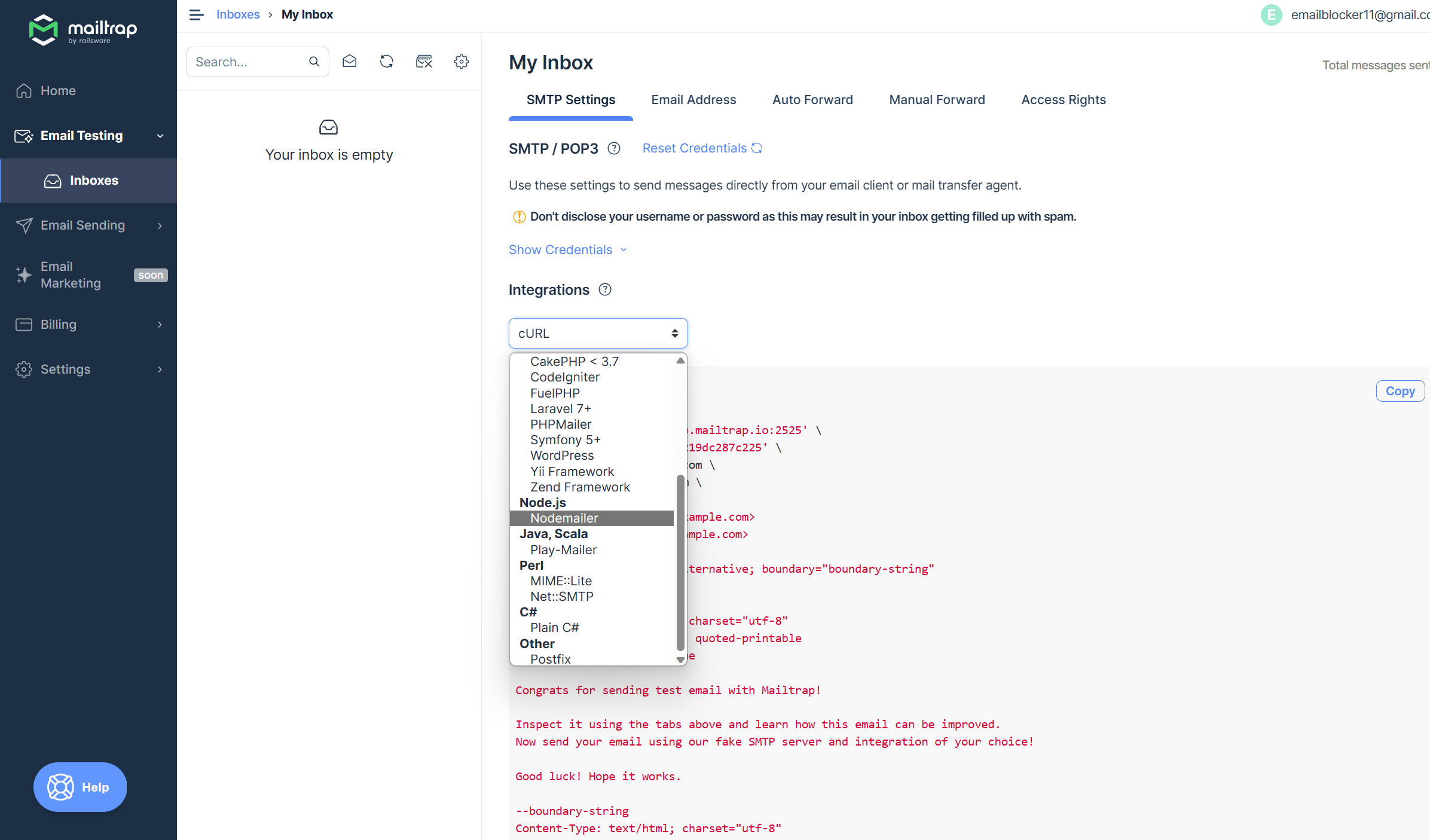
****

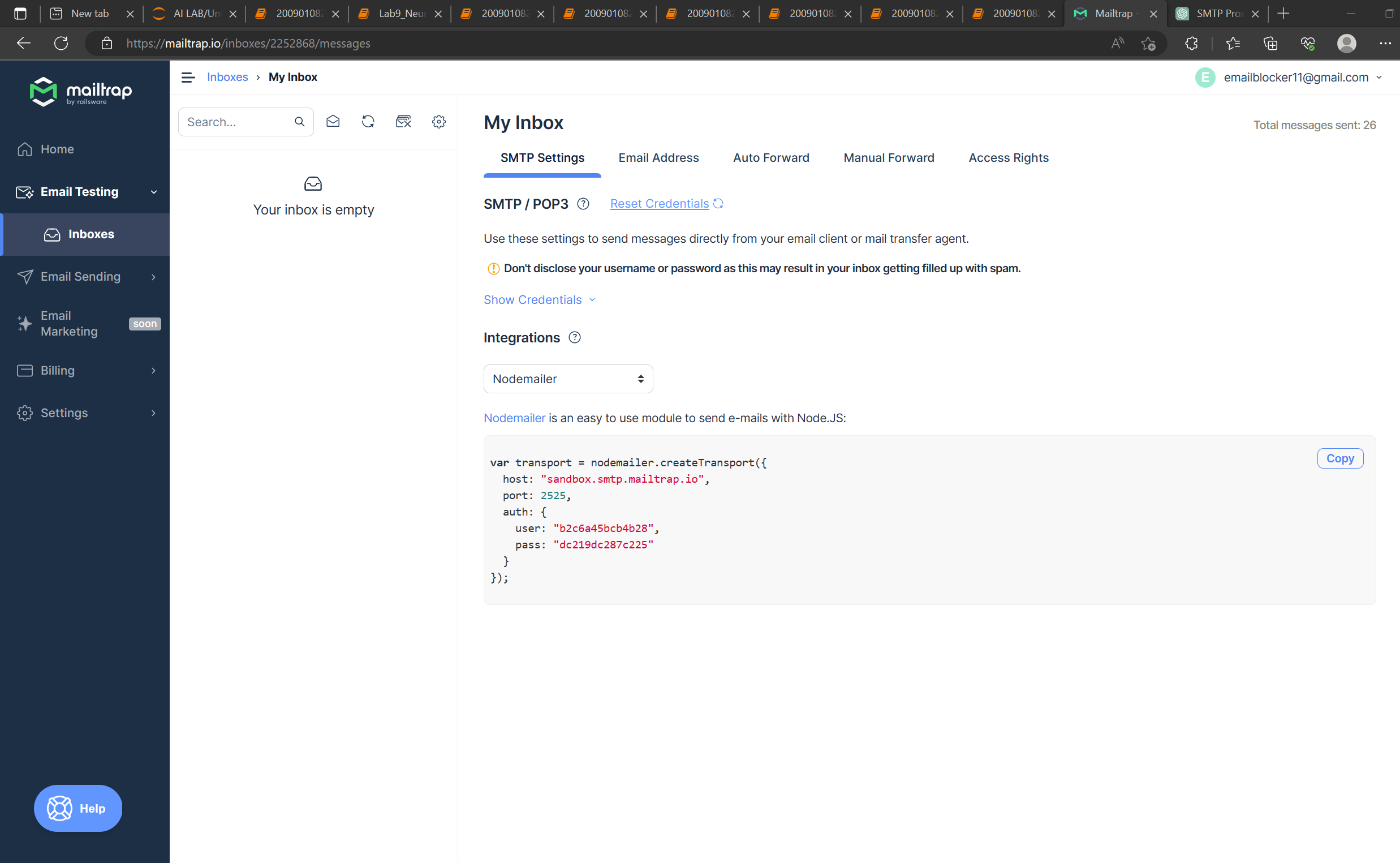
|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  |  | |
| **TITLE** | **:** | **Create a SMTP server proxy - that blocks emails from .in domains and other spam domains as listed in block.txt - in Node.JS** | |
|  |  |  | |
| **DATE OF EXPERIMENT** | **:** | **2nd June, 2020** | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
| **NAME** | **:** |  | **MUHAMMAD JAMEEL KHALID** |
| **ROLL NO** | **:** | **200901082** | |

**Introduction:**

The provided code uses the power of Node.js to create a Proxy SMTP server that also features email blocking functionality. This is achieved using a number of libraries to handle various tasks such as SMTP operations, parsing emails, and reading from the file system. It is designed to work with the popular email testing tool Mailtrap, which allows for the safe testing of email functionality during development.

