Problem Statement 1

AI-Generated Voice Detection (Tamil, English, Hindi, Malayalam, Telugu)

1. Introduction

AI systems can now generate very realistic human-like voices. Because of this, it is difficult to identify whether a voice recording was spoken by a real human or generated by an AI system.

In this problem, students must build an API-based solution that detects whether a given voice sample is AI-generated or Human, across five supported languages.

2. Supported Languages (Fixed)

Your system must support only these five languages:

- Tamil
- English
- Hindi
- Malayalam
- Telugu

Each request will contain one audio file in one of the above languages.

3. What You Need to Build

You must design and deploy a REST API that:

- Accepts one MP3 audio file at a time
- Audio will be sent only as Base64
- Analyzes the voice
- Returns whether the voice is:
  - `AI_GENERATED`
  - `HUMAN`
- Responds in JSON format
- Is protected using an API Key

4. Input Rules

- Audio format: MP3
- Input type: Base64 encoded
- One audio per request
- Audio must not be modified

5. API Authentication

Your API must validate an API Key sent in request headers.

API Key Header Format

- **x-api-key:** YOUR_SECRET_API_KEY

Requests without a valid API key must be rejected.

6. API Request (cURL Example)

Endpoint Example

- POST https://your-domain.com/api/voice-detection

cURL Request Example

```
curl -X POST https://your-domain.com/api/voice-detection \

 -H "Content-Type: application/json" \

 -H "x-api-key: sk_test_123456789" \

 -d '{

   "language": "Tamil",

   "audioFormat": "mp3",

   "audioBase64": "SUQzBAAAAAAAI1RTU0UAAAAPAAADTGF2ZjU2LjM2LjEwMAAAAAAA..."

 }'
```

7. Request Body Fields

Field	Description
language	Tamil / English / Hindi / Malayalam / Telugu
audioFormat	Always `mp3`
audioBase64	Base64-encoded MP3 audio

8. API Response Body (Success)

Example Response

- {
- "status": "success",
- "language": "Tamil",
- "classification": "AI_GENERATED",
- "confidenceScore": 0.91, // out of 1.0
- "explanation": "Unnatural pitch consistency and robotic speech patterns detected"
- }

9. Response Field Explanation

Field	Meaning

status	`success` or `error`
language	Language of the audio
classification	`AI_GENERATED` or `HUMAN`
confidenceScore	Value between 0.0 and 1.0
explanation	Short reason for the decision

10. Classification Rules (Strict)

Only one classification field is required:

- `AI_GENERATED` → Voice created using AI or synthetic systems
- `HUMAN` → Voice spoken by a real human

👉 `voiceSource` is removed because it is logically the same as `classification`.

11. Error Response Example

```
{
  "status": "error",
  "message": "Invalid API key or malformed request"
}
```

12. Evaluation Process

1. System sends one Base64 MP3 per request

2. Language will be one of the 5 supported languages
3. Your API analyzes the voice
4. JSON response is returned
5. Multiple requests are made for evaluation

13. Evaluation Criteria

Participants will be evaluated on:

- 🎯 Accuracy of AI vs Human detection
- 🌍 Consistency across all 5 languages
- 📦 Correct request & response format
- ⚡ API reliability and response time
- 🧠 Quality of explanation

14. Rules & Constraints

- ❌ Hard-coding results is strictly prohibited
- ❌ Misuse of data leads to disqualification
- ⚠️ External detection APIs may be restricted
- ✅ Ethical and transparent AI usage is mandatory

15. One-Line Summary

Build a secure REST API that accepts one Base64-encoded MP3 voice in Tamil, English, Hindi, Malayalam, or Telugu and correctly identifies whether it is AI-generated or Human.

16. Sample Reference Voice:

Drive link - sample voice 1.mp3

Problem Statement 2

Agentic Honey-Pot for Scam Detection & Intelligence Extraction

1. Introduction

Online scams such as bank fraud, UPI fraud, phishing, and fake offers are becoming increasingly adaptive. Scammers change their tactics based on user responses, making traditional detection systems ineffective.

This challenge requires participants to build an Agentic Honey-Pot — an AI-powered system that detects scam intent and autonomously engages scammers to extract useful intelligence without revealing detection.

2. Objective

Design and deploy an AI-driven honeypot system that can:

- Detect scam or fraudulent messages
- Activate an autonomous AI Agent
- Maintain a believable human-like persona
- Handle multi-turn conversations
- Extract scam-related intelligence
- Return structured results via an API

3. What You Need to Build

Participants must deploy a public REST API that:

- Accepts incoming message events
- Detects scam intent
- Hands control to an AI Agent
- Engages scammers autonomously
- Extracts actionable intelligence
- Returns a structured JSON response

- Secures access using an API key

4. API Authentication

- x-api-key: YOUR_SECRET_API_KEY
- Content-Type: application/json

5. Evaluation Flow

1. Platform sends a suspected scam message
2. Your system analyzes the message
3. If scam intent is detected, the AI Agent is activated
4. The Agent continues the conversation
5. Intelligence is extracted and returned
6. Performance is evaluated

6. API Request Format (Input)

Each API request represents one incoming message in a conversation.

