Online Examination GUI

PYTHON PROGRAMMING PROJECT REPORT

By

THODUPUNURI ABHINAV

22EEB0B07

MOHIT VARMA VADAPALLI

22EEB0B08

DEVAGUPTAPU SRIDHAR

22EEB0B09

Department of Electrical and Electronics Engineering

NATIONAL INSTITUTE OF TECHNOLOGY

(An Institute of National Importance)

WARANGAL

TELANGANA, 506004

Nov, 2023

Introduction

The Online Examination GUI is a Python-based project developed using the Kivy framework for creating a graphical user interface (GUI). The purpose of the project is to provide a platform for conducting online exams where users can log in, answer a set of questions, and receive their marks. The project uses Kivy for the frontend, pandas for data manipulation, and integrates with an Excel file for storing user information and exam results.

Kivy Module Integration

The Online Examination project harnesses the Kivy module to streamline the creation of a dynamic and user-friendly graphical interface. Key highlights of Kivy's role in the project include:

Widget-Based Design:

Utilizes Kivy's widget system for constructing interactive elements like TextInput, Button, and custom widgets such as TitleLabel.

Layout Management:

Implements Kivy's layout managers (BoxLayout, GridLayout) to organize and structure widgets within the application window.

Event Handling:

Leverages Kivy's event handling mechanism to respond to user actions, such as button presses for credential verification and exam navigation.

Dynamic UI Updates:

Exploits Kivy's ability to dynamically update the user interface based on real-time user interactions and system events.

Scrollable Interface:

Implements a scrollable interface using Kivy's ScrollView for a smooth user experience, particularly beneficial in handling a list of questions and options.

Graphics and Styling:

Customizes widget appearance by adjusting features like font size, color, and background color using Kivy's styling capabilities.

Screen Management:

Integrates Kivy's ScreenManager to seamlessly transition between the login screen and the exam screen, optimizing user flow.

Kivy's versatility enhances the Online Examination GUI, providing an intuitive and visually appealing interface for users as they navigate through the exam process.

In this project, the pandas module plays a crucial role in handling and manipulating tabular data, particularly in the context of loading and updating user information and exam results stored in an Excel file. Here are some relevant aspects of the pandas module that you can highlight in your project:

Data Loading

You utilize pandas to read data from an Excel file into a DataFrame. This provides a convenient way to handle tabular data, allowing for easy querying and manipulation. For example:

```
try: df = pd.read_excel(self.excel_file) except FileNotFoundError: print("Error: Excel file not found.") except
Exception as e: print("Error loading data from Excel:", e)
```

This code snippet demonstrates how pandas is used to load data from an Excel file into a DataFrame, providing a structured and efficient way to work with the information stored in the file.

Data Manipulation

Once the data is loaded into a DataFrame, you can use pandas to perform various data manipulation tasks. In your project, you update the 'Marks' column based on the user's email after evaluating exam results:

```
df.loc[df['Email'] == self.user_email, 'Marks'] = self.marks df.to_excel(self.excel_file, index=False)
```

Here, pandas allows you to easily locate the row corresponding to a specific user email and update the 'Marks' column with the calculated marks. The DataFrame is then saved back to the Excel file, reflecting the changes made.

Iterating Through Rows

In the part of your code where you load data from the Excel file into the email_rollno_mapping dictionary, you use iterrows() to iterate through the rows of the DataFrame:

```
email_rollno_mapping = {row[email_column]: row[rollno_column] for _, row in df.iterrows()}
```

