ChatBotAI

Project Specification

Modular AI Toolkit & Integration Platform

Version 0.5

February 2026

A collection of standalone, independently importable modules for Telegram, Gmail, OpenAI, and Pinecone - designed to be composed into agentic workflows.

Table of Contents

1. Project Overview
2. Architecture
3. Deliverables
   3.1 Telegram Integration (tg/)
   3.2 Gmail Integration (gmail/)
   3.3 RAG Agent (agent/)
   3.4 Pinecone Toolkit (tools/pinecone/)
   3.5 OpenAI Ingestion (tools/openai/)
4. Tasks Completed
5. Technical Specifications
6. Configuration
7. Dependencies
8. File Manifest

1. Project Overview

ChatBotAI is a modular toolkit of standalone API functions, processing pipelines, and AI agent components. Every module is independently importable and designed to be composed into agentic workflows in a separate project.

The project provides production-ready integrations for Telegram, Gmail, OpenAI, and Pinecone, connected through a consistent 3-stage pipeline architecture using JSON queue files as the inter-process communication layer.

Key Principles

- Standalone modules - each .py file can be imported and used independently
- Decoupled pipeline - triggers, handlers, and actions run as separate processes
- JSON queue data layer - inter-stage communication via JSON files on disk
- UUID correlation IDs - every message is tracked across the full pipeline
- Provider-agnostic - embedding functions, LLM providers, and vector stores are pluggable
- No namespace collisions - package named tg/ to avoid shadowing python-telegram-bot

2. Architecture

Repository Structure

```
ChatBotAI/
+-- tg/              Telegram Bot - API functions & message pipeline
+-- gmail/           Gmail - API functions & email pipeline
+-- agent/           RAG chatbot - OpenAI + Pinecone retrieval
+-- tools/
|   +-- pinecone/    Vector database toolkit & CLI
|   +-- openai/      Embedding & knowledge base ingestion
+-- _config files/   Configuration templates
+-- credentials/     OAuth tokens (gitignored)
```

Pipeline Architecture

Both tg/ and gmail/ implement a 3-stage pipeline. Each stage is a standalone script that reads from and writes to JSON queue files:

```
triggers/              handlers/              actions/
(external input)  -->  (process & respond)  -->  (send output)
                  |                          |
         trigger_queue.json          reply_queue.json
```

Stage	Telegram	Gmail
Triggers	Bot listener	Inbox poller (IMAP/API)
Handlers	Build reply payloads	Build reply payloads

Actions	Send via Bot API	Send via Gmail API

3. Deliverables

3.1 Telegram Integration (tg/)

Standalone API wrappers and a message processing pipeline for Telegram bots.

Module	Description
api/send_message.py	Send a text message (sync + async)
api/send_typing.py	Send typing indicator (sync + async)
api/get_me.py	Get bot info (sync + async)
utils/config.py	Load config from JSON or env vars
utils/chat_logger.py	JSONL audit logger for updates
utils/queue_manager.py	JSON queue: load, save, append, clear
triggers/bot.py	Bot listener - queues incoming messages
handlers/build_replies.py	Trigger queue -> reply queue
actions/send_replies.py	Reply queue -> send via Bot API

3.2 Gmail Integration (gmail/)

Standalone Gmail API functions and an email processing pipeline using OAuth2.

Module	Description
utils/auth.py	OAuth2 flow: authorize, refresh, build service
utils/parser.py	Parse messages: MIME, HTML, headers, attachments
utils/queue_manager.py	JSON queue manager (same interface as tg/)
api/get_email.py	Fetch single email by ID with parsing
api/list_emails.py	Search/list with Gmail query syntax
api/send_email.py	Send email with attachments, CC, BCC, HTML
api/reply_email.py	Reply to existing thread
api/modify_labels.py	Mark read/unread, archive, trash, star
api/get_attachments.py	Download attachments from a message
triggers/poll_inbox.py	Poll unread inbox -> trigger_queue.json
handlers/build_replies.py	Trigger queue -> reply queue
actions/send_replies.py	Reply queue -> send via Gmail API

3.3 RAG Agent (agent/)

Retrieval-Augmented Generation chatbot. Embeds user questions, retrieves context from a Pinecone vector store, and generates responses via OpenAI.

