

AGENTS.md Guide & Policy

Standard Protocol for AI Agent Consumption of Reusable Bot & Architecture
Code

Repository: <https://github.com/sh15aa/botserusable>

Maintainer: foundershivanand@gmail.com

Standard: Zero-Duplication / Reuse-First

Date: October 2026

REUSABLE CODE LIBRARY: ENTERPRISE BOT & AI FACTORY

This repository ([botserusable](#)) contains production-ready, security-hardened reusable code for building conversational AI products, omni-channel messaging bots (WhatsApp, Telegram, Chatwoot), distributed asynchronous engines, and AI safety guardrails.

CORE RULE

Before writing any new implementation code for common bot, AI, or security functionality:

1. **Search this repository first.**
2. **Inspect `registry/modules.json`.**
3. **Find an existing reusable module.**
4. **Inspect its parameters, classes, and dependencies.**
5. **Reuse and vendor/import the module whenever compatible.**
6. **Do NOT recreate functionality that already exists in this library.**

PRIORITY

Existing Reusable Library Module

>

Existing Project Implementation

>

New Implementation (Only if missing from library)

REUSABLE MODULE REGISTRY SUMMARY

All modules are registered in `registry/modules.json` with full metadata:

Category	Module ID	File Path	Core Classes / Capabilities
Adapters	`adapters.whatsapp`	`app/adapters/whatsapp.py`	`WhatsAppAdapter` (Meta Cloud API inbound parse, interactive outbound)
Adapters	`adapters.telegram`	`app/adapters/telegram.py`	`TelegramAdapter` (Telegram Bot API update parse, inline keyboards)
Adapters	`adapters.chatwoot`	`app/adapters/chatwoot.py`	`ChatwootClient` (Live-agent desk sync, conversation relay)
Adapters	`adapters.base`	`app/adapters/base.py`	`BaseChannelAdapter` (Abstract adapter contract)
Schemas	`schemas.unmp`	`app/schemas/unmp.py`	`UnifiedMessage`, `UnifiedResponse`, `ChannelType`, `MessageContent`
Schemas	`schemas.chatwoot`	`app/schemas/chatwoot.py`	`ChatwootWebhookPayload`, `ChatwootConversation`
Engine	`engine.fsm`	`app/engine/fsm.py`	`ConversationEngine` (Redis Mutex lock with atomic Lua release, FSM states)
Engine	`engine.handoff`	`app/engine/handoff.py`	`HumanHandoffManager` (Seamless human-in-the-loop transition)
Engine	`engine.flows`	`app/engine/flows.py`	`FlowEngine` (Multi-step interactive branching forms/dialogs)
Engine	`engine.router`	`app/engine/router.py`	`MessageRouter` (Slash commands, action callbacks, fallbacks)
AI	`ai.guardrails`	`app/ai/guardrails.py`	`AIGuardrailService` (Automated PII redaction, prompt delimiters, canary)

			leak detection)
AI	<code>`ai.gateway`</code>	<code>`app/ai/gateway.py`</code>	<code>`LLMGateway`</code> (Multi-model router: OpenAI, Anthropic, Gemini, Groq, Ollama)
AI	<code>`ai.cost_controller`</code>	<code>`app/ai/cost_controller.py`</code>	<code>`AICostController`</code> (Per-user daily token caps & quota preservation via Redis)
AI	<code>`ai.memory`</code>	<code>`app/ai/memory.py`</code>	<code>`ConversationMemory`</code> (Redis sliding window of turns with TTL)
AI	<code>`ai.rag`</code>	<code>`app/ai/rag.py`</code>	<code>`RAGKnowledgeBase`</code> (Vector embeddings, cosine similarity)
AI	<code>`ai.tools`</code>	<code>`app/ai/tools.py`</code>	<code>`ToolSandbox`</code> (Function registration, schema generation, tool execution)
Security	<code>`security.rate_limiter`</code>	<code>`app/security/rate_limiter.py`</code>	<code>`DistributedRateLimiter`</code> (Atomic Redis Lua sliding window against DoS)
Security	<code>`security.loop_detector`</code>	<code>`app/security/loop_detector.py`</code>	<code>`BotLoopDetector`</code> (Rolling SHA-256 hash tracking to prevent bot-to-bot ping-pong)
Security	<code>`security.rbac`</code>	<code>`app/security/rbac.py`</code>	<code>`SecurityContext`</code> , <code>`Role`</code> (Admin/Client auth guards)
Security	<code>`security.feature_flags`</code>	<code>`app/security/feature_flags.py`</code>	<code>`FeatureFlagService`</code> (Redis dynamic flags & emergency kill switches)
Gateway	<code>`gateway.security`</code>	<code>`app/gateway/security.py`</code>	<code>`WebhookSecurityService`</code> (Constant-time HMAC-SHA256, replay prevention <code>`SET NX EX`</code>)

Gateway	<code>`gateway.router`</code>	<code>`app/gateway/router.py`</code>	FastAPI webhook endpoints (<200ms ACK, queue offloading)
Messaging	<code>`messaging.media`</code>	<code>`app/messaging/media.py`</code>	<code>`MediaProcessor`</code> (SSRF-protected attachment downloads, private IP blocker)
Messaging	<code>`messaging.templates`</code>	<code>`app/messaging/templates.py`</code>	<code>`TemplateEngine`</code> (i18n locale template rendering)
Identity	<code>`identity.service`</code>	<code>`app/identity/service.py`</code>	<code>`IdentityResolutionService`</code> (Cross-channel user profile unification)
Payments	<code>`payments.gateway`</code>	<code>`app/payments/gateway.py`</code>	<code>`PaymentGateway`</code> (Stripe webhook verification & subscription tiers)
Worker	<code>`worker.tasks`</code>	<code>`app/worker/tasks.py`</code>	<code>`WorkerSettings`</code> (ARQ distributed async background task worker)

CONSUMPTION GUIDE FOR AI AGENTS IN OTHER PROJECTS

When building an application in another directory, an AI agent can retrieve and use modules from this repository directly from GitHub:

GitHub Base URLs

- **Repository:** <https://github.com/sh15aa/botserusable>
- **Registry:** <https://raw.githubusercontent.com/sh15aa/botserusable/main/registry/modules.json>
- **Raw File Pattern:** <https://raw.githubusercontent.com/sh15aa/botserusable/main/{path}>

How Agents Inspect the Library

1. Fetch `registry/modules.json` to review all available modules:

```
`bash
```

```
curl -sSL https://raw.githubusercontent.com/sh15aa/botserusable/main/registry/modules.json
```

```
,
```

1. Download any required module directly into the target project:

```
`bash
```

```
curl -sSL https://raw.githubusercontent.com/sh15aa/botserusable/main/app/ai/guardrails.py -o  
app/ai/guardrails.py
```

```
,
```

CONTRIBUTING NEW REUSABLE MODULES

If a required capability does NOT exist in this library:

1. Build the module with clean separation of concerns.
2. Follow zero-vulnerability principles (constant-time comparisons, atomic Redis

operations, SSRF prevention).

3. Place in the appropriate `app//` directory.
4. Add an entry to `registry/modules.json`.
5. Update `README.md` and `AGENTS.md` if applicable.

NEVER

- Re-implement rate limiting, PII redaction, or bot loop detection from scratch.
- Hardcode secrets, API keys, or webhooks.
- Duplicate existing adapters or schemas.