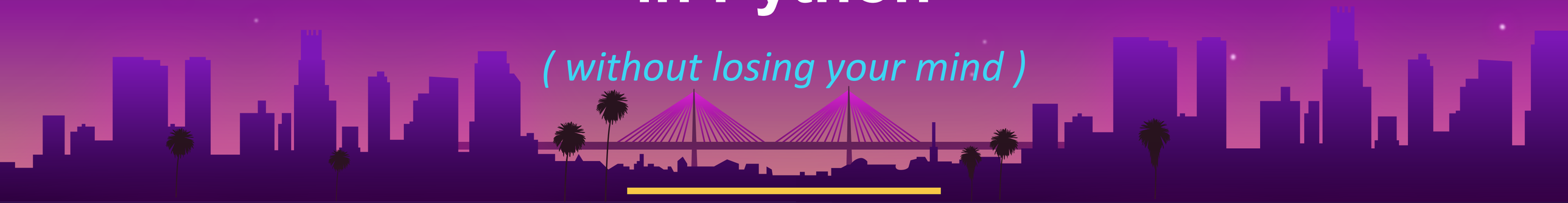
PyCon US 2026 · Long Beach

How to Build Your First Real-Time Voice Agent in Python

(without losing your mind)



Elizabeth Fuentes Leone ● Camila Hinojosa Añez

Hi! We're Camila & Elizabeth

Two builders who like making voice AI feel less intimidating.

Camila Hinojosa Añez

Community Builder

Software Engineer

🎓 Final-year **Civil Engineering in Computer Science** at UANDES, Chile. Researches eye-tracking in VR applied to financial decisions.

