M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur, Tamilnadu.

A Minor Project Report
On

REDESIGNING A PUBLIC SQUARE

Under the guidance of

Ms. SRIMATHI V

CORPORATE TRAINER-IBM

Submitted by

1.R MONISHA - 927623BAD064
2.S NANDHINI -927623BAD066
3.M MUKESH -927623BAD065

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)
KARUR – 639113

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

INTRODUCTION

PROBLEM STATEMENT:

Develop a Python program that automatically sends birthday wishes via email to individuals based on their birth dates. The program should retrieve the birth dates from a database or a CSV file, generate personalized messages, and send them out at the appropriate times. Additionally, the program should be able to handle error cases such as incorrect email addresses or missing birth dates.

OBJECTIVES:

The objective for automatic birthday mail sending in Python could be to create a program that automatically sends out birthday emails to a list of contacts on their birthdays. This involves tasks like reading birthdays from a database or file, formatting and composing personalized emails, and using a library like smtplib to send the emails automatically.

1.Automate Communication: Enable the automated delivery of personalized birthday wishes to individuals, reducing the need for manual intervention and ensuring timely greetings.

2. Personalization: Customize each email with the recipient's name and possibly other relevant details to create a more meaningful and engaging message.

3. Efficiency: Streamline the process of sending birthday greetings by leveraging Python's automation capabilities, thereby saving time and effort for the sender.

4. Accuracy: Ensure that birthday emails are sent accurately and reliably, with minimal errors or discrepancies in the recipient list and message content.

5. Scalability: Design the system to handle a growing number of recipients and birthdays efficiently, accommodating potential increases in workload without significant performance degradation.

6. Error Handling: Implement robust error handling mechanisms to address issues

such as invalid email addresses, failed deliveries, or other unexpected errors, ensuring the reliability of the system.

7. Security: Safeguard sensitive information such as email credentials and personal data, adhering to best practices for data protection and privacy.

8. Flexibility: Provide options for customization and configuration, allowing users to adjust settings such as email templates, delivery schedules, and data sources according to their preferences and requirements.

9. Monitoring and Reporting: Incorporate features for monitoring the system's performance and generating reports on email delivery status, allowing users to track the effectiveness of their birthday greetings campaign.

CHAPTER-2

EXISTING AND PROPOSED SYSTEM

EXISTING SYSTEM:

Creating a system for automatically sending birthday emails in Python involves several steps:

1.Storing Birthdays: You need a way to store the birthdays and corresponding email addresses. This can be done using a CSV file, a database, or any other suitable storage method.

2.Reading Birthdays: Your Python script will read the stored birthdays and check if today matches any of the birthdays.

3. Sending Emails: If there are any matches, the script will send an email using an email service (like Gmail, SMTP, etc.).

1. Create a CSV File for Birthdays:

Create a CSV file named birthdays.csv with the following structure:

csv
Name, Email, Birthday
John Doe,john@example.com,2000-05-15
Jane Smith,jane@example.com,1995-07-23

PROPOSED SYSTEM:

To create a system for automatically sending birthday emails in Python, you need to follow several steps. First, store user data, including names, emails, and birthdays, in a CSV file. Then, write a Python script that reads this data using the Pandas library and checks if any birthdays match the current date using the datetime module. For sending emails, use the smtplib library to configure your email server and send personalized birthday messages. Ensure to replace placeholders with your actual email credentials and SMTP server details. Finally, automate the script to run daily using a task scheduler like Task Scheduler on Windows or cron jobs on Linux/Mac. This setup ensures that birthday emails are sent out automatically each day.

1. Setup and Requirements

- Python
- SMTP library (smtplib)
- Pandas for reading data (optional)
- Datetime for date handling

2. Steps to Implement

1. Read User Data: Store user data (name, email, birthday) in a CSV or database.
2. Check for Today's Birthdays: Compare today's date with stored birthdays.
3. Send Emails: Use smtplib to send emails to users whose birthday is today.

CHAPTER-3

METHODOLOGY

METHODOLOGY:

Creating an automated system for sending birthday emails in Python involves several steps. Here is a detailed methodology to guide you through the process:

1. Data Storage
Objective: Store user information including names, email addresses, and birthdates in a structured format.

