Build Your Own AI Gmail Assistant

Using Gmail API, Gemini AI & GitHub Actions

By Muhammad Saad

X: @msaad_alam

About This Guide

This guide will teach you how to build an AI-powered Gmail Assistant that automatically reads unread emails, classifies them using Gemini AI, applies Gmail labels, and generates draft replies directly inside Gmail.

The entire workflow runs in the cloud using GitHub Actions, meaning there is no need for a VPS, dedicated server, or paid automation platform.

Whether you're a developer, freelancer, creator, entrepreneur, or simply someone who wants a cleaner inbox, this guide will walk you through the complete setup from start to finish.

By the end of this guide, you will have a fully automated email assistant running inside your Gmail account.

Requirements

Before starting, make sure you have:

- A Gmail account
- A GitHub account
- Access to Google Cloud Console
- A Gemini API key
- Basic familiarity with copying files and using a web browser

No prior experience with Gmail APIs, OAuth, GitHub Actions, or AI automation is required.

Table of Contents

1. Introduction
2. Why This Project Exists
3. Features
4. How The Workflow Works
5. System Architecture
6. Files Included
7. Create a GitHub Repository
8. GitHub Codespaces Setup
9. Local Machine Setup (Optional)
10. Gemini API Setup
11. Create a Google Cloud Project
12. Enable Gmail API
13. Configure OAuth Consent Screen
14. Create OAuth Credentials
15. Download credentials.json
16. Understanding OAuth
17. Generate token.json
18. GitHub Secrets Setup
19. GitHub Actions Workflow Setup
20. First Successful Run
21. Understanding Gmail Labels
22. Troubleshooting Common Errors
23. Customization & Future Improvements
24. Final Notes
25. Need Help?

---

1. Introduction

Email is one of the most important communication tools in modern business. Every day we receive newsletters, notifications, meeting requests, collaboration offers, cold outreach, invoices, account alerts, and messages that require a response.

As inboxes grow, manually reviewing and organizing emails becomes increasingly time-consuming.

This project solves that problem by creating an AI-powered Gmail Assistant that automatically reads unread emails, classifies them into useful categories, applies Gmail labels, and generates draft replies whenever appropriate.

The assistant uses:

- Gmail API
- Gemini AI
- GitHub Actions

to create a fully automated workflow that runs directly in the cloud.

No VPS is required.

No paid automation platform is required.

No complex infrastructure is required.

The goal of this project is simplicity. Even if you have never worked with Gmail APIs, OAuth, GitHub Actions, or AI automation before, you can follow this guide and have everything running successfully.

What This Assistant Can Do

- Read unread Gmail messages
- Classify emails using Gemini AI
- Automatically apply Gmail labels
- Generate draft replies
- Create drafts directly inside Gmail
- Detect collaboration opportunities
- Detect meeting requests
- Organize newsletters and marketing emails
- Handle notifications and account alerts
- Separate cold outreach from important conversations
- Run automatically in the cloud
- Work without a VPS
- Work without n8n

- Operate with minimal ongoing cost

---

2. Why This Project Exists

Managing email manually is repetitive work.

Many incoming emails fall into predictable categories. A collaboration request is different from a meeting invitation. A security alert is different from a newsletter. A cold sales email is different from a message that actually requires a response.

Humans are very good at understanding these differences, but modern AI models can now perform these tasks surprisingly well.

The idea behind this project was simple:

Instead of manually checking every unread email, let AI perform the first layer of organization.

The workflow automatically reviews unread emails and decides where they belong.

If an email requires a response, the system can even generate a draft reply that appears directly inside Gmail.

Another goal was to avoid expensive automation platforms.

Many people build similar systems using automation tools such as n8n, Zapier, or Make. While these tools are powerful, they often introduce monthly costs, hosting requirements, or additional complexity.

GitHub Actions provides a simpler alternative.

Because GitHub Actions runs directly inside your repository, the workflow can operate automatically in the cloud without requiring a VPS or paid server.

The result is a lightweight, affordable, and beginner-friendly AI email assistant.

3. Features

The AI Gmail Assistant is designed to automate the most repetitive parts of email management while remaining simple to operate and maintain.

Feature	Included
AI Email Classification	✅
AI Draft Generation	✅
Gmail Labels	✅
GitHub Actions Automation	✅
Cloud-Based Execution	✅
No VPS Required	✅
No n8n Required	✅
Automatic Label Creation	✅
Customizable Prompts	✅

Automatic Email Classification

Every unread email is analyzed using Gemini AI and assigned to the most appropriate category.

Available categories include:

- To Respond
- Collaboration
- Meeting
- Marketing
- Cold Email
- Notification
- Review Manually
- AI Processed
- AI Draft Created

This allows your inbox to stay organized automatically.

Automatic Gmail Labels

The workflow creates and manages Gmail labels automatically.

Labels help separate important emails from newsletters, sales outreach, notifications, and other messages.

AI-Powered Draft Replies

For emails that require a response, Gemini AI generates a professional draft reply.

The draft appears directly inside Gmail, allowing you to review, edit, and send it whenever you choose.

Smart Response Logic

The assistant follows specific rules when generating drafts.

Examples include:

- Asking for project requirements before providing quotes.
- Requesting additional information for collaboration opportunities.
- Asking for availability and time zones when scheduling meetings.
- Avoiding invented pricing or unavailable information.

Cloud-Based Automation

The workflow runs using GitHub Actions.

There is no need to keep a computer running.

There is no need for a VPS.

There is no need for a separate automation platform.