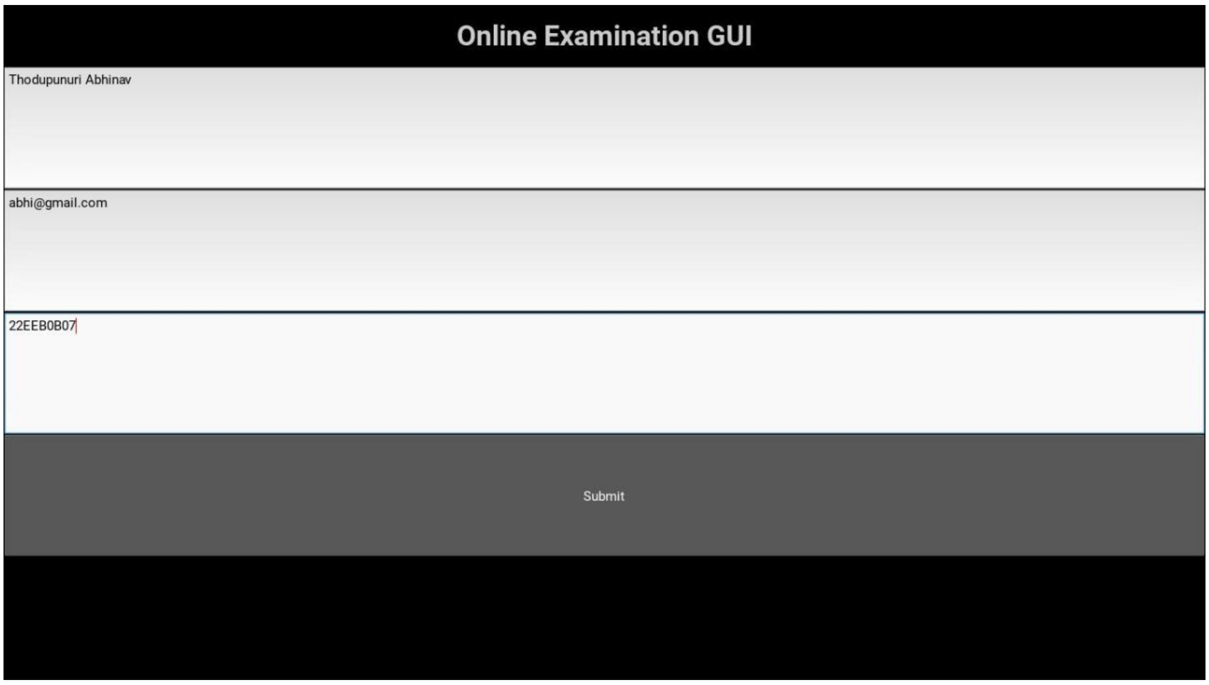
This showcases how pandas simplifies the process of iterating through rows and extracting relevant information, which is particularly useful when building a mapping of email addresses to roll numbers.

Conclusion

In summary, the pandas module in your project serves as a powerful tool for data manipulation and analysis. It streamlines the process of loading, updating, and extracting information from tabular data, contributing to the efficiency and functionality of your online examination system.

Project Overview

The project consists of two main screens: the login screen and the exam screen. The login screen collects user information such as name, email, and roll number. After successful authentication, users move to the exam screen, where they answer a series of questions. The user's answers are then evaluated, and the marks are stored in an Excel sheet.

