Software Design Document
AI Therapy chatbot with Anonymous peer support

Version 1.0 | Date 03 feb 2025 | By: Sameer Mardani

I. INTRODUCTION

1. purpose

   Design a secure, AI-driven chatbot to combat loneliness and anxiety using free/open-source tools and Python-centric backend, including:

   Objective

   - AI Therapy : Meta's LLaMA (open-source LLM) for empathetic conversations
   - Anonymous Peer Chat: Omegle-Like text/voice sessions with safety features
   - Crisis Detection: Escalation to human professionals.
   - Cost-Free Moderation: Open-source image/text moderation models.
2. scope

   MVP Features:

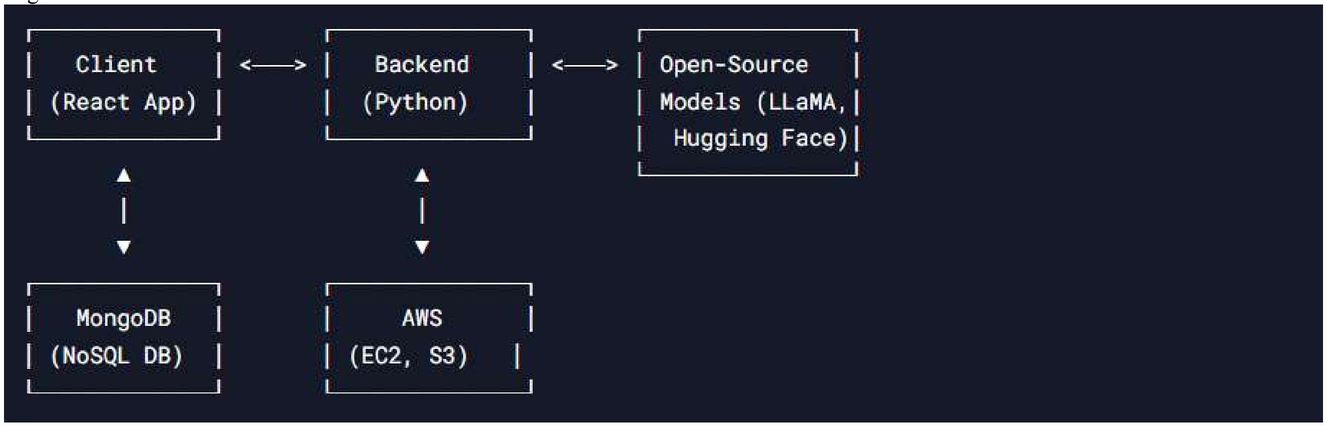
   - AI therapy chatbot (LLaMA 2/3 via Hugging Face).
   - Anonymous peer-to-peer chat (3 free sessions)
   - Image/text moderation (free alternatives)
   - Crisis detection and escalation

   Exclusions: Video chat, multilingual support (post-MVP)

   Exclusions: Paid third-party APIs (e.g., OpenAI, Google Vision).

II. System Architecture

1. High-level architecture



2. Component Breakdown

Component	Technology	Responsibility
Frontend	React,javascript	User interface for chat, AI interactions
Backend	Python (FastAPI)	Handle chat routing, AI integration, auth
Database	MongoDB	Store anonymized chat logs, user sessions
AL/ML	LLaMA 2/3 (Meta)	Generate therapy responses via Hugging Face
Safety/Moderation	Hugging Face Pipelines	Detect NSFW images/text, crisis escalation
Deployment	Docker, AWS EC2/S3	Host backend/frontend, store data

III. Detailed Component Design

1. Frontend (React/JavaScript)

   Anonymous Chat UI

   - Match users by topic (e.g., "loneliness")
   - Text/voice chat with session timers (15 mins/session)

AI Therapy Interface

- Chat window with LLaMA responses.
- Crisis button to connect to human help (hotline numbers)

2. Backend (Flask/django)

APIS:

- POST /api/chat/ai: Fetch LLaMA response via Hugging Face.
- POST /api/chat/peer: Match users for anonymous cha
- POST /api/moderation/image: Blur NSFW images with open source models

Authentication

- Firebase Anonymous Auth for guest users.
- Anonymous sessions with UUIDs (no personal data).

3. AI/ML Module

Workflow

- User sends message → Backend calls LLaMA via Hugging Face's transformers library.
- LLaMA generates response using a custom system prompt:

Python

```
system_prompt = "You are a compassionate, non-judgmental AI therapist. Use active listen
ing and avoid medical advice."
```

- Sentiment analysis with Hugging Face's pipeline for crisis detection.
- generates response using system prompt:"Act as a supportive therapist. Do NOT give medical advice."
- Sentiment analysis flags crisis triggers (e.g., "suicide").