Customizable Prompts

Both the email classification and draft generation prompts can be customized.

You can easily modify:

- Writing style
- Labels
- Confidence thresholds
- Response behavior
- Business rules

Beginner Friendly

The project was intentionally designed to be simple.

Once configured, the workflow runs automatically with minimal maintenance.

Automatically creates missing Gmail labels

The workflow automatically creates these labels if they do not already exist.

No manual Gmail label setup is required.

---

4. How The Workflow Works

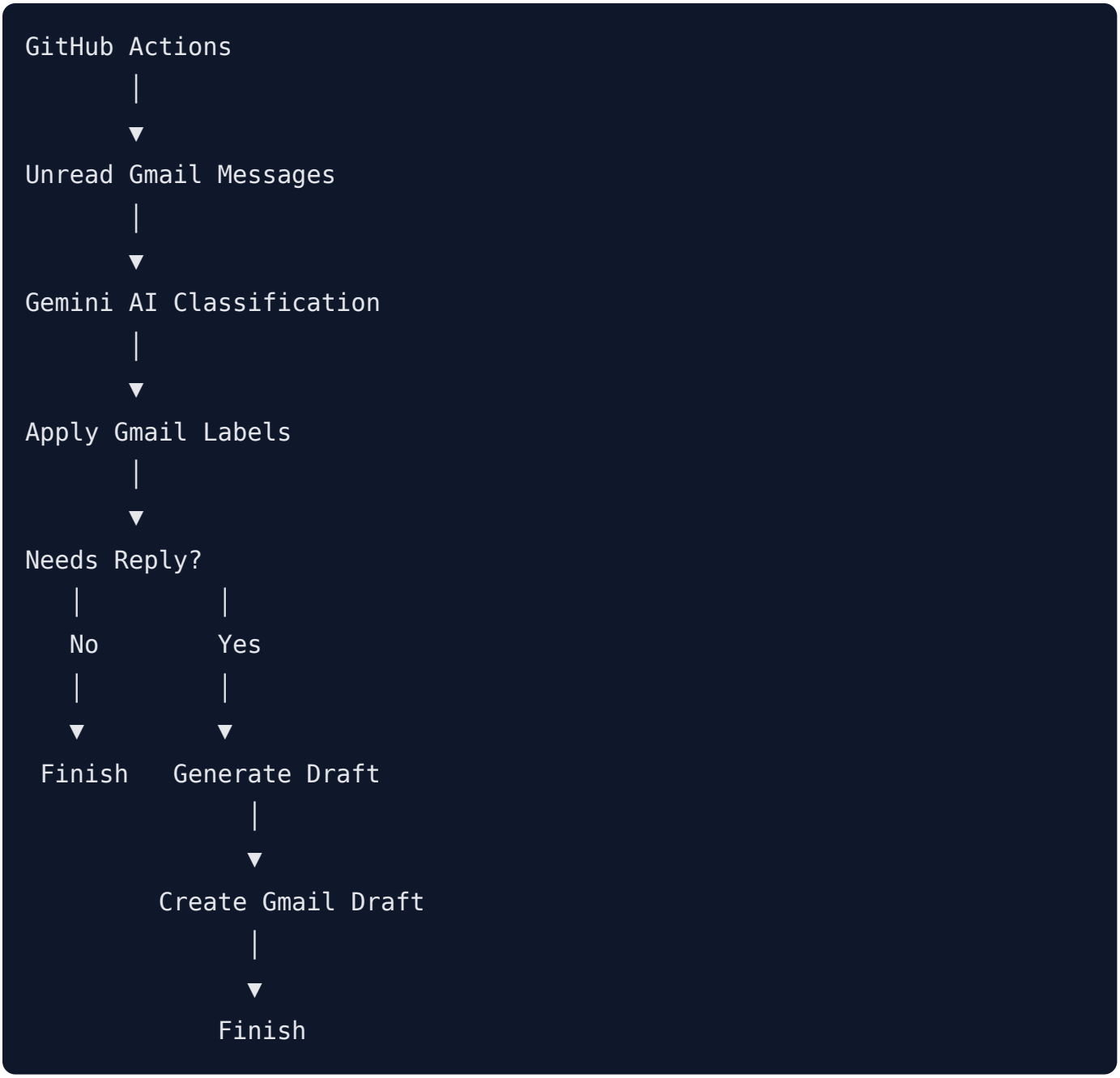
Understanding the overall workflow makes the setup process much easier.

The assistant follows a straightforward sequence:

1. GitHub Actions starts the workflow.
2. The workflow connects to Gmail using the Gmail API.
3. Unread emails are retrieved.
4. Gemini AI analyzes the email.
5. The email is classified.
6. Gmail labels are applied.
7. If a response is required, Gemini generates a draft reply.
8. A Gmail draft is created.
9. The email is marked as processed.
10. The workflow waits until the next scheduled run.

Processing Flow

The following diagram shows the overall process:



Why This Approach Works

Instead of trying to fully automate sending emails, this workflow only creates drafts.

This gives you complete control over outgoing messages while still saving significant time.

You remain responsible for the final review and sending process.

---

5. System Architecture

The project consists of four major components.

Gmail API

The Gmail API provides access to:

- Inbox messages
- Labels
- Draft creation
- Message metadata

Without the Gmail API, the workflow would not be able to interact with your Gmail account.

Gemini AI

Gemini is responsible for:

- Understanding email content
- Classifying emails
- Determining whether a reply is needed
- Generating draft responses

The AI serves as the decision-making layer of the system.

GitHub Actions

GitHub Actions acts as the automation engine.

