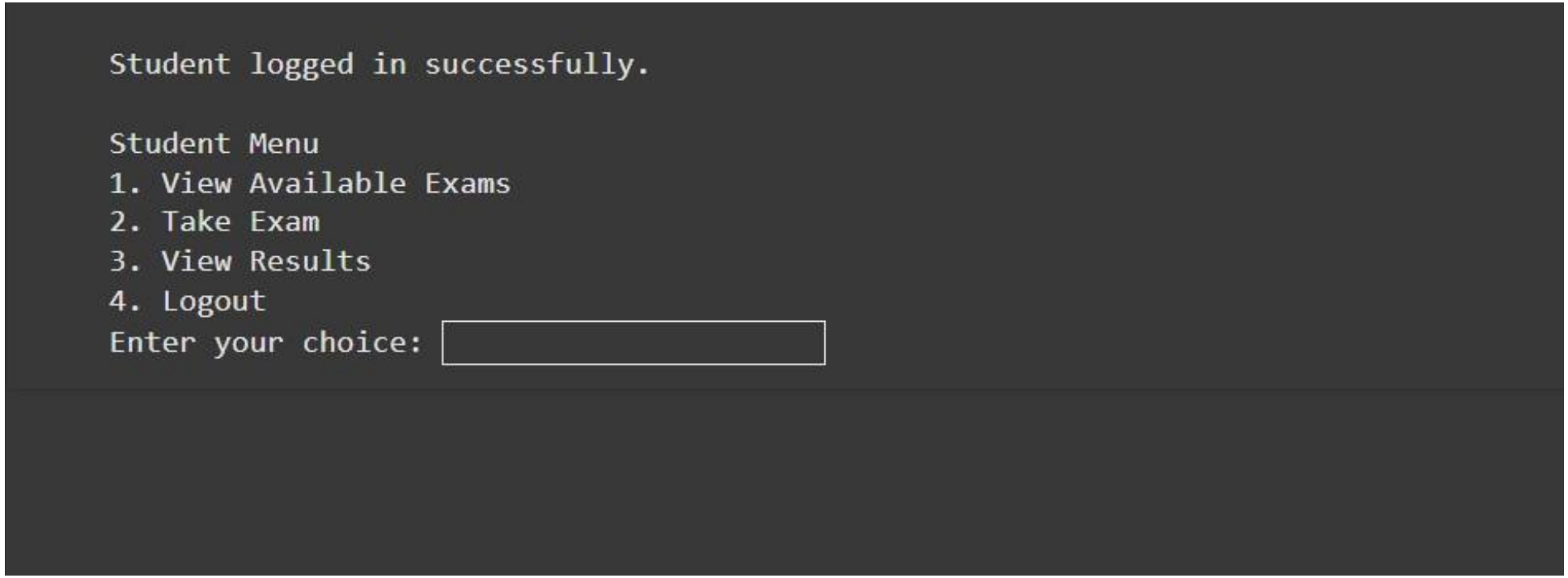
After sign in as a admin, the examiner can access admin Functionality

```
Admin logged in successfully.

Admin Menu
1. Add Exam
2. Delete Exam
3. Add Question to Exam
4. View Exams
5. View Questions in Exam
6. Edit Question in Exam
7. View All Student Results
8. Logout
Enter your choice:
```

How to Use

After sign in as a student, the student can access student's Functionality



How to Use

Let’s have a detailed look how this system works

Google Colab - Online Exam System

OOP Concepts

Abstract Method

```
import os
from abc import ABC, abstractmethod

# Abstract class for shared user behavior
class User(ABC):
    def __init__(self, username, password):
        self._username = username  # Encapsulation: Protecting username
        self._password = password  # Encapsulation: Protecting password

    @abstractmethod
    def display_info(self):
        pass  # Polymorphism: Enforced implementation in subclasses
```

OOP Concepts

Polymorphism

```python
    def display_info(self):
        """Display admin information."""
        print(f"Admin: {self._username}")  # Polymorphism: Different display for Admin


    def display_info(self):
        """Display student information."""
        print(f"Student: {self._username} (Department: {self.department})")  # Polymorphism: Different
                                                                                display for Student
```

OOP Concepts

Inheritance

```python
# Admin class inheriting from User class
class Admin(User):
    def __init__(self, username, password):
        super().__init__(username, password)
        self.exams = {}  # Dictionary to store exam data
        self.student_results = {}  # Dictionary to store student results

    def add_exam(self, exam_name, department_name):
        """Add a new exam to the system."""
        if exam_name in self.exams:
            print("Exam already exists.")
        else:
            self.exams[exam_name] = {"department": department_name, "questions": []}
            print(f"Exam '{exam_name}' under department '{department_name}' added successfully.")
```

OOP Concepts

Encapsulation

```python
class User(ABC):
    def __init__(self, username, password):
        self._username = username  # Encapsulation: Protecting username
        self._password = password  # Encapsulation: Protecting password

    @abstractmethod
    def display_info(self):
        pass  # Polymorphism: Enforced implementation in subclasses

    def login(self, username, password):
        return self._username == username and self._password == password
```

Functions, Conditional Statements, Methods and overall Basics of Python

Error Handle

To give the best user experience I handled error case carefully.

- For **empty** input or **invalid** input

```
    def view_available_exams(self, all_exams):
        """View available exams for the student's department."""
        available_exams = {name: data for name, data in all_exams.items() if data['department'] ==
self.department}

        if not available_exams:
            print("No exams available for your department.")
            return []

        print("Available Exams:")
        for i, (exam_name, exam_data) in enumerate(available_exams.items(), 1):
            print(f"  {i}. {exam_name} (Department: {exam_data['department']},
{len(exam_data['questions'])} questions)")

        return list(available_exams.keys())
```

Problems

- Indentation
- File Handling – Separate txt file for admin data, exam data and student data

Conclusion

By this project

- Easy, user friendly application for conducting exams efficiently
- It's a 'Skeleton Structure' that can be developed later with Graphical User Interface and Database Integration
- I gained clear concept of Python Programming Language

Here is the GitHub link of the code - Online Exam System

THANK YOU