2-Day Zero-Cost AI Mental Health Assistant Hackathon Plan

Team Size: 6 members
Timeline: 48 hours
Budget: $4 GPT-4 API credits + $100 AWS free tier
Goal: Working demo with voice-to-plan-to-delivery pipeline

Executive Summary

This plan transforms your ambitious AI mental health assistant into a **working demo in 48 hours** using **100% free tools** (except $4 in GPT API credits). We'll use AWS free tier ($100 credits), Supabase free plan, Vercel free hosting, self-hosted n8n, and open-source Whisper. By the end, you'll have a complete system: users record voice → transcription → AI generates personalized plan → automated delivery via Telegram.

Free Tools Stack (Detailed Breakdown)

1. Supabase (Database + Auth + Storage) - FREE FOREVER

- **What you get free:**
  - 500 MB database storage (enough for 1000s of transcripts)
  - 1 GB file storage (for audio files)
  - 2 free projects
  - Unlimited API requests
  - 50,000 monthly active users
  - Built-in authentication (email, social logins)
  - Real-time subscriptions
- **Why it's perfect:** No credit card required, won't charge you unexpectedly, includes everything you need
- **Setup time:** 5 minutes
- **Limits that matter:** 500 MB DB is plenty for hackathon; if exceeded, project pauses (no charges)
- **Sign up:** https://supabase.com

2. Vercel (Frontend Hosting) - FREE FOREVER

- **What you get free (Hobby plan):**
  - Unlimited websites
  - 100 GB bandwidth/month
  - Automatic HTTPS
  - Global CDN
  - Serverless functions (100 GB-hours compute)
  - Automatic deployments from Git

- **Why it's perfect:** Zero-config Next.js deployment, instant previews, professional URLs
- **Setup time:** 2 minutes (connect GitHub)
- **Sign up:** https://vercel.com

3. AWS EC2 (Backend + Whisper) - **$100 FREE CREDITS**

- **What you get with free credits:**
  - t2.micro instance (1 vCPU, 1 GB RAM) - free for 12 months
  - t3.medium (2 vCPU, 4 GB RAM) for Whisper - covered by $100 credits
  - 30 GB EBS storage
  - 15 GB outbound data transfer
- **Why we use it:** Need compute for FastAPI backend + Whisper transcription
- **Cost estimate:** t3.medium = $0.0416/hour × 48 hours = $2 (well within $100)
- **Setup time:** 15 minutes

4. n8n (Automation) - **FREE SELF-HOSTED**

- **Cloud vs Self-hosted:**
  - n8n Cloud Free: Only 200 executions/month (NOT enough)
  - Self-hosted: UNLIMITED executions (we'll use this)
- **Where to host:** Same AWS EC2 instance as backend (no extra cost)
- **What you get:** Unlimited workflows, unlimited executions, all integrations
- **Setup time:** 10 minutes with Docker

5. Faster-Whisper (Transcription) - **FREE OPEN-SOURCE**

- **Why not OpenAI Whisper API:** Costs $0.006/minute (would eat your $4 budget fast)
- **Faster-Whisper benefits:**
  - 4x faster than original Whisper
  - Runs on CPU (t3.medium can handle it)
  - Same accuracy as OpenAI's version
  - Zero per-use cost
- **Model size:** Use "medium" model (1.5 GB) - best speed/accuracy tradeoff
- **Performance:** Can transcribe 1 minute of audio in ~15 seconds on t3.medium

6. LLM API (Planner Agent) - **100% FREE via Agent Router**

- **Agent Router (FREE API key):**
  - Sign up at https://agentrouter.org for free API key
  - Access to multiple LLMs: GPT-4o-mini, Claude Sonnet, Gemini Pro, Llama 3
  - Completely free for hackathons and development
  - Automatic failover between models for reliability
  - No credit card required, no usage limits for reasonable use
- **Cost savings:**
  - Your $4 GPT credits: SAVED for emergencies/production
  - Agent Router handles all plan generation during hackathon
  - Can generate unlimited plans during 48-hour period
- **Implementation:**
  - Replace OpenAI API calls with Agent Router endpoint
  - Same request/response format (OpenAI-compatible)
  - Automatic model selection for best availability

    – Built-in rate limiting and error handling

7. WhatsApp Bot (Emergency Notifications) - FREE WITH TWILIO TRIAL

- **Twilio WhatsApp Sandbox (FREE for hackathon):**
  - $15 trial credit included (no payment required initially)
  - Send WhatsApp messages to pre-approved contacts
  - Perfect for emergency alerts to loved ones
  - ~1000 free messages with trial credit
- **Setup:** 10 minutes to configure Twilio sandbox
- **What you get:** Send WhatsApp messages to family/friends when mental health crisis detected
- **Cost:** $0 for hackathon (trial credit), $0.005/message after ($5 for 1000 messages)

8. Spotify Integration (Music Therapy) - FREE API

- **Spotify Web API (FREE):**
  - Free API access with Spotify account (no Spotify Premium required)
  - Create and play mood-based playlists
  - Search for calming/uplifting music
  - Control playback if user has Spotify installed
- **Setup:** 15 minutes (create Spotify Developer app)
- **What you get:** Automatically play therapeutic music when low mood detected
- **Limitation:** User must have Spotify app installed; free tier has ads

7. WhatsApp Bot (Emergency Notifications) - FREE WITH TWILIO TRIAL

- **Twilio WhatsApp Sandbox (FREE for hackathon):**
  - $15 trial credit included (no payment required initially)
  - Send WhatsApp messages to pre-approved contacts
  - Perfect for emergency alerts to loved ones
  - ~1000 free messages with trial credit
- **Setup:** 10 minutes to configure Twilio sandbox
- **What you get:** Send WhatsApp messages to family/friends when mental health crisis detected
- **Cost:** $0 for hackathon (trial credit), $0.005/message after ($5 for 1000 messages)

8. Spotify Integration (Music Therapy) - FREE API

- **Spotify Web API (FREE):**
  - Free API access with Spotify account (no Spotify Premium required)
  - Create and play mood-based playlists
  - Search for calming/uplifting music
  - Control playback if user has Spotify app installed
- **Setup:** 15 minutes (create Spotify Developer app)
- **What you get:** Automatically play therapeutic music when low mood detected
- **Limitation:** User must have Spotify app installed; free tier has ads

9. Additional Free Tools

Tool	Purpose	Free Tier
GitHub	Code hosting, CI/CD	Unlimited public repos
Docker Hub	Container registry	1 free private repo
Postman	API testing	Free forever plan
Sentry	Error tracking	5,000 events/month
Cloudflare	DNS (optional)	Free tier
Twilio	WhatsApp messaging	$15 trial credit
Spotify Developer	Music API	Free with Spotify account
Agent Router	LLM API gateway	Free for hackathons

Table 1: Supporting tools for development and deployment

Architecture (Detailed Flow)

Figure 1: System Architecture - Voice to Plan Delivery

Step-by-step data flow:

1. **User records 60-second audio** via Next.js frontend (MediaRecorder API)
2. **Frontend uploads** audio blob to FastAPI backend (hosted on EC2)
3. **Backend stores** audio in Supabase Storage, creates `audio_logs` entry
4. **Background worker** picks up job, downloads audio
5. **Faster-Whisper** transcribes audio to text (15 seconds for 1-min audio)
6. **Sentiment analyzer** extracts mood vector (anxious/calm/sad/energetic) + crisis detection
7. **GPT-4o-mini** receives [transcript + mood] and generates structured plan JSON
8. **Crisis detection:** If severe distress detected (keywords: suicide, harm, hopeless):
    - Send immediate WhatsApp alerts to emergency contacts
    - Display crisis hotline numbers in app
    - Skip normal plan generation
9. **Music therapy:** If moderate/low mood detected:
    - Backend calls Spotify API to create calming playlist
    - Frontend displays "Play Music" button with embedded Spotify player
    - Auto-play if user has granted permission
10. **Backend saves** plan to Supabase `plans` table
11. **Backend triggers** n8n webhook with plan data
12. **n8n workflow** formats message, sends via WhatsApp (Twilio) to user
13. **User receives** plan on WhatsApp with actionable steps
14. **Scheduler** (n8n cron) sends check-in messages at planned times

---

Database Schema (Supabase)

```
-- Users table (managed by Supabase Auth)
-- No custom users table needed

-- Audio logs
CREATE TABLE audio_logs (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID REFERENCES auth.users(id),
storage_path TEXT NOT NULL,
duration_seconds INTEGER,
status TEXT DEFAULT 'uploaded', -- uploaded/processing/completed/failed
created_at TIMESTAMP DEFAULT now()
);

-- Transcripts
CREATE TABLE transcripts (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
audio_id UUID REFERENCES audio_logs(id),
text TEXT NOT NULL,
language TEXT DEFAULT 'en',
confidence FLOAT,
mood_vector JSONB, -- {anxiety: 0.7, calm: 0.3, energy: 0.6}
created_at TIMESTAMP DEFAULT now()
);

-- Plans
CREATE TABLE plans (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
transcript_id UUID REFERENCES transcripts(id),
plan_json JSONB NOT NULL,
delivery_status TEXT DEFAULT 'pending', -- pending/sent/failed
whatsapp_message_id TEXT,
spotify_playlist_url TEXT,
crisis_alert_sent BOOLEAN DEFAULT false,
created_at TIMESTAMP DEFAULT now()
);

-- Emergency contacts
CREATE TABLE emergency_contacts (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID REFERENCES auth.users(id),
name TEXT NOT NULL,
phone_number TEXT NOT NULL, -- E.164 format: +1234567890
relationship TEXT, -- mom, dad, friend, therapist
priority INTEGER DEFAULT 1, -- 1=primary, 2=secondary
created_at TIMESTAMP DEFAULT now()
);

-- Reminders
CREATE TABLE reminders (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
plan_id UUID REFERENCES plans(id),
message TEXT NOT NULL,
scheduled_time TIMESTAMP NOT NULL,
sent_status TEXT DEFAULT 'pending',
sent_at TIMESTAMP,
created_at TIMESTAMP DEFAULT now()
);
```

Team Roles (6 People, 2 Days)

Person	Role	Key Responsibilities
Person 1	Product Manager (You)	Priorities, acceptance tests, demos
Person 2	Frontend Dev	Next.js UI, audio recording, auth
Person 3	Backend Dev	FastAPI, endpoints, queue management
Person 4	AI/ML Engineer	Whisper setup, GPT prompts, mood analysis
Person 5	DevOps	AWS setup, Docker, deployment, monitoring
Person 6	Automation	n8n workflows, Telegram bot, scheduling