Its responsibilities include:

- Running the workflow automatically
- Executing Python code
- Loading secrets
- Scheduling inbox checks

Because GitHub Actions runs in the cloud, no VPS is required.

Python Application

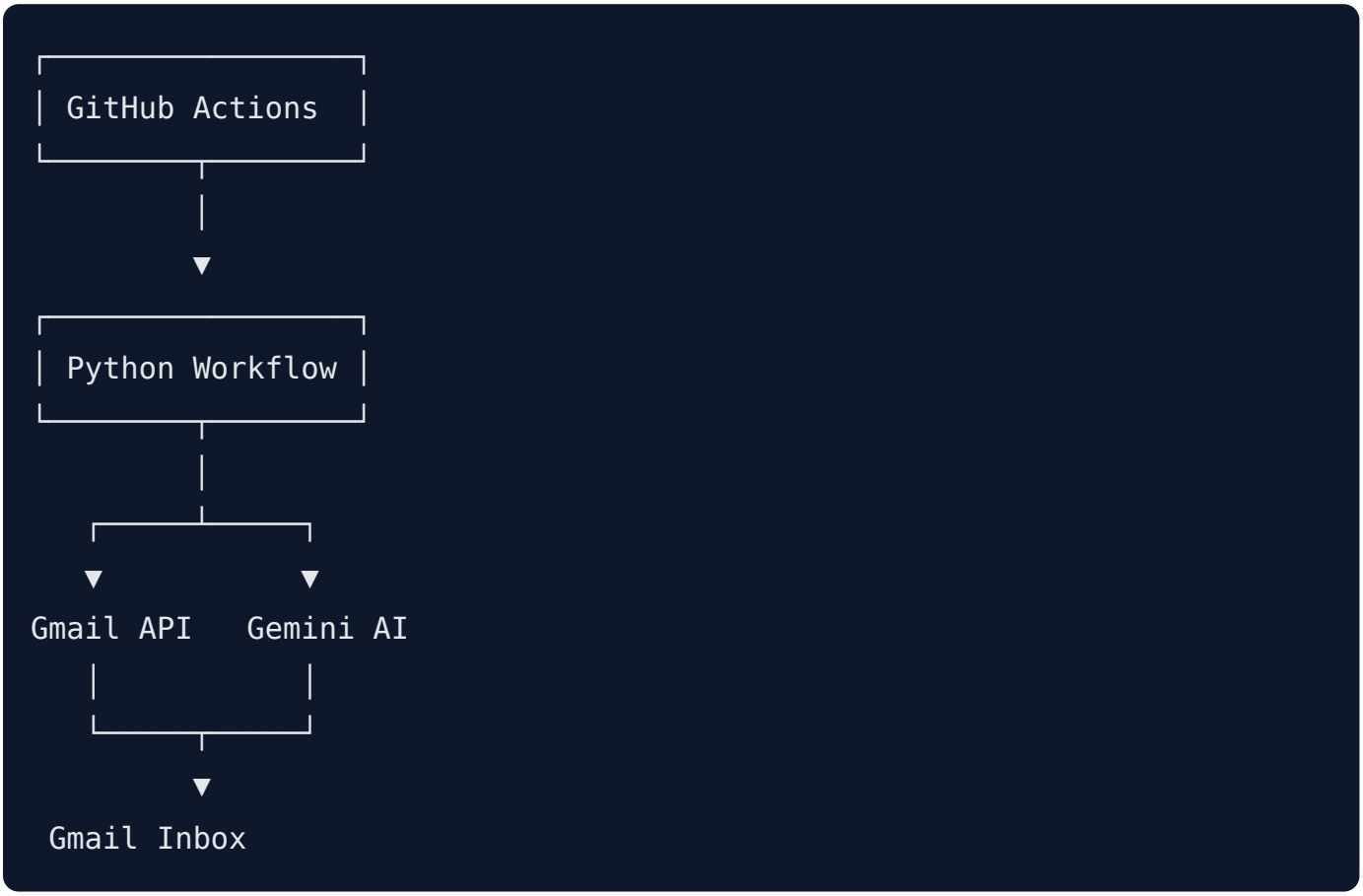
The Python code connects everything together.

It handles:

- Gmail communication

- AI requests
- Label management
- Draft creation
- Workflow logic

Architecture Diagram



---

6. Files Included

The project is intentionally split into separate files to keep the code organized and easy to maintain.

config.py

Stores project-wide configuration values.

Examples include:

- Gemini model name
- Email labels

- Confidence threshold
- Signature settings

This is usually the first file you modify when customizing the workflow.

gmail_utils.py

Contains all Gmail-related functionality.

Responsibilities include:

- Connecting to Gmail
- Reading messages
- Creating labels
- Applying labels
- Creating drafts

ai_classifier.py

Responsible for classifying emails using Gemini AI.

The classifier determines:

- Email category
- Confidence score
- Whether a response is needed

draft_generator.py

Responsible for generating draft replies.

It builds prompts and sends them to Gemini.

The generated text becomes a Gmail draft.

main.py

The main entry point of the application.

This file coordinates the entire workflow.

When GitHub Actions runs the project, it executes this file.

requirements.txt

Contains all required Python dependencies.

GitHub Actions installs these packages automatically before running the workflow.

.gitignore

Prevents sensitive files and unnecessary files from being uploaded to GitHub.

Examples include:

- token.json
- Python cache files
- Environment files

.github/workflows/gmail-ai.yml

The GitHub Actions workflow file.

This file controls:

- Schedule timing
- Manual execution
- Python setup
- Workflow execution

It is responsible for keeping the automation running in the cloud.

