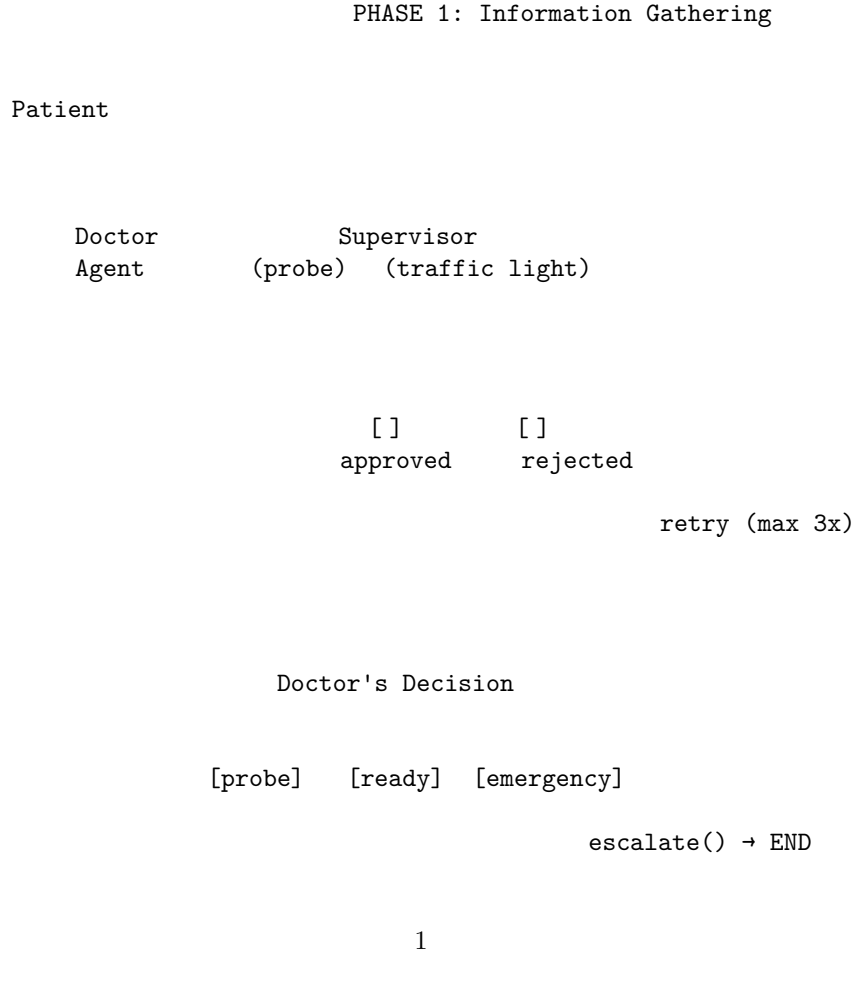
# System Design Document: AI Primary Care Consultation

## Overview

Amigo is a two-phase AI primary care chat that routes each patient message through specialized agents and a simple state machine. In Phase 1, a Doctor agent gathers symptoms and decides whether to keep probing, escalate for emergency care, or transition to assessment. Probe responses are quality-gated by a Supervisor with up to three retries (feedback is fed back into the Doctor). In Phase 2, a Counselor explains the assessment and plan in a structured format, or answers follow-up questions directly. The API keeps chat state (phase, collected info, assessment, plan) and signals emergencies and phase transitions to the UI.

---

## Architecture



PHASE 2

ask more questions

PHASE 2: Assessment & Plan

Patient

Counselor Agent

Explain assessment
Walk through plan
Answer follow-up questions

---

## Supervisor as Traffic Light

The Supervisor does NOT generate content. It only gives: - **Green** (`approved: true`) → Doctor's probe response goes to patient - **Red** (`approved: false`) → Doctor regenerates response using feedback

Notes: - Supervisor validation is only applied to `probe` responses. - `ready` and `emergency` responses bypass validation and proceed immediately.

Doctor generates response

Supervisor

[] []

Doctor tries again (up to 3x, with feedback)

After 3 failures: escalate()

Send to patient

---

## Retry Logic (in /api/chat)

```
let approved = false;
let attempts = 0;
let supervisorFeedback;

while (!approved && attempts < 3) {
  attempts++;

  // Doctor generates response (with optional feedback)
  doctorDecision = await runDoctor(conversation, collectedInfo, supervisorFeedback);

  // Ready or emergency skip validation
  if (doctorDecision.type === "ready" || doctorDecision.type === "emergency") {
    approved = true;
    break;
  }

  // Supervisor checks probe responses only
  supervisorResult = await runSupervisor(
    doctorDecision.response,
    doctorDecision.type,
    conversation,
    doctorDecision.assessment,
    doctorDecision.plan
  );

  approved = supervisorResult.approved;
  if (!approved) {
    supervisorFeedback = supervisorResult.reason || "Fix supervisor constraints";
  }
}

if (!approved) {
  await escalate("Supervisor validation failed after 3 attempts");
}
```

---

## Agent Responsibilities

| Agent | What it does | Output |
|---|---|---|
| **Doctor** | Talks to patient, gathers symptoms, decides next step | `{type, response, assessment?, plan?}` |
| **Supervisor** | Reviews Doctor's probe response, approve or reject | `{approved: true/false, reason?}` |
| **Counselor** | Explains assessment/plan or answers follow-ups | `{mode: "plan"\|"answer", assessment?, treatment_plan?, follow_up?, answer?}` |

---

## Doctor Decision Types

| Type | Meaning | What happens |
|---|---|---|
| `probe` | Need more info | Ask another question, stay in Phase 1 |
| `ready` | Have enough info | Provide assessment, move to Phase 2 |
| `emergency` | Urgent symptoms detected | Call `escalate()`, end conversation |

---

## Counselor Modes (Phase 2)

- **plan mode**: Structured assessment + treatment plan (3 items) + follow-up.
- **answer mode**: Direct answer when the user asks a specific question.
- The API formats plan mode into `ASSESSMENT / TREATMENT_PLAN / FOLLOW_UP` sections and normalizes the closing line.

---

## Chat State & API Response

State fields tracked by the API: - `phase`: `"collecting" | "counseling" | "escalated" | "completed"` (completed is currently unused) - `collectedInfo`: user messages collected in Phase 1 - `assessment`, `plan`: saved when the Doctor returns `ready`

API response fields used by the UI: - `response`: assistant message to display - `newState`: updated `ChatState` - `isEmergency`: highlights emergency/escalation messages - `showPhaseTransition`: triggers the Phase 1 → Phase 2 transition banner - `counselor`: raw `CounselorResult` for structured rendering

---

## File Structure

```
src/
  lib/
     types.ts                # Phase, ChatState, DoctorDecision, SupervisorResult
     agents.ts               # runDoctor(), runSupervisor(), runCounselor(), escalate()
     counselorFormatting.ts # normalizeTreatmentPlan()
  components/
     PhaseIndicator.tsx     # Shows Phase 1 vs Phase 2 visually
     ChatMessage.tsx        # Shows "Dr. Amigo" or "Counselor" label
     ChatInput.tsx
  app/
      api/chat/route.ts      # Main logic: doctor → supervisor → response
      page.tsx
```

---

## Escalation Triggers

| Trigger | When |
|---|---|
| Emergency symptoms | Doctor returns `type: "emergency"` |
| 3 supervisor rejections | Probe responses fail validation 3x |
| System error | Catch block in API |

**Stub (replace in production):**

```typescript
export async function escalate(reason: string): Promise<void> {
  console.log(`[ESCALATION] ${reason}`);
  // POST to external service here
}
```