🤖 Jarvis — Simple Task Automation Agent (Text-Only)

Jarvis is a lightweight automation assistant you can run from your terminal (and optionally from Streamlit).
It supports:

- 📧 **Send Emails** (via Gmail App Passwords)
- ⏰ **Set Reminders** (with optional email alert)
- 🗂️ **Organize Files** (clean up your Downloads, dry-run supported)
- 💬 **Chat** (optional — can be disabled by admin)

This project is designed to be **beginner-friendly** and secure (no secrets in code).

✨ What We Built (at a glance)

1. **Text-only agent** that parses natural commands.
2. **Rule-based intent parser** with **LLM fallback** (optional).
3. **Gmail email sender** using **App Password** (secure).
4. **Reminders** using `schedule` (with optional email send).
5. **File organizer** that groups files by type (with dry-run).
6. Optional **Streamlit UI** (and how to disable OpenAI safely).
7. Proper **`.gitignore`, secrets handling, and GitHub push protection** steps.

🗂️ Folder Structure

```
your-repo/
├─ main.py                 # Entry point (text-only loop)
├─ intent_agent.py         # Rule-based + optional LLM fallback
├─ email_agent.py          # Gmail send via SMTP + App Password
├─ reminder_agent.py       # Daily time-based reminders
├─ file_agent.py           # File organizer (with dry-run)
├─ chat_agent.py           # Chat; returns "disabled" if no API key
├─ .gitignore              # Ignore venv, __pycache__, .env, etc.
├─ requirements.txt        # Minimal dependencies
└─ README.md               # This doc
```

🛠️ Prerequisites

- **Python 3.10+** (Windows: installed from Microsoft Store or python.org)
- A **Gmail** account + **App Password**
- (Optional) **OpenAI API key** (only if you enable chat/LLM fallback)

- **Git** (if you want to push to GitHub)

🔒 Gmail App Password (one-time)

1. Go to **Google Account → Security**
2. Turn on **2-Step Verification**
3. Search **App passwords** → Create a new one (**App: Mail, Device: Windows**)
4. Copy the 16-char password (looks like `abcd efgh ijkl mnop`)

You'll use this in your environment variables as `GMAIL_PASSWORD`.

🌱 Setup (Windows / PowerShell)

```
# 1) Clone your repo (or create folder)
git clone https://github.com/<you>/Simple-Task-Automation-Agent.git
cd Simple-Task-Automation-Agent

# 2) Create & activate virtual environment
python -m venv venv
.\venv\Scripts\Activate.ps1

# 3) Install dependencies
pip install -r requirements.txt
```

requirements.txt (minimal)

```
schedule
python-dotenv
streamlit      # optional, only if you build a UI
openai         # optional, only if you enable chat/LLM fallback
```

If you are **disabling Chat/LLM**, you can remove `openai` and `streamlit` from the file.

🔑 Environment Variables

Create a `.env` file in your project root:

```
GMAIL_EMAIL=yourgmail@gmail.com
GMAIL_PASSWORD=your_app_password
# OPENAI_API_KEY=sk-...   # Only if you want to enable chat/LLM
```

We **never** hardcode secrets in code. `.env` is **ignored** by git (via `.gitignore`).

▶️ Run the Agent (text-only)

```
.\venv\Scripts\Activate.ps1
python main.py
```

You'll see:

```
🤖 Jarvis Text Assistant Ready.
Type `help` to see commands. Type `exit` to quit.
```

💬 Example Commands (copy–paste)

Emails

```
send email to someone@example.com Hello how are you
send happy birthday to someone@example.com
```

Reminders

```
remind me to drink water at 9am
remind me at 23:20 to stretch
remind me to pray at 7pm email me
remind me to send report at 9am to my mail someone@example.com
```

Time can be `HH:MM` (24h) or simple `"9am"`, `"7pm"`. The interpreter normalizes common patterns.