7. Create a GitHub Repository

Before setting up any APIs or authentication, we need a GitHub repository to store the project files.

This repository will contain:

- Python source code
- GitHub Actions workflow
- Configuration files
- Documentation

It will also serve as the cloud environment that runs the automation.

Create a New Repository

1. Log in to GitHub.
2. Click the **+** icon in the top-right corner.
3. Select **New repository**.
4. Enter a repository name.

Example:

```
gmail-ai-assistant
```

5. Set the repository visibility.

Recommended:

```
Private
```

6. Click **Create repository**.

Repository Settings

Setting	Recommended Value
Visibility	Private
README	Optional
License	Optional
Git Ignore	None

Upload Project Files

Upload the provided project files:

```
config.py
gmail_utils.py
ai_classifier.py
draft_generator.py
main.py
requirements.txt
.gitignore
.github/workflows/gmail-ai.yml
```

After uploading the files, commit and push the changes.

Recommended Repository Structure

```
gmail-ai-assistant/
│
├── config.py
├── gmail_utils.py
├── ai_classifier.py
├── draft_generator.py
├── main.py
├── requirements.txt
├── .gitignore
│
└── .github
    └── workflows
        └── gmail-ai.yml
```

> Tip: Keep the repository private because it contains automation logic and will later store references to your Gmail and Gemini configuration.

---

8. GitHub Codespaces Setup

This guide uses GitHub Codespaces because it is the easiest option for beginners.

I personally used GitHub Codespaces while building this project.

What Is GitHub Codespaces?

GitHub Codespaces is a cloud-based development environment that runs entirely in your browser.

Instead of installing Python, Git, and development tools locally, GitHub provides a ready-to-use environment.

Benefits

Benefit	Description
No Installation	Everything runs in the browser
Free Tier Available	Suitable for this project
Accessible Anywhere	Work from any computer
Fast Setup	Ready in minutes
VS Code Experience	Familiar development environment

Create a Codespace

1. Open your repository.
2. Click the green **Code** button.
3. Select the **Codespaces** tab.
4. Click **Create codespace on main**.

GitHub will prepare a cloud development environment.

This may take a few minutes during the first launch.

Verify Python Installation

Open the terminal and run:

```
python --version
```

Expected output:

```
Python 3.x.x
```

Install Project Dependencies

Run:

```
pip install -r requirements.txt
```

GitHub Codespaces will download and install all required packages.

Verify Everything Works

Run:

```
python main.py
```

At this stage, the script may fail because Gmail authentication has not been configured yet.

This is expected.

We will configure authentication in later chapters.

---

9. Local Machine Setup (Optional)

If you prefer, you can run the project locally instead of using GitHub Codespaces.

The setup process is almost identical.

Requirements

Install:

- Python 3.10 or newer
- Git
- Visual Studio Code (recommended)

Clone Repository

Open a terminal and run:

```
git clone https://github.com/YOUR_USERNAME/gmail-ai-assistant.git
```

Move into the project directory:

```
cd gmail-ai-assistant
```

Create Virtual Environment

Windows:

```
python -m venv venv
venv\Scripts\activate
```

macOS / Linux:

```
python3 -m venv venv
source venv/bin/activate
```

Install Dependencies

```
pip install -r requirements.txt
```

Verify Installation

```
python main.py
```

Authentication errors are expected at this stage because Gmail setup has not yet been completed.

Codespaces vs Local Machine

Feature	Codespaces	Local Machine

Setup Speed	Fast	Medium
Installation Required	No	Yes
Browser-Based	Yes	No
Works Anywhere	Yes	Depends
Recommended For Beginners	Yes	No

For beginners, GitHub Codespaces is strongly recommended.

---

10. Gemini API Setup

Gemini AI is responsible for:

- Reading email content
- Understanding context
- Classifying messages
- Generating draft replies

Without Gemini, the assistant cannot make decisions.

Create a Gemini API Key

1. Visit Google AI Studio.
2. Sign in with your Google account.
3. Create a new API key.
4. Copy the generated key.

Keep this key secure.

Anyone with access to the key can use your Gemini quota.

Store the API Key

Do not place the key directly in your code.

Later in this guide, we will store it safely using GitHub Secrets.

Test Your API Key

A simple test request can verify that the API key works correctly.

The project will automatically perform API requests once configured.

Security Best Practices

Do	Don't
Store keys in GitHub Secrets	Hardcode keys in Python files
Keep repository private	Commit API keys to GitHub
Rotate compromised keys	Share keys publicly

Common Gemini API Errors

Error	Cause
Invalid API Key	Incorrect key
Quota Exceeded	Usage limit reached
Authentication Error	Key missing or revoked
Permission Denied	API access not enabled

Once your Gemini API key is ready, the next step is creating a Google Cloud project and enabling Gmail API access.

11. Create a Google Cloud Project

Before Gmail can communicate with our application, we need a Google Cloud project.

This project will contain the Gmail API configuration and OAuth credentials required for authentication.

Why Do We Need a Google Cloud Project?

Google requires every application that accesses Gmail data to be registered.

The Google Cloud project acts as the container for:

- Gmail API access
- OAuth configuration
- Authentication credentials

Without it, the workflow cannot access your inbox.

Create a New Project

1. Visit the Google Cloud Console.
2. Sign in with your Google account.
3. Click the project selector near the top of the page.
4. Click **New Project**.
5. Enter a project name.

Example:

```
gmail-ai-assistant
```

6. Click **Create**.

Google will take a few moments to create the project.

