

## 1. Chat Logic (Brain of the App)

```
// This function decides what the AI says back to you
const generateAIResponse = (userText: string = "", autoSpeak: boolean = false) => {
  setTimeout(() => {
    let responseText = "";
    let responseMood = "comforting";
    const lowerText = userText.toLowerCase();

    // TAMIL LANGUAGE LOGIC IN
    if (currentLang === 'ta-IN') {
      if (lowerText.includes("joke") || lowerText.includes("சிரிப்பு")) {
        // Tamil Jokes Database
        const jokes = [
          "டாக்டர்: உங்க உயிருக்கு இனிமே எந்த ஆபத்தும் இல்ல...",
          "ஆசிரியர்: 'காகம்' என்பதன் எதிர்ச்சொல் என்ன? மாணவன்: 'போகும்' சார்! 🤣",
          ];
        responseText = jokes[Math.floor(Math.random() * jokes.length)];
        responseMood = "funny";
      } else if (lowerText.includes("sad")) {
        responseText = "கவலைப்படாதீர்கள்! எல்லாம் விரைவில் சரியாகிவிடும். 💙";
        responseMood = "sad";
      }
    }
    // ENGLISH LANGUAGE LOGIC US
    else {
      if (lowerText.includes("sad")) {
        responseText = "I'm so sorry you're feeling this way. Would a warm bear hug help?";
        responseMood = "sad";
      } else if (lowerText.includes("joke")) {
        responseText = "Why did the scarecrow win an award? Because he was outstanding in his field! 🎃";
        responseMood = "funny";
      }
    }

    // Add message to chat and speak it out
    setMessages(prev => [...prev, { text: responseText, sender: "ai", mood: responseMood }]);
    if (autoSpeak) speakText(responseText);
  }, 1500);
};
```

## **2. Voice Input Logic (Ears of the App)**

```
/ This listens to your microphone
useEffect(() => {
  const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  recognitionRef.current = new SpeechRecognition();

  // Set language based on user choice (English or Tamil)
  recognitionRef.current.lang = currentLang === 'ta-IN' ? 'ta-IN' : 'en-US';

  recognitionRef.current.onresult = (event) => {
    // Convert voice audio to text
    let fullTranscript = "";
    for (let i = 0; i < event.results.length; ++i) {
      fullTranscript += event.results[i][0].transcript;
    }
    setTranscript(fullTranscript); // Update screen with text
  };
}, [currentLang]);
```

## Chat.tsx

```
import { useState, useRef, useEffect } from "react";
import { Link, useLocation } from "wouter";
import { PolarBear } from "@/components/ui/PolarBear";
import { Button } from "@/components/ui/button";
import { ScrollArea } from "@/components/ui/scroll-area";
import { motion, AnimatePresence } from "framer-motion";
import { Mic, Send, ChevronLeft, Settings, Volume2 } from "lucide-react";

interface Message {
  id: number;
  text: string;
  sender: "user" | "ai";
  mood?: "happy" | "sad" | "funny" | "neutral" | "comforting";
}

export default function Chat() {
  const [messages, setMessages] = useState<Message[]>([
    { id: 1, text: "Hi there! I noticed you might be feeling a bit down. Want to talk about it?", sender: "ai", mood: "happy" }
  ]);
  const [inputValue, setInputValue] = useState("");
  const [isTyping, setIsTyping] = useState(false);
  const scrollRef = useRef<HTMLDivElement>(null);
  const [, setLocation] = useLocation();

  // Get language from URL
  const searchParams = new URLSearchParams(window.location.search);
  const currentLang = searchParams.get("lang") || "en-US";

  // Parse query params for incoming voice messages
  useEffect(() => {
    const incomingMessage = searchParams.get("message");
    const shouldAutoSpeak = searchParams.get("autoSpeak") === "true";

    if (incomingMessage) {
      // Clear params to prevent re-sending on refresh
      const newUrl = new URL(window.location.href);
      newUrl.searchParams.delete("message");
      newUrl.searchParams.delete("autoSpeak");
      window.history.replaceState({}, "", newUrl.toString());
    }
  });

  // Add the message
```

```

const newMsg: Message = { id: Date.now(), text: incomingMessage, sender: "user" };
setMessages(prev => [...prev, newMsg]);
setIsTyping(true);

// Trigger AI response with the user's message text
generateAIResponse(incomingMessage, shouldAutoSpeak);
}

}, []);

// Auto-scroll to bottom
useEffect(() => {
  if (scrollRef.current) {
    scrollRef.current.scrollTop = scrollRef.current.scrollHeight;
  }
}, [messages]);

const speakText = (text: string) => {
  if ('speechSynthesis' in window) {
    window.speechSynthesis.cancel();
    const utterance = new SpeechSynthesisUtterance(text);
    const voices = window.speechSynthesis.getVoices();

    // Try to find appropriate voice based on language
    let preferredVoice;
    if (currentLang === 'ta-IN') {
      preferredVoice = voices.find(v => v.lang.includes('ta'));
    } else {
      preferredVoice = voices.find(voice =>
        (voice.name.includes("Female") || voice.name.includes("Samantha") ||
        voice.name.includes("Google US English"))
        && !voice.name.includes("Zira")
      );
    }

    if (preferredVoice) {
      utterance.voice = preferredVoice;
    }

    utterance.pitch = 1.1;
    utterance.rate = 0.9;
    window.speechSynthesis.speak(utterance);
  }
};