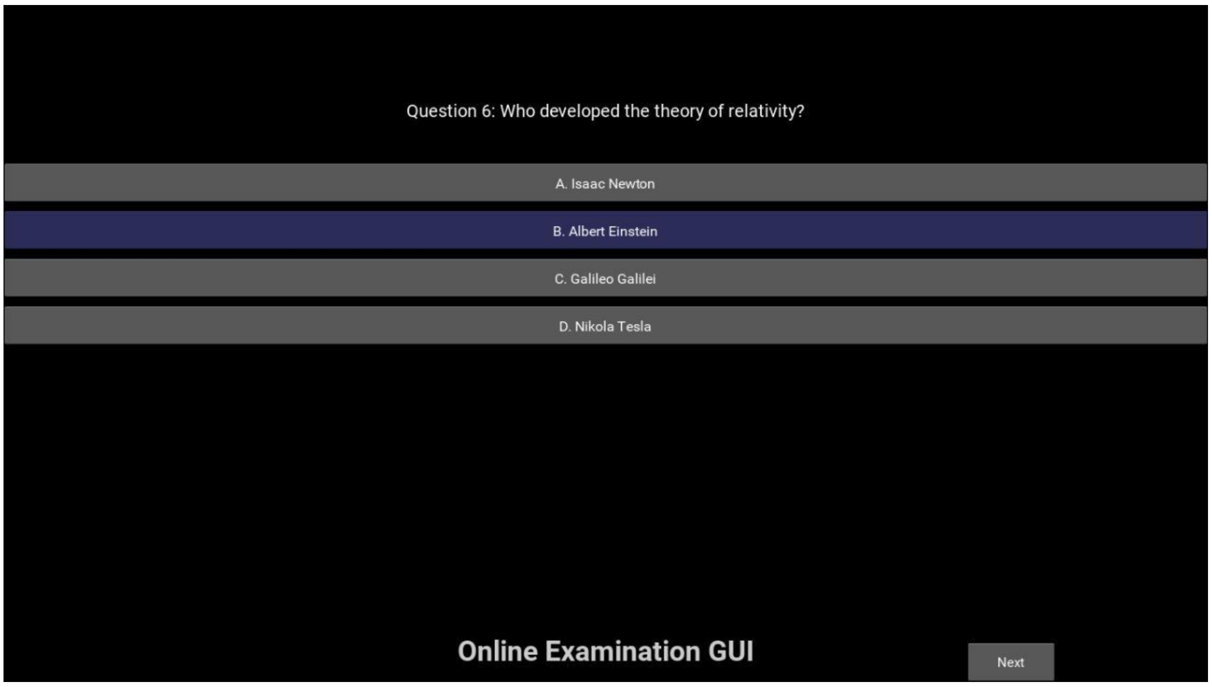


Thodupunuri Abhinav

abhi@gmail.com

22EEB0B07

LOGIN SCREEN


Question 6: Who developed the theory of relativity?