Select Your Project

After the project is created:

1. Open the project selector.
2. Choose your newly created project.

Make sure the correct project is selected before continuing.

What We'll Do Next

Now that the project exists, we can enable the Gmail API and configure authentication.

---

12. Enable Gmail API

The Gmail API allows our Python application to:

- Read unread emails
- Create Gmail labels
- Apply labels to messages
- Create draft replies

Without enabling the API, none of these actions will work.

Enable Gmail API

1. Inside Google Cloud Console, open your project.
2. Navigate to **APIs & Services**.
3. Click **Library**.
4. Search for:

```
Gmail API
```

5. Open the Gmail API page.
6. Click **Enable**.

After a few seconds, the API will become available to your project.

Verify the API Is Enabled

You should now see Gmail API listed under:

```
APIs & Services → Enabled APIs & Services
```

If Gmail API appears in the list, you're ready to continue.

> Tip: Most Gmail authentication errors later in the setup process are caused by forgetting this step.

---

13. Configure OAuth Consent Screen

Google requires users to grant permission before an application can access Gmail data.

The OAuth Consent Screen is what users see when authorizing the application.

Since this project is intended for personal use, the setup is simple.

Create the Consent Screen

1. Open:

```
APIs & Services → OAuth Consent Screen
```

2. Select:

```
External
```

3. Click **Create**.

App Information

Fill in the following fields:

Field	Example
App Name	Gmail AI Assistant
User Support Email	Your Gmail Address
Developer Contact Email	Your Gmail Address

Click **Save and Continue**.

Scopes

Click:

```
Add or Remove Scopes
```

Add:

```
https://www.googleapis.com/auth/gmail.modify
```

and

```
https://www.googleapis.com/auth/gmail.compose
```

These permissions allow the assistant to:

- Read emails
- Apply labels
- Create drafts

Click **Save and Continue**.

Test Users

Add the Gmail account that will use the workflow.

Example:

```
yourname@gmail.com
```

Click **Save and Continue**.

Review the settings and finish the setup.

---

14. Create OAuth Credentials

Now we need credentials that allow the Python application to authenticate with Gmail.

Create OAuth Client

1. Open:

```
APIs & Services → Credentials
```

2. Click:

```
Create Credentials
```

3. Select:

```
OAuth Client ID
```

Application Type

Choose:

```
Desktop Application
```

This is important.

Do NOT choose:

- Web Application
- Service Account

The workflow uses OAuth credentials generated from a Desktop Application.

Name the Credential

Example:

```
gmail-ai-assistant
```

Click **Create**.

Google will generate the OAuth credentials.

Download the Credentials File

Click:

```
Download JSON
```

Save the file.

Rename it to:

```
credentials.json
```

This file will be used later when generating the Gmail authentication token.

Keep it safe.

Do not upload it publicly.

Do not share it with anyone.

---

15. Download credentials.json

By now you should have a file similar to:

```
credentials.json
```

Open the file and verify it contains information similar to:

```
{
  "installed": {
    "client_id": "...",
    "project_id": "...",
    "auth_uri": "...",
    "token_uri": "...",
    "client_secret": "...",
    "redirect_uris": [
      "http://localhost"
    ]
  }
}
```

The exact values will be different for every user.

Important Security Notes

Never commit this file to a public repository.

Never send this file to anyone.

Treat it like a password.

Anyone with access to this file may be able to impersonate your application.

Where to Store It

For now, place the file in the root directory of your project:

```
gmail-ai-assistant/
│
├── credentials.json
├── config.py
├── main.py
└── ...
```

We will use this file in the next chapter to generate:

```
token.json
```

which will allow GitHub Actions to access Gmail on your behalf.

16. Understanding OAuth

Before generating our authentication token, it is helpful to understand what is actually happening behind the scenes.

Many beginners see files like:

```
credentials.json
```

and

```
token.json
```

without knowing what they do.

Fortunately, the concept is much simpler than it looks.

How Gmail Authentication Works

Google does not allow applications to access your Gmail account automatically.

Instead, Google requires you to explicitly grant permission.

The process works like this:

```
You
 |
 ▼
Google Login Screen
 |
 ▼
Grant Permission
 |
 ▼
Application Receives Access
```

Once permission is granted, Google creates credentials that allow the application to act on your behalf.

What Is credentials.json?

The file:

```
credentials.json
```

contains information about your Google Cloud application.

Think of it as the application's identity card.

It tells Google:

- Which application is requesting access
- Which project owns the application
- Which OAuth settings should be used

This file does NOT provide Gmail access by itself.

It only identifies the application.

What Is token.json?

The file:

```
token.json
```

contains the authorization granted by your Google account.

Think of it as a permission slip.

This file tells Google:

```
Yes, I approve this application.
```

Without token.json, the workflow cannot access Gmail.

Access Tokens and Refresh Tokens

Inside token.json are special values provided by Google.

Access Token

The access token is used for Gmail API requests.

However, it expires after a period of time.

Refresh Token

The refresh token allows Google to issue a new access token automatically.

This is what keeps the workflow running without requiring you to log in repeatedly.

Why Tokens Sometimes Expire

In some situations Google may revoke or invalidate a token.

Examples include:

- Changing OAuth settings
- Deleting OAuth credentials
- Removing application permissions

- Regenerating credentials
- Security-related account changes

When this happens, a new token.json must be generated.

Summary

At this point:

```
credentials.json
```

identifies the application.

And:

```
token.json
```

authorizes the application to access Gmail.

In the next chapter we will generate token.json.

