Phase 1: Database Creation & Enhancement

1. **Source Data Collection**
    - Gather **official documentation** from White paper, documents about HD wallet, multi-chain support, security models about 2FA, biometric auth.
    - (Create anticipated qna of product guides, setup, transactions, security)
    - Include **crypto security white papers and audit reports**.
    - Collect **API documentation** for wallet services.
    - Compile **common troubleshooting guides** (e.g., transaction failures, sync issues).
2. **Structured Knowledge Base**
    - Extract **Q&A pairs** covering:
        - Wallet **features & specifications**
        - **Security protocols & best practices**
        - **Transaction workflows**
        - **Error solutions**
        - **Integration guides** with dApps/DeFi protocols
    - **Remove all general knowledge** (unrelated to crypto wallets).

Phase 2: Pre-Training Modifications (Restrict Knowledge)

3. **Masked Domain Training**
    - Train **only on the curated database** (no general text).
    - Use **MLM (Masked Language Modeling)** to reinforce domain terms:
        - python
        - Copy
        - Download
        - `input_text = "To recover a [MASK] wallet, you need your seed phrase."`
4. **Vocabulary Pruning**
    - **Reduce tokenizer vocabulary** to **wallet-specific terms only**.
    - **Remove general language tokens** (e.g., "cake," "weather").
    - **Create a custom tokenizer** (prevents off-topic word generation).

Phase 3: Fine-Tuning (Enforce Strict Responses)

5. **Supervised Fine-Tuning**
    - Train on **exact Q&A pairs** from the database.
    - Add **rejection examples**:
        - python
        - Copy

        - Download

```
{
    "question": "How to cook pasta?",
    "answer": "I specialize in crypto wallets. Ask about transactions, security, or integrations."
}
```

6. **Train a Smaller, Constrained Model**
    - Use **knowledge distillation** to create a **smaller model** that **only reproduces database content**.
    - Ensures **no hallucination beyond the knowledge base**.

Phase 4: Deployment & Response Control

7. **Two-Stage Pipeline**
    - **Stage 1: Binary Classifier** (Wallet-related or not?)
        - If **not wallet-related** → **Auto-reject**.
    - **Stage 2: Vector Similarity Search**
        - Compare user query vs. database embeddings (e.g., `SentenceTransformer`).
        - Only answer if **match confidence > 90%**.
8. **Response Templates**
    - Format answers to cite sources:
        - *"According to Ledger's documentation: [answer]."*
        - *"The official procedure is: [step-by-step]."*
9. **Strict Confidence Thresholds**
    - Reject **low-confidence responses** (<90% match to database).
    - Log **rejected queries** for future expansion.

Phase 5: Maintenance & Updates

10. **Automated Documentation Scrapers**

    - Monitor **wallet GitHub repos, docs, forums** for updates.
    - **Retrain model monthly** with new data.
11. **Human Review Loop**
    - Flag **novel/unanswered questions** for expert review.
    - Gradually **expand database** based on real user queries.

Key Restriction Mechanisms

Stage	Technique	Purpose
Pre-Training	Masked domain training, vocab pruning	Remove general knowledge
Fine-Tuning	Q&A pairs + rejection examples	Teach model to say "I don’t know"
Inference	Binary classifier + similarity search	Block off-topic answers
Post-Processing	Confidence thresholds + templates	Ensure answers are database-backed

-