Table 2: Team structure and ownership

Cross-functional pairing:

- Frontend + Backend: API contract design (Hour 2)
- ML + Backend: Transcription worker integration (Day 1 PM)
- DevOps + All: Deployment support (ongoing)
- Automation + Backend: Webhook security (Day 2 AM)

Hour-by-Hour Timeline (48 Hours)

Day 1: Foundation + Core Features (24 hours)

Hours 0-4: Setup Sprint

- **Hour 0-1: Environment Setup (Everyone)**
  - Create GitHub monorepo: `/frontend, /backend, /infra`
  - Sign up for Supabase, Vercel, AWS
  - Install Docker Desktop on all machines
  - Create shared Slack/Discord channel
  - Clone starter templates (Next.js, FastAPI)
- **Hour 1-2: Infrastructure (DevOps + Backend)**

    - Launch AWS EC2 t3.medium instance
    - Install Docker, Docker Compose
    - Set up security groups (ports 80, 443, 8000)
    - Deploy nginx reverse proxy
    - Create Supabase project
- **Hour 2-3: Database (Backend + DevOps)**
    - Run SQL schema on Supabase
    - Enable Row Level Security (RLS)
    - Create API keys
    - Test connection from backend
- **Hour 3-4: Auth Setup (Frontend + Backend)**
    - Configure Supabase Auth (email + Google)
    - Implement login/signup in Next.js
    - Create JWT middleware in FastAPI
    - Test end-to-end auth flow

Hours 4-8: Core Backend

- **Hour 4-6: Audio Upload (Backend + Frontend)**
    - Frontend: Implement MediaRecorder, create upload UI
    - Backend: `POST /api/upload-audio` endpoint
    - Store audio in Supabase Storage
    - Create `audio_logs` entry with status
    - Return job ID to frontend
- **Hour 6-8: Worker Queue (Backend + DevOps)**
    - Set up Redis container on EC2
    - Implement background worker using RQ (Redis Queue)
    - Create `/transcribe` job handler
    - Test job enqueue and processing

CHECKPOINT: Audio upload working, jobs queuing ✓

Hours 8-12: Transcription Pipeline

- **Hour 8-10: Faster-Whisper Setup (ML Engineer + DevOps)**
    - Install faster-whisper in Docker container
    - Download "medium" model (1.5 GB)
    - Create Python wrapper: `transcribe(audio_path) -> text`
    - Test with sample audio files
- **Hour 10-12: Integration (ML + Backend)**
    - Worker downloads audio from Supabase Storage
    - Calls Whisper transcription
    - Saves transcript to `transcripts` table
    - Updates `audio_logs` status to "completed"
    - Frontend polls for status updates

CHECKPOINT: End-to-end transcription working ✓

Hours 12-16: AI Planner

- **Hour 12-14: Mood Analysis (ML Engineer)**
    - Use TextBlob or VADER for sentiment
    - Extract mood scores: anxiety, calm, energy, sadness

    - Store in mood_vector JSON field
    - Test with various transcripts
- **Hour 14-16: GPT Planner Agent + Crisis Detection (ML + Backend)**
    - Design prompt template (see below)
    - Implement crisis keyword detection BEFORE GPT call:
        * Keywords: suicide, kill myself, end it all, no point, harm, hopeless
        * If detected: skip plan, trigger emergency workflow
    - Implement generate_plan(transcript, mood) function
    - Parse JSON response from GPT-4o-mini
    - Validate schema, handle errors
    - If mood score < 3/10: Add Spotify playlist recommendation
    - Save plan to plans table
- **Hour 16-17: Spotify Integration (ML + Backend)**
    - Implement Spotify OAuth flow
    - Create create_mood_playlist(mood_vector) function:
        * Low mood → search "calming meditation relaxing"
        * Anxious → search "peaceful ambient nature sounds"
        * Sad → search "uplifting hopeful positive"
    - Return playlist URL and save to plans.spotify_playlist_url
    - Frontend: Add Spotify player embed
- **Hour 17-18: Emergency Contact System (Backend)**
    - Create UI for adding emergency contacts (name, phone, relationship)
    - Validate phone numbers (E.164 format)
    - Implement send_crisis_alert(user_id) function
    - Test WhatsApp delivery to multiple contacts

GPT Prompt Template (Updated with Music Therapy):

You are a compassionate mental health assistant. Given a user's voice transcript and mood analysis,
create a personalized wellness plan. CRITICAL: Do not provide medical diagnosis or replace professional help. If severe crisis detected, the system will handle emergency protocols separately.

INPUT:
Transcript: {transcript}
Mood: {mood_vector}
Spotify Available: {spotify_connected}

OUTPUT (JSON):
{
"summary": "Brief empathetic assessment of user's state",
"mood_severity": 1-10, // 1=crisis, 5=moderate, 10=great
"goals": ["goal1", "goal2", "goal3"],
"activities": [
{
"title": "Activity name",
"description": "What to do",
"duration_minutes": 15,
"difficulty": "easy|medium|hard",
"time_of_day": "morning|afternoon|evening|anytime",
"category": "breathing|movement|social|creative|rest"

}
],
"music_recommendation": {
"needed": true/false,
"mood_target": "calming|uplifting|energizing|grounding",
"search_query": "specific spotify search terms"
},
"reminders": [
{
"message": "Reminder text",
"delay_hours": 2
}
],
"emergency_resources": [
"National Suicide Prevention Lifeline: 988",
"Crisis Text Line: Text HOME to 741741"
]
}

Be specific, actionable, and empathetic. Always include emergency resources.

Crisis Detection Logic (Pre-GPT):

Run BEFORE calling GPT

def detect_crisis(transcript: str) -> bool:
crisis_keywords = [
'suicide', 'kill myself', 'end it all', 'want to die',
'no point living', 'harm myself', 'can't go on',
'better off dead', 'hopeless', 'give up on life'
]

```
transcript_lower = transcript.lower()
for keyword in crisis_keywords:
    if keyword in transcript_lower:
        return True
return False
```

If crisis detected:

if detect_crisis(transcript):
# Skip normal plan generation
# Send immediate alerts to emergency contacts
send_crisis_alerts(user_id)
# Display crisis resources in app
return crisis_response()

CHECKPOINT: Plans generating from transcripts ✓

Hours 16-20: Frontend Dashboard

- **Hour 16-18: Plan Display (Frontend)**
  - Create `/dashboard` page
  - Fetch and display user's plans
  - Show activities with cards/list view
  - Add "Mark Complete" buttons
- **Hour 18-20: Status Updates (Frontend)**
  - Real-time status: "Uploading" → "Transcribing" → "Generating Plan"
  - Progress bar or spinner
  - Show transcript when ready
  - Polish UI with Tailwind

Hours 20-24: Automation Setup

- **Hour 20-21: n8n Deployment (Automation + DevOps)**
  - Deploy n8n on EC2 with Docker Compose
  - Set up nginx subdomain: n8n.yourdomain.com
  - Create admin credentials
  - Verify web interface accessible
- **Hour 21-22: WhatsApp + Spotify Setup (Automation)**
  - Create Twilio account, get $15 trial credit
  - Set up WhatsApp Sandbox: https://console.twilio.com/sandbox
  - Join sandbox with emergency contacts' WhatsApp numbers
  - Create Spotify Developer App: https://developer.spotify.com/dashboard
  - Get Client ID and Client Secret
  - Test sending WhatsApp message via Twilio API
  - Test Spotify API playlist creation
  - Store credentials in environment
- **Hour 22-24: n8n Workflows (Automation + Backend)**
  - **Workflow 1 - Normal Plan Delivery:**
    - Create webhook endpoint in n8n
    - Backend: Add `call_n8n_webhook(plan)` function
    - n8n: Format plan as WhatsApp message
    - n8n: Send via Twilio WhatsApp API
  - **Workflow 2 - Crisis Alert:**
    - Separate webhook for emergencies
    - Query emergency_contacts table
    - Send immediate WhatsApp alerts to all contacts
    - Include user's location (if permitted)
  - **Workflow 3 - Music Integration:**
    - Backend calls Spotify API on low mood detection
    - Create playlist based on mood (calming/uplifting)
    - Return playlist URL to frontend
  - Test all three workflows end-to-end

END OF DAY 1: Core pipeline functional ✓

Day 2: Polish + Testing + Demo Prep (24 hours)

Hours 24-28: Reminders System

- **Hour 24-26: Scheduler (Automation)**
  - Create n8n cron workflow (runs every 5 minutes)
  - Query reminders table for due reminders
  - Send via Telegram
  - Update sent_status to "sent"
- **Hour 26-28: Reminder Creation (Backend + Automation)**
  - Parse reminders array from GPT plan
  - Calculate scheduled_time = now + delay_hours
  - Insert into reminders table
  - Test scheduled delivery

Hours 28-32: Testing & Bug Fixes

- **Hour 28-30: End-to-End Testing (Everyone)**
  - Test complete flow: record → plan → Telegram
  - Try edge cases: long audio, silence, background noise
  - Test error handling: network failures, API errors
  - Fix critical bugs
- **Hour 30-32: Error Handling (Backend + DevOps)**
  - Add try-catch blocks everywhere
  - Implement retry logic for API calls
  - Add Sentry error tracking
  - Create fallback responses

Hours 32-36: UX Polish

- **Hour 32-34: Frontend Polish (Frontend)**
  - Responsive design (mobile-first)
  - Loading states and animations
  - Error messages for users
  - Add demo audio samples
- **Hour 34-36: Content & Copy (PM + Frontend)**
  - Write landing page copy
  - Add instructions: "Record 30-60 seconds about how you feel"
  - Create privacy policy page
  - Add GitHub link and team info

Hours 36-40: Monitoring & Observability

- **Hour 36-38: Logging (Backend + DevOps)**
  - Structured logging (JSON format)
  - Log all API requests/responses
  - Log transcription timings
  - Add CloudWatch Logs integration
- **Hour 38-40: Monitoring Dashboard (DevOps)**
  - Create simple Grafana dashboard (optional)
  - Monitor: API response times, transcription queue length
  - Set up email alerts for errors