---

17. Generate token.json

Now that we understand OAuth, we can generate the authentication token used by the workflow.

Create generate_token.py

Create a new file named:

```
generate_token.py
```

Paste the following code:

```
from google_auth_oauthlib.flow import InstalledAppFlow

SCOPES = [
    "https://www.googleapis.com/auth/gmail.modify",
    "https://www.googleapis.com/auth/gmail.compose",
]

flow = InstalledAppFlow.from_client_secrets_file(
    "credentials.json",
    SCOPES,
)

flow.redirect_uri = "http://localhost"

auth_url, _ = flow.authorization_url(
    access_type="offline",
    prompt="consent",
)

print("\nOPEN THIS URL:\n")
print(auth_url)

code = input("\nPASTE AUTHORIZATION CODE HERE:\n")

flow.fetch_token(code=code)

creds = flow.credentials

with open("token.json", "w") as f:
    f.write(creds.to_json())

print("\nSUCCESS! token.json created.")
```

Save the file.

Run the Script

In the terminal, run:

```
python generate_token.py
```

The script will print a Google authorization URL.

Example:

```
OPEN THIS URL:

https://accounts.google.com/...
```

Authorize the Application

1. Copy the URL.
2. Open it in your browser.
3. Sign in with your Gmail account.
4. Review the requested permissions.
5. Click **Allow**.

After approval, Google will redirect you to a URL similar to:

```
http://localhost/?state=...
&code=4/0AeoWuM-XXXXXXXXXXX
&scope=...
```

Copy the Authorization Code

From the browser address bar, copy the value after:

```
code=
```

and before

```
&scope
```

Example:

```
4/0AX4XfW...
```

Return to the terminal.

Paste the code when prompted.

```
PASTE AUTHORIZATION CODE HERE:
```

Press Enter.

Successful Result

If everything works correctly, you should see:

```
SUCCESS! token.json created.
```

A new file named:

```
token.json
```

will appear in your project folder.

Verify token.json

You should now have:

```
gmail-ai-assistant/
│
├── credentials.json
├── token.json
├── main.py
├── config.py
└── ...
```

Congratulations.

You have successfully authenticated the Gmail Assistant.

Common Problems

Error: Missing redirect_uri

This usually happens when OAuth credentials are configured incorrectly.

Make sure:

```
Desktop Application
```

was selected when creating OAuth credentials.

Error: invalid_grant

This usually means:

- Token expired
- Token revoked
- OAuth configuration changed

The solution is usually to generate a new token.json.

Error: mismatching_state

This can happen when:

- Multiple authorization tabs are opened
- The authorization flow is restarted midway

Close all authorization tabs and run the script again.

Error: token.json Not Created

Verify:

- credentials.json exists
- OAuth credentials are valid
- Correct authorization code was pasted

Once token.json has been created successfully, we are ready to move everything into GitHub Secrets so the workflow can run completely in the cloud.

---

18. GitHub Secrets Setup

The workflow uses GitHub Secrets to securely store sensitive information.

Instead of uploading secrets directly to the repository, GitHub encrypts and protects them.

This keeps credentials safe while still allowing GitHub Actions to use them.

Open Repository Settings

Navigate to:

```
Repository
→ Settings
→ Secrets and Variables
→ Actions
```

Click:

```
New Repository Secret
```

We will create three secrets.

Secret 1: GEMINI_API_KEY

Name:

```
GEMINI_API_KEY
```

Value:

```
Your Gemini API Key
```

This allows Gemini AI to classify emails and generate drafts.

Secret 2: GMAIL_CREDENTIALS

Name:

```
GMAIL_CREDENTIALS
```

Open:

```
credentials.json
```

Copy the entire contents.

Paste everything into the secret value.

Save the secret.

Secret 3: GMAIL_TOKEN_JSON

Name:

```
GMAIL_TOKEN_JSON
```

Open:

```
token.json
```

Copy the entire contents.

Paste everything into the secret value.

Save the secret.

Verify Your Secrets

You should now have:

```
GEMINI_API_KEY

GMAIL_CREDENTIALS

GMAIL_TOKEN_JSON
```

stored in GitHub Secrets.

At this point the project is fully configured and ready for GitHub Actions.

19. GitHub Actions Workflow Setup

One of the biggest advantages of this project is that it runs automatically in the cloud.

Instead of keeping your computer online 24/7, GitHub Actions runs the workflow for you.

What Is GitHub Actions?

GitHub Actions is GitHub's built-in automation platform.

It can:

- Run Python scripts
- Execute scheduled tasks
- Install dependencies
- Access GitHub Secrets
- Run entirely in the cloud

For this project, GitHub Actions checks Gmail on a schedule and processes new emails automatically.

Workflow File

The project includes:

```
.github/workflows/gmail-ai.yml
```

This file controls how and when the automation runs.

Schedule Configuration

Inside the workflow file you will find a cron schedule.

Example:

```
schedule:
  - cron: "*/10 * * * *"
```

This means:

```
Run every 10 minutes
```

You can adjust the frequency if desired.

Examples:

```
*/5   = Every 5 minutes
*/10  = Every 10 minutes
*/15  = Every 15 minutes
*/30  = Every 30 minutes
```

Manual Execution

The workflow also supports manual runs.

Inside GitHub:

```
Actions
→ Gmail AI Assistant
→ Run Workflow
```

This is useful for testing changes immediately.

First Push

Commit and push all project files to GitHub.

Once pushed, GitHub Actions will automatically detect the workflow file.

Verify Workflow Detection

Open:

```
Actions
```

You should see:

```
Gmail AI Assistant
```

listed as an available workflow.

