Security Implementation Plan

Security is integrated across all project phases and layers to ensure the chatbot platform remains safe, compliant, and resilient against real-world threats. The security design aligns with **OWASP Top 10** and modern **AI system protection standards**.

Security Objectives

- Protect user and business data during storage, processing, and communication.
- Prevent unauthorized access or abuse through robust authentication and rate limiting.
- Ensure AI and API components cannot be exploited or injected with malicious inputs.
- Maintain system visibility and control through secure logging and monitoring.

Implementation Details

1. Authentication & Authorization

- Implement **OAuth2** and **JWT-based** authentication for all users and businesses.
- Use **short-lived tokens** and refresh mechanisms to prevent token reuse.
- Enforce **Role-Based Access Control (RBAC)** for business admins and platform users.

2. Data Protection

- Enforce **HTTPS/TLS** for all communication (frontend, backend, APIs, integrations).
- Encrypt sensitive data (chat logs, credentials, business info) in **PostgreSQL (pgcrypto)**.
- Use environment variables and secret management in CI/CD to protect API keys.

3. Input Validation & AI Security

- Validate and sanitize all user inputs with **Pydantic** schemas.
- Prevent **prompt injection** and ensure AI models only access intended data.
- Filter and review AI responses before displaying them to users.

4. API & Network Security

- Apply **rate limiting and throttling** on key endpoints to prevent abuse.
- Enforce **CORS with domain whitelisting** for integrations (WhatsApp, Facebook, Web).
- Use **Docker container isolation** and vulnerability scans (Trivy, Bandit).

5. Monitoring & Testing

- Enable **structured security logging** (Loguru) for authentication and API events.
- Mask sensitive data in logs.
- Automate **OWASP ZAP** and **Bandit** scans in GitHub CI/CD pipeline.
- Conduct periodic **penetration testing** and performance checks before release.

OWASP Top10 s

1. Authentication & Session Security (A07: Identification & Authentication Failures)

Already covered: OAuth2 + JWT.
Add:

- **Token rotation & short TTL** (short-lived JWTs, refresh tokens stored securely).
- **Device/session revocation** — allow logout from all sessions.
- **Secure password policy** + account lockout on repeated failed logins.
- Use **FastAPI's OAuth2PasswordBearer** securely (no token in URL).

---

2. Access Control & Authorization (A01: Broken Access Control)

- Enforce **RBAC (Role-Based Access Control)** between businesses, admins, and normal users.
- Prevent access to chat histories or admin routes unless authorized.
- Use **path- and object-level access control** (e.g., users can only view their own chat logs).
- Validate all incoming API requests against permissions.

---

3. Data Protection & Privacy (A02: Cryptographic Failures)

- Encrypt sensitive data (e.g., chat logs, business info) in **PostgreSQL using pgcrypto**.
- Never store plain tokens, passwords, or API keys.
- Use **.env** for credentials (never hardcode).
- Force HTTPS for all communications, including AI API calls.

---

4. Injection & Input Validation (A03: Injection, A05: Security Misconfiguration)

- Sanitize **all incoming user messages** before processing (no injection into AI prompts).
- Use **parameterized queries** in PostgreSQL.
- Add **input schema validation** in FastAPI with Pydantic.

- Escape all user-provided content on the frontend (prevent XSS).
- Filter file uploads (if added later) by MIME type.

---

5. Prompt Injection & AI Layer Security (Emerging category, OWASP LLM Top 10)

Since you're using LLMs (Gemini/Qwen):

- Implement **input/output sanitization** before passing data to AI.
- Restrict model access to only what's needed (no sensitive data in prompts).
- Add a **safety layer** that checks responses before returning them to users.
- Limit what the AI can access through LangChain (no unrestricted file or system calls).

---

6. API Security (A08: Software and Data Integrity Failures)

- Require **API keys** or OAuth2 for external integrations (WhatsApp, Facebook, etc.).
- Add **Rate Limiting** + **Throttling** per endpoint.
- Use **CORS** with strict domain whitelisting.
- Validate all payloads with schemas (avoid mass assignment vulnerabilities).

---

7. Dependency & Infrastructure Security (A06: Vulnerable & Outdated Components)

- Use Dependabot or Safety CLI to scan dependencies weekly.
- Regularly update FastAPI, React, and Hugging Face libraries.
- Keep Docker images minimal and use **non-root users**.
- Scan Docker images for vulnerabilities (e.g., Trivy).

---

8. Logging & Monitoring (A09: Security Logging & Monitoring Failures)

- Implement **structured logging** (e.g., loguru in FastAPI).
- Log all login attempts, API errors, and suspicious activity.
- Mask sensitive data in logs (tokens, passwords).
- Set up **alerting** on failed login bursts or rate-limit triggers.

---

9. Server & Config Security (A05: Security Misconfiguration)

- Enforce **Content Security Policy (CSP)** in React.
- Disable directory listing and debug mode.
- Use secure headers (HSTS, X-Frame-Options, X-Content-Type-Options).
- Manage secrets securely (GitHub Actions Secrets + .env + Docker secrets).

---

10. Business Logic & Abuse Prevention (A10: SSRF, A04: Insecure Design)

- Validate that bots can’t be abused to spam or flood messages.
- Limit AI response length and frequency.
- Validate business IDs and message origins.
- Use rate limiting per user/IP to stop abuse of free tiers or spam attacks.