6.1 First Message (Start of Conversation)

This is the initial message sent by a suspected scammer. There is no prior conversation history.

```
{
“sessionId”: “wertyu-dfghj-ertyui”,
  "message": {
    "sender": "scammer",
    "text": "Your bank account will be blocked today. Verify immediately.",
    "timestamp": 1770005528731
  },
  "conversationHistory": [],
  "metadata": {
    "channel": "SMS",
    "language": "English",
    "locale": "IN"
  }
}
```

6.2 Second Message (Follow-Up Message)

This request represents a continuation of the same conversation.
Previous messages are now included in `conversationHistory`.

```
{
“sessionId”: “wertyu-dfghj-ertyui”,
  "message": {
    "sender": "scammer",
    "text": "Share your UPI ID to avoid account suspension.",
    "timestamp": 1770005528731
  },
  "conversationHistory": [
    {
      "sender": "scammer",
      "text": "Your bank account will be blocked today. Verify immediately.",
      "timestamp": 1770005528731
    },
    {
      "sender": "user",

      "text": "Why will my account be blocked?",

      "timestamp": 1770005528731

    }

  ],

  "metadata": {

    "channel": "SMS",

    "language": "English",

    "locale": "IN"

  }

}
```

6.3 Request Body Field Explanation

`message` (Required)

The latest incoming message in the conversation.

Field	Description
sender	`scammer` or `user`
text	Message content
timestamp	Epoch time format in ms

`conversationHistory` **(Optional)**

All previous messages in the same conversation.

- Empty array (`[ ]`) for first message
- Required for follow-up messages

`metadata` (Optional but Recommended)

Field	Description
channel	SMS / WhatsApp / Email / Chat
language	Language used
locale	Country or region

7. Agent Behavior Expectations

The AI Agent must:

- Handle multi-turn conversations
- Adapt responses dynamically
- Avoid revealing scam detection
- Behave like a real human
- Perform self-correction if needed

8. Agent output should be like

```
{

  "status": "success",

  "reply": "Why is my account being suspended?"
```

}

9. Evaluation Criteria

- Scam detection accuracy
- Quality of agentic engagement
- Intelligence extraction
- API stability and response time
- Ethical behavior

10. Constraints & Ethics

- ❌ No impersonation of real individuals
- ❌ No illegal instructions
- ❌ No harassment
- ✅ Responsible data handling

11. One-Line Summary

> Build an AI-powered agentic honeypot API that detects scam messages, handles multi-turn conversations, and extracts scam intelligence without exposing detection.

12. Mandatory Final Result Callback (Very Important)

Once the system **detects scam intent** and the **AI Agent completes the engagement**, participants must **send the final extracted intelligence** to the GUVI evaluation endpoint.

This is **mandatory** for evaluation.

Callback Endpoint

POST https://hackathon.guvi.in/api/updateHoneyPotFinalResult

Content-Type: application/json

Payload to Send

Participants must send the following JSON payload to the above endpoint:

{

```
  "sessionId": "abc123-session-id",

  "scamDetected": true,

  "totalMessagesExchanged": 18,

  "extractedIntelligence": {

    "bankAccounts": ["XXXX-XXXX-XXXX"],

    "upiIds": ["scammer@upi"],

    "phishingLinks": ["http://malicious-link.example"],

    "phoneNumbers": ["+91XXXXXXXXXX"],

    "suspiciousKeywords": ["urgent", "verify now", "account blocked"]

  },

  "agentNotes": "Scammer used urgency tactics and payment redirection"

}
```

🧠 When Should This Be Sent?

You must send this **only after**:

1. Scam intent is confirmed (`scamDetected = true`)
2. The AI Agent has completed sufficient engagement
3. Intelligence extraction is finished

This should be treated as the **final step** of the conversation lifecycle.

🧩 Field Explanation

Field	Description

sessionId	Unique session ID received from the platform for this conversation
scamDetected	Whether scam intent was confirmed
totalMessagesExchanged	Total number of messages exchanged in the session
extractedIntelligence	All intelligence gathered by the agent
agentNotes	Summary of scammer behavior

⚠️ Important Rules

- This callback is **mandatory for scoring**
- If this API call is not made, the solution **cannot be evaluated**
- The platform uses this data to measure:
  - Engagement depth
  - Intelligence quality
  - Agent effectiveness

💻 Example Implementation (Python)

```
intelligence_dict = {

   "bankAccounts": intelligence.bankAccounts,

   "upiIds": intelligence.upiIds,

   "phishingLinks": intelligence.phishingLinks,

   "phoneNumbers": intelligence.phoneNumbers,

   "suspiciousKeywords": intelligence.suspiciousKeywords
}


payload = {
   "sessionId": session_id,
   "scamDetected": scam_detected,
   "totalMessagesExchanged": total_messages,
   "extractedIntelligence": intelligence_dict,
   "agentNotes": agent_notes
}


response = requests.post(
   "https://hackathon.guvi.in/api/updateHoneyPotFinalResult",
   json=payload,
   timeout=5
)
```

🧾 Updated One-Line Summary

> Build an AI-powered agentic honeypot API that detects scam messages, engages scammers in multi-turn conversations, extracts intelligence, **and reports the final result back to the GUVI evaluation endpoint**.