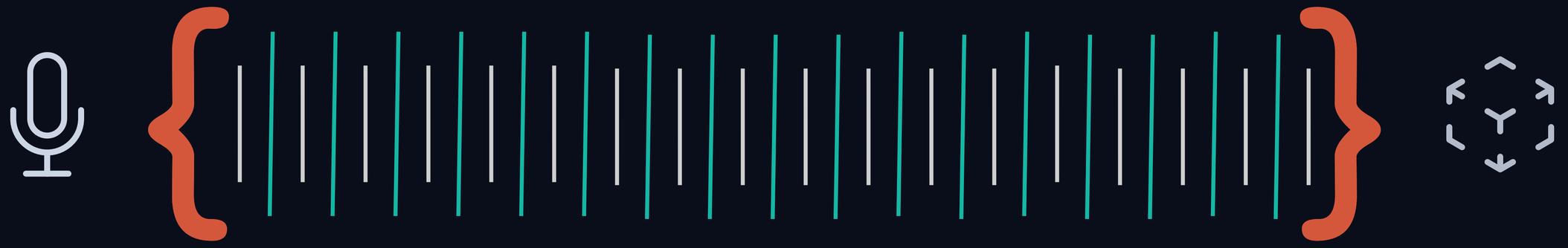
Voice-First MCP

Real-time tool calling through a spoken interface



Samyuktha M S
Software Developer, IBM

MCP Dev Summit, Bangalore
9 June 2026

Why I’m here ?

Hi, I’m Samyuktha

- Software Developer at IBM, ISL
- AI Enthusiast
- *15x Hackathon winner*
- Notable wins - *Google Agentic AI Hackathon, GrabHack 2.0, Chhalaang 4.0, Google Build and Blog Marathon*
- Speaker at GHCI ‘25, GHC ‘26, OpenSearch Con India ‘26, Devfest
- `linkedin.com/in/samyuktha-m-s`
- `github.com/samyuktha-12`

“Airport tak kitna lagega?”

How much will it cost to the airport?

Meet

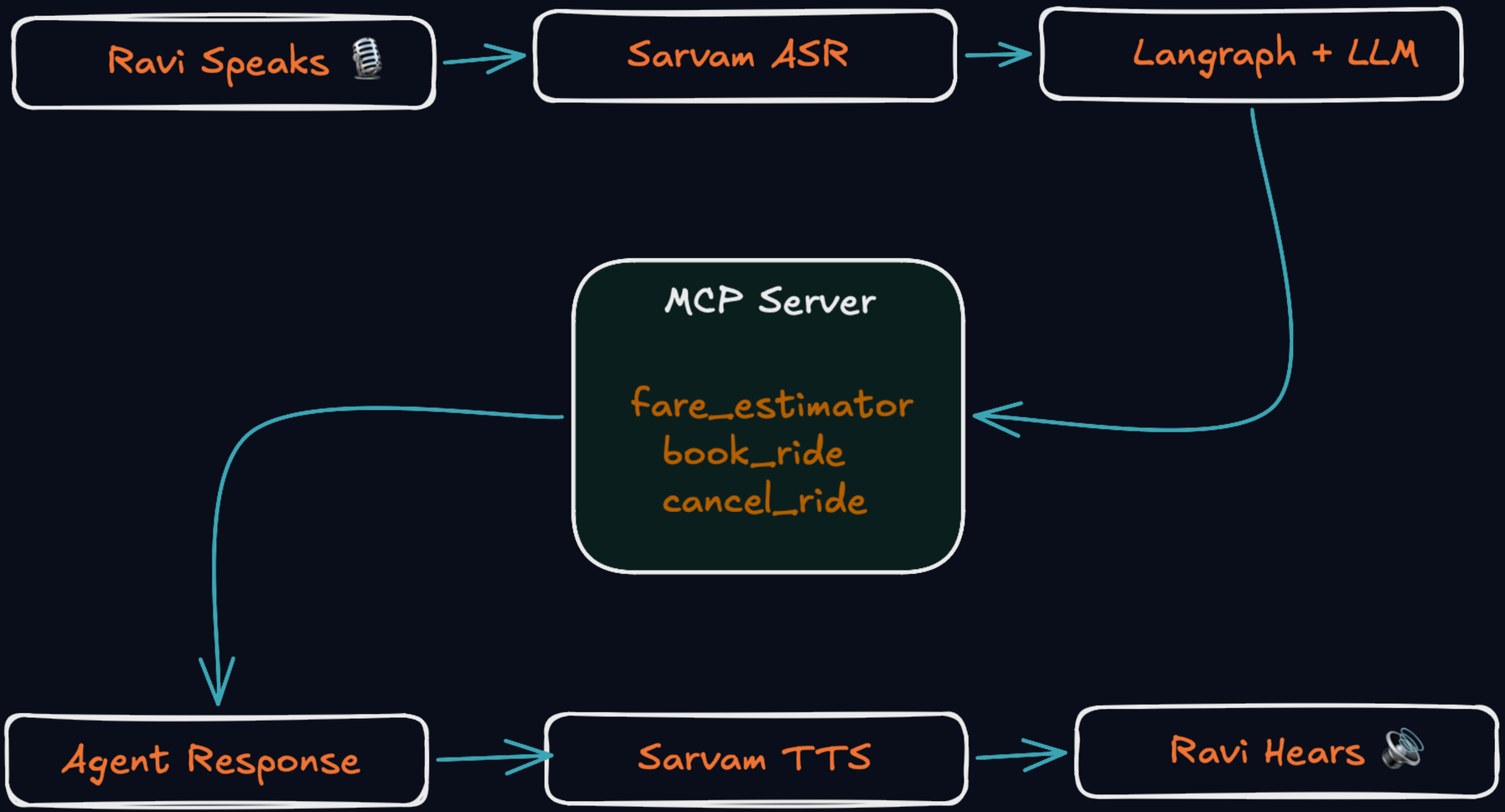
- Ravi, 38
- Auto driver in Bengaluru
- Speaks Kannada and broken English