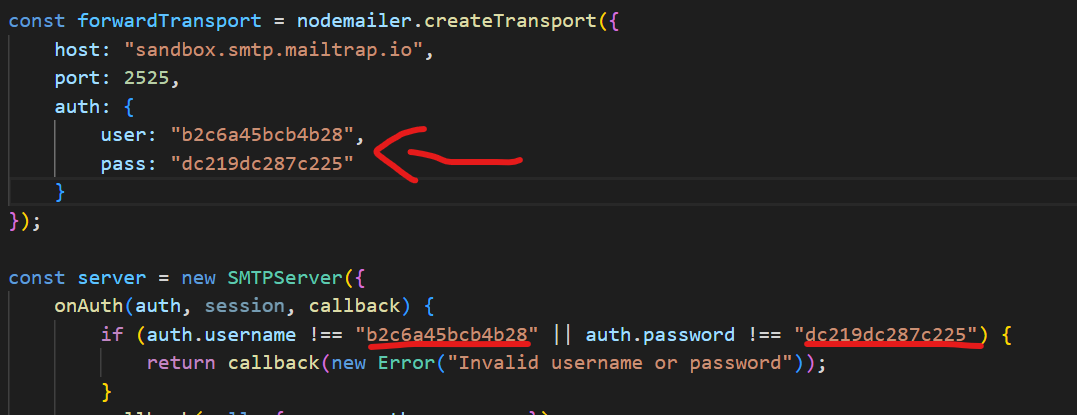
To make this task work with you. You have to sign up to mailtrap.com then copy the code shown in the screenshots





Then change the user and password in according to the above ss code into your proxy and emaiTest.js respectively as shown in screen shots below

**Proxy.js**



**emailTest.js**

**A picture containing text, screenshot, font

Description automatically generated**

**Libraries Used:**

* **smtp-server**

The smtp-server library is used to create the SMTP server. It provides an interface to handle different SMTP events such as authentication and data transfer. In our code, it is used to create a server that listens on port 2525 and handles authentication and incoming data.

* **nodemailer**

nodemailer is a Node.js library that makes it easy to send emails from your applications. It supports various transport methods, SMTP being one of them. In our application, it's used to forward incoming emails from the proxy SMTP server to the final destination, which in this case is a sandbox on mailtrap.io.

* **fs**

The fs module is a built-in Node.js library for handling file system operations. Here, it's used to read a blocklist file that contains a list of top-level domains (TLDs), domains, and specific email addresses that should be blocked.

* **mailparser**

mailparser is a library that can parse raw email data into a more usable JavaScript object. This is necessary because when data comes into the SMTP server, it's in raw form, and we need to extract the sender's email and domain for our blocklist functionality.

**Working of the Proxy SMTP Server:**

This proxy server uses the SMTP protocol to receive emails and forward them to another SMTP server. In this case, we are using Mailtrap as the final destination SMTP server.

The proxy server authenticates the incoming connection using the onAuth method, and if the username and password match the expected values, it lets the connection through.

Once the server receives email data, it parses the raw email using the mailparser library to extract the sender's email and domain. It then checks these against a blocklist. The blocklist, stored in a text file, contains TLDs, domains, and specific email addresses that should be blocked.

If the sender's TLD, domain, or email is found in the blocklist, the server returns an error indicating the email was blocked. If the sender's details are not in the blocklist, the email is forwarded to the Mailtrap SMTP server using nodemailer.

