**FEEDBACK COLLECTION SYSTEM**

**DONE**-**BY**

AKSHAY S

**AGENDA**

* Aim of the project
* Problem Statement
* Overview of the project
* Project Description
* Scope
* Technologies & Methodologies
* Functionalities
* Input Versatility with Error Handling & Exception Handling
* Code Implementation
* Results & Outcomes
* Conclusion

**AIM OF THE PROJECT**

The primary objective of the project is to develop a comprehensive feedback collection system that enhances user engagement and satisfaction. The main goals include:

* Developing a Comprehensive Feedback Collection System
* Streamlining Feedback Collection Process
* Enhancing Stakeholder Communication
* Improving Overall User Satisfaction
* Providing a Centralized Platform for Managing Feedback
  + Feedback Attributes
  + Validation and Error Handling
  + Feedback Storage

**Developing a Comprehensive Feedback Collection System**

Creating a robust system to gather and analyse feedback effectively.

**Streamlining Feedback Collection Process**

Simplifying the process for users to provide feedback seamlessly.

**Enhancing Stakeholder Communication**

Improving communication channels between users and system administrators.

**Improving Overall User Satisfaction**

Implementing feedback to enhance services and user experience.

**Providing a Centralized Platform for Managing Feedback**

**Feedback Attributes**

Name, Email, Topics Covered, Feedback, Performance Rating.

**Validation and Error Handling**

Ensuring data integrity and user input validation.

**Feedback Storage**

Storing feedback securely for analysis and improvement.

**PROBLEM STATEMENT**

In the educational context, traditional methods of managing feedback and administrative tasks often lead to inefficiencies, errors, and communication gaps. The primary challenges include:

* **Manual Feedback Collection:** Education fields (School, College) rely on manual processes for gathering feedback from stakeholders, which can be time-consuming and prone to errors.
* **Communication Breakdowns:** Fragmented communication channels between teachers, students, parents, and administrators hinder effective information sharing and collaboration.
* **Data Handling Issues:** Lack of real-time data and analytics prevents educators and administrators from making informed decisions about student performance, resource allocation, and school operations.
* **Need for Modernization:** There is a pressing need to modernize school management practices by implementing automated systems that streamline administrative tasks, improve communication, and provide actionable insights through data-driven analysis.

**OVERVIEW OF THE PROJECT**

The project aims to develop a comprehensive feedback collection system for educational institutions to address existing challenges in administrative tasks, communication, and data management. Key aspects of the project include:

* **Automated Feedback Collection:** Implementing a user-friendly interface to streamline the process of gathering feedback from stakeholders such as students, parents, and staff.
* **Enhanced Communication Channels:** Establishing centralized communication channels to facilitate effective information sharing among teachers, students, parents, and administrators.
* **Real-Time Data Analytics:** Providing robust analytics and reporting functionalities to empower educators with insights into student performance, resource utilization, and operational efficiency.
* **Centralized Information Management:** Creating a centralized platform for managing student feedback, academic records, attendance, and communication records to improve data accuracy and accessibility.
* **Modernization of Educational Administration:** Modernizing traditional administrative practices by automating routine tasks, minimizing errors, and enhancing overall efficiency within educational institutions.
* **User-Focused Design:** Designing the system with a focus on user experience, ensuring intuitive navigation, clear feedback submission processes, and responsive support for diverse stakeholder needs.
* **Continuous Improvement:** Supporting continuous improvement in educational quality and institutional effectiveness through data-driven decision-making and feedback-driven enhancements.

**PROJECT DESCRIPTION**

The project involves developing a feedback collection system using Python, focusing on core functionalities to streamline the process of gathering feedback, enhancing communication, and improving overall efficiency in managing feedback data. The key functionalities of the system include:

* Automated Feedback Collection
* Subject-Specific Feedback
* Feedback Data Management
* Performance Rating
* Enhanced Communication
* Data Validation and Error Handling
* Feedback Persistence

These functionalities will be implemented using Python programming language along with appropriate data structures and algorithms to ensure efficiency and scalability. For example, feedback data can be managed using dictionaries or lists to store user information, topics covered, feedback text, performance ratings, and staff names. Input validation and error handling will be incorporated to maintain data accuracy and provide a seamless user experience. The system will also feature communication capabilities to notify relevant stakeholders about new feedback submissions and updates.