- Last year his cooperative gave him a number to call.
- A voice agent.
- Three tools — fare, book, cancel. *Powered by MCP under the hood.*

Then I gave the number to real drivers like Ravi. It broke.

Four different ways

The Naive Architecture



Break #1: The 3.2 Second Silence

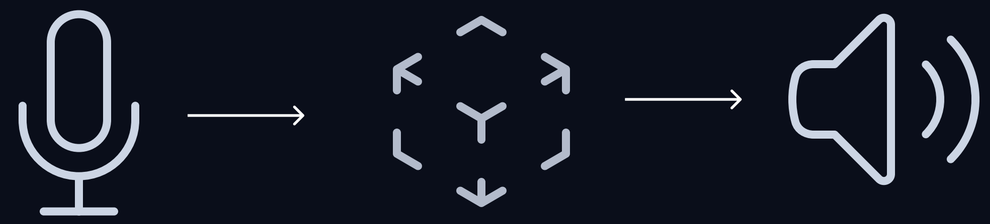
```
00:00.0   Ravi:    "Airport tak kitna lagega?"

00:01.4   ……       [VAD: silence detected]

00:02.1   ……       [Tool call dispatched]

00:03.8   Ravi:    "Hello? Suna nahi?"

00:04.6   Agent:   "About 420 rupees."
```



3.2 seconds of dead air. Ravi re-asked at 3.8s — before the agent even finished thinking.

Where the Time Went

Stage	Latency	What's happening	Verdict
VAD silence wait	800ms	Waiting to confirm end-of-speech	TUNABLE
ASR finalization	400ms	Locking partial transcript into final	PARTIAL
LLM tool decision	600ms	Choosing tool + extracting args	FIXED
fare_estimator	1100ms	Network + surge calc on partner API	FIXED
TTS first byte	300ms	Generating first audio chunk	STREAM
TOTAL	3200ms	p50 over 1,200 real calls	p95: 4,800ms

Two-thirds of the wait is sequential dead time the user hears as silence.
Only ~900ms is unavoidable compute.

THE INSIGHT - Don’t wait. Speculate.

Sarvam emits a partial transcript every ~120ms with a stability score. By 60% of the utterance, intent is usually decidable.

t= 240ms	"airport"	stab 0.41
t= 480ms	"airport tak"	stab 0.74 ◀ fire tool: fare_estimator confidence 0.83
t= 900ms	"airport tak kitna lagega"	stab 0.94

Fire when stability > 0.70 and tool confidence > 0.70.
Cancel the in-flight task if the user keeps talking.

