MAIL AUTOMATION USING DJANGO FRAMEWORK

Sona S[1], Thashmigaa E M[2], Menaha C[3]

[1] *Sona S Student, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India*
[2] *Thashmigaa E M Student, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India*
[3] *Menaha C Student, Biotechnology, Bannari Amman Institute of Technology, Tamil Nadu, India*

ABSTRACT

In today's fast-paced world, organizations and individuals are constantly seeking ways to optimize their workflow and improve efficiency. One area that often demands significant time and resources is mail management. Traditional methods of handling mail can be laborious and prone to errors. However, with the advent of mail automation, a revolutionary approach has emerged, transforming the way we handle mail and streamlining communication processes. Mail automation refers to the use of technology to automate various aspects of mail management. Advanced algorithms and optical character recognition (OCR) technology can quickly analyze and categorize mail based on predefined rules, ensuring that it reaches the appropriate recipient in a timely manner. This automation process not only saves time but also minimizes the risk of misplaced or lost mail. Moreover, mail automation enables rapid response management. Incoming mail can be digitized and categorized, making it easier to track and assign tasks to the relevant personnel. Automated notifications and reminders can be sent to responsible individuals, ensuring timely responses and avoiding unnecessary delays. This streamlined approach allows organizations to improve customer service, resolve issues promptly, and maintain a high level of professionalism in their communication. Integrating automation systems with existing mail infrastructure can be complex, requiring careful planning and coordination. Organizations must adopt robust security measures, such as encryption and access controls, to protect the integrity and confidentiality of mail data. In businesses, it enables seamless mail room operations, reducing manual labor and providing real-time tracking of mail items. Even individuals can benefit from mail automation by simplifying personal mail management and minimizing the risk of missing important correspondence. Mail automation offers a transformative solution for managing mail efficiently. With its ability to automate sorting, routing, scanning, and response management, organizations and individuals can optimize their workflow, reduce errors, and enhance communication processes. By embracing mail automation technologies, we can usher in a new era of streamlined and efficient mail management, facilitating increased productivity and improved customer satisfaction.

Keyword*: Optical character recognition, Digitized*

1. INTRODUCTION

Django Framework: Model-View-Controller (MVC) is an architectural design pattern that is used by the Django framework, a high-level Python web framework. It has integrated help for managing email as well as other web development chores.

Email Features in Django: Simple Mail Transfer Protocol (SMTP) email backend: Django includes a built-in SMTP email backend that enables you to send emails using an SMTP server. In the settings.py file of your Django project, you can customize the SMTP settings.

Email Templates: Using its templating mechanism, Django enables you to develop reusable email templates. This makes it simple to maintain standardized email content and formatting.

Email Sending: To send emails from your application, you can use Django's send mail() function or more sophisticated tools like the Email Message class.

Scheduled Email Sending: To plan and automate the sending of emails at specified times or in reaction to specific events, use Django's built-in tasks or third-party libraries like Celery.

Email logging: Django has the ability to record sent emails, which is useful for monitoring email activity and troubleshooting problems.

The necessity for dependable and effective email communication within web applications is the context for building mail automation using the Django framework. Web applications and their users typically communicate via email, and automating email-related processes has the following benefits:

Guarantees consistency in the format and branding of emails sent from the application, improving the user experience and professionalism.
User Engagement: Automated emails can be used to engage users at different points in their connection with the programme, such as when a user signs up for the first time, when an order is confirmed, or when a user becomes inactive.
Security: Including email verification and password-reset options improves user account security by lowering the possibility of unauthorized access.
Scalability: Manual email management becomes impractical as a programme expands. The solution can scale without considerably increasing the administrative overhead by automating email procedures.
Personalization: Mail automation can make it possible to create emails with content that is specifically suited to each user's behaviors and preferences.
Communication that is Scheduled: Some emails, like newsletters or reminders, must be delivered at specific times or intervals. Scheduling and distributing these emails are made easier via automation.
In general, the motivation for mail automation with Django stems from the need to build secure, dependable, and effective email communication processes within online applications, improving user experience and streamlining administrative work for developers and administrators.