If it appears, GitHub has successfully detected the automation.

Important Note About Scheduling

GitHub does not always run scheduled workflows at the exact minute specified.

Small delays are normal.

This behavior is controlled by GitHub and is expected.

---

20. First Successful Run

Now it's time to test the complete system.

This is the moment where everything comes together.

Run the Workflow

Navigate to:

```
Actions
→ Gmail AI Assistant
→ Run Workflow
```

Select:

```
main
```

and click:

```
Run Workflow
```

GitHub will start a new workflow run.

Watch the Logs

Click the running workflow.

You should see logs similar to:

```
Installing dependencies...

Connecting to Gmail...

Loading Gemini AI...

Processing emails...
```

Successful Label Creation

During the first run you should see messages similar to:

```
Label exists: To Respond
Label exists: Collaboration
Label exists: Meeting
Label exists: Marketing
Label exists: Cold Email
Label exists: Notification
Label exists: Review Manually
Label exists: AI Processed
Label exists: AI Draft Created
```

This confirms the Gmail labels are available.

Automatic Label Creation

You do not need to manually create any of the Gmail labels before running the workflow.

During startup, the assistant automatically checks whether the required labels exist.

If a label is missing, it will be created automatically.

This includes:

- To Respond
- Collaboration
- Meeting
- Marketing
- Cold Email
- Notification
- Review Manually
- AI Processed
- AI Draft Created

For example, if your Gmail account does not already contain a label named:

```
Collaboration
```

the workflow will create it automatically during its first run.

This makes the setup process much easier because no manual Gmail label configuration is required.

You can simply complete the setup, run the workflow, and allow the assistant to prepare the inbox structure automatically.

Successful Email Processing

Example:

```
Found 3 email(s)

Processing email...

Classification:
Meeting

Generating draft...

Draft created successfully.
```

Check Gmail

Open Gmail and verify:

- Labels exist
- Emails are categorized correctly
- Drafts appear when needed

If everything works, congratulations.

Your AI Gmail Assistant is now operational.

What Happens Next?

Once the workflow is running successfully:

1. New emails arrive.
2. GitHub Actions starts automatically.
3. Emails are classified.
4. Labels are applied.
5. Draft replies are generated.
6. Gmail stays organized automatically.

From this point onward, the system largely manages itself.

---

21. Understanding Gmail Labels

The labels are the foundation of the workflow.

They allow Gmail to remain organized and searchable.

The workflow automatically creates these labels if they do not already exist.

No manual Gmail label setup is required.

User-Facing Labels

To Respond

Used for emails that require a reply.

Examples:

- Client questions
- Business inquiries
- Follow-ups

Collaboration

Used for:

- Partnerships
- Sponsorships
- Affiliate opportunities
- Guest posts
- Creator outreach

Meeting

Used for:

- Calls
- Consultations
- Scheduling discussions
- Calendar coordination

Marketing

Used for:

- Newsletters
- Product announcements
- Promotional content
- Company updates

Cold Email

Used for:

- Sales pitches
- SEO services
- Lead generation offers

- Agency outreach

Notification

Used for:

- Login alerts
- Verification codes
- Security notices
- Account updates
- Automated system emails

Review Manually

Used when AI is uncertain or when human review is recommended.

Examples:

- Legal documents
- Contracts
- Invoices
- Financial discussions
- Complex business matters

Internal Workflow Labels

These labels are used by the automation itself.

AI Processed

This label prevents the same email from being processed repeatedly.

Without it, the workflow could analyze the same message every time it runs.

AI Draft Created

This label prevents duplicate drafts.

If a draft already exists, the workflow knows not to generate another one.

Example Classifications

Email Subject	Assigned Label

Partnership Opportunity	Collaboration
Let's Schedule a Call	Meeting
Your Verification Code	Notification
New Product Release	Marketing
We Can Improve Your SEO	Cold Email
Project Inquiry	To Respond

Label Summary

Label	Purpose
To Respond	Requires a reply
Collaboration	Partnership opportunities
Meeting	Scheduling and calls
Marketing	Newsletters and promotions
Cold Email	Sales outreach
Notification	Alerts and system emails
Review Manually	Human review recommended
AI Processed	Prevents reprocessing
AI Draft Created	Prevents duplicate drafts

Why Labels Matter

As your inbox grows, labels become increasingly valuable.

Instead of searching through hundreds or thousands of emails, Gmail automatically organizes everything into meaningful categories.

This saves time and makes important emails easier to find.

22. Troubleshooting Common Errors

Even with a correct setup, occasional issues can occur.

Fortunately, most problems are easy to solve once you know what causes them.

Error: Token Has Expired or Been Revoked

Example:

```
google.auth.exceptions.RefreshError:
invalid_grant: Token has been expired or revoked.
```

Cause

Google no longer trusts the existing token.

Common reasons include:

- OAuth credentials were recreated
- OAuth settings changed
- Google revoked access
- Token became invalid

Solution

Generate a new:

```
token.json
```

using:

```
python generate_token.py
```

Then update:

```
GMAIL_TOKEN_JSON
```

inside GitHub Secrets.

Error: token.json Not Found

Example:

```
FileNotFoundError:
No such file or directory: token.json
```

Cause

The workflow cannot find the authentication token.

Solution

Verify that:

```
token.json
```

exists locally.

Also verify:

```
GMAIL_TOKEN_JSON
```

contains the full contents of token.json.

Error: credentials.json Not Found

Example:

```
FileNotFoundError:
No such file or directory: credentials.json
```

Cause

OAuth credentials are missing.

