**INTERNSHIP PROJECT REPORT**

**ON**

**“WEB DEVELOPMENT”**

**TOPIC: MASS-MAIL-DISPATCHER**

**SUBMITTED BY:**

**AJIT KUMAR,**

**1BY22MC003,**

**MCA**

**UNDER THE GUIDANCE OF**

**Y. VISHNUARDHAN**

**PERFORMED AT**

**EXPOSYS DATA LABS**

P.M R. Residency

Gmund Floor, No-5/3 Sy. No. 10/6-1

Doddaballapur Main Road

Yelahanka Bengaluru, Karnataka 560064

**DURATION OF INTERNSHIP**

**ONE MONTH**

**ABSTRACT**

The Mass-Mail Dispatcher is a web-based application designed to revolutionize the process of sending bulk emails. This innovative system serves as a comprehensive tool for seamlessly dispatching emails to a multitude of recipients with efficiency and ease. The primary aim of this project is to provide users with a sophisticated platform capable of handling extensive email distribution while ensuring precision and accuracy in email delivery. By allowing users to upload recipient lists via CSV files, the system streamlines the process, offering meticulous sorting of both valid and invalid email addresses.

Through meticulous validation procedures, the application meticulously identifies and segregates valid and invalid email addresses, facilitating a user-friendly interface where users can effortlessly access and manage their recipient lists. The application empowers users with a structured and organized display of validated emails, thereby enabling efficient email dispatch based on user preferences. This system's robust architecture and user-centric functionalities redefine the conventional email dispatching process, catering to the diverse needs of users seeking a reliable and efficient mass-mailing solution.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL.NO.** | **TITLE** | **PAGE** |
| 1 | INTRODUCTION | 01 |
| 2 | EXISTING METHOD | 02 |
| 3 | PROPOSED METHOD AND SYSTEM ARCHITECTURE | 03 |
| 4 | METHODOLOGY | 06 |
| 5 | IMPLEMENTATION | 08 |
| 6 | CONCLUSION | 15 |

1. **INTRODUCTION**

The introduction to the Mass-Mail Dispatcher unveils a pioneering solution poised to transform the landscape of email dissemination. In today's digital age, efficient communication through emails stands as a cornerstone for businesses, organizations, and individuals alike. However, the traditional methods of sending bulk emails often pose challenges, especially when managing extensive recipient lists. The introduction of this web-based application marks a significant leap forward, addressing the inherent complexities in managing and dispatching emails to numerous recipients.

The system's genesis stems from the necessity to streamline and simplify the process of sending mass emails, ensuring a seamless and structured approach to handling vast recipient lists. By harnessing the power of technology, this application aims to revolutionize how emails are distributed, offering users an intuitive and efficient platform.

Through a user-friendly interface, the Mass-Mail Dispatcher endeavours to optimize the email dispatching process, enhancing precision and reliability while catering to the diverse needs of users seeking a dependable mass-mailing solution. This introduction sets the stage for a comprehensive exploration of how this innovative system redefines the paradigm of email dispatching, promising enhanced efficiency, accuracy, and convenience in the realm of electronic communication.

1. **EXISTING METHOD**

In the realm of mass email distribution, the existing methods often entail cumbersome and time-consuming processes. Traditional approaches to sending bulk emails require manual handling of recipient lists, often stored in CSV (Comma-Separated Values) files. These conventional methods lack the sophistication to efficiently parse through extensive email lists, resulting in challenges when segregating valid and invalid email addresses.

Without dedicated tools, users grapple with the arduous task of sifting through these lists manually, verifying and sorting email addresses based on their validity. The absence of a structured system to detect and manage valid and invalid emails presents significant hurdles in ensuring the successful delivery of mass emails. Moreover, the lack of streamlined processes contributes to potential errors and inefficiencies, leading to suboptimal communication outcomes.

The current methods fall short in providing users with a comprehensive and user-friendly approach to managing vast recipient databases, often resulting in time wastage and reduced effectiveness in email dispatching. This absence of an optimized system to handle mass emails prompts the need for an innovative solution that streamlines the process, enhances accuracy, and facilitates efficient communication.

1. **PROPOSED METHOD**

The proposed method presents an innovative and user-centric approach to streamline mass email distribution. Leveraging modern web-based technologies, the system aims to revolutionize the way bulk emails are managed and dispatched. The cornerstone of this method lies in the development of a user-friendly application – the Mass-Mail Dispatcher. This application offers a seamless platform for sending mass emails to a vast array of recipients uploaded through CSV files.