    - Document debugging procedures

Hours 40-44: Demo Preparation

- **Hour 40-42: Demo Script (PM + Everyone)**
    - Write 5-minute demo script
    - Prepare 3 test scenarios:
        1. Anxious user → calming activities
        2. Low-energy user → energizing plan
        3. Happy user → maintenance activities
    - Record demo videos as backup
    - Test on fresh devices
- **Hour 42-44: Presentation Slides (PM + Frontend)**
    - Create 10-slide deck:
        1. Problem statement
        2. Solution overview
        3. Architecture diagram
        4. Live demo
        5. Technical highlights
        6. Impact/use cases
        7. Future roadmap
        8. Team & stack
    - Practice demo 3 times

Hours 44-48: Buffer & Contingency

- **Hour 44-46: Final Bug Fixes**
    - Address any remaining issues
    - Performance optimization if needed
    - Database query optimization
- **Hour 46-48: Documentation**
    - README with setup instructions
    - Architecture documentation
    - API documentation (Postman collection)
    - Video demo recording

Critical Setup Instructions

1. AWS EC2 Setup (15 minutes)

Launch instance

aws ec2 run-instances
--image-id ami-0c55b159cbfafe1f0 \ # Ubuntu 22.04 LTS
--instance-type t3.medium
--key-name your-key
--security-group-ids sg-xxx
--subnet-id subnet-xxx

SSH into instance

```
ssh -i your-key.pem ubuntu@ec2-xx-xx-xx-xx.compute.amazonaws.com
```

Install Docker

```
sudo apt update
sudo apt install -y docker.io docker-compose
sudo usermod -aG docker ubuntu
```

Install Python 3.11

```
sudo apt install -y python3.11 python3-pip
```

Clone your repo

```
git clone https://github.com/yourteam/hackathon-project.git
cd hackathon-project
```

2. Backend Deployment (Docker Compose)

Create `docker-compose.yml`:

version: '3.8'

services:
backend:
build: ./backend
ports:
- "8000:8000"
environment:
- SUPABASE_URL=$SUPABASE_URL - SUPABASE_KEY =${SUPABASE_KEY}
- OPENAI_API_KEY=${OPENAI_API_KEY}
- REDIS_URL=redis://redis:6379
depends_on:
- redis
volumes:
- ./audio_cache:/app/audio_cache

worker:
build: ./backend
command: python worker.py
environment:
- SUPABASE_URL=$SUPABASE_URL - SUPABASE_KEY =${SUPABASE_KEY}
- REDIS_URL=redis://redis:6379
depends_on:
- redis
volumes:

```
- ./models:/app/models # Whisper models cache
- ./audio_cache:/app/audio_cache

redis:
image: redis:7-alpine
ports:
- "6379:6379"

n8n:
image: n8nio/n8n:latest
ports:
- "5678:5678"
environment:
- N8N_BASIC_AUTH_ACTIVE=true
- N8N_BASIC_AUTH_USER=admin
- N8N_BASIC_AUTH_PASSWORD=hackathon2024
- WEBHOOK_URL=https://your-domain.com/
volumes:
- n8n_data:/home/node/.n8n

volumes:
n8n_data:
```

Deploy:

```
docker-compose up -d
```

3. Faster-Whisper Setup

Add to `backend/requirements.txt`:

```
fastapi0.104.1
uvicorn0.24.0
supabase2.0.0
requests2.31.0 # For Agent Router API calls
redis5.0.1
rq1.15.1
faster-whisper0.9.0
textblob0.17.1
spotipy==2.23.0 # For Spotify integration
```

Create `backend/transcribe.py`:

```
from faster_whisper import WhisperModel
```

Download model on first run (1.5 GB)

```
model = WhisperModel("medium", device="cpu", compute_type="int8")

def transcribe_audio(audio_path: str) -> dict:
segments, info = model.transcribe(audio_path, beam_size=5)
```

```
text = " ".join([segment.text for segment in segments])

return {
    "text": text,
    "language": info.language,
    "duration": info.duration
}
```

4. Frontend Deployment (Vercel)

In frontend directory

npm install
npm run build

Deploy to Vercel

vercel --prod

Or connect GitHub repo to Vercel dashboard for auto-deploys

5. Supabase Setup

1. Go to https://supabase.com/dashboard
2. Create new project (choose free tier)
3. Go to SQL Editor, run the schema from above
4. Go to Settings → API → Copy Project URL and anon key
5. Add to `.env`:

SUPABASE_URL=https://xxxxx.supabase.co
SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

5a. Agent Router Setup (NEW - 5 minutes)

1. Go to https://agentrouter.org
2. Sign up with email (no credit card required)
3. Navigate to Dashboard → API Keys
4. Click "Create New Key" → Name it "Hackathon Mental Health App"
5. Copy the API key (starts with `ar_`)
6. Add to `.env`:

AGENT_ROUTER_API_KEY=ar_xxxxxxxxxxxxxxxxxxxxxxxxxxxx
AGENT_ROUTER_API_KEY=ar_your_free_api_key_here

6. Twilio WhatsApp Setup

Step 1: Create Twilio Account (5 minutes)

1. Go to https://www.twilio.com/try-twilio
2. Sign up with email (no credit card required initially)
3. Verify phone number
4. Get $15 trial credit automatically

Step 2: WhatsApp Sandbox Setup (5 minutes)

1. Go to Console → Messaging → Try it out → Send a WhatsApp message
2. You'll see a join code like: "join abc-xyz"
3. Send this code from YOUR WhatsApp to the sandbox number shown
4. Add emergency contacts: Have them send the join code too
5. Copy credentials from Console → Account → API keys:

```
TWILIO_ACCOUNT_SID=ACxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
TWILIO_AUTH_TOKEN=your_auth_token
TWILIO_WHATSAPP_NUMBER=+14155238886 # Sandbox number
```

Step 3: Test Sending Message

curl -X POST "https://api.twilio.com/2010-04-01/Accounts/$TWILIO_ACCOUNT_SID/Messages.json"
--data-urlencode "From=whatsapp:+14155238886"
--data-urlencode "To=whatsapp:+1234567890"
--data-urlencode "Body=Hello from hackathon! □"
-u $TWILIO_{A}CCOUNT_{S}ID$:TWILIO_AUTH_TOKEN

Important Notes:

- Sandbox allows messaging pre-approved numbers only (perfect for demo)
- For production, you need WhatsApp Business API approval (takes 1-2 weeks)
- Trial credit: $0.005/message = 3,000 messages with $15

7. Spotify API Setup

Step 1: Create Spotify Developer App (5 minutes)

1. Go to https://developer.spotify.com/dashboard
2. Log in with Spotify account (create free one if needed)
3. Click "Create App"
4. Fill in:
   - App name: "Mental Health Music Helper"
   - Description: "Plays therapeutic music based on mood"
   - Redirect URI: `http://localhost:3000/api/spotify/callback`
5. Check agreement boxes, click "Create"
6. Copy credentials:

```
SPOTIFY_CLIENT_ID=your_client_id_here
SPOTIFY_CLIENT_SECRET=your_client_secret_here
```

Step 2: Get User Authorization (10 minutes)

1. Implement OAuth flow in backend (see code snippet below)
2. User clicks "Connect Spotify" in your app
3. Redirects to Spotify login
4. App receives access token
5. Store token in database for user

Step 3: Test Playlist Creation

curl -X GET "https://api.spotify.com/v1/search?q=calming music&type=playlist&limit=1"
-H "Authorization: Bearer YOUR_ACCESS_TOKEN"

Important Notes:

- Free Spotify API access (no Premium required for API)
- User needs Spotify app installed to play music
- Access token expires after 1 hour (implement refresh token)
- Free tier users will hear ads (Premium users won't)

Demo Script (5 Minutes)

Slide 1: Problem (30 seconds)
"Mental health support is expensive and not always accessible. 67% of people don't seek help due to cost or availability barriers."

Slide 2: Solution (30 seconds)
"We built an AI-powered mental health assistant that listens to how you feel and creates a personalized wellness plan in under 2 minutes."

Slide 3: Live Demo (2 minutes)

1. Open app on phone (or show pre-recorded demo)
2. Log in / sign up
3. Click microphone button
4. Record: "I'm feeling really anxious today. I have a big presentation tomorrow and can't stop thinking about it. My heart is racing and I can't focus."
5. Show transcription appearing (15 seconds)
6. Show plan generation (10 seconds)
7. Display generated plan on screen with activities:
    - 5-minute breathing exercise
    - 15-minute walk outside
    - Write down 3 preparation steps
    - Practice presentation once
8. Switch to WhatsApp (phone), show received message with plan and Spotify link
9. Click Spotify link, show it opens playlist in Spotify app
10. Show reminder notification scheduled for 2 hours later
11. **Demo emergency feature:** Record crisis message → Show WhatsApp alerts sent to emergency contacts (use test contacts, blur real numbers)

Slide 4: Technical Architecture (1 minute)
"Built with modern stack: Next.js frontend, FastAPI backend, Faster-Whisper for transcription, GPT-4o-mini for plan generation, n8n for automation. Everything deployed on AWS free tier and Vercel."

Slide 5: Impact (30 seconds)
"Provides immediate support 24/7, personalized to individual needs, completely private, and accessible to anyone with a phone."

Slide 6: Q&A (30 seconds)

Hackathon Presentation Talking Points

What to emphasize:

- **Technical sophistication:** "We integrated 5 different AI/automation systems seamlessly"
- **Practical deployment:** "It's actually deployed and working - not just localhost"
- **Real-time processing:** "From voice to personalized plan in under 2 minutes"
- **Privacy-first:** "No data sent to unnecessary third parties, audio deleted after transcription"
- **Cost efficiency:** "Entire stack runs on free tier - accessible to nonprofits"
- **Scalability:** "Architecture can handle 1000s of users with minimal cost increase"

Questions judges might ask (and answers):

Q: "How accurate is the transcription?"
A: "Faster-Whisper achieves 95%+ accuracy on clear audio. We tested with various accents and background noise levels. For medical-grade use, we'd add confidence thresholds and manual review."

Q: "What about data privacy and HIPAA compliance?"
A: "Currently this is a demo. For production, we'd: 1) Encrypt audio at rest and in transit, 2) Implement proper consent flows, 3) Add audit logging, 4) Use HIPAA-compliant hosting. The architecture supports this with minimal changes."