4. Safety Module (Free alternatives)

Image Moderation:

- User uploads image → Google Vision API checks for NSFW content → Blur if unsafe.

Python

```
from transformers import pipeline
nsfw_detector = pipeline("image-classification", model="Falconsai/nsfw_image_detection")
def blur_nsfw(image_path):
    result = nsfw_detector(image_path)
    if result[0]['label'] == 'nsfw':
        return apply_blur(image_path)
```

Text Moderation:

- Use Hugging Face's toxicity-check pipeline for harmful language.

Python

```
toxicity_check = pipeline("text-classification", model="martin-ha/toxic-comment-model")
def is_toxic(text):
    result = toxicity_check(text)
    return result[0]['label'] == 'toxic'
```

IV. Data Design

1. Database Schema (MongoDB)

Users Collection (Anonymous):

JSON

```json
{
  "_id": "anon_123",
  "sessions_used": 2,
  "chat_history": [
    { "message": "I feel lonely", "timestamp": "2024-02-01T12:00:00Z" }
  ]
}
```

Chat Logs Collection (Ephemeral):

JSON

```json
{
  "session_id": "peer_abc",
  "messages": [
    { "text": "Hi!", "sender": "anon_123", "timestamp": "..." }
  ],
  "expireAt": "2024-02-02T12:00:00Z" // Auto-delete after 24h
}
```

2. Data Flow
   - user sends message → Backend saves to MongoDB.
   - AI processes message → Response saved to user's chat history
   - Images → NSFW detector → Blurred version stored in AWS S3.

V. Interfaces

1. **External Tools**

   LLaMA Integration: Use Hugging Face's transformers library.

   Hugging Face Models: Pre-trained pipelines for NSFW detection and toxicity.

   Twilio/Pusher: Real-time chat (if time permits)

   User Interfaces

   - Chat window

     Input field, send button, session timer.

     Crisis button (redirects to hotline).

   - Admin Dashboard (Future):

     Monitor active sessions, flagged content.

VI. Safety & Ethics

1. **Crisis Escalation:**

   Immediate redirection to hotlines (e.g., Crisis Text Line)

2. Privacy

   No personal data stored (anonymous IDs only)

   End-to-end encryption for peer chats

   Ephemeral chat logs (auto-delete in 24h).

3. Bias Mitigation

Audit LLaMA responses for fairness.

VII. Deployment Plan

1. **Containerization:**

    Dockerize Python backend + React frontend.

    dockerfile

```
# Backend Dockerfile
FROM python:3.9
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

2. AWS SETUP

EC2 Instance: Host Docker containers/(use GPU instances if needed for LLaMA).

S3 Bucket: Store blurred images.

Elastic Beanstalk: Auto-scaling (if needed).

Domain: Use Route 53 for custom domain (e.g., therapybot.com).

VIII. Testing Strategy

1. **Unit Tests:**

    Verify LLaMA response generation.

    Test NSFW detection accuracy.

2. Integration Tests:

End-to-end chat flow (AI + peer)

3. User Testing:

Recruit 5 beta testers for feedback.

IX. Timeline & Milestones

Dates	Task	Tips
Feb 5–9	Frontend Setup - UI/UX design (React) - Anonymous chat interface (mock APIs)	Use tools like Figma for prototyping. Start with mock data to avoid backend dependency.
Feb 10–14	Backend Development - Python/Flask/Django setup - Llama integration - Sync APIs with frontend	Document API specs early. Use Postman to test endpoints.
Feb 15–18	Moderation Features - Integrate Hugging Face API - Add content filters	Test moderation in parallel with chat functionality.
Feb 19–23	Integration & Testing - Connect frontend + backend - User testing + bug fixes	Involve testers early. Use Jest/Cypress for automated tests.
Feb 24–28	Deployment & Polish - Deploy to AWS/Heroku - Final tweaks + docs	Use Docker for containerization. Write a deployment checklist.
Mar 1–2	Buffer Days - Relax or add minor enhancements	Avoid major changes now!

X. Risks & Mitigations

Risk	Mitigation
LLaMA's computational cost	Use smaller 7B/13B parameter models.
Open-source model accuracy	Test with diverse datasets pre-launch
AWS free-tier limits	Monitor usage; optimize resource allocation

THE END

Never let yourself panic. No matter how bad it is, you find a way to stay calm, and keep your wits.