File Organizer

```
organize files in "C:\Users\YourName\Downloads"
organize files in "C:\Users\YourName\Downloads" dry run
```

Chat (if enabled)

```
what is software
who are you
tell me about AI
```

🧠 How It Works (line-by-line idea)

1) `main.py`

- Starts a simple loop: waits for your text input
- Calls `interpret(text)` from `intent_agent.py`
- For each returned action, calls `dispatch_one(intent, slots)`
- Handles missing info by asking follow-up questions (email address/message)
- Prints results (and sends email / schedules reminders / organizes files)

2) `intent_agent.py`

- **Rule-based** patterns for emails, reminders, file commands
- **LLM fallback** (optional) using OpenAI if `OPENAI_API_KEY` is set
- Extracts `intent` + `slots` as JSON-ish dict:
- `[{"intent":"send_email","slots":{"to":"x@y.com","subject":"...","message":"..."}}]`
- Normalizes "spoken" emails like `name at gmail dot com → name@gmail.com`

3) `email_agent.py`

- Uses Python's `smtplib` to login with `GMAIL_EMAIL` + `GMAIL_PASSWORD`
- Sends simple plaintext emails

4) `reminder_agent.py`

- Uses the `schedule` library to run a job daily at `HH:MM`
- When time hits, prints the reminder and (optionally) emails it to you

5) `file_agent.py`

- Walks a given directory and moves files into type folders:
  - `PDFs/`, `Images/`, `Executables/`, `Archives/`, `Others/`
- **Dry run** prints what it would do without moving anything

6) `chat_agent.py`

- **Safe by default**: if `OPENAI_API_KEY` is missing → returns "⚠️ `Chat feature disabled by admin (no OpenAI API key configured).`"
- If enabled, uses `gpt-4o-mini` to answer general questions

🔐 Security: Avoid Leaking Keys to GitHub

- Make sure `.gitignore` contains:
- `venv/`
- `__pycache__/`
- `.env`
- `memory.json`
- If GitHub rejects a push because a secret was committed, use:
- `git filter-repo --path main.py --invert-paths --force`
- `git add main.py`
- `git commit -m "Re-add clean main.py (removed secret from history)"`
- `git push -u origin main --force`
- **Rotate** any leaked API key immediately (delete and create a new one).

🌐 Optional: Streamlit UI (web)

1. Create `app.py`:

```
import streamlit as st
from main import interpret, dispatch_one  # reuse the same logic

st.set_page_config(page_title="Jarvis", page_icon="🤖", layout="centered")
st.title("🤖 Jarvis — Automation Assistant")

cmd = st.text_input("Type a command")
if st.button("Run") or cmd:
    actions = interpret(cmd)
    for a in actions or []:
        dispatch_one(a.get("intent"), a.get("slots", {}))
```

2. Add to `requirements.txt`:

```
streamlit
```

3. Locally:

```
streamlit run app.py
```

4. On Streamlit Cloud:

- Connect GitHub repo
- App file: `app.py`
- **Secrets** (if using email/LLM):
- `GMAIL_EMAIL = "yourgmail@gmail.com"`
- `GMAIL_PASSWORD = "app_password"`
- `# OPENAI_API_KEY = "sk-..."  (optional)`

If you **don't** want LLM costs, **omit** `OPENAI_API_KEY`. `chat_agent.py` will show "disabled by admin".

🧪 Troubleshooting

`pip : command not found` (PowerShell)
Use full path python to install pip packages:

```
.\venv\Scripts\Activate.ps1
python -m pip install -r requirements.txt
```

`The recipient address <happy> is not valid`
Your email pattern is wrong. Say:

```
send email to someone@example.com Hello
send happy birthday to someone@example.com
```

Reminder didn’t trigger

- Keep the program running (the scheduler checks every second)
- Use 24-hour `HH:MM` format for `set_reminder`

Streamlit: OpenAIError (api_key missing)

- Add key in Streamlit **Secrets** or keep chat disabled (safe default)

🗺️ Roadmap (next steps you can add)

- Web scrapers + RAG for “Autonomous Research Assistant”
- Persistent chat memory (SQLite)
- Google Calendar integration for reminders
- Better NER for names / emails (e.g., `phonenumbers`, `dateparser`)
- Unit tests