Implementation:
- Use a CSV file or a database to store user data. A CSV file is a simple and effective way to manage this data. Below is an example of the CSV structure

2. Reading User Data
Objective: Read and process user data to check for birthdays.

Implementation:
- Utilize the Pandas library to read data from the CSV file.
- Example code to read the CSV file:
  python
  import pandas as pd

  # Load the user data from a CSV file
  users = pd.read_csv('users.csv')

3. Check for Today's Birthdays
Objective: Compare the current date with stored birthdates to identify birthdays.

Implementation:
- Use the datetime module to get the current date.
- Compare the current date with each user's birthday.
- Example code to get today's date and check for birthdays:
  python
  from datetime import datetime

  # Get today's date

```
today = datetime.now().strftime('%Y-%m-%d')
```

4. Sending Emails
Objective: Send personalized birthday emails to users whose birthday is today.

Implementation:
- Use the smtplib library to configure the SMTP server and send emails.
- Create a function to send emails.

5. Automating the Script
Objective: Schedule the script to run daily to check for birthdays and send emails automatically.

Implementation:
- Use a task scheduler to run the script daily.
- Windows: Use Task Scheduler.

Security Considerations:
- Credentials Management: Store email credentials securely using environment variables or a secure vault, rather than hardcoding them in the script.

- Error Handling: Implement robust error handling to manage issues such as invalid email addresses or network errors.

CHAPTER-4

RESULT & ANALYSIS

RESULT & ANALYSIS:

Upon implementing the automatic birthday email sending system in Python, we can expect the following results:

1. Daily Email Dispatch: The script will run daily, checking the user data for any birthdays matching the current date and sending personalized birthday emails to those users.

2. Email Delivery: Users whose birthdays are today will receive an email wishing them a happy birthday. The emails will be personalized with the recipient's name.

3. Logging and Reporting: Each email sent will be logged to the console, providing a record of successful and failed email dispatches.

Accuracy and Reliability:
- Accuracy: The system accurately identifies users whose birthdays match the current date and sends emails to those users.
- Reliability: The script is scheduled to run daily using a task scheduler or cron job, ensuring consistent execution.

2. Performance:
- Efficiency: The script efficiently reads the CSV file, checks for birthdays, and sends emails. Performance is dependent on the size of the user data; for small to moderate datasets, the script performs well.
- Scalability: For larger datasets, optimizations may be required. For instance, using a database instead of a CSV file for faster data retrieval and indexing.

3. Error Handling:
- Common Issues:
 - SMTP Authentication Errors: Incorrect email credentials or configuration can lead to authentication failures.
 -Invalid Email Addresses: Malformed or invalid email addresses in the user data can cause email sending to fail.
- Solutions: Implement robust error handling to catch and log exceptions, providing insights into issues that need resolution.

4. Security Considerations:
- Credential Management: Avoid hardcoding email credentials in the script. Use environment variables or a secure credentials management system.
- Data Protection: Ensure user data (email addresses, names, birthdays) is stored and transmitted securely to protect user privacy.

5. User Experience:
- Personalization: Personalizing the email content with the recipient's name enhances user experience and engagement.
- Timeliness: Sending emails on the exact birthday demonstrates attentiveness and can

positively impact user satisfaction.

CHAPTER-5

CONCLUSION

CONCLUSION:

The implementation of an automatic birthday email sending system in Python is an effective solution for maintaining and enhancing user engagement through personalized communication. By leveraging libraries such as Pandas for data handling, datetime for date comparisons, and smtplib for email dispatch, the system ensures timely and accurate delivery of birthday greetings.

This automation not only saves time but also provides a reliable way to ensure no user's birthday is overlooked. The system's robustness can be further enhanced through proper error handling, security measures for credential management, and scalability improvements by integrating with a relational database.

Overall, the automatic birthday email sending system offers a significant improvement in user experience, demonstrating attentiveness and appreciation. Future enhancements such as email content customization, advanced analytics, and user preference management can further elevate the effectiveness and user satisfaction derived from this system.

\