```

```

const generateAIResponse = (userText: string = "", autoSpeak: boolean = false) => {
  setTimeout(() => {
    let responseText = "";
    let responseMood: "happy" | "sad" | "funny" | "neutral" | "comforting" = "comforting";
    const lowerText = userText.toLowerCase();

    if (currentLang === 'ta-IN') {
      // TAMIL RESPONSES
      if (lowerText.includes("joke") || lowerText.includes("சிரிப்பு") || lowerText.includes("நகைச்சுவை")) {
        const jokes = [
          "டாக்டர்: உங்க உயிருக்கு இனிமே எந்த ஆபத்தும் இல்ல. நோயாளி: அப்ப நான் வீட்டுக்குப் போகலாமா? டாக்டர்: இல்ல... ஏன்னா, உங்க உயிர எமன்கிட்ட இருந்து காப்பாத்திட்டேன். ஆனா என் பீஸ் கட்டாம என்கிட்ட இருந்து தப்பிக்க முடியாது! 😊",
          "ஆசிரியர்: 'காகம்' என்பதன் எதிர்ச்சொல் என்ன? மாணவன்: 'போகும்' சார்! 🐶",
          "ஓருவர்: ஏன் சார் உங்க நாய் குறைச்சுக்கிட்டே இருக்கு? மற்றொருவர்: அதுக்கு தமிழ் தெரியாது, அதான் ஆங்கிலத்துல 'Bark' பண்ணுது! 🐕"
        ];
        responseText = jokes[Math.floor(Math.random() * jokes.length)];
        responseMood = "funny";
      } else if (lowerText.includes("sad") || lowerText.includes("கவலை")) {
        responseText = "கவலைப்படாதீர்கள்! எல்லாம் விரைவில் சரியாகிவிடும். நான் உங்களுடன் இருக்கிறேன். 💙";
        responseMood = "sad";
      } else {
        responseText = "நான் கேட்கிறேன். சொல்லுங்கள், இன்று உங்கள் நாள் எப்படி இருந்தது?";
        responseMood = "neutral";
      }
    } else if (currentLang === 'tanglish') {
      // TANGLISH RESPONSES
      if (lowerText.includes("joke") || lowerText.includes("comedy")) {
        const jokes = [
          "Teacher: Dei, why are you late? Student: Sir, late aah vandhalum latest aah varuven! 😊",
          "Friend 1: Machan, why is the math book sad? Friend 2: Because it has too many problems da! 😭"
        ];
        responseText = jokes[Math.floor(Math.random() * jokes.length)];
        responseMood = "funny";
      } else if (lowerText.includes("sad") || lowerText.includes("worry") || lowerText.includes("feel")) {
        responseText = "I'm sorry to hear that. Is there anything I can do to help you feel better?";
        responseMood = "neutral";
      }
    }
  });
}