The proposed architecture encompasses an intuitive user interface where users can upload CSV files containing recipient email addresses. Upon upload, the system automatically detects, segregates, and lists both valid and invalid email addresses. This unique feature ensures a structured and efficient approach to managing recipient lists, allowing users to readily identify and eliminate invalid email addresses, thereby enhancing the accuracy and effectiveness of the email distribution process.

By employing advanced email validation algorithms, the system meticulously sifts through the uploaded CSV files, categorizing emails based on their validity. Valid emails are distinctly presented to the user, allowing for easier selection and usage in the email dispatching process. Simultaneously, the system highlights and lists the invalid email addresses, enabling users to rectify or remove erroneous entries, thereby ensuring a higher success rate in email delivery.

The proposed method endeavours to empower users by offering a comprehensive tool that not only simplifies mass email dispatching but also enhances the overall user experience. The aim is to provide a structured, efficient, and accurate means of managing vast recipient databases, enabling users to communicate seamlessly with a broad audience while mitigating the challenges posed by erroneous or invalid email addresses. This innovative approach is poised to revolutionize mass email communication, offering a refined and streamlined method for efficient email dispatching.

* 1. **System Architecture Overview**

**Frontend:**

- HTML, CSS, JavaScript for the user interface.

- Upload CSV feature allowing users to select and upload files.

- Display area for valid and invalid email lists.

- Compose Email section for users to input sender email, subject, and message.

**Backend (Flask-based):**

- Receive and handle requests from the frontend.

- Endpoints for CSV file processing and email dispatch.

- Validate uploaded CSV files and extract email data.

- Separate valid and invalid emails for display.

- Utilize SMTP to send emails to valid recipients.

**SMTP Server:**

- Used to send emails from the application.

- Authentication with sender's email credentials.

- Processes composed emails and dispatches them to the valid recipients.

**Data Flow:**

1. User uploads CSV file via the frontend.

2. Backend receives the file, processes it to extract emails.

3. Email validation separates valid and invalid emails.

4. Valid emails get composed into emails with user-defined content.

5. SMTP server sends composed emails to the valid recipient list.

**Tools and Technologies:**

- Frontend: HTML, CSS, JavaScript.

- Backend: Python with Flask framework.

- Email Sending: SMTP with SSL for secure communication.

The architecture functions by facilitating CSV uploads, extracting emails, validating them, composing emails, and utilizing SMTP to dispatch them to valid recipients, providing a structured and controlled mass-mailing system.

1. **METHODOLOGY**

Our approach to developing the Mass-Mail Dispatcher project follows a comprehensive methodology encompassing various stages of planning, development, testing, deployment, and user engagement.

This methodical journey involves understanding requirements, designing, implementing, testing, deploying, training, and maintaining the system.

1. **Requirement Analysis**:
   * We initiated the project by engaging with stakeholders to comprehend their needs for a mass-mailing solution. This step helped outline essential functionalities like CSV upload, email validation, and a user-friendly interface.
2. **Design Phase**:
   * **Frontend Development**:
     + The design process commenced with the creation of a visually appealing and intuitive user interface, incorporating HTML, CSS, and JavaScript. This stage focused on structuring sections for composing emails, uploading CSV files, and displaying email validation results.
   * **Backend Development**:
     + Parallel to the frontend, the backend was constructed using Flask, delineating server-side components responsible for handling user requests, processing CSV uploads, extracting email data, and executing validation processes.
3. **Implementation**:
   * **Frontend Implementation**:
     + Through coding and integration of HTML/CSS/JavaScript, we developed features for handling CSV uploads, email validation, and displaying email results to users.
   * **Backend Implementation**:
     + Utilizing Python with Flask, we scripted modules to process uploaded CSV files and facilitate email dispatch using libraries like **smtplib** and **EmailMessage**.
4. **Testing**:
   * **Unit and Integration Testing**:
     + Extensive testing was conducted at various levels to verify the functionality and interaction of individual system components. This encompassed rigorous checks on CSV parsing, email validation algorithms, and seamless data flow between frontend and backend systems.
5. **Deployment**:
   * The application was deployed on a suitable platform supporting Flask applications, with careful attention to server configurations and security measures. Additionally, the SMTP server was configured for secure and efficient email dispatch.
6. **User Training and Acceptance**:
   * Detailed user documentation, along with informative training sessions, were organized to familiarize users with the system’s functionalities and features, ensuring ease of adoption and utilization.
7. **Maintenance and Updates**:
   * Ongoing monitoring and regular updates are pivotal in maintaining system reliability and adaptability, aligning with evolving email protocols and security standards.

This systematic approach guides the development process, ensuring a robust, user-friendly, and dependable Mass-Mail Dispatcher system from inception to deployment and beyond.

