## Goals / Target:

- Ultra-low latency (target: <50ms)
- Stealth operation (undetectable during screen sharing or process tracking)
- Industrial-grade deployment (secure, offline-capable, and license-controlled)

---

## 🧱 Modified Project Structure (Low Latency + Stealth)

```
project-root/
│
├── /frontend-overlay/       ← Dev 1: Piyush
│   ├── overlay.rs           ← Tauri (Rust) or C++ UI
│   ├── styles.css           ← Lightweight styling
│   └── config.json          ← Theme, license, expiry
│
├── /screenshot-engine/      ← Dev 2: Modi & Tripathi
│   ├── capture_native.rs    ← Native OS APIs (Win32, Quartz, X11)
│   ├── ocr_native.cpp       ← PaddleOCR C++ or EasyOCR GPU
│   └── auth.rs              ← Local license + expiry check
│
├── /nlp-engine/             ← Dev 3: Nayan
│   ├── inference.onnx       ← MiniLM / DistilBERT model
│   ├── runtime.rs           ← ONNX/TensorRT wrapper
│   └── cache.db        ← FAISS or SQLite for fast lookup
│
└── /utils/
    ├── obfuscation.rs      ← Anti-debug, process hiding
    └── self_destruct.rs    ← Expiry-based cleanup
```

---

## 👤 Dev 1: UI/UX + Floating Overlay

### 🔨 Responsibilities:

- Build GPU-rendered overlay using Tauri (Rust) or C++.
- Ensure frameless, transparent, and excluded from screen capture.
- Integrate:
  - Screenshot trigger
  - Input bar
  - Response display
  - License expiry check

### 💡 Tech:

- Tauri + Rust or C++ with Qt
- SetWindowDisplayAffinity (Windows), CGShieldingWindowLevel (macOS)
- GPU rendering via Vulkan/DirectX

---

## 👤 Dev 2: Screenshot + OCR Engine

🔨 **Responsibilities:**

- Implement native screen capture (no Electron).
- Use PaddleOCR C++ or EasyOCR with GPU for fast text extraction.
- Handle license validation and expiry logic.

💡 **Tech:**

- Native OS APIs (Win32, Quartz, X11)
- PaddleOCR C++ (compiled binary)
- Local license file + encrypted expiry timestamp

---

## 👤 Dev 3: NLP Inference + Caching

🔨 **Responsibilities:**

- Replace GPT API with local ONNX/TensorRT model.
- Use MiniLM or DistilBERT for fast contextual inference.
- Implement FAISS or SQLite for caching frequent terms.
- Add self-destruct logic post-expiry.

💡 **Tech:**

- ONNX Runtime or TensorRT
- FAISS for vector search
- SQLite for logs and cache
- Obfuscation tools (UPX, Obfuscator-LLVM)

## 🔐 Security & Stealth Enhancements

| Feature | Implementation |
|---|---|
| Undetectable UI | GPU overlay, native window flags |
| Process Hiding | Rename binaries, obfuscate metadata |
| Offline Operation | No network calls, local models |
| License Control | Encrypted license file + expiry |
| Self-Destruct | Delete binaries/config after expiry |
| Tamper Detection | Hash checks, anti-debugging hooks |

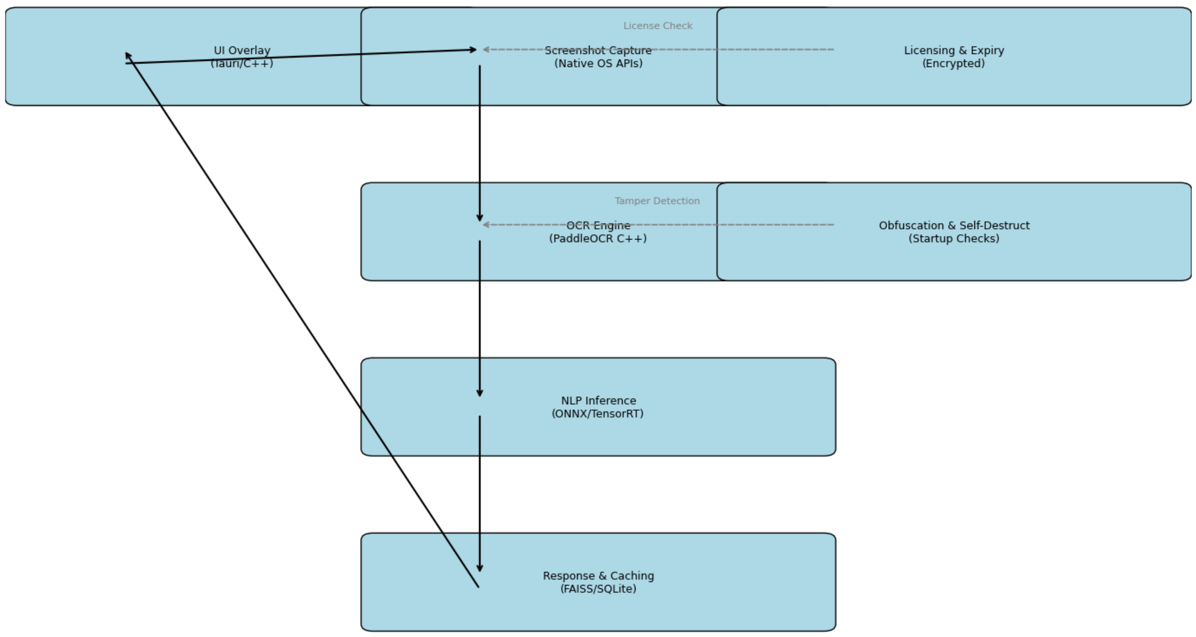
---

## 📊 Visual System Diagram Update

Here’s your updated system diagram tailored for a low-latency, stealth-first, industrial-grade application:

---

## 🔍 Diagram Highlights



- **UI Overlay (Tauri/C++): Always-on-top, GPU-rendered, excluded from screen capture.**
- **Screenshot Capture: Native OS APIs for ultra-fast, undetectable screen grabs.**
- **OCR Engine: PaddleOCR C++ for high-speed, local text recognition.**
- **NLP Inference: ONNX/TensorRT running MiniLM or DistilBERT for instant context.**
- **Response & Caching: FAISS or SQLite for sub-ms retrieval of known terms.**

- **Licensing & Expiry: Encrypted license file with time-bound access control.**
- **Obfuscation & Self-Destruct: Startup checks for tampering, expiry-triggered cleanup.**