2. LITERATURE SURVEY

S.NO	**Journal Paper Title with Author**	**Works carried out**	**Information gathered relevant to your project**
1.	E-Mail Assistant – Automation of E-Mail Handling and Management using Robotic Process AutomationArpit Khare;Sudhakar Singh;Richa Mishra;Shiv Prakash;Pratibha Dixit 2022 International Conference on Decision Aid Sciences and Applications (DASA)	Robotics process	Mail automation
2.	Secure official document mail systems for office automation Chung-Huang Yang;So-Lin Yen;Hwang David Liu;Kuei Liu;Bor-Shenn Jeng;Kung-Yao Chan;Min-Shin Chang;Yu-Ling Cheng;Jo-Ling Liang;Don-Min Shien	Network and automation	Security
3.	Research of an E-mail forensic and analysis system based on visualization Fanlin Meng;Shunxiang Wu;Junbin Yang;Genzhen Yu 2009 Asia-Pacific Conference on	Data visualization	Mail automation

	Computational Intelligence and Industrial Applications (PACIIA)		
4	Voice Automation Mail System for Visually Impaired D Malathi;S Gopika;Dev Awasthi;Dorathi Jayaseeli 2023 International Conference on Networking and Communications (ICNWC)	Natural Language Processing	Mail automation with voice feature
5	"Mailroom - An Open Source Mail Automation Platform" (GitHub Repository): URL: https://github.com/18F/mailroom	This project aims to automate email processing and archiving using Django and Elasticsearch.	Mail automation, Django framework

3. METHODOLOGY

3.1 PROBLEM IDENTIFICATION

In today's fast-paced business environment, effective communication is paramount for the success of any organization. Email communication, in particular, is a cornerstone of modern business operations. However, managing emails efficiently can be a daunting task, often leading to productivity bottlenecks, missed opportunities, and communication breakdowns. To address these challenges, we propose the development of a comprehensive mail automation system using the Django framework.

3.2 PROPOSED WORK

1. Setting up Django Configure the database settings to define the database engine, name, user, and password.
   To manage CSS, JavaScript, and other static components, configure static file handling.
   • Set up email settings, such as default sender information and SMTP server information.
2. Models of databases
   • Create tables for users, email templates, and other data using Django's Object-Relational Mapping (ORM).
   • Create connections between models, such as foreign keys for information about users.
3. Email Templates
   • Use the template tags and filters provided by Django to generate dynamic email content.
   • For greater email client compatibility, think about creating email templates in both HTML and plain text.
4. User Verification and Registration
   • Implement tokens or links that are specific to each user to implement email verification.
   • Configure user registration and user verification views and URL patterns.
5. Password Reset
   • To increase security, make sure password reset links have a defined amount of time before expiring.
   • Use user authentication when requesting password resets.
6. Pre-planned emails
   • Use a scheduler to automate the sending of scheduled emails, such as Django Celery.
   • Ensure that sent emails are reliably stored and processed.
7. Integration with Email Services - Based on business demands, budget, and scalability requirements, select an email delivery service. - Configure the proper API keys, authentication, and configurations for the email service.
8. Logging and Monitoring - Record email sending status, errors, and exceptions using Django's built-in logging framework. - Use monitoring tools like log aggregation to proactively track email-related issues.
9. Testing and Quality Assurance - Create test cases for various email workflows, taking into account both successful and unsuccessful circumstances. - For automated testing, take into account employing testing libraries like Django's TestCase.
10. User input and Optimisation - Create channels for users to give input on email correspondence. - Review user comments and data analytics frequently to improve email strategy and content.
11. Cross-Platform Compatibility - To guarantee uniform display and operation, test email templates on multiple desktop and mobile platforms as well as well-known email clients (such as Outlook, Gmail, and Apple Mail).