```

```

        responseText = "Hey, don't worry pa! Everything will be alright. Just chill panu, I'm here
for you! 🐻";
        responseMood = "comforting";
    } else {
        responseText = "Super ji! Sollunga, what else is happening? I'm listening.";
        responseMood = "happy";
    }
} else {
    // ENGLISH RESPONSES (Existing Logic)
    if (lowerText.includes("sad") || lowerText.includes("unhappy") ||
lowerText.includes("depressed") || lowerText.includes("cry")) {
        responseText = "I'm so sorry you're feeling this way. I'm here for you. Would a warm
bear hug help?";
        responseMood = "sad";
    } else if (lowerText.includes("happy") || lowerText.includes("great") ||
lowerText.includes("good") || lowerText.includes("excited")) {
        responseText = "That makes me so happy to hear! Keep shining! ✨ What made your
day so special?";
        responseMood = "happy";
    } else if (lowerText.includes("joke") || lowerText.includes("laugh") ||
lowerText.includes("funny")) {
        const jokes = [
            "Why did the scarecrow win an award? Because he was outstanding in his field! 🎉",
            "What do you call a bear with no teeth? A gummy bear! 🐻",
            "Why don't scientists trust atoms? Because they make up everything!"
        ];
        responseText = jokes[Math.floor(Math.random() * jokes.length)];
        responseMood = "funny";
    } else {
        const randomResponses = [
            "I'm listening. Tell me more about that.",
            "That's interesting. How does that make you feel?",
            "I'm here for you, no matter what.",
            "Can you tell me more?"
        ];
        responseText = randomResponses[Math.floor(Math.random() *
randomResponses.length)];
        responseMood = "neutral";
    }
}

setMessages(prev => [...prev, {
    id: Date.now() + 1,
    text: responseText,
}
]
)

```

```

    sender: "ai",
    mood: responseMood
  ]]);
  setIsTyping(false);

  if (autoSpeak) {
    speakText(responseText);
  }
}, 1500);
};

const handleSend = () => {
  if (!inputValue.trim()) return;

  const newMsg: Message = { id: Date.now(), text: inputValue, sender: "user" };
  setMessages(prev => [...prev, newMsg]);
  const textToSend = inputValue;
  setInputValue("");
  setIsTyping(true);

  generateAIResponse(textToSend, false);
};

const currentBearState = isTyping ? "listening" : (messages[messages.length - 1]?.mood ===
"funny" ? "happy" : messages[messages.length - 1]?.mood || "comforting");

return (
  <div className="flex flex-col h-screen bg-background overflow-hidden">
    {/* Header */}
    <header className="p-4 flex items-center justify-between bg-white/50 backdrop-blur-md border-b z-10">
      <Link href={`/emotion-check`}>
        <Button variant="ghost" size="icon" className="rounded-full">
          <ChevronLeft className="w-6 h-6" />
        </Button>
      </Link>
      <div className="flex flex-col items-center">
        <span className="font-heading font-bold text-lg">MagizhBuddy</span>
        <span className="text-xs text-green-500 font-medium">Online & Listening</span>
      </div>
      <Link href="/settings">
        <Button variant="ghost" size="icon" className="rounded-full">
          <Settings className="w-6 h-6" />
        </Button>
      </Link>
    </header>
  </div>
);

