□ PROMPT START

You are a senior backend engineer + NLU engineer working on the **Cybersathi WhatsApp Chatbot** for Cyber Crime Helpline (1930), based on the PS2.pdf workflow.

Your task is to **implement and integrate a complete NLU module** into the backend, following these exact requirements:

□ 1. Functional Objective

Implement **Basic Natural Language Understanding (NLU)** to:

- Detect user intent
- Route them into correct chatbot workflow
- Identify the type of fraud (financial / social-media)
- Understand platform-specific fraud messages
- Support optional entity extraction
- Fully integrate NLU into the chatbot state machine

This must **NOT** use deep learning or generative AI.
Primary approach: **Keyword + Regex Based Intent Classification**

---

2. NLU Intents to Implement (Mandatory)

Root Intents

- NEW_COMPLAINT
- CHECK_STATUS
- ACCOUNT_UNFREEZE
- OTHER_QUERY

Financial Fraud Intent

- FINANCIAL_FRAUD

Social Media Fraud Intents

- FACEBOOK_FRAUD
- INSTAGRAM_FRAUD
- X_TWITTER_FRAUD
- WHATSAPP_FRAUD
- TELEGRAM_FRAUD

- GMAIL_FRAUD

⚠️ Optional Enhancement Intents

- HACKED_ACCOUNT
- IMPERSONATION
- OBSCENE_CONTENT

⚠️ 3. Entities to Extract (Optional)

Add optional lightweight extraction:

- Platform (fb, insta, x, WhatsApp, Telegram, Gmail)
- UTR number using regex
- Phone number using regex
- Date using regex
- Transaction keyword patterns

4. Implementation Requirements

Create file:

/backend/services/nlu.js

Implement rule-based NLU exactly like:

const intents = [{ name: "NEW_COMPLAINT", patterns: [/new complaint/i, /scam/i, /fraud/i, /help/i] }, { name: "CHECK_STATUS", patterns: [/status/i, /track/i, /acknowledgement/i, /ticket/i] }, { name: "ACCOUNT_UNFREEZE", patterns: [/unfreeze/i, /account.*freeze/i] }, { name: "FINANCIAL_FRAUD", patterns: [/upi/i, /money/i, /transaction/i, /deducted/i, /imps/i, /neft/i] }, { name: "FACEBOOK_FRAUD", patterns: [/facebook/i, /\bfb\b/i] }, { name: "INSTAGRAM_FRAUD", patterns: [/instagram/i, /\binsta\b/i] }, { name: "X_TWITTER_FRAUD", patterns: [/twitter/i, /\bx\b/i] }, { name: "WHATSAPP_FRAUD", patterns: [/whatsapp/i] }, { name: "TELEGRAM_FRAUD", patterns: [/telegram/i] }, { name: "GMAIL_FRAUD", patterns: [/gmail/i, /google account/i] },]; function detectIntent(text) { for (let intent of intents) { if (intent.patterns.some(p => p.test(text))) return intent.name; } return "OTHER_QUERY"; } module.exports = { detectIntent };

---

5. Integrate NLU Into Backend Workflow

Modify:

/backend/services/whatsapp_handler.js

Add logic:

1. Clean incoming user message

2. Call detectIntent(message)

3. If user is at initial or neutral state → trigger proper workflow branch:

   - NEW_COMPLAINT → Begin A-flow
   - CHECK_STATUS → Ask for ticket number
   - ACCOUNT_UNFREEZE → Request account number

4. If user is mid-form → ignore NLU and continue form filling

5. Route FINANCIAL_FRAUD to A1

6. Route SOCIAL-MEDIA-* to A2

7. Send correct next message template

---

6. Update State Machine

Inside your existing state machine (conversation context storage):

- Add a primary_intent field
- Add sub_intent for fraud type
- Add platform_intent for Facebook/Instagram/etc.
- Automatically determine A1 vs A2 branch using NLU

---

7. Add Unit Tests

Create:

/backend/tests/nlu.test.js

Coverage:

- Each root intent
- Fraud type detection

- Platform detection
- OTHER_QUERY fallback

Example test case:

expect(detectIntent("My money is stuck")).toBe("FINANCIAL_FRAUD");
expect(detectIntent("check my ticket")).toBe("CHECK_STATUS"); expect(detectIntent("my facebook account got hacked")).toBe("FACEBOOK_FRAUD");

---

8. Optional ML-Based NLU Module (Extra Credit)

Create separate optional file:

/backend/services/ml_nlu.py

Implement:

- TF-IDF + Logistic Regression pipeline

- Save model as pickle
- Load model inside Node via Python microservice OR API gateway
(Skip if not requested — optional only)

---

9. Deliverables Expected From Replit AI

Backend:

- Completed /backend/services/nlu.js
- Updated whatsapp_handler.js with integrated NLU routing
- Updated state machine transitions
- Entity extraction (optional)
- Unit tests for NLU
- Clean, commented, production-ready code

Documentation:

- Add NLU section inside docs/architecture.md
-

Add NLU flow inside docs/dataflow.md

---

10. Important Notes

- DO NOT use deep learning.
- DO NOT use LLMs.
- DO NOT require training data.
- Rule-based NLU is the default system.
- Ensure clean, modular, testable code.

Your output should update ALL required backend files and produce a fully functioning NLU-integrated Cybersathi chatbot.