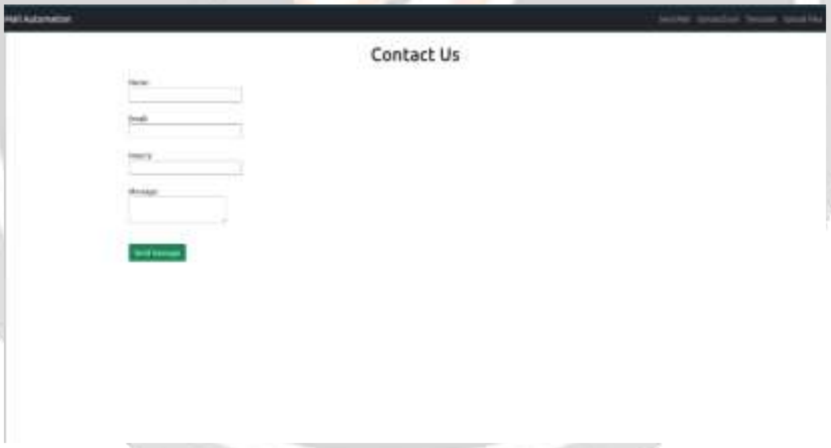
This methodology offers a thorough strategy for using the Django framework to develop mail automation, covering a range of topics from design to implementation and ongoing maintenance. Throughout the project lifecycle, it places a focus on best practises, testing, compliance, and user engagement optimisation.

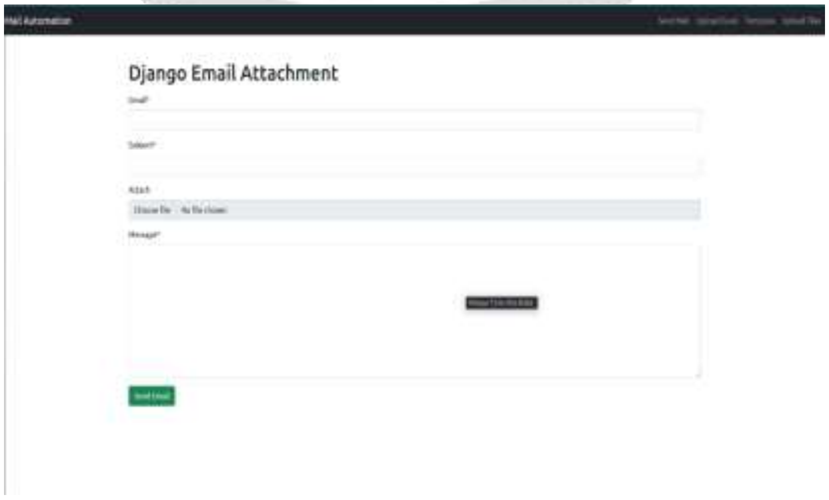
3.3. PROPOSED WORK MODULES

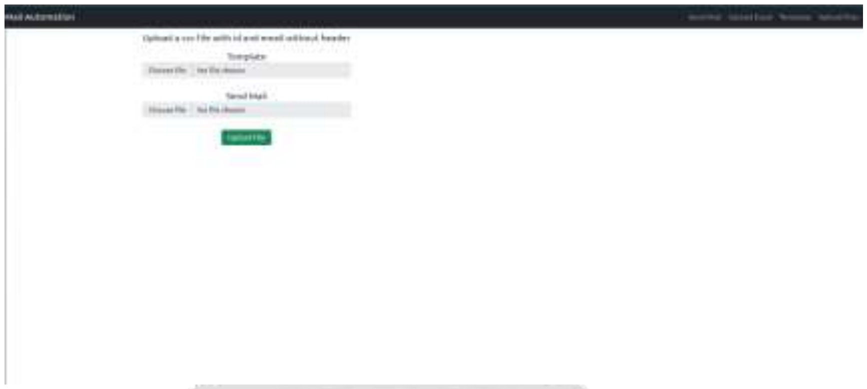
Users interact with the system through the user interface (UI), which is the front-end component of the programme. Users can configure and track their email automation tasks on a web-based dashboard.
API Layer: Django has a robust API structure that enables communication between the system and external services. The management of API requests from diverse customers falls under the purview of this layer.
Email Service Integration: Integrating with an email service provider, such as SMTP for sending and receiving emails, is necessary for the system. The system's ability to communicate with the email infrastructure depends on these integrations.
Database Layer: Django offers a layer of abstraction for working with databases. For tasks, contacts, email templates, automation rules, and other pertinent data, we created database models.
Users must register and log in to access the system. This process is known as authentication and authorization. To protect the application, appropriate processes for authentication and authorization must be in place.

