ENTER SOFT SECURITY – TECHNICAL INTERNSHIP ASSIGNMENT

AI Code Remediation Microservice (LLM + Local Inference + Optional RAG)

Assignment Version: 1.0
Confidential – For Candidate Evaluation Only

---

1. Introduction

Entersoft Security is developing an AI-powered Remediation-as-a-Service (RaaS) platform that generates secure code fixes using advanced Large Language Models (LLMs) and automated tooling.
This assignment evaluates your ability to design, develop, and test a microservice that uses a **locally-run open-source LLM** to analyze and correct vulnerable code snippets.

The task is intentionally designed to assess practical engineering skills including model serving, prompt engineering, backend development, retrieval augmentation (optional), and system-level thinking.

---

2. Objective of the Assignment

The objective is to build a **local AI Code-Fix Microservice** that:

1. Runs an open-source coder/instruction model locally.
2. Exposes a FastAPI endpoint /local_fix.
3. Generates a secure version of the input code along with a diff and explanation.
4. Logs key metrics such as token usage and latency.
5. (Optional, but strongly beneficial) Implements a minimal RAG-style retriever to include contextual remediation guidelines.

Candidates may complete this assignment using CPU only; GPU is optional but preferred.

---

3. Scope of Work

3.1 Mandatory Requirements

The candidate must complete the following:

A. Local LLM Model Inference

Run any open-source LLM locally. Candidates may choose from:

- Qwen2.5-Coder (1.5B, 7B)

- StarCoder2 (3B, 7B)
- DeepSeek-Coder (1.3B or 6.7B)
- Mistral-7B-Instruct
- Any other open coder model you prefer

Inference may be implemented using:

- vLLM
- Hugging Face transformers
- TGI (optional)

GPU usage is optional; CPU-only systems are acceptable.

B. FastAPI Microservice

Implement a FastAPI service with a POST endpoint:

POST /local_fix

Input JSON:

```
{
  "language": "java",
  "cwe": "CWE-89",
  "code": "<vulnerable snippet>"
}
```

Output JSON:

```
{
  "fixed_code": "…",
  "diff": "…",
  "explanation": "…",
  "model_used": "…",
  "token_usage": {
     "input_tokens": 0,
     "output_tokens": 0
  },
  "latency_ms": 0
}
```

C. Logging & Instrumentation

- Log input token count
- Log output token count
- Log total response latency
- Print or store logs (CSV, console, or file)

D. Testing Script

Provide a script test_local.py which:

- Sends at least three test vulnerabilities to the API
- Displays responses clearly
- Records latency

3.2 Optional (Recommended) Requirements

These components are not mandatory but will significantly strengthen your evaluation.

A. Mini Retrieval-Augmented Generation (RAG) Component

Implement a minimal retriever using FAISS or SentenceTransformers.

Example Directory Structure:

```
recipes/
  sql_injection.txt
  hardcoded_secret.txt
  xss_dom_based.txt
  ssrf_basic.txt
  jwt_validation_issue.txt
```

Steps:

1. Embed all recipe files
2. On request, compute similarity
3. Retrieve top-1 file
4. Inject its content into the LLM prompt as context

B. Dockerization

Provide a Dockerfile to run the service.

C. Unit Tests

Provide basic unit tests validating:

- Model loading
- API response schema
- RAG retrieval logic

---

4. Deliverables

The candidate must submit a public GitHub repository named:
ai-codefix-assignment-<yourname>

The repository must contain:

1. Complete FastAPI application
2. Local model inference code
3. test_local.py script
4. requirements.txt
5. README.md including:
    - Setup instructions
    - How the model was run
    - Example inputs and outputs
    - Observations about performance
    - Assumptions and limitations

Optional:

- RAG implementation
- Docker configuration

---

5. Evaluation Criteria

Category	**Weightage**	**Description**
Local Model Inference	30%	Ability to run an open LLM locally; correct usage of libraries; stable inference
API Functionality	15%	Correct endpoint, schema, and JSON outputs
Prompt Design	15%	Quality, structure, clarity, and consistency of prompt
RAG (Optional)	20%	Implementation of retrieval and integration into prompt
Diff & Explanation Quality	10%	Accuracy and clarity of generated fix
Logging & Metrics	10%	Token usage, latency tracking, observability

Total: 100 points

---

6. Submission Deadline

All submissions must be completed and uploaded within **5 days** from the date the assignment is shared.

Shortlisted candidates will be contacted for a brief technical discussion and final evaluation.

---

7. Disclaimers & Conditions

1. **Confidentiality:**
   All materials, code snippets, instructions, and communications shared by Entersoft Security are confidential.
   This assignment and any derivative work **may not be posted publicly, shared, or reused** beyond the recruitment process.

2. **Use of External Tools:**
   Candidates may use open-source or publicly available LLMs.
   Proprietary or paid APIs (such as OpenAI GPT-4) may be used but are **not required**.

3. **Original Work:**
   All submitted work must be original.
   Plagiarism or the use of auto-generated boilerplate without understanding will lead to disqualification.

4. **Intellectual Property:**
   Entersoft Security makes no claim on source code produced for the purposes of this

assignment.
However, candidates must not include any proprietary or confidential code from previous employers or academic institutions.

5. **Hardware Requirements:**
   GPU usage is optional.
   Solutions must run on CPU-only environments unless GPU access is stated.

6. **Fair Use Clause:**
   The assignment is purely for evaluating candidate proficiency and will not be used as production code.

---

8. Contact

For queries or clarifications, candidates may reach out to the recruitment team at:
careers@entersoftsecurity.com

Entersoft Information Systems Pvt. Ltd.

K.S. Chakradhar.

Sri Chakradhar Kurmapu
CEO