By incorporating these functionalities, the feedback collection system aims to enhance user engagement, streamline the feedback process, and provide actionable insights for continuous improvement within educational institutions.

**SCOPE**

The Feedback Collection System project aims to develop an efficient, user-friendly platform for collecting, managing, and analysing feedback from various stakeholders in an educational environment. The system targets the needs of students, parents, and staff, ensuring streamlined administrative tasks, enhanced communication, and overall improvement in feedback handling and utilization.

**TECHNOLOGIES & METHODOLOGIES**

* **Programming Language**: Python 3
* **Data Structures:** Lists and dictionaries to store and manage feedback data.
* **Regular Expressions (re):** For email validation to ensure correct email formats.
* **File Handling:** To save feedback data to a text file for long-term storage and retrieval.
* **Exception Handling:** To manage and provide informative error messages for invalid inputs.
* **Modular Design:** Using classes and functions to encapsulate different functionalities and ensure maintainability and scalability of the system.

By leveraging these technologies and methodologies, the Feedback Collection System aims to provide an efficient and reliable solution for managing feedback within educational institutions, enhancing communication, and improving overall operational efficiency.

**FUNCTIONALITIES**

**Automated Feedback Collection**

* Develop a user-friendly interface to facilitate easy submission of feedback from stakeholders, including students, parents, and staff.
* Collect detailed feedback including the name, email, topics covered, feedback text, performance rating, and relevant staff member.

**Subject-Specific Feedback**

* Allow users to select from predefined subjects (Aptitude, Communication, Domain) and provide feedback specific to the chosen subject.
* Display relevant staff members for each subject to ensure targeted and accurate feedback collection.

**Feedback Data Management**

* Use data structures like lists and dictionaries to store and manage feedback data efficiently.
* Ensure easy retrieval, update, and iteration over feedback records for analysis and reporting purposes.

**Performance Rating**

* Enable users to rate performance on a scale of 1 to 5 stars, ensuring a quantitative measure of performance.
* Validate the performance rating to ensure it falls within the specified range and is accurately recorded.

**Enhanced Communication**

* Provide a centralized platform for managing and viewing feedback data, facilitating effective communication among stakeholders.
* Integrate with email or messaging services to notify relevant stakeholders about new feedback submissions and updates.

**Data Validation and Error Handling**

* Implement robust input validation to ensure data integrity, including checks for valid email formats and alphabetic-only names.
* Handle exceptions gracefully, providing informative error messages and prompts for correcting input errors.

**Feedback Persistence**

* Save feedback data to a file for long-term storage and retrieval, ensuring feedback records are well-organized and easily accessible.
* Implement mechanisms to append new feedback to existing records, maintaining a comprehensive history of feedback submissions.

**User-Friendly Interaction**

* Design the user interface to be intuitive and easy to navigate, ensuring a smooth user experience.
* Provide clear instructions and feedback prompts to guide users through the feedback submission process.

**Staff Selection**

* Display a list of available staff members based on the selected subject to ensure that feedback is directed to the appropriate individual.
* Allow users to select the relevant staff member from a predefined list for each subject.

**Review and Analysis**

* Implement functionalities for administrators to review and analyse collected feedback, gaining insights into areas of improvement and performance.
* Generate reports and summaries based on feedback data to inform decision-making and enhance educational outcomes.

**INPUT VERSATILITY WITH ERROR HANDLING & EXCEPTION HANDLING**

The Feedback Collection System is designed to handle diverse types of input data while incorporating robust error handling and exception mechanisms to maintain data integrity and provide a smooth user experience. Here’s a summary of how the system manages various inputs and ensures accurate data collection:

**Name Validation**

* **Requirement:** The name must contain only alphabetic characters.
* **Implementation:** The program continuously prompts the user until a valid alphabetic name is entered.

