The Challenge - Full Stack Track

Existing chat applications are either too intrusive (strict identity verification) or unsafe (unrestricted anonymity).

Your Goal: Build a full-stack web application that balances privacy and safety by implementing **"Controlled Anonymity."** Users remain anonymous, but AI verification and device fingerprinting prevent abuse.

🚀 MVP Core Features

1. Anonymous Onboarding & AI Verification

- **No PII Required:** No email or phone numbers.
- **Device Fingerprinting:** Generate a unique ID stored locally (LocalStorage/IndexedDB).
- **Gender Verification:**
  - **Camera Access Only:** Users must capture a selfie in real-time. **Gallery uploads are strictly disabled** to prevent catfishing.
  - **AI Classification:** A Python model or API classifies gender (Male/Female).
  - **Privacy Rule:** The image is deleted **immediately** after classification; only the result is stored.

2. Pseudonymous Profiles

- **Minimal Context:** Users set a temporary Nickname and Short Bio (1–2 lines).
- **Visual Privacy:** No profile pictures displayed in chat; only text context is visible.

3. Intelligent Matching & Queue

- **Real-Time Queue:** Users are matched based on availability and filter settings.
- **Filters:** Male / Female / Any.
- **Logic:** Enforce cooldowns to prevent rapid re-queuing (spam prevention).

4. Ephemeral Chat Experience

- **Architecture:** 1-to-1 real-time chat via WebSockets.
- **Data Policy:** Chat history is stored locally (or not at all) and cleared upon session end.
- **Actions:** Report user, Leave chat, Next match.

5. Fairness & Usage Limits

- **Freemium Simulation:** Limit specific gender filters (e.g., 5 specific matches/day).
- **Reset Logic:** Limits reset daily based on the Device ID.

⚙️ Tech Stack & Architecture

Component	Recommended Tech	Function
Frontend	React / Next.js / Vue	UI, Chat Interface, Local Persistence
Backend	Python (FastAPI / Django) or Node.js	API, Auth Logic, Limit Enforcement
Real-Time	Python `websockets` / Django Channels or Socket.IO	Live Matching & Message Relay
AI / ML	Python (TensorFlow / PyTorch) or Free API	Gender Classification (Stateless)
Database	Redis (Queue) + Mongo/PG	Fast Matchmaking & Metadata Storage

---

🏫 Evaluation Criteria

The judges will prioritize **logic, safety, and system architecture** over visual polish.

1. **25%** - Matching & Queue Logic
2. **20%** - Privacy & Anonymity Design (Data handling)
3. **20%** - Gender Verification Flow
4. **15%** - UX & Chat Flow
5. **10%** - Abuse Prevention (Limits/Reporting)
6. **10%** - System Architecture

🔧 Deliverables

- **GitHub Repository** (Public, with README).
- **Working Demo** (Local or Deployed).
- **Architecture Diagram** (Visualizing the Queue + Socket flow).
- **Short Video Walkthrough:**
  - Onboarding (Demonstrate Live Camera Capture) → AI Verification → Profile Setup → Chat Match.
- **Documentation (PDF):** Brief explanation of the "Delete-after-verify" logic and device ID implementation.

⚠️ **Critical Constraints:**

- **This is NOT a dating app.** Focus on conversation.
- **Zero Retention:** User images must **never** be permanently stored.
- **Safety First:** Fairness and privacy architecture are more important than 100% AI accuracy.

🚀 Ready to Build?

The clock is ticking! If you have any doubts regarding the problem statement or submission guidelines, drop a message in the community group.

👉 **[Click to Join the WhatsApp Group]** *May the best solution win!*