Q: "How do you prevent misuse or harm?"
A: "We have guardrails: 1) GPT prompt includes 'do not provide medical diagnosis', 2) All plans include crisis hotline numbers, 3) We detect keywords like 'suicide' and provide immediate resources, 4) Plans are complementary to professional help, not replacements."

Q: "What's your business model?"
A: "Three paths: 1) B2C freemium ($5/month premium features), 2) B2B licensing to therapists/clinics, 3) Insurance partnerships for preventive care. Our costs scale efficiently."

Q: "How does this compare to existing apps like Headspace or Calm?"
A: "Those are content libraries. We're personalized AI coaching. Every plan is unique to the user's current state. Comparable to: Replika (companionship) + Youper (CBT therapy) + Calm (meditation), but voice-first and integrated."

Risk Mitigation & Troubleshooting

Common Issues & Solutions

Issue	Solution	Prevention
Whisper too slow	Use "small" model instead of "medium" or add GPU instance	Profile early (Hour 10)
GPT API rate limit	Implement exponential backoff	Add rate limiter (50/min)
EC2 out of credits	Stop non-essential services	Monitor AWS billing dashboard
Supabase DB full	Delete old audio files	Auto-delete audio after 7 days
n8n webhook unreachable	Check security groups, nginx config	Test with curl immediately
Audio upload fails	Check CORS settings, file size limits	Test with different audio formats
Frontend won't deploy	Check build errors, env vars	Deploy early (Hour 18)

Table 3: Common failure modes and recovery strategies

Contingency Plans

If Whisper setup fails (Hour 8-10):

- Fallback: Use AssemblyAI free tier (5 hours free transcription)
- Sign up: https://www.assemblyai.com
- Replace transcribe function with API call
- Continue with rest of pipeline

If Agent Router fails (extremely rare):

- Fallback to your $4 OpenAI credits (already configured)
- Switch to GPT-3.5-turbo (5x cheaper than GPT-4)
- Use Claude via Anthropic free tier (offers initial credits)
- Agent Router has built-in failover, so this is unlikely

If AWS credits insufficient:

- Move backend to Railway ($5 trial credit)
- Use Render free tier (750 hours/month)
- Move Whisper to Modal Labs (30 free GPU hours/month)

If demo fails live:

- Show pre-recorded video
- Have backup Loom recording
- Show screenshots of successful runs

Testing Checklist (Critical Path)

Must-Test Before Demo

- [ ] **Auth flow:** Sign up → Log in → Log out
- [ ] **Audio recording:** Works on mobile browser (Chrome/Safari)
- [ ] **Upload:** Files reach Supabase Storage
- [ ] **Transcription:** Text appears correctly
- [ ] **Plan generation:** Valid JSON, sensible recommendations
- [ ] **WhatsApp delivery:** Message received within 30 seconds
- [ ] **Spotify integration:** Playlist link works, opens in Spotify app
- [ ] **Emergency contacts:** Add/edit contacts UI works
- [ ] **Crisis detection:** Test keywords trigger emergency workflow
- [ ] **Emergency alerts:** Multiple contacts receive WhatsApp alerts
- [ ] **Error handling:** Graceful failure messages shown
- [ ] **Mobile responsive:** UI works on phone screen
- [ ] **Demo account:** Pre-created account with sample data + emergency contacts
- [ ] **WhatsApp sandbox:** All demo phone numbers joined sandbox

Nice-to-Have Tests

- [ ] Edge case: 5-second audio (too short)
- [ ] Edge case: 3-minute audio (long)
- [ ] Edge case: Silent audio
- [ ] Multiple concurrent users
- [ ] Reminder scheduling accuracy
- [ ] Plan history view

Post-Hackathon Next Steps

If you want to continue after winning:

1. **Week 1:** Add user feedback loop (thumbs up/down on plans)
2. **Week 2:** Implement plan history and progress tracking
3. **Week 3:** Add mood journal with trend visualization
4. **Week 4:** Implement therapist dashboard for professional oversight
5. **Month 2:** Get beta users (10-50 people)
6. **Month 3:** Apply to Y Combinator / accelerators
7. **Month 6:** Pursue HIPAA compliance and clinical partnerships

---

Final Checklist (Print This)

4 Hours Before Demo

- [ ] All services running on EC2
- [ ] Frontend deployed to Vercel with production URL
- [ ] Test complete flow 3 times successfully
- [ ] Demo Telegram account set up and receiving messages
- [ ] Presentation slides finalized
- [ ] Team practiced demo together
- [ ] Backup video recorded
- [ ] GitHub repo cleaned up (remove secrets, add README)
- [ ] All team members can explain architecture
- [ ] Battery/power backup for laptop

1 Hour Before Demo

- [ ] Test on phone (not just laptop)
- [ ] Verify internet connection stable
- [ ] Open all necessary tabs/apps
- [ ] Close unnecessary applications
- [ ] Log into demo account
- [ ] Clear browser cache/cookies
- [ ] Have backup laptop ready
- [ ] Charge all devices 100%

During Demo

- [ ] PM introduces (30s)
- [ ] Frontend dev drives demo (2m)
- [ ] ML engineer explains AI (30s)
- [ ] Backend dev shows architecture (30s)
- [ ] PM closes with impact (30s)
- [ ] Team answers questions together

Budget Breakdown (Actual Costs)

Item	Cost	Notes
Supabase	$0.00	Free tier forever
Vercel	$0.00	Free tier forever
AWS EC2 (48h)	~$2.00	t3.medium × 48 hours
Agent Router API	$0.00	Free for hackathons
Domain (optional)	$0.00	Use Vercel subdomain
n8n	$0.00	Self-hosted
Twilio WhatsApp	$0.00	$15 trial credit (3000 msgs)
Spotify API	$0.00	Free with Spotify account
Total	$2.00	Out of $104 available

Table 4: Actual projected costs for 48-hour hackathon

You'll have $102 left over + $15 Twilio credit + $4 GPT credits as emergency backup!

Cost breakdown after hackathon (if you continue):

- WhatsApp messages: $0.005/message = $5 for 1,000 messages
- For 10 users × 3 messages/day × 30 days = 900 messages = $4.50/month
- Agent Router: Continue free or upgrade to paid tier if needed
- Everything else remains free

Cost breakdown after hackathon (if you continue):

- WhatsApp messages: $0.005/message = $5 for 1,000 messages
- For 10 users × 3 messages/day × 30 days = 900 messages = $4.50/month
- Everything else remains free

Success Criteria (Definition of Done)

Your project is **demo-ready** when:

1. User can record 30-60 seconds of audio through web interface
2. Audio is transcribed to text with 90%+ accuracy
3. GPT generates a structured plan with 3-5 activities
4. Plan is delivered to Telegram within 2 minutes of recording
5. At least 1 scheduled reminder is sent successfully
6. System handles 5 concurrent users without crashing
7. All code is pushed to GitHub with basic documentation
8. Team can explain every component in architecture

Key Contacts & Resources

During hackathon, if stuck:

- **Supabase Discord:** https://discord.supabase.com (response in ~30 min)
- **n8n Forum:** https://community.n8n.io (helpful community)
- **FastAPI Docs:** https://fastapi.tiangolo.com (excellent docs)
- **Faster-Whisper GitHub:** https://github.com/SYSTRAN/faster-whisper (examples)
- **Stack Overflow:** Tag with specific tech (fast responses)

Useful starter repos:

- Next.js + Supabase Auth: https://github.com/supabase/auth-helpers
- FastAPI + Background Tasks: https://fastapi.tiangolo.com/tutorial/background-tasks/
- n8n Telegram Bot Example: https://n8n.io/integrations/telegram/

Motivation & Final Words

You have everything you need to build something incredible in 48 hours:

- **Free tools** that are better than most paid options
- **AI models** that work reliably
- **A clear plan** with hour-by-hour guidance
- **A passionate team** of 6 talented people

Remember:

- Done is better than perfect
- Deploy early, deploy often
- Test on real devices, not just localhost
- Judges care about demos that work, not code beauty
- Your idea solves a real problem - believe in it

You got this! Build something amazing! ▯

Appendix: Code Snippets

A. FastAPI Audio Upload Endpoint

```
from fastapi import FastAPI, UploadFile, Depends
from supabase import create_client, Client
import os

app = FastAPI()
supabase: Client = create_client(
os.getenv("SUPABASE_URL"),
os.getenv("SUPABASE_KEY")
)

@app.post("/api/upload-audio")
async def upload_audio(file: UploadFile, user_id: str):
# Save to Supabase Storage
```

file_path = f"audio/{user_id}/{file.filename}"
storage_response = supabase.storage.from_("audio-files").upload(
file_path,
await file.read()
)

```
# Create database entry
audio_log = supabase.table("audio_logs").insert({
    "user_id": user_id,
    "storage_path": file_path,
    "status": "uploaded"
}).execute()

# Queue transcription job
from redis import Redis
from rq import Queue

redis_conn = Redis.from_url(os.getenv("REDIS_URL"))
q = Queue(connection=redis_conn)
q.enqueue("worker.transcribe_job", audio_log.data[0]["id"])

return {"audio_id": audio_log.data[0]["id"], "status": "queued"}
```

B. Background Worker

worker.py

import os
from supabase import create_client
from transcribe import transcribe_audio
import openai

supabase = create_client(
os.getenv("SUPABASE_URL"),
os.getenv("SUPABASE_KEY")
)

def transcribe_job(audio_id: str):
# Get audio log
audio = supabase.table("audio_logs").select("*").eq("id", audio_id).single().execute()

```
# Download from storage
audio_data = supabase.storage.from_("audio-files").download(audio.data["stora
local_path = f"/tmp/{audio_id}.webm"
with open(local_path, "wb") as f:
    f.write(audio_data)

# Transcribe
result = transcribe_audio(local_path)

# Get mood
from textblob import TextBlob
blob = TextBlob(result["text"])
mood_vector = {
    "polarity": blob.sentiment.polarity,
    "subjectivity": blob.sentiment.subjectivity
}

# Save transcript
transcript = supabase.table("transcripts").insert({
    "audio_id": audio_id,
    "text": result["text"],
    "language": result["language"],
    "mood_vector": mood_vector
}).execute()

# Generate plan
plan = generate_plan(result["text"], mood_vector)

# Save plan
plan_record = supabase.table("plans").insert({
    "transcript_id": transcript.data[0]["id"],
    "plan_json": plan
}).execute()

# Trigger n8n
import requests
requests.post(
    "http://n8n:5678/webhook/new-plan",
    json={"plan_id": plan_record.data[0]["id"], "plan": plan}
)

# Update status
supabase.table("audio_logs").update({"status": "completed"}).eq("id", audio_id).e

# Clean up
os.remove(local_path)
```

def generate_plan(transcript: str, mood: dict):
# Using Agent Router - free API for hackathons
import requests

```
response = requests.post(
    "https://api.agentrouter.org/v1/chat/completions",
    headers={
        "Authorization": f"Bearer {os.getenv('AGENT_ROUTER_API_KEY')}",
        "Content-Type": "application/json"
    },
    json={
        "model": "gpt-4o-mini",  # Agent Router auto-selects best available
        "messages": [
            {"role": "system", "content": PLANNER_PROMPT},
            {"role": "user", "content": f"Transcript: {transcript}\nMood: {mood}"}
        ],
        "response_format": {"type": "json_object"}
    }
)
return response.json()["choices"][0]["message"]["content"]
```