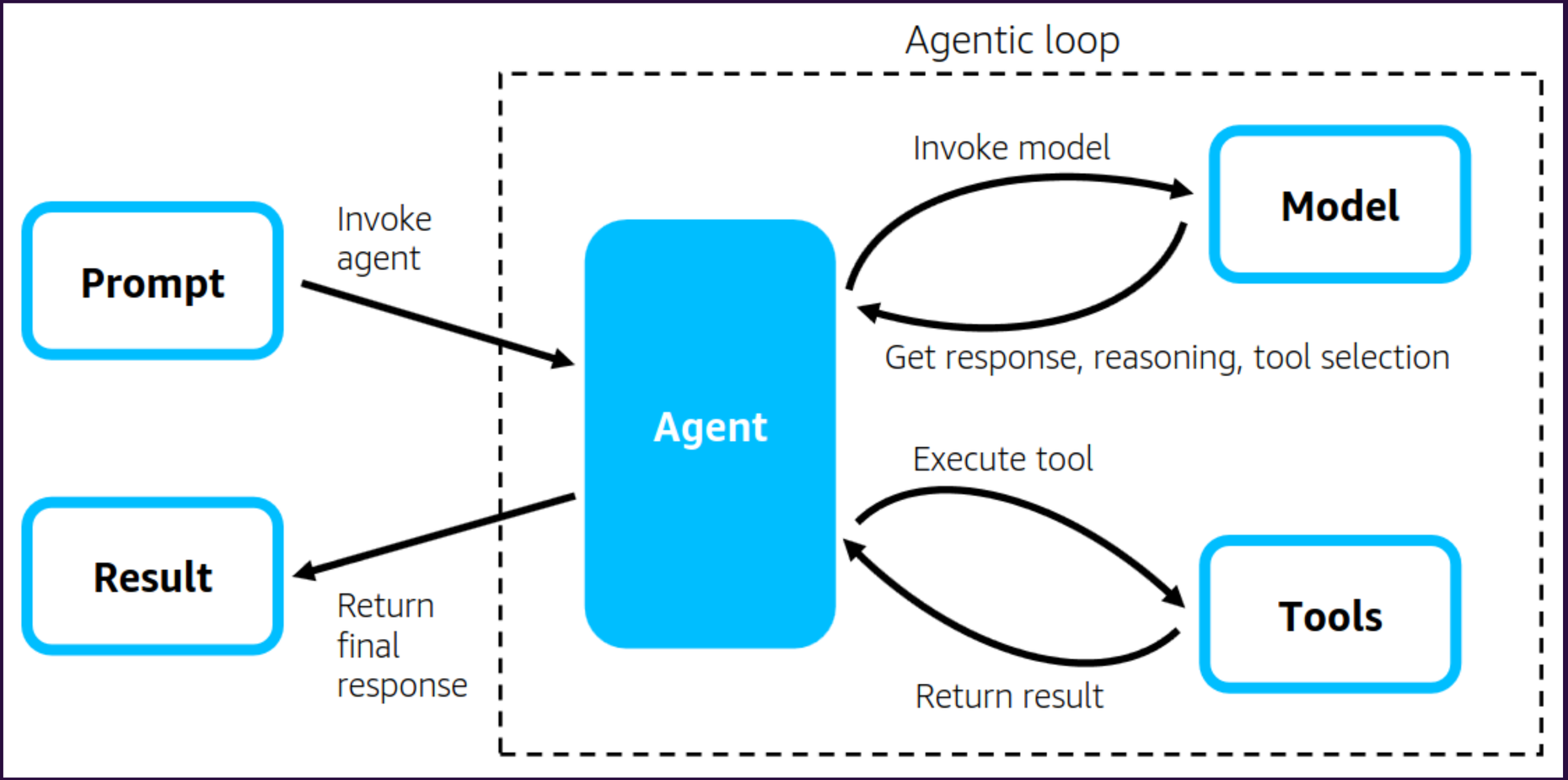
Elizabeth Fuentes Leone

Developer Advocate · AI / ML

RAG · agentic workflows

🎶 Helps developers ship **production-ready AI**. Bridges cutting-edge research and real-world code through tutorials, demos, and talks.

What is an Agent?



A loop: the model picks a tool, the tool runs, the result feeds back. Until done.

🤖 **Agent = model + tools + a loop.** *Reasoning happens in the model. Effects happen in the tools.*

What is a voice agent?

An agent that holds spoken dialogue and acts autonomously, in real time.

🗣️ **Voice agent = agent + ears + mouth.** *Same loop as before, just with audio at both ends.*

Spoken dialogue

Systems that hold spoken conversations with users and perform tasks autonomously.

Audio I/O

Voice in, voice out. Is the surface of the interaction.

Language understanding

Models that comprehend intent and reason about it.

Real-time interaction

Responds and acts as the conversation happens.

Voice Agentic Pipeline

Three swappable stages. Maximum flexibility and control.

Three stages, fully decoupled.
Mix-and-match best-of-breed services.

1

ASR

Automatic Speech Recognition
Voice → text

2

LLM

Large Language Model
Understand → respond

3

TTS

Text To Speech
Text → voice

Maximum flexibility. Pick the most accurate ASR for your language, an LLM that supports tools, or a custom TTS voice. each component is independent and replaceable.

Speech-to-Speech (S2S)

One model handles everything. Voice in → voice out. No text in between.

⚡ **Speed and naturalness win.** *You give up granular control. Pick S2S when latency matters most.*

Advantage

Preserves prosody and reduces latency. No intermediate text conversion step.

Trade-off

Less control: harder to debug, harder to extend with custom tools.

The components, end to end

Seven pieces working together to coordinate real-time conversation.