```

```

        </Link>
    </header>

    {/* Bear Area - Fixed at top of chat content */}
    <div className="h-48 flex-shrink-0 flex items-center justify-center bg-gradient-to-b
from-blue-50/50 to-transparent">
        <div className="w-40 h-40">
            <PolarBear state={currentBearState as any} />
        </div>
    </div>

    {/* Chat Messages */}
    <ScrollArea className="flex-1 p-4" ref={scrollRef}>
        <div className="space-y-4 pb-4">
            <AnimatePresence>
                {messages.map((msg) => (
                    <motion.div
                        key={msg.id}
                        initial={{ opacity: 0, y: 10, scale: 0.9 }}
                        animate={{ opacity: 1, y: 0, scale: 1 }}
                        className={`flex ${msg.sender === "user" ? "justify-end" : "justify-start"}`}
                    >
                        <div className={`${flex flex-col ${msg.sender === "user" ? "items-end" :
"items-start"}'}`}>
                            <div
                                className={`max-w-[85%] p-4 rounded-2xl text-sm leading-relaxed shadow-sm relative group
${msg.sender === "user"
? "bg-primary text-white rounded-br-none"
: "bg-white text-gray-800 rounded-bl-none border border-gray-100"}`}
                            >
                                {msg.text}
                            
```

```

        )}
      </div>
    </div>
  </motion.div>
)}
{isTyping && (
  <motion.div
    initial={{ opacity: 0 }}
    animate={{ opacity: 1 }}
    className="flex justify-start"
  >
    <div className="bg-white px-4 py-3 rounded-2xl rounded-bl-none border border-gray-100 flex gap-1 items-center shadow-sm">
      <span className="w-2 h-2 bg-gray-300 rounded-full animate-bounce" />
      <span className="w-2 h-2 bg-gray-300 rounded-full animate-bounce delay-75" />
      <span className="w-2 h-2 bg-gray-300 rounded-full animate-bounce delay-150" />
    </div>
  </motion.div>
)
}
</AnimatePresence>
</div>
</ScrollArea>

/* Input Area */
<div className="p-4 bg-white border-t pb-8">
  <div className="flex gap-2 items-center bg-gray-50 p-2 rounded-full border border-gray-200 focus-within:ring-2 focus-within:ring-primary/20 transition-all">
    <Link href={`/voice?lang=${currentLang}`}>
      <Button size="icon" variant="ghost" className="rounded-full text-gray-500 hover:text-primary hover:bg-blue-50">
        <Mic className="w-5 h-5" />
      </Button>
    </Link>
    <input
      value={inputValue}
      onChange={(e) => setInputValue(e.target.value)}
      onKeyDown={(e) => e.key === "Enter" && handleSend()}
      placeholder={currentLang === 'ta-IN' ? "செய்தியை தட்டச்ச செய்யவும்..." : "Type a message..."}
      className="flex-1 bg-transparent border-none outline-none text-sm px-2"
    />
    <Button
      size="icon"
      className="rounded-full w-10 h-10 shrink-0"

```

```

        onClick={handleSend}
        disabled={!inputValue.trim()}
      >
    <Send className="w-4 h-4" />
  </Button>
</div>
</div>
</div>
);
}

```

## EmotionCheckin.tsx

```

import { useState } from "react";
import { Link, useLocation } from "wouter";
import { PolarBear } from "@/components/ui/PolarBear";
import { EmotionButton } from "@/components/ui/EmotionButton";
import { motion } from "framer-motion";
import { ChevronLeft, Globe } from "lucide-react";
import { Button } from "@/components/ui/button";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";

const emotions = [
  { id: "happy", emoji: "😊", label: "Happy", color: "#FEF3C7" }, // amber-100
  { id: "excited", emoji: "🤩", label: "Excited", color: "#FCE7F3" }, // pink-100
  { id: "neutral", emoji: "😐", label: "Okay", color: "#F3F4F6" }, // gray-100
  { id: "sad", emoji: "😢", label: "Sad", color: "#DBEAFE" }, // blue-100
  { id: "stress", emoji: "😡", label: "Stressed", color: "#EDE9FE" }, // violet-100
  { id: "angry", emoji: "😠", label: "Angry", color: "#FEE2E2" }, // red-100
];

export default function EmotionCheckin() {
  const [, setLocation] = useLocation();
  const [lang, setLang] = useState("en-US");

  const handleEmotionSelect = (emotionId: string) => {
    setLocation(`/chat?mood=${emotionId}&lang=${lang}`);
  }
}