C. Next.js Audio Recorder Component

'use client';

import { useState, useRef } from 'react';

export default function AudioRecorder() {
const [isRecording, setIsRecording] = useState(false);
const [audioBlob, setAudioBlob] = useState<Blob | null>(null);

const mediaRecorderRef = useRef<MediaRecorder | null>(null);
const chunksRef = useRef<Blob[]>([]);

const startRecording = async () => {
const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
const mediaRecorder = new MediaRecorder(stream);
mediaRecorderRef.current = mediaRecorder;
chunksRef.current = [];

```
mediaRecorder.ondataavailable = (e) => {
  chunksRef.current.push(e.data);
};

mediaRecorder.onstop = () => {
  const blob = new Blob(chunksRef.current, { type: 'audio/webm' });
  setAudioBlob(blob);
};

mediaRecorder.start();
setIsRecording(true);
```

};

const stopRecording = () => {
mediaRecorderRef.current?.stop();
setIsRecording(false);
};

const uploadAudio = async () => {
if (!audioBlob) return;

```
const formData = new FormData();
formData.append('file', audioBlob, 'recording.webm');

const response = await fetch('/api/upload-audio', {
  method: 'POST',
  body: formData,
  headers: {
    'Authorization': `Bearer ${localStorage.getItem('token')}`
  }
});
```

```
  const data = await response.json();
  console.log('Upload successful:', data);
```

```
};

return (

<button
onClick={isRecording ? stopRecording : startRecording}
className={px-6 py-3 rounded-full ${ isRecording ? 'bg-red-500' : 'bg-blue-500' } text-white}
>
{isRecording ? 'Stop Recording' : 'Start Recording'}
</button>
{audioBlob && (

Upload & Process

)}

);
}
```

D. n8n Workflows Configuration

Workflow 1: Normal Plan Delivery (WhatsApp)

```
{
"nodes": [
{
"name": "Webhook - New Plan",
"type": "n8n-nodes-base.webhook",
"position": [250, 300],
"parameters": {
"path": "new-plan",
"responseMode": "onReceived"
}
},
{
"name": "Format WhatsApp Message",
"type": "n8n-nodes-base.function",
"position": [450, 300],
"parameters": {
"functionCode": "const plan = items[0].json.plan;\nconst spotifyLink =
items[0].json.spotify_playlist_url || '';\nconst musicSection = spotifyLink ? \\n\\n▯ *Music
Therapy*\\n${spotifyLink} : '';\nconst message = ▯ *Your Wellness
Plan*\\n\\n${plan.summary}\\n\\n▯ *Activities:*\\n${plan.activities.map((a, i) => $i + 1. *$
{a.title}* (${a.duration_minutes} min)\n {musicSection}\n\n▯ You've got this!`;\nreturn
[{json: {message, to: items[0].json.user_phone}}];"
}
},
{
"name": "Send WhatsApp",
"type": "n8n-nodes-base.twilio",
"position": [650, 300],
"parameters": {
"operation": "send",
"from": "whatsapp:+14155238886",
"to": "whatsapp:={{json.message}}"
},
"credentials": {
"twilioApi": "twilio_credentials"
}
}
]
}
```

Workflow 2: Crisis Alert (Emergency Contacts)

```
{
"nodes": [
{
"name": "Webhook - Crisis",
"type": "n8n-nodes-base.webhook",
"position": [250, 300],
"parameters": {
"path": "crisis-alert",
"responseMode": "onReceived"
}
},
{
"name": "Get Emergency Contacts",
"type": "n8n-nodes-base.supabase",
"position": [450, 300],
"parameters": {
"operation": "getAll",
"table": "emergency_contacts",
"filterType": "manual",
"filters": {
"conditions": [
{
"column": "user_id",
"operator": "eq",
"value": "={{{user_name} has indicated they're experiencing a mental health
crisis.\n\nTime: json.to}}",
"message": "={{$json.message}}"
}
}
]
}
```

Workflow 3: Spotify Playlist Creator (Called from Backend)

This happens in backend Python code, not n8n

backend/spotify_helper.py

import requests
import base64
from typing import Dict

def get_spotify_token() -> str:
"""Get Spotify API access token"""
client_id = os.getenv("SPOTIFY_CLIENT_ID")
client_secret = os.getenv("SPOTIFY_CLIENT_SECRET")

```
auth_str = f"{client_id}:{client_secret}"
auth_bytes = auth_str.encode("utf-8")
auth_base64 = base64.b64encode(auth_bytes).decode("utf-8")

response = requests.post(
    "https://accounts.spotify.com/api/token",
    headers={"Authorization": f"Basic {auth_base64}"},
    data={"grant_type": "client_credentials"}
)
return response.json()["access_token"]
```

def create_mood_playlist(mood_vector: dict) -> str:
"""Create Spotify playlist based on mood"""
token = get_spotify_token()

```
# Determine search query based on mood
if mood_vector.get("anxiety", 0) > 0.6:
    query = "peaceful calming meditation ambient"
elif mood_vector.get("sadness", 0) > 0.6:
    query = "uplifting hopeful positive encouraging"
elif mood_vector.get("energy", 0) < 0.3:
    query = "motivational energizing workout pump up"
else:
    query = "relaxing chill peaceful nature sounds"
```

```
# Search for playlists
response = requests.get(
    f"https://api.spotify.com/v1/search?q={query}&type=playlist&limit=1",
    headers={"Authorization": f"Bearer {token}"}
)

playlists = response.json()["playlists"]["items"]
if playlists:
    return playlists[0]["external_urls"]["spotify"]
return None
```

Good luck, team! You're going to crush this hackathon! ▯

2-Day Zero-Cost AI Mental Health Assistant Hackathon Plan

Team Size: 6 members
Timeline: 48 hours
Budget: $4 GPT-4 API credits + $100 AWS free tier
Goal: Working demo with voice-to-plan-to-delivery pipeline

Executive Summary

This plan transforms your ambitious AI mental health assistant into a **working demo in 48 hours** using **100% free tools** (except $4 in GPT API credits). We'll use AWS free tier ($100 credits), Supabase free plan, Vercel free hosting, self-hosted n8n, and open-source Whisper. By the end, you'll have a complete system: users record voice → transcription → AI generates personalized plan → automated delivery via Telegram.

Free Tools Stack (Detailed Breakdown)

1. Supabase (Database + Auth + Storage) - FREE FOREVER

- **What you get free:**
  - 500 MB database storage (enough for 1000s of transcripts)
  - 1 GB file storage (for audio files)
  - 2 free projects
  - Unlimited API requests
  - 50,000 monthly active users
  - Built-in authentication (email, social logins)
  - Real-time subscriptions
- **Why it's perfect:** No credit card required, won't charge you unexpectedly, includes everything you need
- **Setup time:** 5 minutes

- **Limits that matter:** 500 MB DB is plenty for hackathon; if exceeded, project pauses (no charges)
- **Sign up:** https://supabase.com

2. Vercel (Frontend Hosting) - FREE FOREVER

- **What you get free (Hobby plan):**
  - Unlimited websites
  - 100 GB bandwidth/month
  - Automatic HTTPS
  - Global CDN
  - Serverless functions (100 GB-hours compute)
  - Automatic deployments from Git
- **Why it's perfect:** Zero-config Next.js deployment, instant previews, professional URLs
- **Setup time:** 2 minutes (connect GitHub)
- **Sign up:** https://vercel.com

3. AWS EC2 (Backend + Whisper) - $100 FREE CREDITS

- **What you get with free credits:**
  - t2.micro instance (1 vCPU, 1 GB RAM) - free for 12 months
  - t3.medium (2 vCPU, 4 GB RAM) for Whisper - covered by $100 credits
  - 30 GB EBS storage
  - 15 GB outbound data transfer
- **Why we use it:** Need compute for FastAPI backend + Whisper transcription
- **Cost estimate:** t3.medium = $0.0416/hour × 48 hours = $2 (well within $100)
- **Setup time:** 15 minutes

4. n8n (Automation) - FREE SELF-HOSTED

- **Cloud vs Self-hosted:**
  - n8n Cloud Free: Only 200 executions/month (NOT enough)
  - Self-hosted: UNLIMITED executions (we'll use this)
- **Where to host:** Same AWS EC2 instance as backend (no extra cost)
- **What you get:** Unlimited workflows, unlimited executions, all integrations
- **Setup time:** 10 minutes with Docker

5. Faster-Whisper (Transcription) - FREE OPEN-SOURCE

- **Why not OpenAI Whisper API:** Costs $0.006/minute (would eat your $4 budget fast)
- **Faster-Whisper benefits:**
  - 4x faster than original Whisper
  - Runs on CPU (t3.medium can handle it)
  - Same accuracy as OpenAI's version
  - Zero per-use cost
- **Model size:** Use "medium" model (1.5 GB) - best speed/accuracy tradeoff
- **Performance:** Can transcribe 1 minute of audio in ~15 seconds on t3.medium

6. GPT-4 API (Planner Agent) - $4 BUDGET

- **Cost calculation:**
  - GPT-4o-mini: $0.15 per 1M input tokens, $0.60 per 1M output tokens
  - Average request: 500 input + 1000 output tokens
  - Cost per plan: ~$0.0007
  - Your $4 = ~5,700 plan generations (more than enough)
- **How to conserve credits:**
  - Use GPT-4o-mini (not GPT-4) - 10x cheaper, still great quality
  - Only call for final plan generation (not testing)
  - Batch multiple test transcripts