Audio capture

WebRTC streams mic audio

Voice Activity

Detects start / stop of speech

Speech Recognition

Spoken audio → text

Language model

Understands & responds

Tool execution

Performs actions

Speech synthesis

Text → audio

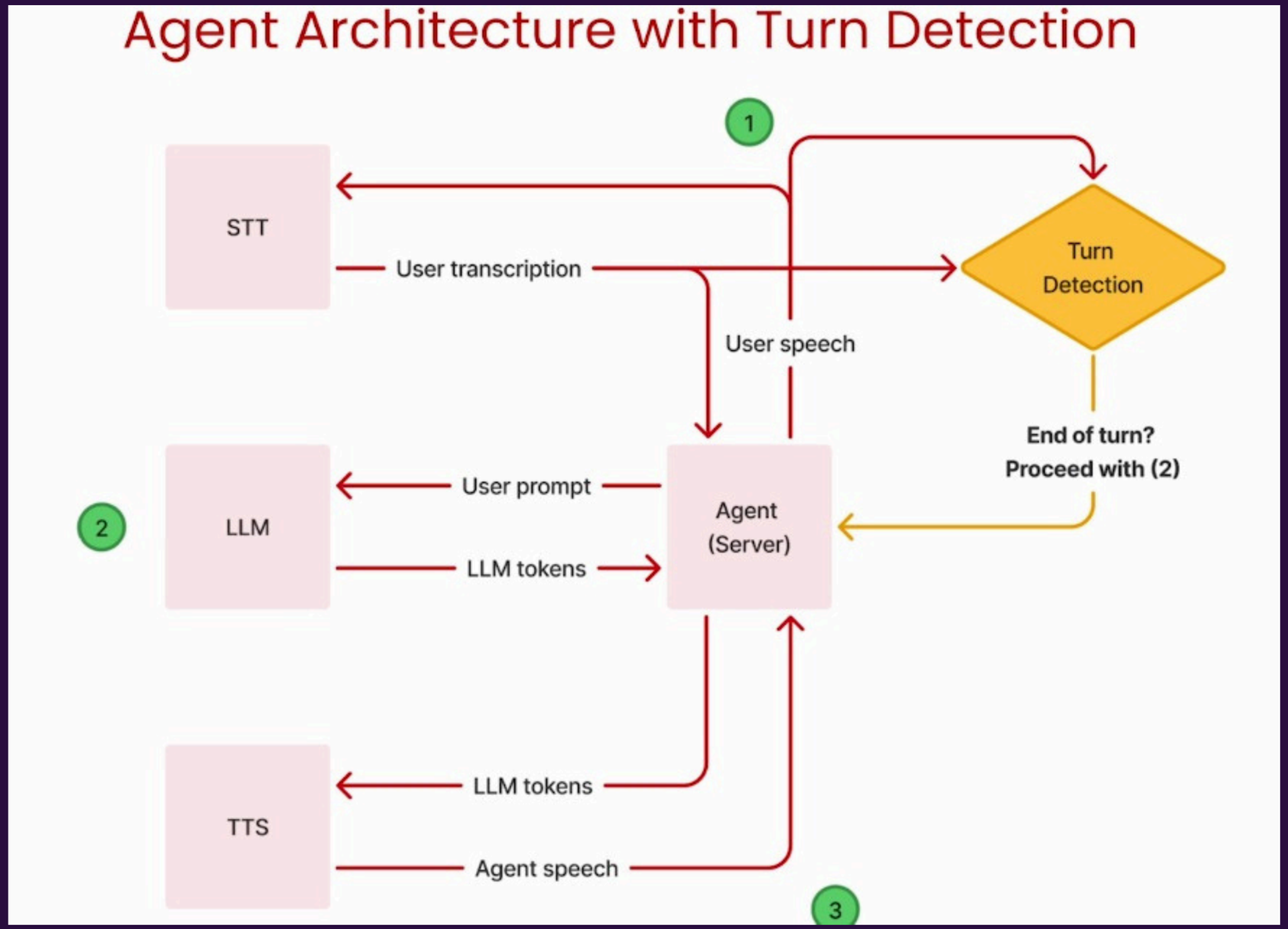
Orchestration

Coordinates everything

Lots of moving parts. *Next up —> a framework that wires them all together for you.*

Knowing when the user stops talking

The orchestration server decides: is the user done? Then it advances the pipeline.



Source: deeplearning.ai Building AI Voice Agents for Production

Pipecat: wires the components for you

Open-source Python framework for real-time voice and multimodal AI agents.

What it gives you

- Connects LLMs, STT and TTS into a single pipeline
- Real-time audio and video transports built-in
- Conversational latency tuned out of the box
- Phone bots, web assistants, interactive characters

Pipecat

frames flow top → bottom

open-source · Python

Pipecat handles the glue. *You focus on what the agent says, not on how the frames flow.*

Best practices

Five habits that make the difference between a demo and a deployable agent.

Optimize latency

Every saved millisecond compounds across the conversation.