Module	Description
memory.py	Conversation history: load, save, clear, append, trim
context.py	Embed questions + query Pinecone for context
chat.py	OpenAI chat: chat(), chat_simple(), build_messages()
prompt.py	System prompt loader: .txt, .docx, or inline
agent.py	Orchestrator: memory + context + chat in one flow

3.4 Pinecone Toolkit (tools/pinecone/)

Full-featured, self-contained Pinecone vector database toolkit. Can be copied into any project. Includes a unified CLI.

Module	Description
config.py	PineconeConfig: from_json(), from_env()
client.py	Authenticated client/index factory
vector_store.py	Upsert, query (filters), batch, fetch, stats
index_manager.py	Create, delete, list, describe indexes
embeddings.py	embed_text(), embed_batch(), make_embed_fn()
parser.py	Parse .docx, .txt, .csv into chunks
fetch.py	fetch_vectors(), fetch_one(), vector_exists()
namespace_manager.py	List, delete, copy, stats for namespaces
backup.py	Export/import to JSON, metadata-only export
cli.py	Unified CLI: index, vectors, namespace, backup

3.5 OpenAI Ingestion (tools/openai/)

One-file runner for embedding .docx knowledge bases and upserting to Pinecone. Supports interactive and CLI modes with auto index dimension validation.

4. Tasks Completed

v0.3 - Telegram Refactor

- Restructured telegram/ into modular tg/ package with triggers/, api/, handlers/, actions/, utils/
- Built standalone API wrappers: send_message, send_typing, get_me (sync + async)
- Implemented JSON queue pipeline: trigger_queue.json and reply_queue.json
- Created shared utilities: config loader, JSONL audit logger, generic queue manager
- Added UUID correlation IDs to track messages across pipeline stages
- Renamed input/ to triggers/ for clarity

Agent Modules

- Built agent/memory.py - conversation history management with configurable max pairs
- Built agent/context.py - Pinecone vector search with OpenAI embeddings
- Built agent/chat.py - OpenAI chat completion with RAG context injection
- Built agent/prompt.py - system prompt loader supporting .txt, .docx, and inline strings
- Refactored agent/agent.py to use new modular components

Documentation

- Created README.md in every package and subfolder (16 files)
- Wrote top-level README.md with setup, structure, and usage documentation
- Added docstrings to every public function and class

Repo-Wide Fixes & Optimisations

- CRITICAL: Renamed telegram/ to tg/ to resolve namespace collision with python-telegram-bot
- Added missing __init__.py for agent/ and tools/openai/
- Moved logging.basicConfig() from module level to main() functions
- Added null guard on message/text in bot trigger handler
- Propagated trigger entry UUIDs through to reply queue entries
- Fixed agent load_config to private _load_config to avoid import conflicts
- Fixed tools/pinecone/cli.py relative path to use absolute project root path
- Fixed make_embed_fn return type annotation
- Updated all internal imports from telegram.* to tg.*
- Updated .gitignore for tg/ paths and runtime data

v0.4 - Gmail Integration

- Built gmail/utils/auth.py - full OAuth2 flow with token caching and auto-refresh
- Built gmail/utils/parser.py - MIME tree walker, HTML stripping, attachment metadata
- Built gmail/api/ - 6 standalone API modules: get, list, send, reply, labels, attachments
- Built gmail/triggers/poll_inbox.py - polls unread inbox, queues messages, marks as read
- Built gmail/handlers/build_replies.py - processes trigger queue with pluggable reply logic
- Built gmail/actions/send_replies.py - sends threaded replies or new emails from queue
- Added Gmail config to config.example.json and .gitignore