```

```

};

return (
  <div className="min-h-screen bg-background p-6 flex flex-col relative">
    <div className="flex items-center justify-between mb-8">
      <div className="flex items-center">
        <Link href="/">
          <Button variant="ghost" size="icon" className="rounded-full">
            <ChevronLeft className="w-6 h-6" />
          </Button>
        </Link>
        <span className="font-heading font-bold text-lg ml-2">Check In</span>
      </div>

      {/* Language Selector */}
      <Select value={lang} onChange={setLang}>
        <SelectTrigger className="w-[140px] rounded-full bg-white border-gray-200">
          <Globe className="w-4 h-4 mr-2 text-gray-500" />
          <SelectValue placeholder="Language" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="en-US">English</SelectItem>
          <SelectItem value="ta-IN">தமிழ் (Tamil)</SelectItem>
          <SelectItem value="tanglish">Tanglish</SelectItem>
        </SelectContent>
      </Select>
    </div>

    <motion.div
      initial={{ opacity: 0, y: 10 }}
      animate={{ opacity: 1, y: 0 }}
      className="flex-1 flex flex-col items-center"
    >
      <h2 className="text-3xl font-heading font-bold text-center mb-2">
        {lang === 'ta-IN' ? 'இன்று நீங்கள் எப்படி உணர்கிறீர்கள்?' : 'How are you feeling today?'}
      </h2>
      <p className="text-muted-foreground text-center mb-8">
        {lang === 'ta-IN' ? 'பொருத்தமான ஒன்றைத் தட்டவும்' : 'Tap the one that fits best'}
      </p>

      <div className="w-40 h-40 mb-8">
        <PolarBear state="listening" />
      </div>
    
```

```

        </div>

        <div className="grid grid-cols-2 gap-4 w-full max-w-md">
          {emotions.map((emotion) => (
            <EmotionButton
              key={emotion.id}
              {...emotion}
              onClick={() => handleEmotionSelect(emotion.id)}
            />
          )));
        </div>
      </motion.div>
    </div>
  );
}

```

### 3. Notfoundtxt

```

import { Card,CardContent } from "@/components/ui/card";
import { AlertCircle } from "lucide-react";

export default function NotFound() {
  return (
    <div className="min-h-screen w-full flex items-center justify-center bg-gray-50">
      <Card className="w-full max-w-md mx-4">
        <CardContent className="pt-6">
          <div className="flex mb-4 gap-2">
            <AlertCircle className="h-8 w-8 text-red-500" />
            <h1 className="text-2xl font-bold text-gray-900">404 Page Not Found</h1>
          </div>

          <p className="mt-4 text-sm text-gray-600">
            Did you forget to add the page to the router?
          </p>
        </CardContent>
      </Card>
    </div>
  );
}

```

### 4. Setting.tsx

```
import { Link } from "wouter";
```

```

import { Button } from "@/components/ui/button";
import { ChevronLeft, User, MessageCircle, Bell, Info, LogOut } from "lucide-react";
import { motion } from "framer-motion";

export default function Settings() {
  const menuItems = [
    { icon: User, label: "Profile", href: "#" },
    { icon: MessageCircle, label: "Chat History", href: "#" },
    { icon: Bell, label: "Notifications", href: "#" },
    { icon: Info, label: "About MagizhBuddy", href: "#" },
  ];

  return (
    <div className="min-h-screen bg-background p-6">
      <div className="flex items-center mb-8">
        <Link href="/chat">
          <Button variant="ghost" size="icon" className="rounded-full">
            <ChevronLeft className="w-6 h-6" />
          </Button>
        </Link>
        <span className="font-heading font-bold text-lg ml-2">Settings</span>
      </div>

      <div className="space-y-6">
        <div className="bg-white p-6 rounded-3xl shadow-sm border border-gray-100 flex items-center gap-4">
          <div className="w-16 h-16 bg-blue-100 rounded-full flex items-center justify-center text-2xl">
            <UserIcon />
          </div>
          <div>
            <h3 className="font-heading font-bold text-xl">Guest User</h3>
            <p className="text-muted-foreground text-sm">MagizhBuddy Friend</p>
          </div>
        </div>

        <div className="space-y-3">
          {menuItems.map((item, index) => (
            <motion.button
              key={index}
              whileHover={{ scale: 1.02 }}
              whileTap={{ scale: 0.98 }}
              className="w-full bg-white p-4 rounded-2xl shadow-sm border border-gray-100 flex items-center justify-between group">
              <div>
                {item.icon}
              </div>
              <div>
                {item.label}
              </div>
            </motion.button>
          ))}
        </div>
      </div>
    </div>
  );
}