FIX 01 - The speculative dispatcher

```python
class SpeculativeDispatcher:
    def __init__(self, mcp_client):
        self.client = mcp_client
        self.in_flight: dict[str, asyncio.Task] = {}

    async def on_partial(self, p: Partial):
        if p.stability < 0.70:
            return
        intent = await self.client.predict_tool(p.text)
        if intent.confidence < 0.70:
            return
        self.in_flight[p.id] = asyncio.create_task(
            self.client.call_tool(intent.tool, intent.args)
        )

    async def on_user_continues(self, pid: str):
        if task := self.in_flight.pop(pid, None):
            task.cancel()
```

The Result

- Latency 3,200ms → 1,400ms (−56%)
- Wasted calls 0% → 11%
- Net win 1,800ms saved per call

Break #2: The double booking

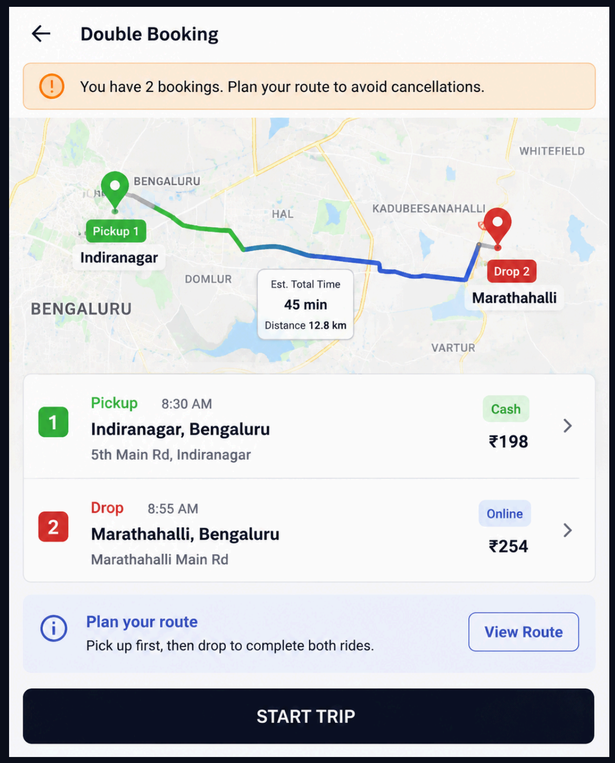
```
00:00.0   Ravi:     "Indiranagar se Whitefield book kar do"

00:00.5   ……        [dispatcher fires: book_ride to Whitefield]

00:01.2   Ravi:     "Ruko ruko! Whitefield nahi, Marathahalli"

00:01.8   ……        [new request received, but first ride is committed]

00:02.6   Agent:    "Your ride to Whitefield is confirmed.
                     Also booking to Marathahalli."
```



Two rides booked. One driver dispatched to Whitefield. ₹340 in cancellation fees, 2 minutes of an irate customer call

THE DIAGNOSIS - MCP can’t catch this

THE MCP TOOL CALL INTERFACE

```
client → server   tools/call
{ name: "book_ride", arguments: {...} }

server → client   tools/call response
{ content: [...], isError: false }
```

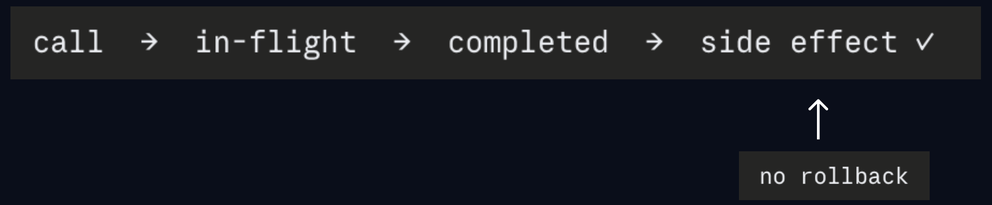
WHICH TOOLS BREAK

IDEMPOTENT	fare_estimator	safe to retry
SIDE-EFFECT	book_ride · payment	needs commit