7. Telegram Bot (Delivery) - FREE FOREVER

- **Why Telegram (not WhatsApp):**
  - Telegram Bot API: 100% free, unlimited messages
  - WhatsApp: Requires Twilio ($15 minimum), business verification
  - Email: Works but less engaging for hackathon demo
- **Setup:** Create bot with @BotFather in 2 minutes
- **What you get:** Send messages, buttons, rich media - all free

8. Additional Free Tools

Tool	Purpose	Free Tier
GitHub	Code hosting, CI/CD	Unlimited public repos
Docker Hub	Container registry	1 free private repo
Postman	API testing	Free forever plan
Sentry	Error tracking	5,000 events/month
Cloudflare	DNS (optional)	Free tier

Table 5: Supporting tools for development and deployment

Architecture (Detailed Flow)

Figure 2: System Architecture - Voice to Plan Delivery

Step-by-step data flow:

1. **User records 60-second audio** via Next.js frontend (MediaRecorder API)
2. **Frontend uploads** audio blob to FastAPI backend (hosted on EC2)
3. **Backend stores** audio in Supabase Storage, creates `audio_logs` entry
4. **Background worker** picks up job, downloads audio
5. **Faster-Whisper** transcribes audio to text (15 seconds for 1-min audio)
6. **Sentiment analyzer** extracts mood vector (anxious/calm/sad/energetic)
7. **GPT-4o-mini** receives [transcript + mood] and generates structured plan JSON
8. **Backend saves** plan to Supabase `plans` table
9. **Backend triggers** n8n webhook with plan data
10. **n8n workflow** formats message, sends via Telegram Bot API
11. **User receives** plan on Telegram with actionable steps

12. **Scheduler** (n8n cron) sends reminder messages at planned times

---

Database Schema (Supabase)

```
-- Users table (managed by Supabase Auth)
-- No custom users table needed

-- Audio logs
CREATE TABLE audio_logs (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID REFERENCES auth.users(id),
storage_path TEXT NOT NULL,
duration_seconds INTEGER,
status TEXT DEFAULT 'uploaded', -- uploaded/processing/completed/failed
created_at TIMESTAMP DEFAULT now()
);

-- Transcripts
CREATE TABLE transcripts (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
audio_id UUID REFERENCES audio_logs(id),
text TEXT NOT NULL,
language TEXT DEFAULT 'en',
confidence FLOAT,
mood_vector JSONB, -- {anxiety: 0.7, calm: 0.3, energy: 0.6}
created_at TIMESTAMP DEFAULT now()
);

-- Plans
CREATE TABLE plans (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
transcript_id UUID REFERENCES transcripts(id),
plan_json JSONB NOT NULL,
delivery_status TEXT DEFAULT 'pending', -- pending/sent/failed
telegram_message_id TEXT,
created_at TIMESTAMP DEFAULT now()
);

-- Reminders
CREATE TABLE reminders (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
plan_id UUID REFERENCES plans(id),
message TEXT NOT NULL,
scheduled_time TIMESTAMP NOT NULL,
sent_status TEXT DEFAULT 'pending',
sent_at TIMESTAMP,
created_at TIMESTAMP DEFAULT now()
);
```

Team Roles (6 People, 2 Days)

Person	Role	Key Responsibilities
Person 1	Product Manager (You)	Priorities, acceptance tests, demos
Person 2	Frontend Dev	Next.js UI, audio recording, auth
Person 3	Backend Dev	FastAPI, endpoints, queue management
Person 4	AI/ML Engineer	Whisper setup, GPT prompts, mood analysis
Person 5	DevOps	AWS setup, Docker, deployment, monitoring
Person 6	Automation	n8n workflows, Telegram bot, scheduling

Table 6: Team structure and ownership

Cross-functional pairing:

- Frontend + Backend: API contract design (Hour 2)
- ML + Backend: Transcription worker integration (Day 1 PM)
- DevOps + All: Deployment support (ongoing)
- Automation + Backend: Webhook security (Day 2 AM)

Hour-by-Hour Timeline (48 Hours)

Day 1: Foundation + Core Features (24 hours)

Hours 0-4: Setup Sprint

- **Hour 0-1: Environment Setup (Everyone)**
  - Create GitHub monorepo: `/frontend`, `/backend`, `/infra`
  - Sign up for Supabase, Vercel, AWS
  - Install Docker Desktop on all machines
  - Create shared Slack/Discord channel
  - Clone starter templates (Next.js, FastAPI)
- **Hour 1-2: Infrastructure (DevOps + Backend)**
  - Launch AWS EC2 t3.medium instance
  - Install Docker, Docker Compose
  - Set up security groups (ports 80, 443, 8000)
  - Deploy nginx reverse proxy
  - Create Supabase project
- **Hour 2-3: Database (Backend + DevOps)**
  - Run SQL schema on Supabase
  - Enable Row Level Security (RLS)

    - Create API keys
    - Test connection from backend
- **Hour 3-4: Auth Setup (Frontend + Backend)**
    - Configure Supabase Auth (email + Google)
    - Implement login/signup in Next.js
    - Create JWT middleware in FastAPI
    - Test end-to-end auth flow

Hours 4-8: Core Backend

- **Hour 4-6: Audio Upload (Backend + Frontend)**
    - Frontend: Implement MediaRecorder, create upload UI
    - Backend: `POST /api/upload-audio` endpoint
    - Store audio in Supabase Storage
    - Create `audio_logs` entry with status
    - Return job ID to frontend
- **Hour 6-8: Worker Queue (Backend + DevOps)**
    - Set up Redis container on EC2
    - Implement background worker using RQ (Redis Queue)
    - Create `/transcribe` job handler
    - Test job enqueue and processing

CHECKPOINT: Audio upload working, jobs queuing ✓

Hours 8-12: Transcription Pipeline

- **Hour 8-10: Faster-Whisper Setup (ML Engineer + DevOps)**
    - Install faster-whisper in Docker container
    - Download "medium" model (1.5 GB)
    - Create Python wrapper: `transcribe(audio_path) -> text`
    - Test with sample audio files
- **Hour 10-12: Integration (ML + Backend)**
    - Worker downloads audio from Supabase Storage
    - Calls Whisper transcription
    - Saves transcript to `transcripts` table
    - Updates `audio_logs` status to "completed"
    - Frontend polls for status updates

CHECKPOINT: End-to-end transcription working ✓

Hours 12-16: AI Planner

- **Hour 12-14: Mood Analysis (ML Engineer)**
    - Use TextBlob or VADER for sentiment
    - Extract mood scores: anxiety, calm, energy, sadness
    - Store in `mood_vector` JSON field
    - Test with various transcripts
- **Hour 14-16: GPT Planner Agent (ML + Backend)**
    - Design prompt template (see below)
    - Implement `generate_plan(transcript, mood)` function
    - Parse JSON response from GPT-4o-mini
    - Validate schema, handle errors
    - Save plan to `plans` table

GPT Prompt Template:

You are a mental health assistant. Given a user's voice transcript and mood analysis, create a personalized wellness plan.

INPUT:
Transcript: {transcript}
Mood: {mood_vector}

OUTPUT (JSON):
```
{
"summary": "Brief assessment of user's state",
"goals": ["goal1", "goal2", "goal3"],
"activities": [
{
"title": "Activity name",
"description": "What to do",
"duration_minutes": 15,
"difficulty": "easy|medium|hard",
"time_of_day": "morning|afternoon|evening|anytime"
}
],
"reminders": [
{
"message": "Reminder text",
"delay_hours": 2
}
]
}
```

Be specific, actionable, and empathetic.

CHECKPOINT: Plans generating from transcripts ✓

Hours 16-20: Frontend Dashboard

- **Hour 16-18: Plan Display (Frontend)**
  - Create `/dashboard` page
  - Fetch and display user's plans
  - Show activities with cards/list view
  - Add "Mark Complete" buttons
- **Hour 18-20: Status Updates (Frontend)**
  - Real-time status: "Uploading" → "Transcribing" → "Generating Plan"
  - Progress bar or spinner
  - Show transcript when ready
  - Polish UI with Tailwind

Hours 20-24: Automation Setup

- **Hour 20-21: n8n Deployment (Automation + DevOps)**
  - Deploy n8n on EC2 with Docker Compose
  - Set up nginx subdomain: n8n.yourdomain.com
  - Create admin credentials

    - Verify web interface accessible
- **Hour 21-22: Telegram Bot (Automation)**
    - Create bot with @BotFather
    - Get bot token
    - Test sending messages via API
    - Store bot token in environment
- **Hour 22-24: n8n Workflow (Automation + Backend)**
    - Create webhook endpoint in n8n
    - Backend: Add `call_n8n_webhook(plan)` function
    - n8n: Format plan as Telegram message
    - n8n: Send via Telegram Bot API
    - Test end-to-end delivery

END OF DAY 1: Core pipeline functional ✓

Day 2: Polish + Testing + Demo Prep (24 hours)

Hours 24-28: Reminders System

- **Hour 24-26: Scheduler (Automation)**
    - Create n8n cron workflow (runs every 5 minutes)
    - Query `reminders` table for due reminders
    - Send via Telegram
    - Update `sent_status` to "sent"
- **Hour 26-28: Reminder Creation (Backend + Automation)**
    - Parse `reminders` array from GPT plan
    - Calculate `scheduled_time` = now + delay_hours
    - Insert into `reminders` table
    - Test scheduled delivery

Hours 28-32: Testing & Bug Fixes

- **Hour 28-30: End-to-End Testing (Everyone)**
    - Test complete flow: record → plan → Telegram
    - Try edge cases: long audio, silence, background noise
    - Test error handling: network failures, API errors
    - Fix critical bugs
- **Hour 30-32: Error Handling (Backend + DevOps)**
    - Add try-catch blocks everywhere
    - Implement retry logic for API calls
    - Add Sentry error tracking
    - Create fallback responses

Hours 32-36: UX Polish

- **Hour 32-34: Frontend Polish (Frontend)**
    - Responsive design (mobile-first)
    - Loading states and animations
    - Error messages for users
    - Add demo audio samples
- **Hour 34-36: Content & Copy (PM + Frontend)**
    - Write landing page copy

    - Add instructions: "Record 30-60 seconds about how you feel"
    - Create privacy policy page
    - Add GitHub link and team info