```

```

    >
    <div className="flex items-center gap-4">
      <div className="w-10 h-10 bg-gray-50 rounded-full flex items-center justify-center text-gray-600 group-hover:bg-primary/10 group-hover:text-primary transition-colors">
        <item.icon className="w-5 h-5" />
      </div>
      <span className="font-medium text-gray-700">{item.label}</span>
    </div>
    <ChevronLeft className="w-5 h-5 text-gray-300 rotate-180" />
  </motion.button>
)
}
</div>

<Button variant="destructive" className="w-full rounded-2xl h-12 mt-8">
  <LogOut className="w-4 h-4 mr-2" />
  Sign Out
</Button>

<p className="text-center text-xs text-muted-foreground mt-8">
  Version 1.0.0 • Made with ❤️
</p>
</div>
</div>
);
}

```

## 6. Voice.tsx

```

import { useState, useEffect, useRef } from "react";
import { Link, useLocation } from "wouter";
import { PolarBear } from "@/components/ui/PolarBear";
import { Button } from "@/components/ui/button";
import { VoiceWaveform } from "@/components/ui/VoiceWaveform";
import { motion, AnimatePresence } from "framer-motion";
import { X, Mic, Send, Edit2, RotateCcw } from "lucide-react";
import { Textarea } from "@/components/ui/textarea";

export default function Voice() {
  const [isRecording, setIsRecording] = useState(false);
  const [transcript, setTranscript] = useState("");
  const [isEditing, setIsEditing] = useState(false);
  const [location, setLocation] = useLocation();
  const recognitionRef = useRef<any>(null);

```

```

// Get language from URL
const searchParams = new URLSearchParams(window.location.search);
const currentLang = searchParams.get("lang") || "en-US";

useEffect(() => {
  if ('webkitSpeechRecognition' in window || 'SpeechRecognition' in window) {
    const SpeechRecognition = (window as any).SpeechRecognition || (window as any).webkitSpeechRecognition;
    recognitionRef.current = new SpeechRecognition();
    recognitionRef.current.continuous = true;
    recognitionRef.current.interimResults = true;
    // Map Tanglish/Tamil to appropriate codes, default to US English
    recognitionRef.current.lang = currentLang === 'ta-IN' ? 'ta-IN' : (currentLang === 'tanglish' ? 'en-IN' : 'en-US');

    recognitionRef.current.onresult = (event: any) => {
      // Rebuild transcript from scratch to avoid duplication
      let fullTranscript = "";
      for (let i = 0; i < event.results.length; ++i) {
        fullTranscript += event.results[i][0].transcript;
      }
      setTranscript(fullTranscript);
    };

    recognitionRef.current.onerror = (event: any) => {
      console.error("Speech recognition error", event.error);
      setIsRecording(false);
    };
  }
}, [currentLang]);

const handleSend = () => {
  const finalMessage = transcript || "I'm feeling a bit overwhelmed today and I just needed someone to talk to./";

  setLocation(`/chat?message=${encodeURIComponent(finalMessage)}&autoSpeak=true&lang=${currentLang}`);
};

const toggleRecording = () => {
  if (!isRecording) {
    setIsRecording(true);
    if (!isEditing) setTranscript("");
  }
};

```

```

if (recognitionRef.current) {
  try {
    recognitionRef.current.start();
  } catch (e) {
    console.error("Speech recognition error:", e);
    simulateRecording();
  }
} else {
  simulateRecording();
}
} else {
  setIsRecording(false);
  if (recognitionRef.current) {
    recognitionRef.current.stop();
  }
}
};

const simulateRecording = () => {
  const simulatedPhrases = [
    "I'm feeling really happy today!",
    "I had a tough day at work...",
    "Can you tell me a joke?",
    "I'm feeling a bit anxious."
  ];
  let i = 0;
  const interval = setInterval(() => {
    setTranscript(simulatedPhrases[0].substring(0, i + 5));
    i += 5;
    if (i > simulatedPhrases[0].length) {
      clearInterval(interval);
      setIsRecording(false);
    }
  }, 200);
};

return (
  <div className="h-screen bg-background flex flex-col items-center justify-between p-6 relative overflow-hidden">
    {/* Background pulses */}
    {isRecording && (
      <>
        <motion.div