Pick efficient models

Trade off size vs quality. Smaller + faster often wins.

Cache repetitive prompts

Use prompt caching for RAG -> same context, lower cost.

Ensure clean audio input

Garbage in, garbage out. Mic quality is foundational.

Build incrementally

Start with the simplest pipeline. Add complexity only when you need it.

Evaluation & monitoring

What to measure once it's running with real users.

Latency per turn

Time from user-stops-talking to bot-starts-talking. This is THE metric.

Functional correctness

Is it actually following instructions? Run scripted scenarios.

Speech errors

Catch mispronunciations and bad transcriptions — they're loud.

State & memory

Does context persist across turns the way you'd expect?

The Pipeline

💡 **The hard part is never the pipeline.** *It's the system prompt and the data your tool can reach.*

Frames flow through processors. You compose the orchestration.

```
pipeline = Pipeline([
    transport.input(),          # audio in   (browser mic)
    aggregators.user(),         # Silero VAD: stop-of-speech
    llm,                        # Nova Sonic: speech in/out
    transport.output(),         # audio out  (speaker)
    aggregators.assistant(),    # capture model's turn
])

task = PipelineTask(pipeline, params=PipelineParams(
    audio_in_sample_rate=16000, audio_out_sample_rate=24000,
    enable_metrics=True, enable_usage_metrics=True,
))
```

How frames flow

Each item is a processor. Audio chunks, text, and tool calls pass through in order.

Compose, don't rewrite

Want to log every word?
→ *insert a processor.*

Want to mute the bot?
→ *filter before output.*

speech → speech, one session

Nova Sonic takes audio in and gives audio back. No intermediate text. Lower latency, preserved prosody.

```python
def create_nova_sonic_service(config):
    audio = AudioConfig(
        input_sample_rate=16000,
        output_sample_rate=24000,
    )

    return AWSNovaSonicLLMService(
        region=config.aws_region,
        audio_config=audio,
        settings=AWSNovaSonicLLMService.Settings(
            model="amazon.nova-2-sonic-v1:0",
            voice="tiffany"           # matthew, amy…
            system_instruction=config.persona,
        ),
    )
```

Make it yours

- **`model`**

  swap the provider if you want :)
  pipeline remains unchanged

- **`voice`**

  tiffany, matthew, amy

- **`system_instruction`**

  personality + rules + closing
  behavior: spend your time here

Same 5-line pipeline. *Different brain = different agent.*

The Tool: an async function the model calls

Anything callable is fair game: SQL query, REST API, RAG retrieval, Spotify SDK.