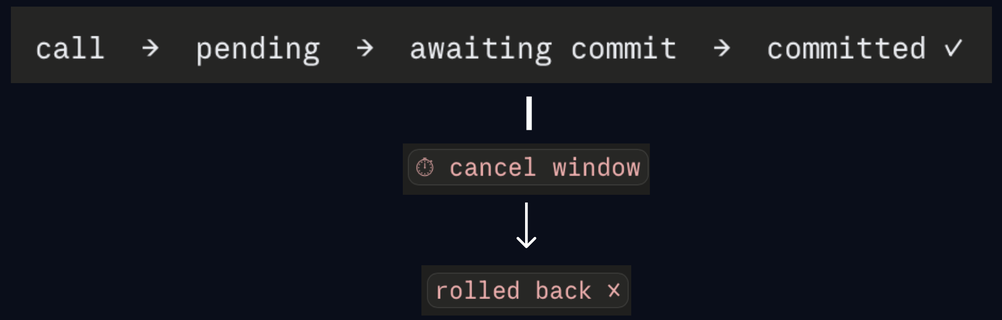
MCP treats both the same. That's the gap.

THE LIFECYCLE GAP

MCP today



What voice needs



THE INSIGHT - The commit boundary

Borrow from databases: two-phase commit.
The tool runs provisionally. The agent decides whether to keep it.

PHASE 1	PREPARE Tool executes in dry-run mode. Returns provisional_id, no side effect committed yet.
PHASE 2	COMMIT (after cancel window) User silent → commit the side effect. User revised → rollback via the tool's declared rollback handler.

Cancel window: 1.5s for bookings, 800ms for fare queries, instant for payments.

FIX 02 - The commit boundary

Tool manifest

```
{
  "name": "book_ride",
  "commit_required": true,
  "cancel_window_ms": 1500,
  "rollback_tool": "cancel_ride"
}
```

The Result

- Double bookings 7 / 100 calls → 0 / 100
- Cancellation fees ₹2,380 / day → ₹0 / day

Dispatcher wrapper

```
async def call_with_commit(client, tool, args):
    manifest = client.get_manifest(tool)
    if not manifest.commit_required:
        return await client.call_tool(tool, args)

    r = await client.call_tool(tool, args, dry_run=True)
    pid = r["provisional_id"]

    await asyncio.sleep(manifest.cancel_window_ms / 1000)

    if user_spoke_during(pid):
        return await client.call_tool(
            manifest.rollback_tool, {"id": pid})
    return await client.call_tool(
        tool, {"id": pid, "commit": True})
```

Break #3: The Robot Voice

Tool returned

```
{
  "fare": 420,
  "surge": 1.5,
  "duration_min": 18
}
```

Ravi heard

"Result colon open brace fare colon four two zero comma surge colon one point five comma duration min colon eighteen close brace."

Valid JSON. Read aloud. Ravi hung up.

FIX 03 - Verbalization in the manifest

Tool authors declare how each field should be spoken. The agent assembles. No JSON ever reaches the user.

```
{
  "name": "fare_estimator",
  "output_schema": {

    "fare": {
      "verbalize": "{value} rupees"
    },

    "surge": {
      "verbalize_if_above": [1.2, "surge pricing is on"]
    },

    "duration_min": {
      "verbalize": "about {value} minutes"
    }
  }
}
```

Ravi now hears:

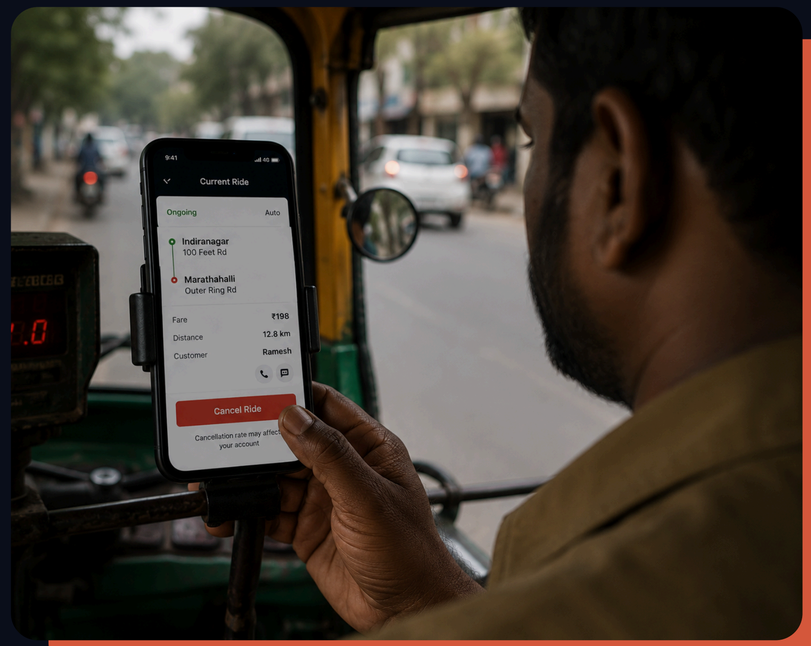
"That'll be 420 rupees, about 18 minutes. Surge pricing is on."