```

```

initial={{ scale: 0.5, opacity: 0.5 }}
animate={{ scale: 2, opacity: 0 }}
transition={{ duration: 2, repeat: Infinity }}
className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2 w-80 h-80
bg-primary/10 rounded-full blur-3xl pointer-events-none"
/>
</>
)}
}

{/* Header */}
<div className="w-full flex justify-between items-center z-10">
  <div className="w-10" /> {/* Spacer */}
  <span className="font-heading font-bold text-lg text-gray-700">
    {currentLang === 'ta-IN' ? 'குரல் அரட்டை' : 'Voice Chat'}
  </span>
  <Link href={`/chat?lang=${currentLang}`}>
    <Button variant="ghost" size="icon" className="rounded-full
    hover:bg-gray-100">
      <X className="w-6 h-6 text-gray-500" />
    </Button>
  </Link>
</div>

{/* Main Content */}
<div className="flex-1 flex flex-col items-center justify-center w-full max-w-md
space-y-8 z-10">

  {/* Bear */}
  <div className="w-48 h-48 relative">
    <PolarBear state={isRecording ? "listening" : "waving"} />
  </div>

  {/* Transcript / Input Area */}
  <motion.div
    layout
    className="w-full bg-white/80 backdrop-blur-sm rounded-3xl p-6 shadow-sm
    border border-gray-100 relative">
    >
      {!isEditing && !transcript ? (
        <div className="text-center space-y-2 py-4">
          <p className="text-xl font-medium text-gray-800">
            {currentLang === 'ta-IN' ? 'பேச மைக்ரோஃபோனைத் தட்டவும்' : 'Tap
            microphone to speak'}
          </p>
      
```

```

<p className="text-sm text-muted-foreground">
  {currentLang === 'ta-IN' ? 'நான் கேட்டுக்கொண்டிருக்கிறேன்...' :
  "I'm listening..."}
</p>
</div>
) : (
<div className="space-y-4">
<div className="flex justify-between items-center mb-2">
<span className="text-xs font-bold uppercase text-muted-foreground tracking-wider">
  {currentLang === 'ta-IN' ? 'உங்கள் செய்தி' : 'Your Message'}
</span>
{!isRecording && (
<Button
  variant="ghost"
  size="sm"
  onClick={() => setIsEditing(!isEditing)}
  className="h-6 text-xs text-primary hover:text-primary/80"
>
<Edit2 className="w-3 h-3 mr-1" /> {currentLang === 'ta-IN' ?
  'திருத்து' : 'Edit'}
</Button>
)}
</div>

{isEditing ? (
<Textarea
  value={transcript}
  onChange={(e) => setTranscript(e.target.value)}
  className="min-h-[100px] text-lg bg-transparent border-gray-200
focus:ring-primary/20 resize-none"
  autoFocus
/>
) : (
<p className="text-lg text-gray-800 font-medium leading-relaxed
min-h-[60px]">
  {transcript}
  {isRecording && <span className="inline-block w-1 h-5 ml-1 bg-primary
animate-pulse align-middle" />}
</p>
)}
{!isRecording && transcript && (
<div className="flex gap-2 pt-2">

```

```

        <Button
            variant="outline"
            className="flex-1 rounded-xl"
            onClick={() => {
                setTranscript("");
                setIsEditing(false);
            }}
        >
            <RotateCcw className="w-4 h-4 mr-2" /> {currentLang === 'ta-IN' ?
        'அழி' : 'Clear'}
    </Button>
    <Button
        className="flex-1 rounded-xl shadow-lg shadow-primary/20"
        onClick={handleSend}
    >
        <Send className="w-4 h-4 mr-2" /> {currentLang === 'ta-IN' ?
    'அனுப்பு' : 'Send'}
    </Button>
</div>
    )}
</div>
    )}
</motion.div>

{/* Waveform */}
<div className="h-16 flex items-center justify-center w-full">
    {isRecording ? (
        <VoiceWaveform isRecording={isRecording} />
    ) : null}
</div>
</div>

{/* Mic Control */}
<div className="pb-8 z-10">
    {!isEditing && (
        <Button
            size="lg"
            className={`w-20 h-20 rounded-full shadow-xl transition-all duration-300
            ${isRecording
                ? 'bg-red-500 hover:bg-red-600 ring-4 ring-red-100'
                : 'bg-primary hover:bg-primary/90 ring-4 ring-blue-100`}
            `}
            onClick={toggleRecording}
        >
    )}