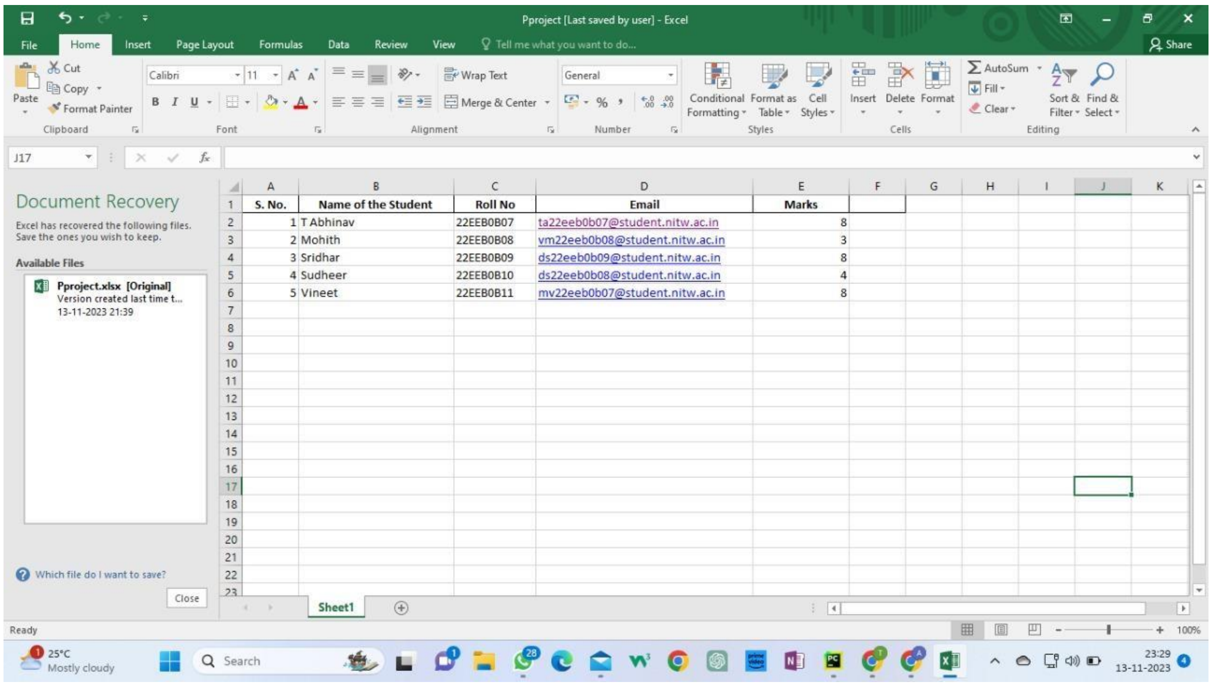
A. Isaac Newton

B. Albert Einstein

C. Galileo Galilei

D. Nikola Tesla

EXAM SCREEN



S. No.	Name of the Student	Roll No	Email	Marks
1	T Abhinav	22EEB0B07	ta22eeb0b07@student.nitw.ac.in	8
2	Mohith	22EEB0B08	vm22eeb0b08@student.nitw.ac.in	3
3	Sridhar	22EEB0B09	ds22eeb0b09@student.nitw.ac.in	8
4	Sudheer	22EEB0B10	ds22eeb0b08@student.nitw.ac.in	4
5	Vineet	22EEB0B11	mv22eeb0b07@student.nitw.ac.in	8

Document Recovery

Excel has recovered the following files. Save the ones you want to keep.

Available Files

Pproject.xlsx [Original]
Version created last time t...
13-11-2023 21:39

<u>**EXCEL SHEET**</u>

System Architecture

The system architecture of the Online Examination GUI project revolves around a client-server model, where the client is the graphical user interface (GUI) developed using the Kivy framework, and the server consists of the backend logic managing user authentication, exam processes, and data storage.

<u>***Components:***</u>

Client (GUI):

The Kivy framework is utilized to create the graphical user interface, providing a cross-platform solution for building interactive applications.

The client side includes the login screen and the exam screen, each serving specific purposes in the user interaction flow.

Server (Backend Logic):

The backend logic handles user authentication, exam processes, and data manipulation.

- <u>User Authentication:</u> During the login phase, the server verifies the entered email and roll number against the stored mapping in the Excel file, granting access only to valid combinations.
- <u>Exam Processes:</u> Once authenticated, the server manages the flow of the exam, loading questions, presenting them to the user, and evaluating their responses.
- <u>Data Manipulation:</u> The backend logic utilizes the pandas library to read and update data in an Excel file. The Excel file serves as a simple database for storing user information and exam results.

Communication:

Communication between the client (GUI) and the server is primarily event-driven.

User actions, such as pressing buttons or submitting answers, trigger events that are processed by the backend logic.

The server communicates with the Excel file to retrieve user information, load questions, and store exam results.

Data Flow:

User inputs (email, roll number) are collected through the GUI.

The client sends authentication requests to the server.

The server validates the credentials against the data in the Excel file.

If authentication is successful, the server allows access to the exam screen.

Exam Flow:

The server loads questions and correct answers from text files.

Questions are presented to the user through the GUI.

User responses are collected and evaluated by the server.

Exam results, including marks, are stored in the Excel file.

Conclusion:

The system architecture effectively manages the flow of user interactions, authentication processes, and exam procedures. While suitable for the project's current scale, future enhancements may involve addressing security concerns, improving scalability, and optimizing performance for more extensive use.

Features and Functionality

- User Authentication: The project authenticates users based on their email and roll number.
- Exam Interface: The exam screen provides a user-friendly interface for answering questions with options.
- Data Storage: User information and exam results are stored in an Excel file, providing a simple and accessible means of data storage.

User Interface Design

The GUI includes text inputs for user information, buttons for navigation, and labels for displaying information. The exam screen features a scrollable layout for questions and options, creating a userfriendly interface.

Database Structure

The project uses an Excel file as a database, with columns such as 'Name,' 'Email,' 'Roll No,' and 'Marks.' The user's marks are updated in the Excel file upon exam submission.

Security Measures

The project includes basic user authentication, ensuring that only users with valid email and roll number combinations can access the exam. However, it is essential to note that using an Excel file as a database may have limitations in terms of security.