**Email Validation**

* **Requirement:** The email must be in a valid format.
* **Implementation:** The program uses a regular expression to validate the email format and prompts the user until a valid email is provided.

**Performance Rating**

* **Requirement:** The performance rating must be between 1 and 5 stars.
* **Implementation:** The program ensures the entered rating is within the valid range and prompts for correct input if an invalid rating is given.

**Subject and Staff Selection**

* **Requirement:** Users must select a valid subject and the corresponding staff member.
* **Implementation:** The program presents a list of subjects and associated staff members, ensuring valid selections through user input validation.

**Comprehensive Input Validation**

* **Requirement:** All fields must be filled with valid data.
* **Implementation:** The program checks all input fields for validity and completeness, raising appropriate errors for any missing or invalid data.

**Handling Additional Feedback Submission**

* **Requirement:** Handle user input for additional feedback submission with clear options (yes/no).
* **Implementation:** The program repeatedly prompts the user to confirm if they want to submit more feedback, accepting only 'yes' or 'no' as valid inputs.

These features collectively ensure that the Feedback Collection System is capable of efficiently handling various types of input while preventing errors and maintaining the integrity of the collected data. The robust error handling mechanisms provide clear guidance to users, making the feedback process user-friendly and reliable.

**CODE IMPLEMENTATION**

To implement the project, we utilize basic Python programming concepts to create a modular and maintainable codebase. We leverage key algorithms and data structures to efficiently manage data processing tasks. The code is organized into modules to ensure modularity and readability, with extensive documentation provided for clarity and future development.

**Code Snippet**

import re

class Feedback:

def \_\_init\_\_(self, name, email, topics\_covered, feedback, performance, staff\_name):

self.name = name

self.email = email

self.topics\_covered = topics\_covered

self.feedback = feedback

self.performance = performance

self.staff\_name = staff\_name

def \_\_str\_\_(self):

return (f"Name: {self.name}\nEmail: {self.email}\nTopics Covered: {self.topics\_covered}\n"

f"Feedback: {self.feedback}\nPerformance: {self.performance} stars\n"

f"Staff: {self.staff\_name}\n{'-'\*40}")

def validate\_feedback(func):

def wrapper(\*args, \*\*kwargs):

feedback\_instance = func(\*args, \*\*kwargs)

if not feedback\_instance.name or not feedback\_instance.email or not feedback\_instance.topics\_covered or not feedback\_instance.feedback or feedback\_instance.performance is None or not feedback\_instance.staff\_name:

raise ValueError("All fields are required!")

if feedback\_instance.performance < 1 or feedback\_instance.performance > 5:

raise ValueError("Performance rating must be between 1 and 5 stars!")

if not re.match(r"[^@]+@[^@]+\.[^@]+", feedback\_instance.email):

raise ValueError("Invalid email format!")

return feedback\_instance

return wrapper

class FeedbackCollection:

def \_\_init\_\_(self):

self.feedback\_list = []

def add\_feedback(self, feedback):

self.feedback\_list.append(feedback)

def \_\_iter\_\_(self):

return iter(self.feedback\_list)

class FeedbackIterator:

def \_\_init\_\_(self, feedback\_list):

self.\_feedback\_list = feedback\_list

self.\_index = 0

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.\_index < len(self.\_feedback\_list):

result = self.\_feedback\_list[self.\_index]

self.\_index += 1

return result

else:

raise StopIteration

def display\_staff\_names(staff\_names):

print("Available Staff:")

for index, name in enumerate(staff\_names, start=1):

print(f"{index}. {name}")

def create\_feedback():

subjects = ["Aptitude", "Communication", "Domain"]

staff\_dict = {

"Aptitude": ["Thanubalini", "Swathi", "Suresh"],

"Communication": ["Roshini", "Mathu", "Deepika"],

"Domain": ["Sasikala","Diana","Harsha"]

