

AI-Powered Email Generator

AI-Powered Email Generator - Detailed Explanation

The **AI-Powered Email Generator** is a **professional email writing assistant** that helps users generate **structured emails** based on predefined templates. Users can **select an email type**, provide **recipient details**, and **include relevant context** to generate well-formatted emails. The system refines the message using **DeepSeek AI via Ollama**, ensuring that the email remains **polished and professional**.

File Structure & Detailed Breakdown

This project consists of four key Python files:

1 `main.py` - The User Interface (Gradio)

This file is responsible for **setting up the user interface** using **Gradio**, allowing users to:

- **Select an AI model** (DeepSeek or other available models).
- **Choose an email template** (Job Application, Business Inquiry, Follow-Up).
- **Provide recipient details and additional context**.
- **Generate a structured email** with a **single click**.

Key Functionalities:

- `get_available_models()` → Fetches available AI models installed in Ollama.
- `create_email(template_type, recipient, details, model)` → Calls the **email generation function** in `email_generator.py`.
- `update_dropdown()` → Refreshes the available models dynamically.

How It Works:

- When the **Generate Email** button is clicked, `create_email()` processes the input and calls the AI model to generate the structured email.
- The AI **follows a predefined format**, ensuring **consistency** and **professionalism**.

◆ Example UI Code:

```
generate_button.click(
    fn=create_email,
    inputs=[template_selector, recipient_input, details_input, model_selector],
    outputs=email_output
)
```

2 `email_generator.py` - AI Email Processing

This file **handles the logic** for constructing the **email content** and interacting with **Ollama's DeepSeek AI** for refinement.

Key Functionalities:

- `run_ollama_prompt(prompt, model)` → Calls **Ollama CLI** to process the email text.
- `generate_email(template_type, recipient, details, model, sender)` → Constructs an email based on user input and the selected **template**.

How It Works:

1. **Retrieves the email template** from `templates.py`.
2. **Formats the email** using placeholders for:
 - **Greeting** (e.g., "Dear [Recipient],")
 - **AI-Generated Content** (based on the template)
 - **Closing Signature**
3. **Passes the formatted email through AI** for further refinement.

◆ Example Email Processing Code:

```
email_body = template.format(
    greeting=EMAIL_FORMAT["greeting"].format(recipient=recipient),
    closing=EMAIL_FORMAT["closing"].format(sender=sender),
    position=details.get("position", "a suitable role"),
    company=details.get("company", "your organization"),
    skills=details.get("skills", "relevant qualifications")
)
```

```
)  
return run_ollama_prompt(email_body, model)
```

templates.py - Predefined Email Templates

This file contains **email structure templates** for various types of professional communication.

Key Functionalities:

- Stores **email formatting templates** for **Job Applications, Business Inquiries, and Follow-Ups**.

How It Works:

- If the user selects "**Job Application**", the AI is guided with this template:

```
"I am writing to express my interest in the {position} role at {company}..."
```

- If the user selects "**Business Inquiry**", the AI uses:

```
"I am reaching out to inquire about {inquiry_details}. I would appreciate any information you can provide."
```

- If the user selects "**Follow-Up**", the system uses:

```
"I wanted to follow up regarding {previous_conversation}."
```

Example Template Code:

```
TEMPLATES = {  
    "job_application": ""  
    {greeting}
```

```
I am writing to express my interest in the {position} role at {company}. With my experience in {skills}, I believe I would be a great fit for your team.
```

```
I have attached my resume for your consideration. I look forward to the opp
```

portunity to discuss my application further.

```
{closing}
""",
}
```

config.py - Email Formatting Rules

This file defines **email formatting rules**, ensuring that generated responses maintain **professionalism and consistency**.

Key Functionalities:

- Stores **greeting & closing structures** for all emails.
- Defines **default sender name** (which can be customized).

How It Works:

- Every AI-generated email **automatically includes**:

```
EMAIL_FORMAT = {
    "greeting": "Dear {recipient},",
    "closing": "Best regards,\n{sender}"
}
```

- This ensures **structured and well-formatted** emails.

How Everything Works Together

- 1** User selects an email type & provides recipient details in the UI (**main.py**).
- 2** The system retrieves the relevant email template (**templates.py**).
- 3** A structured email is generated (**email_generator.py**).
- 4** AI refines and formats the email using DeepSeek (**run_ollama_prompt**).
- 5** The final email is displayed in the Gradio UI.

How to Run the Project

Run the application:

```
python main.py
```

Select an email type, enter details, and generate an email!