Hours 36-40: Monitoring & Observability

- **Hour 36-38: Logging (Backend + DevOps)**
    - Structured logging (JSON format)
    - Log all API requests/responses
    - Log transcription timings
    - Add CloudWatch Logs integration
- **Hour 38-40: Monitoring Dashboard (DevOps)**
    - Create simple Grafana dashboard (optional)
    - Monitor: API response times, transcription queue length
    - Set up email alerts for errors
    - Document debugging procedures

Hours 40-44: Demo Preparation

- **Hour 40-42: Demo Script (PM + Everyone)**
    - Write 5-minute demo script
    - Prepare 3 test scenarios:
        1. Anxious user → calming activities
        2. Low-energy user → energizing plan
        3. Happy user → maintenance activities
    - Record demo videos as backup
    - Test on fresh devices
- **Hour 42-44: Presentation Slides (PM + Frontend)**
    - Create 10-slide deck:
        1. Problem statement
        2. Solution overview
        3. Architecture diagram
        4. Live demo
        5. Technical highlights
        6. Impact/use cases
        7. Future roadmap
        8. Team & stack
    - Practice demo 3 times

Hours 44-48: Buffer & Contingency

- **Hour 44-46: Final Bug Fixes**
    - Address any remaining issues
    - Performance optimization if needed
    - Database query optimization
- **Hour 46-48: Documentation**
    - README with setup instructions
    - Architecture documentation
    - API documentation (Postman collection)
    - Video demo recording

Critical Setup Instructions

1. AWS EC2 Setup (15 minutes)

Launch instance

```
aws ec2 run-instances
--image-id ami-0c55b159cbfafe1f0 \ # Ubuntu 22.04 LTS
--instance-type t3.medium
--key-name your-key
--security-group-ids sg-xxx
--subnet-id subnet-xxx
```

SSH into instance

```
ssh -i your-key.pem ubuntu@ec2-xx-xx-xx-xx.compute.amazonaws.com
```

Install Docker

```
sudo apt update
sudo apt install -y docker.io docker-compose
sudo usermod -aG docker ubuntu
```

Install Python 3.11

```
sudo apt install -y python3.11 python3-pip
```

Clone your repo

```
git clone https://github.com/yourteam/hackathon-project.git
cd hackathon-project
```

2. Backend Deployment (Docker Compose)

Create `docker-compose.yml`:

version: '3.8'

services:
backend:
build: ./backend
ports:
- "8000:8000"
environment:
- SUPABASE_URL=$SUPABASE_{U}RL - SUPABASE_{K}EY =${SUPABASE_KEY}
- OPENAI_API_KEY=${OPENAI_API_KEY}
- REDIS_URL=redis://redis:6379

```
depends_on:
- redis
volumes:
- ./audio_cache:/app/audio_cache

worker:
build: ./backend
command: python worker.py
environment:
- SUPABASE_URL=$SUPABASE_{U}RL - SUPABASE_{K}EY =${SUPABASE_KEY}
- REDIS_URL=redis://redis:6379
depends_on:
- redis
volumes:
- ./models:/app/models # Whisper models cache
- ./audio_cache:/app/audio_cache

redis:
image: redis:7-alpine
ports:
- "6379:6379"

n8n:
image: n8nio/n8n:latest
ports:
- "5678:5678"
environment:
- N8N_BASIC_AUTH_ACTIVE=true
- N8N_BASIC_AUTH_USER=admin
- N8N_BASIC_AUTH_PASSWORD=hackathon2024
- WEBHOOK_URL=https://your-domain.com/
volumes:
- n8n_data:/home/node/.n8n

volumes:
n8n_data:
```

Deploy:

```
docker-compose up -d
```

3. Faster-Whisper Setup

Add to `backend/requirements.txt`:

```
fastapi0.104.1
uvicorn0.24.0
supabase2.0.0
openai1.3.0
redis5.0.1
rq1.15.1
faster-whisper0.9.0
textblob0.17.1
```

Create backend/transcribe.py:

from faster_whisper import WhisperModel

Download model on first run (1.5 GB)

model = WhisperModel("medium", device="cpu", compute_type="int8")

def transcribe_audio(audio_path: str) -> dict:
segments, info = model.transcribe(audio_path, beam_size=5)

```
text = " ".join([segment.text for segment in segments])

return {
    "text": text,
    "language": info.language,
    "duration": info.duration
}
```

4. Frontend Deployment (Vercel)

In frontend directory

npm install
npm run build

Deploy to Vercel

vercel --prod

Or connect GitHub repo to Vercel dashboard for auto-deploys

5. Supabase Setup

1. Go to https://supabase.com/dashboard
2. Create new project (choose free tier)
3. Go to SQL Editor, run the schema from above
4. Go to Settings → API → Copy Project URL and anon key
5. Add to .env:

SUPABASE_URL=https://xxxxx.supabase.co
SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

6. Telegram Bot Setup

1. Open Telegram, search for @BotFather
2. Send `/newbot`
3. Choose name: "Wellness Assistant"
4. Choose username: "your_wellness_bot"
5. Copy token: `1234567890:ABCdefGHIjklMNOpqrsTUVwxyz`
6. Add to `.env`:

```
TELEGRAM_BOT_TOKEN=1234567890:ABCdefGHIjklMNOpqrsTUVwxyz
```

Test sending message:

```
curl -X POST
https://api.telegram.org/bot1234567890:ABCdefGHIjklMNOpqrsTUVwxyz/sendMessage
-d chat_id=YOUR_CHAT_ID
-d text="Hello from hackathon!"
```

Demo Script (5 Minutes)

Slide 1: Problem (30 seconds)
"Mental health support is expensive and not always accessible. 67% of people don't seek help due to cost or availability barriers."

Slide 2: Solution (30 seconds)
"We built an AI-powered mental health assistant that listens to how you feel and creates a personalized wellness plan in under 2 minutes."

Slide 3: Live Demo (2 minutes)

1. Open app on phone (or show pre-recorded demo)
2. Log in / sign up
3. Click microphone button
4. Record: "I'm feeling really anxious today. I have a big presentation tomorrow and can't stop thinking about it. My heart is racing and I can't focus."
5. Show transcription appearing (15 seconds)
6. Show plan generation (10 seconds)
7. Display generated plan on screen with activities:
   - 5-minute breathing exercise
   - 15-minute walk outside
   - Write down 3 preparation steps
   - Practice presentation once
8. Switch to Telegram, show received message
9. Show reminder notification scheduled for 2 hours later

Slide 4: Technical Architecture (1 minute)
"Built with modern stack: Next.js frontend, FastAPI backend, Faster-Whisper for transcription, GPT-4o-mini for plan generation, n8n for automation. Everything deployed on AWS free tier and Vercel."

Slide 5: Impact (30 seconds)
"Provides immediate support 24/7, personalized to individual needs, completely private,

and accessible to anyone with a phone."

Slide 6: Q&A (30 seconds)

Hackathon Presentation Talking Points

What to emphasize:

- **Technical sophistication:** "We integrated 5 different AI/automation systems seamlessly"
- **Practical deployment:** "It's actually deployed and working - not just localhost"
- **Real-time processing:** "From voice to personalized plan in under 2 minutes"
- **Privacy-first:** "No data sent to unnecessary third parties, audio deleted after transcription"
- **Cost efficiency:** "Entire stack runs on free tier - accessible to nonprofits"
- **Scalability:** "Architecture can handle 1000s of users with minimal cost increase"

Questions judges might ask (and answers):

Q: "How accurate is the transcription?"
A: "Faster-Whisper achieves 95%+ accuracy on clear audio. We tested with various accents and background noise levels. For medical-grade use, we'd add confidence thresholds and manual review."

Q: "What about data privacy and HIPAA compliance?"
A: "Currently this is a demo. For production, we'd: 1) Encrypt audio at rest and in transit, 2) Implement proper consent flows, 3) Add audit logging, 4) Use HIPAA-compliant hosting. The architecture supports this with minimal changes."

Q: "How do you prevent misuse or harm?"
A: "We have guardrails: 1) GPT prompt includes 'do not provide medical diagnosis', 2) All plans include crisis hotline numbers, 3) We detect keywords like 'suicide' and provide immediate resources, 4) Plans are complementary to professional help, not replacements."

Q: "What's your business model?"
A: "Three paths: 1) B2C freemium ($5/month premium features), 2) B2B licensing to therapists/clinics, 3) Insurance partnerships for preventive care. Our costs scale efficiently."

Q: "How does this compare to existing apps like Headspace or Calm?"
A: "Those are content libraries. We're personalized AI coaching. Every plan is unique to the user's current state. Comparable to: Replika (companionship) + Youper (CBT therapy) + Calm (meditation), but voice-first and integrated."

Risk Mitigation & Troubleshooting

Common Issues & Solutions

Issue	Solution	Prevention
Whisper too slow	Use "small" model instead of "medium" or add GPU instance	Profile early (Hour 10)
GPT API rate limit	Implement exponential backoff	Add rate limiter (50/min)
EC2 out of credits	Stop non-essential services	Monitor AWS billing dashboard
Supabase DB full	Delete old audio files	Auto-delete audio after 7 days
n8n webhook unreachable	Check security groups, nginx config	Test with curl immediately
Audio upload fails	Check CORS settings, file size limits	Test with different audio formats
Frontend won't deploy	Check build errors, env vars	Deploy early (Hour 18)

Table 7: Common failure modes and recovery strategies

Contingency Plans

If Whisper setup fails (Hour 8-10):

- Fallback: Use AssemblyAI free tier (5 hours free transcription)
- Sign up: https://www.assemblyai.com
- Replace transcribe function with API call
- Continue with rest of pipeline

If GPT credits run out:

- Switch to GPT-3.5-turbo (5x cheaper)
- Use Claude Sonnet free tier (Anthropic offers initial credits)
- Reduce output token limit to 500

If AWS credits insufficient:

- Move backend to Railway ($5 trial credit)
- Use Render free tier (750 hours/month)
- Move Whisper to Modal Labs (30 free GPU hours/month)

If demo fails live:

- Show pre-recorded video
- Have backup Loom recording
- Show screenshots of successful runs

Testing Checklist (Critical Path)

Must-Test Before Demo

- [ ] **Auth flow:** Sign up → Log in → Log out
- [ ] **Audio recording:** Works on mobile browser (Chrome/Safari)
- [ ] **Upload:** Files reach Supabase Storage
- [ ] **Transcription:** Text appears correctly
- [ ] **Plan generation:** Valid JSON, sensible recommendations
- [ ] **Telegram delivery:** Message received within 30 seconds
- [ ] **Error handling:** Graceful failure messages shown
- [ ] **Mobile responsive:** UI works on phone screen
- [ ] **Demo account:** Pre-created account with sample data