}

while True:

print("Select Subject:")

for index, subject in enumerate(subjects, start=1):

print(f"{index}. {subject}")

subject\_index = int(input("Choose subject by number: ")) - 1

if subject\_index < 0 or subject\_index >= len(subjects):

print("Invalid subject selection! Please choose again.")

continue

subject = subjects[subject\_index]

display\_staff\_names(staff\_dict[subject])

staff\_index = int(input("Select Staff by number: ")) - 1

if staff\_index < 0 or staff\_index >= len(staff\_dict[subject]):

raise ValueError("Invalid staff selection!")

staff\_name = staff\_dict[subject][staff\_index]

break

while True:

name = input("Your Name: ")

if name.isalpha():

break

else:

print("Name must contain only alphabetic characters! Please enter a valid name.")

while True:

email = input("Your Email: ")

if re.match(r"[^@]+@[^@]+\.[^@]+", email):

break

else:

print("Invalid email format! Please enter a valid email address.")

topics\_covered = input("Topics Covered Today: ")

feedback = input("Your Feedback: ")

while True:

try:

performance = int(input("Rate Performance (1-5): "))

if 1 <= performance <= 5:

break

else:

print("Performance rating must be between 1 and 5 stars! Please enter a valid rating.")

except ValueError:

print("Invalid input! Please enter a number between 1 and 5.")

return Feedback(name, email, topics\_covered, feedback, performance, staff\_name)

def save\_feedback(feedback\_collection):

with open("feedback.txt", "a") as file:

for feedback in feedback\_collection:

file.write(str(feedback) + "\n")

def main():

feedback\_collection = FeedbackCollection()

while True:

try:

feedback = create\_feedback()

feedback\_collection.add\_feedback(feedback)

except ValueError as e:

print(e)

while True:

another = input("Do you want to add another feedback? (yes/no): ").strip().lower()

if another in ['yes', 'no']:

break

else:

print("Invalid input! Please enter 'yes' or 'no'.")

if another != 'yes':

break

save\_feedback(feedback\_collection)

print("Thank you for your feedback!")

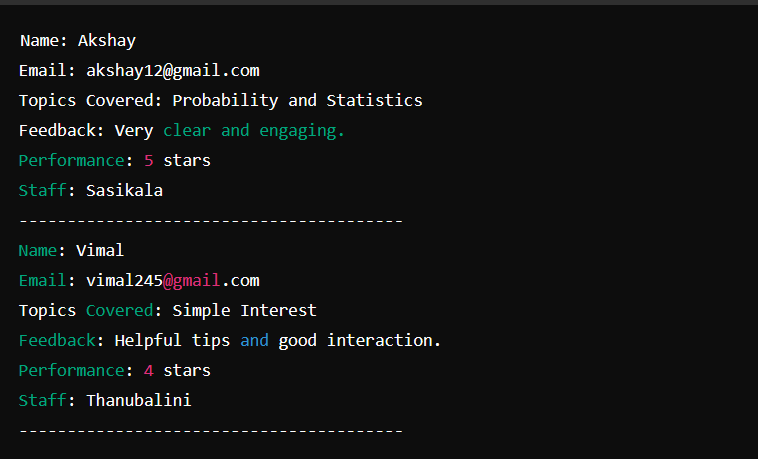
if \_\_name\_\_ == "\_\_main\_\_":

main()

This program demonstrates the core functionalities of a feedback collection system, including user input validation, feedback storage, and modular code organization.

**RESULTS & OUTCOMES**

* Efficient Data Collection
* Input Validation and Error Handling
* User-Friendly Interaction
* Flexible Staff Selection
* Comprehensive Feedback Records
* Persistent Storage
* Iterative Feedback Collection
* Scalability



**CONCLUSION**

The developed feedback collection system effectively streamlines the process of gathering, managing, and analysing feedback from students. The program ensures data integrity through robust validation and exception handling mechanisms. By categorizing feedback based on different subjects and respective staff members, it provides a clear and organized way to track performance and areas for improvement.