1. Describe it to the model

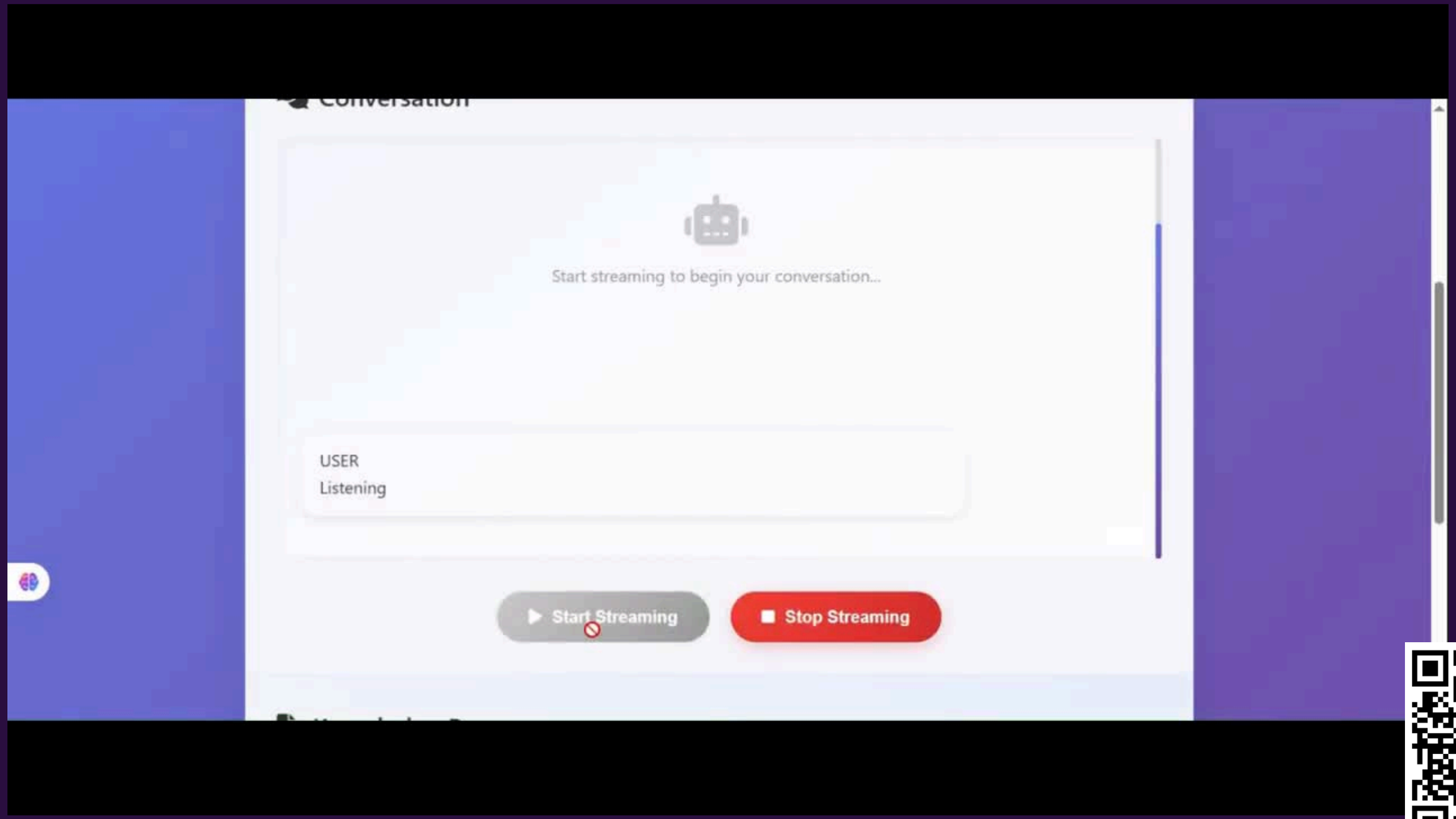
```
SEARCH_STORIES = FunctionSchema(
  name="search_stories",
  description=(
    "Search the story library
     for passages relevant to
     what the student asks…"),
  properties={
    "query": {
      "type": "string",
      "description": "…",
    }
  },
  required=["query"],
)
```

2. Register the handler

```
async def handle_search(params):
    query = params.arguments["query"]
    chunks = await search_stories(
        query, kb_id=cfg.kb_id,
    )
    await params.result_callback({
        "chunks": [
            c.to_dict() for c in chunks
        ]
    })

llm.register_function(
    "search_stories", handle_search,
)
```

🛠️ **Schema = prompt engineering.** The description tells the model ***when*** to reach for your tool.



YouTube: Speak English with Sonic Powered by Amazon Bedrock & Amazon Nova Sonic

bit.ly/practice-speaking

The prototype to production “chasm”