The Result

- Hang-up rate at first response 31% → 4%
- Naturalness score (1-5 mean) 1.8 → 4.2

Break #4: The Silent Failure

```
00:00.0  Ravi:    "Cancel kar do meri last booking"
00:00.4  ……       [cancel_ride → partner API]
00:02.4  ……       [HTTP 503, timeout]
00:02.4  ……       [agent: retrying … ]
00:06.1  ……       [second attempt: 503]
00:06.1  Ravi:    "Hello? Hello?"
00:09.3  Ravi:    [hangs up]
```



The API failed. The agent retried silently. Ravi heard 9 seconds of dead air and left. Errors over voice aren't errors. They're abandonment.

FIX 04 - Error verbalization + fillers

Every tool declares verbal templates per error code. Severity decides whether to retry, fall back, or stop. Fillers play during the wait, so silence never happens.

```
{
  "name": "cancel_ride",
  "errors": {

    "503": {
      "severity": "recoverable",
      "filler": "one sec, let me try again",
      "verbalize": "still having trouble. Want me to retry?"
    },

    "auth_failed": {
      "severity": "hard",
      "verbalize": "I can't access your account. Try the app for now."
    }
  }
}
```

THE FILLER TRICK

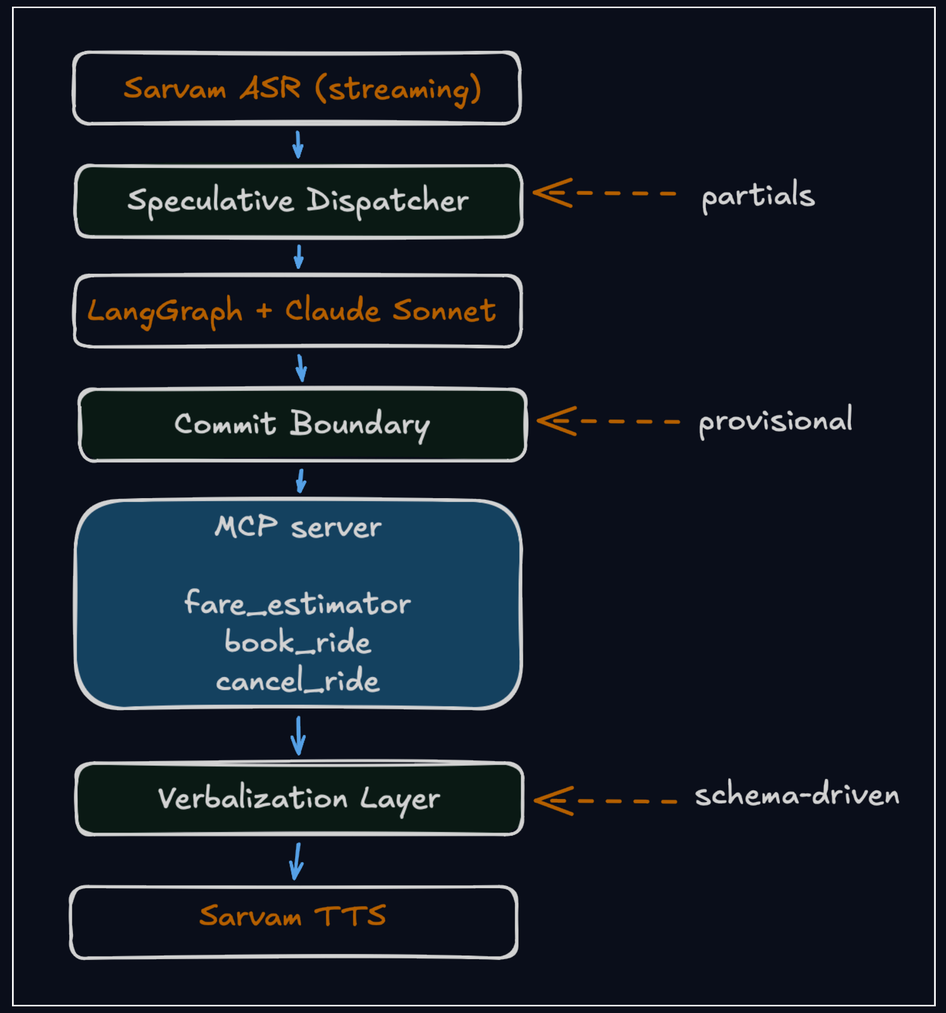
At 600ms after dispatch, play the filler.
Retry behind it. The user hears effort, not silence.