1. **IMPLEMENTATION**

**PYTHON SCRIPT FOR SERVER**

from flask import Flask, request, jsonify, render\_template

from email.message import EmailMessage

import ssl

import smtplib

app = Flask(\_\_name\_\_)

email\_password = 'dgov ecti pefj znvj'

@app.route('/')

def index():

    return render\_template('mail\_index.html')

@app.route('/pass', methods=['POST'])

def pass\_data():

    if request.method == 'POST':

        email\_sender = request.json.get('from\_email')

        subject = request.json.get('subject')

        message = request.json.get('message')

        valid\_emails = request.json.get('valid\_emails')

        if email\_sender and subject and message and valid\_emails:

            send\_emails(email\_sender, subject, message, valid\_emails)

            return jsonify({'message': 'Data received and processed successfully'})

        else:

            return jsonify({'error': 'Missing data'})

def send\_emails(email\_sender, subject, message, valid\_emails):

    context = ssl.create\_default\_context()

    em = EmailMessage()

    em['From'] = email\_sender

    em['Subject'] = subject

    em.set\_content(message)

    to\_email = ', '.join(valid\_emails)

    em['To'] = to\_email

    with smtplib.SMTP\_SSL('smtp.gmail.com', 465, context=context) as smtp:

        smtp.login(email\_sender, email\_password)

        smtp.send\_message(em)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**HTML CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Mass-Mail Dispatcher</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

    <style>

        /\* Combined CSS from both styles.css and styles2.css \*/

        body {

            background-color: #f8f9fa;

        }

        .card {

            width: 100%;

            margin: 0;

            background-color: #fff;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

        }

        .card-header {

            background-color: #007bff;

            color: #fff;

            border-bottom: 1px solid #dee2e6;

        }

        .card-body {

            padding: 30px;

        }

        .form-group label {

            font-weight: bold;

        }

        .btn-primary {

            background-color: #007bff;

            border-color: #007bff;

        }

        .btn-primary:hover {

            background-color: #0056b3;

            border-color: #0056b3;

        }

        .list-group {

            margin-top: 10px;

            height: 239px;

            overflow-y: auto;

            border: 1px solid #ccc;

            border-radius: 5px;

            padding: 10px;

        }

        .list-group-item {

            border-radius: 0;

        }

        /\* Custom styles for dividing into columns \*/

        .row {

            display: flex;

            flex-wrap: wrap;

            margin-right: -5px;

            /\* Reduce the default gutter space \*/

            margin-left: -5px;

            /\* Reduce the default gutter space \*/

        }

        .col-md-6.left-column {

            flex: 0 0 45%; /\* Decrease width of the left column \*/

            max-width: 45%; /\* Decrease width of the left column \*/

            padding-right: 5px;

            /\* Adjust the gutter space \*/

            padding-left: 5px;

            /\* Adjust the gutter space \*/

        }

        .col-md-6.right-column {

            flex: 0 0 55%; /\* Increase width of the right column \*/

            max-width: 55%; /\* Increase width of the right column \*/

            padding-right: 5px;

            /\* Adjust the gutter space \*/

            padding-left: 5px;

            /\* Adjust the gutter space \*/

        }

    </style>

</head>

<body>

    <div class="container-fluid mt-5 px-5">

        <div class="card">

            <h2 class="card-header text-center">Mass-Mail Dispatcher</h2>

            <div class="card-body">

                <div class="row">

                    <!-- Left Column for Composing Email -->

                    <div class="col-md-6 left-column">

                        <!-- Content for left column -->

                        <div class="card">

                            <h4 class="card-header text-center">Compose Email</h4>

                            <div class="card-body">

                                <form id="composeForm">

                                    <div class="form-group">

                                        <label for="to">From:</label>

                                        <input type="email" class="form-control" id="to"

                                            placeholder="Enter sender's email addresses">

                                    </div>

                                    <div class="form-group">

                                        <label for="subject">Subject:</label>

                                        <input type="text" class="form-control" id="subject"

                                            placeholder="Enter email subject">

                                    </div>

                                    <div class="form-group">

                                        <label for="message">Message:</label>

                                        <textarea class="form-control" id="message" rows="6"

                                            placeholder="Compose your email"></textarea>

                                    </div>

                                    <button type="button" class="btn btn-primary btn-block"

                                        onclick="sendEmail()"><h5><b>Send</b></h5></button>

                                </form>

                            </div>

                        </div>

                    </div>

                    <!-- Right Column for Uploading CSV -->

                    <div class="col-md-6 right-column">

                        <!-- Content for right column -->

                        <div class="card">

                            <h4 class="card-header text-center">File Uploading</h4>

                            <div class="card-body">

                                <form id="uploadForm">

                                    <div class="form-group">

                                        <label for="csvFile"><h4><b>Upload CSV File:</b></h4></label>

                                        <input type="file" class="form-control-file" id="csvFile" accept=".csv">

                                    </div>

                                    <button type="button" class="btn btn-primary btn-block"