Excitement and potential

Challenges on the path to production

Meaningful business value

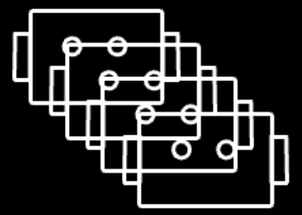
POC

Performance

Scalability

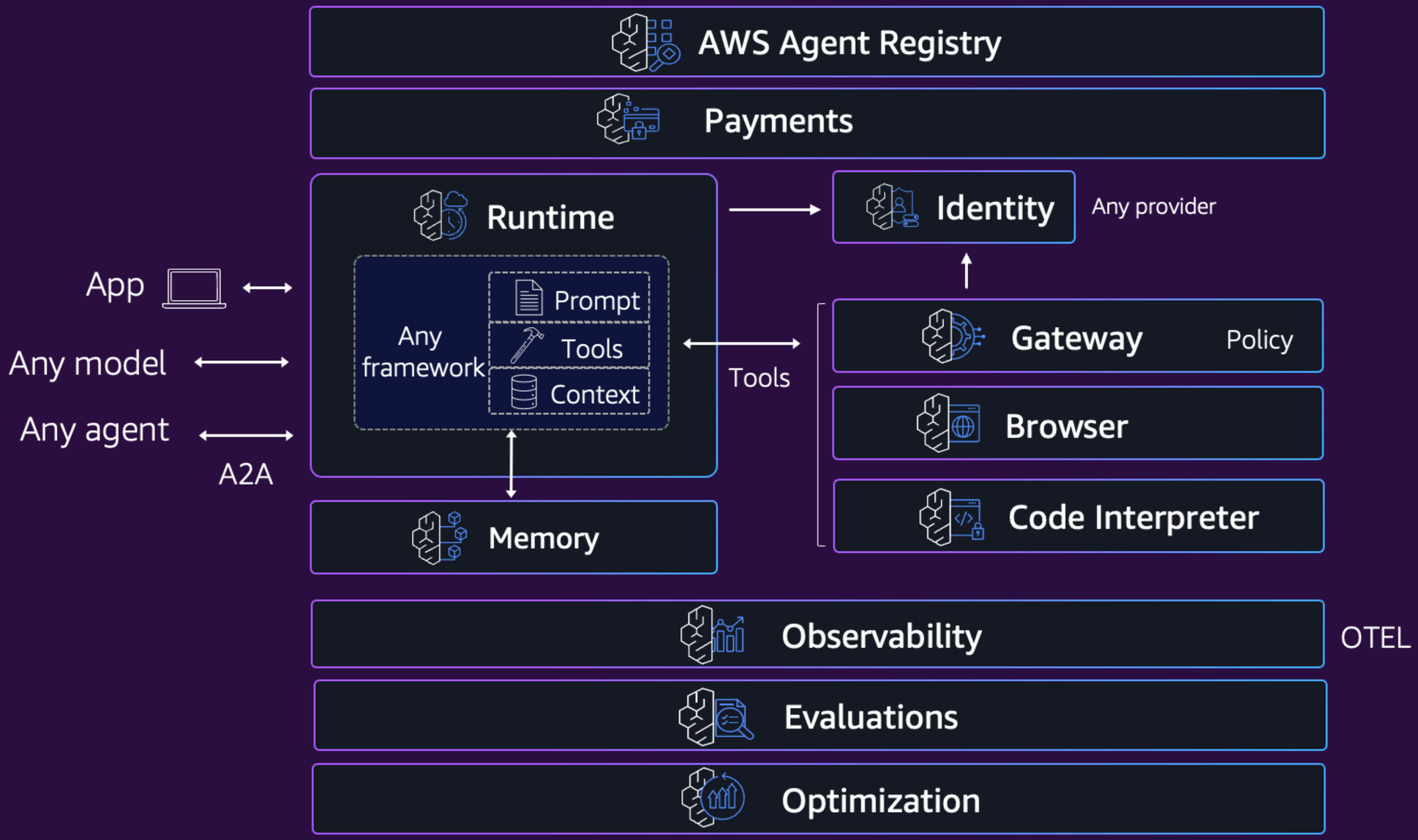
Security

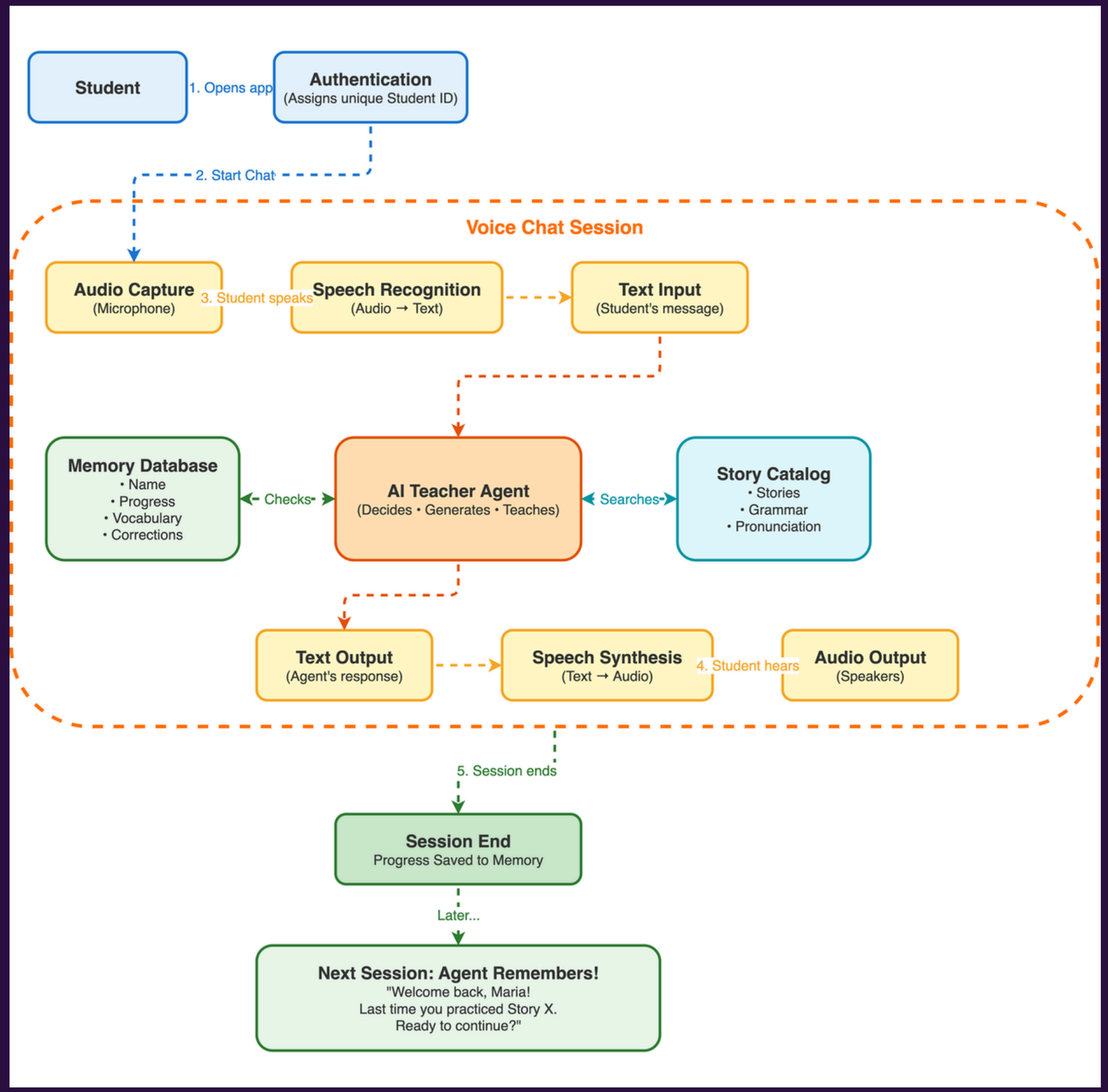
Governance

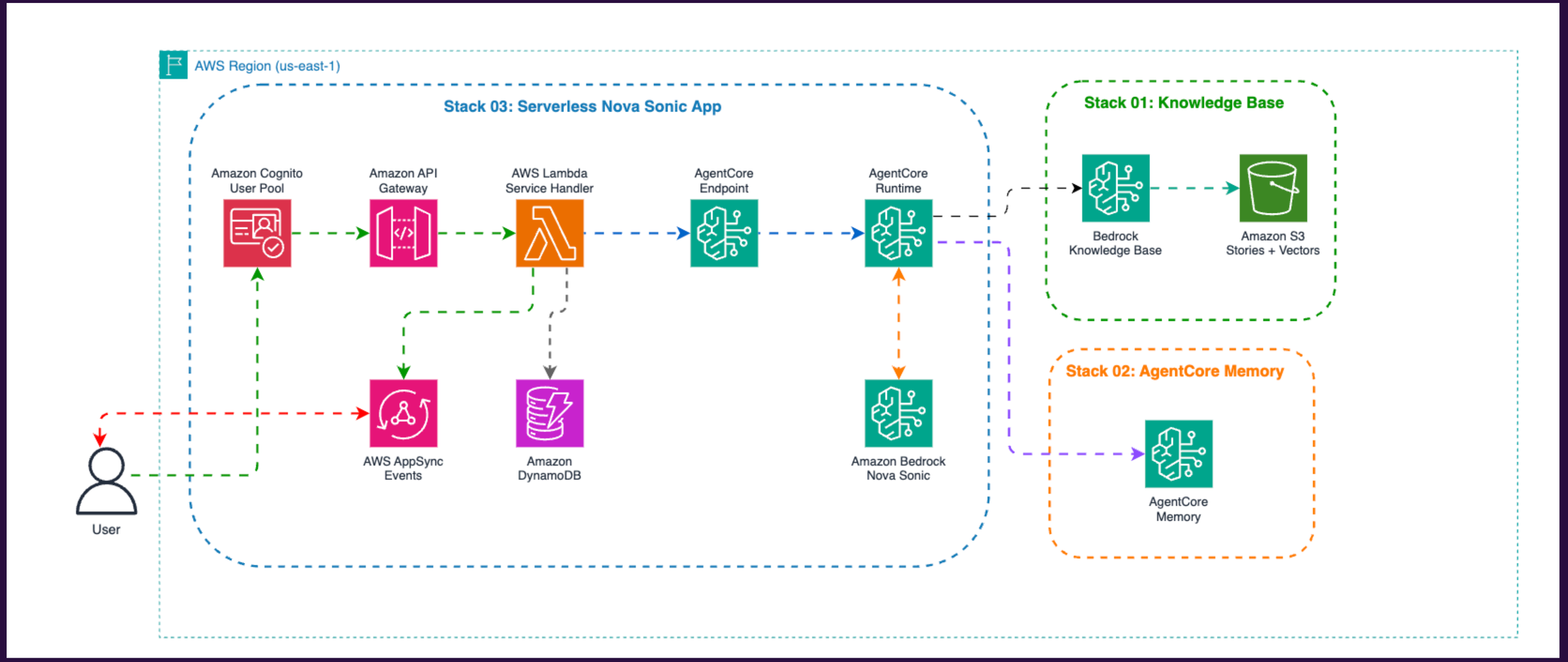


AI production agents

Amazon Bedrock AgentCore Platform




Student
1. Opens app
Authentication
(Assigns unique Student ID)
2. Start Chat
Voice Chat Session
Audio Capture
(Microphone)
3. Student speaks
Speech Recognition
(Audio → Text)
Text Input
(Student's message)