Exam Logic and Workflow

The exam logic involves loading questions and correct answers from text files, presenting them to the user, and evaluating their responses. Marks are calculated based on the correctness of the answers, and the results are stored in the Excel file.

```
def load_questions_and_answers(self):     try:         with
open(self.question_file, 'r') as q_file, open(self.key_file,
'r') as key_file:              lines =
q_file.readlines()
            self.correct_answers = key_file.read().splitlines()
             current_question =
None            current_options =
[]
             for line in
lines:
                line = line.strip()
if line:                     if not
current_question:
current_question = line                        else:
current_options.append(line)
                     if  len(current_options)  ==
4:
                        self.questions.append((current_question,
current_options))
                        current_question = None
current_options = []

        self.correct_answers = [answer if answer else '-1' for answer in
self.correct_answers]

self.display_question()
     except FileNotFoundError:
self.add_widget(Label(text="Error: Files not found"))
```

User Authentication and Authorization

User authentication is handled during the login phase, where the entered email and roll number must match the stored data. Authorization is implicit, as only authenticated users can access the exam screen.

The below part of code makes it possible

```
def verify_credentials(self, instance):
    name = self.name_input.text
email  = self.email_input.text
rollno = self.rollno_input.text

    print("Entered Email:", email)
    print("Entered Roll No:", rollno)
     if name and email and
rollno:
        if email in self.email_rollno_mapping and
self.email_rollno_mapping[email] == rollno:             user_info =
f"Name: {name}\nEmail: {email}\nRoll No: {rollno}"
self.output_label.text = user_info

            # Set the user email for updating marks later
            exam_screen = self.screen_manager.get_screen("exam_screen")
exam_screen.user_email = email

            self.exam_button_callback()
             self.screen_manager.current  =  "exam_screen"            else:
self.output_label.text = "Email and roll number do not match."     else:
        self.output_label.text = "Please fill in all fields"
```

Code Structure and Organization

The code is organized into classes for different components, promoting modularity and readability. The Kivy framework is used for GUI elements, and pandas is employed for data manipulation. The classes used in the project are

TitleLabel (Label):

This class represents a custom label for the title of the application.

It inherits from the Kivy Label class and sets attributes such as text, font size, height, boldness, and colour.

InputForm (BoxLayout):

This class is a container for the input form where users enter their name, email, and roll number.

It inherits from the *Kivy BoxLayout* class and includes text input fields, a submit button, and an output label for displaying messages.

The *verify_credentials* method checks if the entered email and roll number match, and if so, it moves to the exam screen.

ExamScreen (Screen):

This class represents the screen where the exam questions are displayed.

It inherits from the Kivy Screen class and includes methods for loading questions, displaying questions, selecting options, and submitting the exam.

The *load_questions_and_answers* method reads questions and correct answers from files.

The *display_question* method dynamically creates widgets to display the current question and options.

The *select_option* method is a callback for selecting an option.

The *submit_exam* method calculates marks and updates them in an Excel file.

UserInfoApp (App):

This is the main application class that ties everything together.

It inherits from the Kivy App class and includes methods for building the application and loading data from an Excel file.

The build method sets up the screen manager, including the login screen and exam screen.

The *load_data_from_excel* method reads user data (email and roll number mapping) from an Excel file.

These classes collectively create a simple online examination GUI using the Kivy framework. Users enter their credentials on the login screen, and upon successful verification, they move to the exam screen to answer questions. The application calculates and displays the user's marks at the end of the exam.

Future Enhancements

Enhanced Security: Consider implementing a more secure database solution and encryption for sensitive data.

Scalability: The system can be enhanced to handle a larger number of questions, users, and more complex exams.

User Feedback: Include features for providing feedback to users on their performance and explanations for correct answers.

Conclusion

The Online Examination GUI project provides a basic yet functional platform for conducting online exams. It demonstrates the integration of Kivy for GUI development, pandas for data manipulation, and Excel for data storage. Further development and enhancements can make it a more robust and feature-rich solution for online assessments.