```

```

        >
        {isRecording ? (
            <div className="w-8 h-8 bg-white rounded-md" /* Stop Square */
        ) : (
            <Mic className="w-8 h-8 text-white" />
        )}
        </Button>
    )}
</div>
</div>
);
}

```

## 7. Welcome.tsx

```

import { Link } from "wouter";
import { PolarBear } from "@/components/ui/PolarBear";
import { Button } from "@/components/ui/button";
import { motion } from "framer-motion";
import { ArrowRight } from "lucide-react";

export default function Welcome() {
    return (
        <div className="min-h-screen bg-background flex flex-col items-center justify-center p-6 relative overflow-hidden">
            {/* Decorative background circles */}
            <div className="absolute top-[ -10%] right-[ -10%] w-64 h-64 bg-primary/10 rounded-full blur-3xl" />
            <div className="absolute bottom-[ -10%] left-[ -10%] w-64 h-64 bg-accent/20 rounded-full blur-3xl" />

            <motion.div
                initial={{ y: 20, opacity: 0 }}
                animate={{ y: 0, opacity: 1 }}
                transition={{ delay: 0.2 }}
                className="w-full max-w-md flex flex-col items-center text-center space-y-8 z-10"
            >
                <div className="space-y-2">
                    <h1 className="text-4xl font-heading font-bold text-primary">MagizhBuddy</h1>
                    <p className="text-muted-foreground text-lg">Your emotional companion</p>
                </div>
            
```

```

<div className="w-64 h-64">
  <PolarBear state="waving" />
</div>

<div className="bg-white/50 backdrop-blur-sm p-6 rounded-3xl border border-white/50 shadow-sm">
  <p className="text-lg font-medium text-gray-700 leading-relaxed">
    "Hi there! I'm here to listen, chat, and brighten your day. How are you feeling?"
  </p>
</div>

<Link href="/emotion-check">
  <Button
    size="lg"
    className="w-full rounded-full h-14 text-lg font-bold shadow-lg
    shadow-primary/20 hover:shadow-primary/30 transition-all"
  >
    Get Started <ArrowRight className="ml-2 w-5 h-5" />
  </Button>
</Link>
</motion.div>
</div>
);

}

```

## 8. App.tsx

```

import { Switch, Route } from "wouter";
import { queryClient } from "./lib/queryClient";
import { QueryClientProvider } from "@tanstack/react-query";
import { Toaster } from "@components/ui/toaster";
import { TooltipProvider } from "@components/ui/tooltip";
import NotFound from "@pages/not-found";
import Welcome from "@pages/Welcome";
import EmotionCheckin from "@pages/EmotionCheckin";
import Chat from "@pages/Chat";
import Voice from "@pages/Voice";
import Settings from "@pages/Settings";

function Router() {
  return (
    <Switch>
      <Route path="/" component={Welcome} />

```

```

        <Route path="/emotion-check" component={EmotionCheckin} />
        <Route path="/chat" component={Chat} />
        <Route path="/voice" component={Voice} />
        <Route path="/settings" component={Settings} />
        <Route component={NotFound} />
    </Switch>
);
}

function App() {
    return (
        <QueryClientProvider client={queryClient}>
            <TooltipProvider>
                <Toaster />
                <Router />
            </TooltipProvider>
        </QueryClientProvider>
    );
}

export default App;

```

### 9.main.tsx

```

import { createRoot } from "react-dom/client";
import App from "./App";
import "./index.css";

createRoot(document.getElementById("root")!).render(<App />);

```