Memory Database
• Name
• Progress
• Vocabulary
• Corrections
Checks
AI Teacher Agent
(Decides • Generates • Teaches)
Searches
Story Catalog
• Stories
• Grammar
• Pronunciation
Text Output
(Agent's response)
Speech Synthesis
(Text → Audio)
4. Student hears
Audio Output
(Speakers)
5. Session ends
Session End
Progress Saved to Memory
Later...
Next Session: Agent Remembers!
"Welcome back, Maria!
Last time you practiced Story X.
Ready to continue?"


AWS Region (us-east-1)
Stack 03: Serverless Nova Sonic App
Amazon Cognito User Pool
Amazon API Gateway
AWS Lambda Service Handler
AgentCore Endpoint
AgentCore Runtime
Stack 01: Knowledge Base
Bedrock Knowledge Base
Amazon S3 Stories + Vectors
Stack 02: AgentCore Memory
AgentCore Memory
AWS AppSync Events
Amazon DynamoDB
Amazon Bedrock Nova Sonic
User

Live Demo

bit.ly/voice-agent-production

Thank you!

Now go build something that talks back.

Camila Hinojosa Añez

Community Builder · Software Engineer

linkedin.com/in/**camila-hinojosa-anez**

Elizabeth Fuentes Leone

Developer Advocate · AI / ML

linkedin.com/in/**lizfue**

PyCon US 2026 · Long Beach, CA · Friday 5:15 pm · Grand Ballroom A