Solution

Download the OAuth credentials again from Google Cloud Console.

Save them as:

```
credentials.json
```

and update:

```
GMAIL_CREDENTIALS
```

inside GitHub Secrets.

Error: Missing redirect_uri

Example:

```
Access blocked:
Missing required parameter: redirect_uri
```

Cause

OAuth credentials were configured incorrectly.

Solution

When creating OAuth credentials, choose:

```
Desktop Application
```

Do not use:

- Web Application
- Service Account

Error: Mismatching State

Example:

```
MismatchingStateError
```

Cause

The authorization process was interrupted or restarted.

Solution

Close all authorization tabs.

Run:

```
python generate_token.py
```

again and complete the process from the beginning.

Error: Gmail API Not Enabled

Example:

```
403 Access Not Configured
```

Cause

The Gmail API was never enabled.

Solution

Open:

```
Google Cloud Console
→ APIs & Services
→ Library
```

Enable:

```
Gmail API
```

Error: Workflow Not Running Automatically

Cause

GitHub Actions scheduling is not enabled or the workflow file is incorrect.

Solution

Verify:

- Workflow file is inside `.github/workflows`
- Repository contains the workflow
- Scheduled trigger exists
- Actions are enabled

Also remember that GitHub schedules are not always exact and small delays are normal.

Error: Gemini API Authentication Failed

Cause

Invalid API key.

Solution

Verify:

```
GEMINI_API_KEY
```

inside GitHub Secrets.

If necessary, create a new API key and update the secret.

Still Stuck?

Most setup issues can be solved by checking:

1. Gmail API enabled
2. OAuth configured correctly

3. credentials.json valid
4. token.json valid
5. GitHub Secrets updated
6. Workflow file committed and pushed

Carefully reviewing these six items usually resolves the majority of problems.

---

23. Customization & Future Improvements

One of the best things about this project is that it can easily be customized.

The provided workflow is only a starting point.

Add New Labels

You can create additional categories.

Examples:

```
Client
Job Application
Support Ticket
Invoice
Urgent
```

Simply update:

```
config.py
```

and the classification prompt.

Modify Classification Rules

Inside:

```
ai_classifier.py
```

you can change how Gemini makes decisions.

Examples:

- Stricter classification
- More categories
- Different confidence requirements

Change Draft Style

Inside:

```
draft_generator.py
```

you can modify the writing style.

Examples:

- Formal
- Casual
- Friendly
- Corporate
- Technical

Adjust Workflow Frequency

Inside:

```
gmail-ai.yml
```

you can change:

```
cron: "*/10 * * * *"
```

Examples:

```
Every 5 minutes
Every 10 minutes
Every 15 minutes
Every 30 minutes
Every hour
```

Choose a schedule that matches your needs.

Future Improvement Ideas

Some ideas you may want to build in future versions:

Daily Email Summaries

Generate a daily summary of important emails.

Slack Notifications

Send important emails directly to Slack.

Discord Notifications

Receive alerts inside Discord.

Lead Detection

Identify potential clients automatically.

CRM Integration

Push qualified leads into a CRM.

Priority Scoring

Assign priority scores to incoming emails.

Auto Archive

Automatically archive low-value emails.

Calendar Integration

Create calendar events from meeting requests.

Multi-Inbox Support

Manage multiple Gmail accounts from one workflow.

The possibilities are almost endless.

---

24. Final Notes

Congratulations.

You have successfully built an AI-powered Gmail Assistant using:

- Gmail API
- Gemini AI
- Python
- GitHub Actions

without requiring:

- A VPS
- A dedicated server
- n8n
- Zapier
- Make
- Monthly hosting costs

Your assistant can now:

- Read unread emails
- Classify messages intelligently
- Apply Gmail labels
- Generate draft replies
- Organize your inbox automatically
- Run entirely in the cloud

The most important thing to remember is that this project is meant to save time, not remove human judgment.

Always review AI-generated drafts before sending them.

AI is a powerful assistant, but you remain responsible for final decisions and communication.

As AI models continue to improve, workflows like this will become even more capable and valuable.

This project provides a strong foundation that you can continue expanding and customizing to fit your own workflow.

Thank you for taking the time to build it.

I hope it saves you countless hours managing your inbox.

---

25. Need Help?

If you received this guide and need assistance with setup, configuration, or troubleshooting, feel free to reach out.

Contact

You can reach me via X (Twitter) or email:

- **X (Twitter):** @msaad_alam
- **Email:** contact@dnnengineer.com or alamzweb@gmail.com *(Feel free to use these if you don't have an X account or if you run into any errors)*

What You'll Receive

If you message me, I can provide:

- The complete workflow files
- Setup assistance
- The latest project updates
- Future workflow improvements
- Additional guidance if you get stuck

Included Resources

This guide is designed to be used alongside:

Workflow Files

```
config.py
gmail_utils.py
ai_classifier.py
draft_generator.py
main.py
requirements.txt
.gitignore
.github/workflows/gmail-ai.yml
```

Video Tutorial

A complete walkthrough covering the entire setup process from start to finish.

Thank you for reading.

Happy automating! 🚀

— **Muhammad Saad**

Disclaimer

This project and guide are provided for educational purposes only.

Users are responsible for complying with Google's Terms of Service, Gmail API policies, and any applicable laws or regulations.

Always review AI-generated drafts before sending emails.

The workflow is intended to assist with email management and should not replace human judgment.

The author is not responsible for any actions, modifications, or outcomes resulting from the use of this project.