The Result

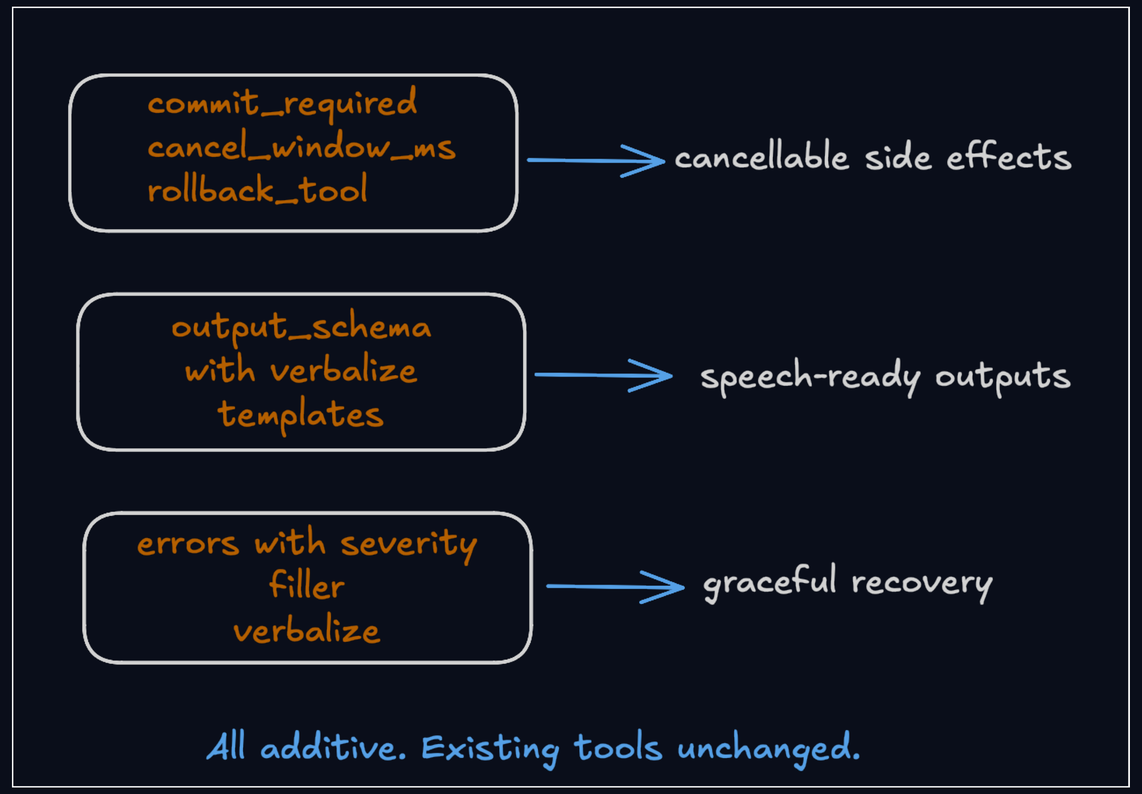
- Hang-ups during retries 43% → 6%
- Avg silence per error 9.3s → 1.1s

Ravi’s full system

ARCHITECTURE



PROPOSED MCP MANIFEST EXTENSIONS



Voice-MCP needs three things the spec doesn't have:
cancellation, verbalization, recovery.

DEMO

Recorded Demo: https://youtu.be/IWCa_JxcHIA

WHAT RAVI TAUGHT ME

01. Voice is not a UI layer
    It's a new MCP contract.

02. Wrap, don't replace.
    Cancellation, verbalization, recovery.

03. Latency is not a feature.
    It's the feature.

The full proposal, code, and demo: github.com/samyuktha-12/voice-mcp

THANK YOU

https://linktr.ee/mcp_dev_summit_blr_samyuktha

All resources linked here