4. CONCLUSIONS

We assess how well email workflows, such as user registration, password reset, transactional emails, and marketing campaigns, have been automated. We also evaluate the efficiency improvements and resource optimization brought about by automation. To determine whether marketing emails are effective, open and click-through rates and other user engagement indicators are examined. We go over the outcomes of the cross-platform testing, focusing on any problems that were encountered and how they were fixed. Finally, we draw attention to the application of security controls and legal compliance.






Mail Automation
Upload a csv file with id and email without header
Template
Choose File No file chosen
Send Mail
Choose File No file chosen
Upload File

6. REFERENCES

[1] E-Mail Assistant – Automation of E-Mail Handling and Management using Robotic Process Automation

[2] Arpit Khare;Sudhakar Singh;Richa Mishra;Shiv Prakash;Pratibha Dixit

[3] 2022 International Conference on Decision Aid Sciences and Applications (DASA)

[4] Design and Implementation of an Email Automation System for E-commerce Platforms" by Abdullahi Yusuf and Jibrin G. Omagbon, International Journal of Emerging Technologies in Engineering Research (IJETER), 2018.

[5] "Email Automation System for Educational Institutions" by G. Sravanthi and G. Shoba Bindhu, International Journal of Advanced Research in Computer Science, 2019.

[6] "Two Scoops of Django 3.x: Best Practices for the Django Web Framework" by Daniel Roy Greenfeld and Audrey Roy Greenfeld

[7] "Django for Professionals" by William S. Vincent - This book provides advanced insights into Django development, including email handling and automation.

[8] Automate the Boring Stuff with Python, 2nd Edition By Al Sweigart · 2019

[9] Django for APIs Build web APIs with Python and Django By William S. Vincent · 2022

[10] The Definitive Guide to Django Web Development Done Right By Adrian Holovaty, Jacob Kaplan-Moss, Jason Gilmore · 2007

[11] Method and system for automated population of message body and address fields based on email subject David A. Cohen, Jayesh B. Patel

[12] Django Documentation on Sending Email - Django's official documentation is an invaluable resource for email automation using the Django framework. You can find comprehensive information and examples here: https://docs.djangoproject.com/en/4.0/topics/email/

[13] https://www.tutorialspoint.com/django/django_sending_emails.html

[14] https://realpython.com/ - Real Python offers various tutorials and articles on Django and email automation. It's a helpful resource for developers looking to implement email automation in Django.

[15] https://essuir.sumdu.edu.ua/handle/123456789/85084

[16] http://hmjournals.com/journal/index.php/JECNAM/article/view/112

[17] http://ceur-ws.org/Vol-1210/LQS_02.pdf

[18] https://www.academia.edu/download/52784064/IJCST-V5I2P71.pdf

[19] https://www.ijrar.org/papers/IJRAR1BQP039.pdf

[20] https://ieeexplore.ieee.org/abstract/document/9711903/

[21] https://ntnuopen.ntnu.no/ntnuxmlui/handle/11250/2563968