**Email Test with the Proxy Server**

In emailTest.js, nodemailer is used to send an email to the proxy server. The SMTP transport is configured to send the email to localhost on port 2525 (the address and port where the proxy server is running).

Once the email is sent, a callback function checks if there was an error while sending the email. If an error occurs, it's logged to the console. If the email is sent successfully, a success message is logged.

**Testing:**

Before testing I want to give an overview of block list working. Block list is divided into three parts and each part is divided with the help of “---”the following is the block list

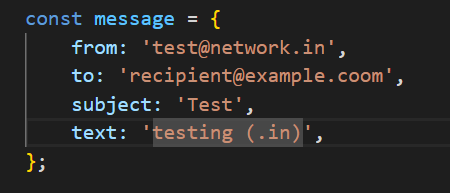
A screenshot of a computer

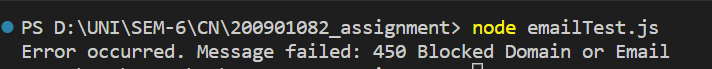
Description automatically generated with medium confidence

1. The first part of the block list is used to block any email or domain which has (.in,.com.org) in the last.
2. The second part of the block list is then used to block any domain (gamil.com,Hotmail.com etc)
3. The last part is used to block list is used to block a specific email ([test@gmail.com](mailto:test@gmail.com) etc)

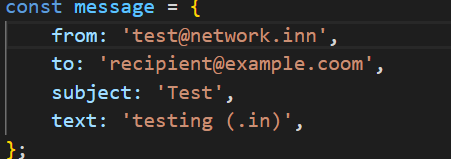
The above are the three according to which my codes are working  
  
**TESTING BLOCKING:**

Testing “**.in**” types emails





Lets change **.in** to **.inn**



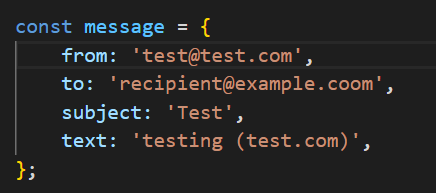


**EMAIL RECIVED:**

**A picture containing text, screenshot, font, software

Description automatically generated**

Now testing on **Domains**

****

****

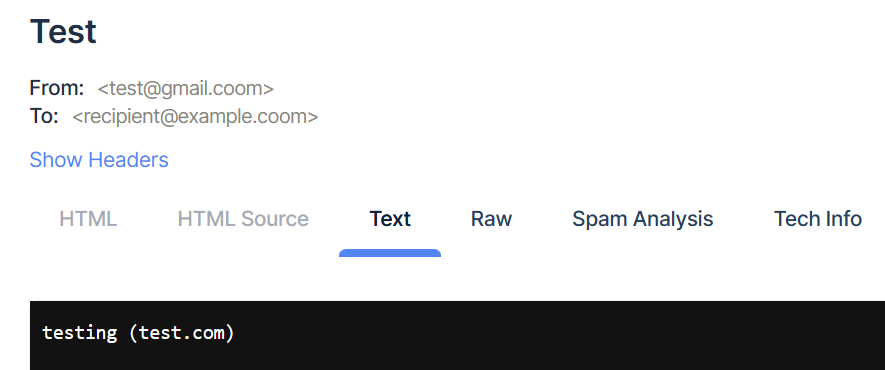
Lets change **test.com** to **gmail.com**

A screen shot of a computer code

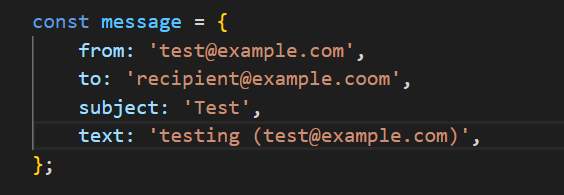
Description automatically generated with low confidence



**EMAIL RECIVED:**



Now testing on **Specific Email**

****

****

Lets change **test@example.com** to **jimmy@gmail.com**

A picture containing text, font, screenshot

Description automatically generated



**EMAIL RECIVED:**

A screenshot of a computer

Description automatically generated with medium confidence