Nice-to-Have Tests

- [ ] Edge case: 5-second audio (too short)
- [ ] Edge case: 3-minute audio (long)
- [ ] Edge case: Silent audio
- [ ] Multiple concurrent users
- [ ] Reminder scheduling accuracy
- [ ] Plan history view

Post-Hackathon Next Steps

If you want to continue after winning:

1. **Week 1:** Add user feedback loop (thumbs up/down on plans)
2. **Week 2:** Implement plan history and progress tracking
3. **Week 3:** Add mood journal with trend visualization
4. **Week 4:** Implement therapist dashboard for professional oversight
5. **Month 2:** Get beta users (10-50 people)
6. **Month 3:** Apply to Y Combinator / accelerators
7. **Month 6:** Pursue HIPAA compliance and clinical partnerships

---

Final Checklist (Print This)

4 Hours Before Demo

- [ ] All services running on EC2
- [ ] Frontend deployed to Vercel with production URL
- [ ] Test complete flow 3 times successfully
- [ ] Demo Telegram account set up and receiving messages
- [ ] Presentation slides finalized
- [ ] Team practiced demo together
- [ ] Backup video recorded
- [ ] GitHub repo cleaned up (remove secrets, add README)
- [ ] All team members can explain architecture
- [ ] Battery/power backup for laptop

1 Hour Before Demo

- [ ] Test on phone (not just laptop)
- [ ] Verify internet connection stable
- [ ] Open all necessary tabs/apps
- [ ] Close unnecessary applications
- [ ] Log into demo account
- [ ] Clear browser cache/cookies
- [ ] Have backup laptop ready
- [ ] Charge all devices 100%

During Demo

- [ ] PM introduces (30s)
- [ ] Frontend dev drives demo (2m)
- [ ] ML engineer explains AI (30s)
- [ ] Backend dev shows architecture (30s)
- [ ] PM closes with impact (30s)
- [ ] Team answers questions together

Budget Breakdown (Actual Costs)

Item	Cost	Notes
Supabase	$0.00	Free tier forever
Vercel	$0.00	Free tier forever
AWS EC2 (48h)	~$2.00	t3.medium × 48 hours
GPT-4o-mini API	~$0.50	For ~700 test requests
Domain (optional)	$0.00	Use Vercel subdomain
n8n	$0.00	Self-hosted
Telegram Bot	$0.00	Free forever
Total	$2.50	Out of $104 available

Table 8: Actual projected costs for 48-hour hackathon

You'll have $101.50 left over!

Success Criteria (Definition of Done)

Your project is **demo-ready** when:

1. User can record 30-60 seconds of audio through web interface
2. Audio is transcribed to text with 90%+ accuracy
3. GPT generates a structured plan with 3-5 activities
4. Plan is delivered to Telegram within 2 minutes of recording
5. At least 1 scheduled reminder is sent successfully
6. System handles 5 concurrent users without crashing
7. All code is pushed to GitHub with basic documentation

8. Team can explain every component in architecture

Key Contacts & Resources

During hackathon, if stuck:

- **Supabase Discord:** https://discord.supabase.com (response in ~30 min)
- **n8n Forum:** https://community.n8n.io (helpful community)
- **FastAPI Docs:** https://fastapi.tiangolo.com (excellent docs)
- **Faster-Whisper GitHub:** https://github.com/SYSTRAN/faster-whisper (examples)
- **Stack Overflow:** Tag with specific tech (fast responses)

Useful starter repos:

- Next.js + Supabase Auth: https://github.com/supabase/auth-helpers
- FastAPI + Background Tasks: https://fastapi.tiangolo.com/tutorial/background-tasks/
- n8n Telegram Bot Example: https://n8n.io/integrations/telegram/

Motivation & Final Words

You have everything you need to build something incredible in 48 hours:

- **Free tools** that are better than most paid options
- **AI models** that work reliably
- **A clear plan** with hour-by-hour guidance
- **A passionate team** of 6 talented people

Remember:

- Done is better than perfect
- Deploy early, deploy often
- Test on real devices, not just localhost
- Judges care about demos that work, not code beauty
- Your idea solves a real problem - believe in it

You got this! Build something amazing! ▯

Appendix: Code Snippets

A. FastAPI Audio Upload Endpoint

```
from fastapi import FastAPI, UploadFile, Depends
from supabase import create_client, Client
import os

app = FastAPI()
supabase: Client = create_client(
os.getenv("SUPABASE_URL"),
os.getenv("SUPABASE_KEY")
)
```

```
@app.post("/api/upload-audio")
async def upload_audio(file: UploadFile, user_id: str):
# Save to Supabase Storage
file_path = f"audio/{user_id}/{file.filename}"
storage_response = supabase.storage.from_("audio-files").upload(
file_path,
await file.read()
)
```

```
# Create database entry
audio_log = supabase.table("audio_logs").insert({
    "user_id": user_id,
    "storage_path": file_path,
    "status": "uploaded"
}).execute()

# Queue transcription job
from redis import Redis
from rq import Queue

redis_conn = Redis.from_url(os.getenv("REDIS_URL"))
q = Queue(connection=redis_conn)
q.enqueue("worker.transcribe_job", audio_log.data[0]["id"])

return {"audio_id": audio_log.data[0]["id"], "status": "queued"}
```

B. Background Worker

worker.py

```
import os
from supabase import create_client
from transcribe import transcribe_audio
import openai

supabase = create_client(
os.getenv("SUPABASE_URL"),
os.getenv("SUPABASE_KEY")
)

def transcribe_job(audio_id: str):
# Get audio log
audio = supabase.table("audio_logs").select("*").eq("id", audio_id).single().execute()
```

```
# Download from storage
audio_data = supabase.storage.from_("audio-files").download(audio.data["stora
local_path = f"/tmp/{audio_id}.webm"
with open(local_path, "wb") as f:
    f.write(audio_data)

# Transcribe
result = transcribe_audio(local_path)

# Get mood
from textblob import TextBlob
blob = TextBlob(result["text"])
mood_vector = {
    "polarity": blob.sentiment.polarity,
    "subjectivity": blob.sentiment.subjectivity
}

# Save transcript
transcript = supabase.table("transcripts").insert({
    "audio_id": audio_id,
    "text": result["text"],
    "language": result["language"],
    "mood_vector": mood_vector
}).execute()

# Generate plan
plan = generate_plan(result["text"], mood_vector)

# Save plan
plan_record = supabase.table("plans").insert({
    "transcript_id": transcript.data[0]["id"],
    "plan_json": plan
}).execute()

# Trigger n8n
import requests
requests.post(
    "http://n8n:5678/webhook/new-plan",
    json={"plan_id": plan_record.data[0]["id"], "plan": plan}
)

# Update status
supabase.table("audio_logs").update({"status": "completed"}).eq("id", audio_id).e

# Clean up
os.remove(local_path)
```

def generate_plan(transcript: str, mood: dict):
response = openai.ChatCompletion.create(
model="gpt-4o-mini",
messages=[
{"role": "system", "content": PLANNER_PROMPT},
{"role": "user", "content": f"Transcript: {transcript}\nMood: {mood}"}
],
response_format={"type": "json_object"}
)
return response.choices[0].message.content

C. Next.js Audio Recorder Component

'use client';

import { useState, useRef } from 'react';

export default function AudioRecorder() {
const [isRecording, setIsRecording] = useState(false);
const [audioBlob, setAudioBlob] = useState<Blob | null>(null);
const mediaRecorderRef = useRef<MediaRecorder | null>(null);
const chunksRef = useRef<Blob[]>([]);

const startRecording = async () => {
const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
const mediaRecorder = new MediaRecorder(stream);
mediaRecorderRef.current = mediaRecorder;
chunksRef.current = [];

```
    mediaRecorder.ondataavailable = (e) => {
      chunksRef.current.push(e.data);
    };

    mediaRecorder.onstop = () => {
      const blob = new Blob(chunksRef.current, { type: 'audio/webm' });
      setAudioBlob(blob);
    };

    mediaRecorder.start();
    setIsRecording(true);
```

};

const stopRecording = () => {
mediaRecorderRef.current?.stop();
setIsRecording(false);
};

const uploadAudio = async () => {
if (!audioBlob) return;

```
  const formData = new FormData();
  formData.append('file', audioBlob, 'recording.webm');

  const response = await fetch('/api/upload-audio', {
    method: 'POST',
    body: formData,
    headers: {
      'Authorization': `Bearer ${localStorage.getItem('token')}`
    }
  });

  const data = await response.json();
  console.log('Upload successful:', data);
```

};

return (

<button
onClick={isRecording ? stopRecording : startRecording}
className={px-6 py-3 rounded-full ${ isRecording ? 'bg-red-500' : 'bg-blue-500' } text-white}
>
{isRecording ? 'Stop Recording' : 'Start Recording'}
</button>
{audioBlob && (

Upload & Process

```
)}

);
}
```

D. n8n Webhook Configuration (JSON)

```
{
"nodes": [
{
"name": "Webhook",
"type": "n8n-nodes-base.webhook",
"position": [250, 300],
"parameters": {
"path": "new-plan",
"responseMode": "onReceived"
}
},
{
"name": "Format Message",
"type": "n8n-nodes-base.function",
"position": [450, 300],
"parameters": {
"functionCode": "const plan = items[0].json.plan;\nconst message = □ Your Wellness
Plan\\n\\n${plan.summary}\\n\\n□ Activities:\\n${plan.activities.map((a, i) => ${i+1}.
a. title({a.duration_minutes} min)).join('\\n')};\nreturn [{json: {message, chat_id:
items[0].json.chat_id}}];"
}
},
{
"name": "Send Telegram",
"type": "n8n-nodes-base.telegram",
"position": [650, 300],
"parameters": {
"operation": "sendMessage",
"chatId": "={{json.message}}"
},
"credentials": {
"telegramApi": "telegram_bot_credentials"
}
}
],
"connections": {
"Webhook": { "main": [[{"node": "Format Message", "type": "main", "index": 0}]] },
"Format Message": { "main": [[{"node": "Send Telegram", "type": "main", "index": 0}]] }
}
}
```

Good luck, team! You're going to crush this hackathon! □