v0.5 - Pinecone Toolkit Expansion

- Built tools/pinecone/embeddings.py - embed_text(), embed_batch(), make_embed_fn()
- Built tools/pinecone/fetch.py - fetch_vectors(), fetch_one(), vector_exists()
- Built tools/pinecone/namespace_manager.py - list, delete, copy, stats for namespaces
- Built tools/pinecone/backup.py - export/import vectors to JSON, metadata-only export
- Expanded parser.py - added .txt (paragraph splitting) and .csv parsing, parse_file() auto-detect
- Expanded vector_store.py - metadata filtering on queries, batch query, min_score, fetch method
- Expanded cli.py - new command groups: vectors fetch/query, namespace, backup
- Updated __init__.py with all new exports

5. Technical Specifications

Language & Runtime

Language: Python 3.10+
Type Hints: PEP 604 union syntax (str | None)
Async Support: Telegram API functions provide sync + async variants

Data Interchange

Queue Format: JSON files (array of objects)
Audit Log: JSONL (append-only, one JSON object per line)
Config: JSON file or environment variables (.env)
Correlation: UUID v4 IDs assigned at trigger, propagated through pipeline

Authentication

Telegram: Bot token via config.json or TELEGRAM_BOT_TOKEN env var
Gmail: OAuth2 with auto-refresh (credentials.json + token.json)
OpenAI: API key via config.json or OPENAI_API_KEY env var
Pinecone: API key via config.json or PINECONE_API_KEY env var

Embedding Models

Model	Dimensions	Alias
text-embedding-3-small	1536	"small"
text-embedding-3-large	3072	"large"

Pinecone CLI Commands

Group	Commands
index	create, delete, list, describe
vectors	stats, upsert, fetch, query, delete, delete-all, update-metadata
namespace	list, stats, delete, copy
backup	export, import

6. Configuration

Settings can be provided via _config files/config.json or environment variables (.env):

Service	Config Key	Env Variable	Required
Telegram	telegram.bot_token	TELEGRAM_BOT_TOKEN	For tg/
OpenAI	openai.api_key	OPENAI_API_KEY	For agent/
Pinecone	pinecone.api_key	PINECONE_API_KEY	For vectors
Gmail	gmail.credentials_file	GOOGLE_CLIENT_ID	For gmail/
Agent	agent.chat_model	OPENAI_CHAT_MODEL	Optional

7. Dependencies

Core

- python-telegram-bot - Telegram Bot API framework
- openai - OpenAI API client (chat, embeddings)
- pinecone - Pinecone vector database client

Gmail

- google-api-python-client - Google API client library
- google-auth-httplib2 - HTTP transport for Google Auth
- google-auth-oauthlib - OAuth2 integration

Parsing

- python-docx - .docx document parsing

Optional

- python-dotenv - .env file loading

8. File Manifest

tg/

```
__init__.py, README.md
api/__init__.py, send_message.py, send_typing.py, get_me.py
triggers/__init__.py, bot.py
handlers/__init__.py, build_replies.py
actions/__init__.py, send_replies.py
utils/__init__.py, config.py, chat_logger.py, queue_manager.py
```

gmail/

```
__init__.py, README.md
api/__init__.py, get_email.py, list_emails.py, send_email.py, reply_email.py, modify_labels.py, get_attachments.py
triggers/__init__.py, poll_inbox.py
handlers/__init__.py, build_replies.py
actions/__init__.py, send_replies.py
utils/__init__.py, auth.py, parser.py, queue_manager.py
```

agent/

```
__init__.py, README.md
agent.py, memory.py, context.py, chat.py, prompt.py
```

tools/pinecone/

```
__init__.py, README.md
config.py, client.py, vector_store.py, index_manager.py
embeddings.py, parser.py, fetch.py, namespace_manager.py, backup.py, cli.py
```

tools/openai/

```
__init__.py, README.md, OpenAI_embeddings.py
```

Root

```
README.md, README.pdf, .gitignore, .env.example
_config files/config.example.json
```