                                        onclick="uploadCSV()">Upload</button>

                                </form>

                                <div class="row mt-4">

                                    <div class="col-md-6">

                                        <h5>Detected Valid Emails (<span id="validCount">0</span>):</h5>

                                        <div id="validEmailsList" class="email-list list-group"></div>

                                    </div>

                                    <div class="col-md-6">

                                        <h5>Detected Invalid Emails (<span id="invalidCount">0</span>):</h5>

                                        <div id="invalidEmailsList" class="email-list list-group"></div>

                                    </div>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

    <!-- Your existing HTML and CSS code -->

    <script>

        function uploadCSV() {

            const fileInput = document.getElementById('csvFile');

            const file = fileInput.files[0];

            const reader = new FileReader();

            reader.onload = function (event) {

                const content = event.target.result;

                // Split content by line breaks to get individual email entries

                const emails = content.split(/\r\n|\n/);

                const validEmails = [];

                const invalidEmails = [];

                const knownTLDs = ["com", "net", "org", "edu", "gov", "in"]; // Add more TLDs as needed

                const emailRegex = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

                emails.forEach((email) => {

                    // Trim whitespace and validate each email with regex and known TLDs

                    const trimmedEmail = email.trim();

                    if (emailRegex.test(trimmedEmail)) {

                        const domainParts = trimmedEmail.split('.');

                        const tld = domainParts[domainParts.length - 1];

                        if (knownTLDs.includes(tld)) {

                            validEmails.push(trimmedEmail);

                        } else {

                            invalidEmails.push(trimmedEmail);

                        }

                    } else if (trimmedEmail !== '') {

                        invalidEmails.push(trimmedEmail);

                    }

                });

                // Display valid and invalid emails in respective divs

                document.getElementById('validEmailsList').innerHTML = validEmails.join('<br>');

                document.getElementById('invalidEmailsList').innerHTML = invalidEmails.join('<br>');

                // Update the count of valid and invalid emails

                document.getElementById('validCount').textContent = validEmails.length;

                document.getElementById('invalidCount').textContent = invalidEmails.length;

            };

            reader.readAsText(file); // Read file as text

        }

        function sendEmail() {

            // Your email sending logic here

            const fromEmail = document.getElementById('to').value;

            const subject = document.getElementById('subject').value;

            const message = document.getElementById('message').value;

            // Retrieve the valid emails from the right column

            const validEmails = document.getElementById('validEmailsList').innerText.split('\n');

            // Make a POST request to the Flask server

            fetch('/pass', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json'

                },

                body: JSON.stringify({

                    from\_email: fromEmail,

                    subject: subject,

                    message: message,

                    valid\_emails: validEmails

                })

            }).then(response => {

                if (response.ok) {

                    alert('Emails sent successfully!');

                } else {

                    console.error('Error sending data.');

                }

            }).catch(error => {

                console.error('Error:', error);

            });

        }

    </script>

</body>

</html>

1. **CONCLUSION**

The development of the Mass-Mail Dispatcher system stands as a significant stride towards addressing the need for a streamlined and efficient tool for managing mass emails. Throughout this project, a comprehensive understanding of stakeholder requirements was achieved, leading to the creation of a robust and user-friendly application.

The system's core functionality revolves around facilitating mass email dispatch, enabling users to upload CSV files containing recipient email addresses and subsequently validating these addresses for accuracy and relevance. The intuitive interface design allows for seamless navigation between composing emails and managing CSV uploads. The application smartly distinguishes between valid and invalid email addresses, enhancing user convenience and ensuring the integrity of the dispatched emails.

By employing a meticulous methodology encompassing requirement analysis, iterative design, and rigorous testing, we ensured that the Mass-Mail Dispatcher meets user expectations while adhering to industry standards and best practices. The utilization of Flask for the backend, coupled with HTML, CSS, and JavaScript for the frontend, enabled the development of a robust and responsive system.

The iterative development approach enabled continuous improvements, ensuring that the system aligns with user needs and remains adaptable to future enhancements. User engagement and feedback have been instrumental in refining the system's features, usability, and overall performance.

In conclusion, the Mass-Mail Dispatcher presents a user-centric, efficient, and scalable solution for managing mass emails. Its successful development reflects our commitment to delivering innovative and practical tools that cater to the evolving needs of users, ensuring seamless communication and efficiency in email distribution processes.