Mental Health Patient Management System — Full spec, workflows, tech stack & AI prompts

> A compact, actionable specification to build your web app: registration/login, role-based dashboards (admin / doctor / patient), teleconsultation (WebRTC), face-based emotion analysis, wellness trackers, content recommendations, offline appointments, analytics and admin controls — plus ready-to-use AI prompts.

---

1. Project goals (single-sentence)

Build a secure, multilingual mental-health patient management web app with doctor–patient telecommunication (chat/audio/video), daily wellness tracking, emotion analysis (face), content recommendations from doctors, and admin tools for resource & user management.

---

2. High-level features (by role)

Patient

- Signup (name, age, gender, preferred language, location, phone, password, optional email, profile photo)
- Login: email/phone + password + optional 2FA; optional face-recognition check for emotional state & extra security
- Dashboard (localized): profile photo, language toggle
- Daily wellness tracker: sleep hours, meditation minutes, mood rating, custom habits
- Stress & mood visualizations: daily, weekly, monthly
- Personal emotional-support chatbot (AI) for basic coping strategies and journaling
- Telecommunication: text chat, audio call, video call (WebRTC + signalling via sockets)
- Resource library: videos, audio, articles, books, music recommended by doctor
- Appointments: online (tele) & offline (in-person) booking and calendar; request calls to doctors
- To-do list: morning plan + night review (mark tasks done)

Doctor

- Apply for account (request with identity proof, CV/experience, available timeslots, profile photo)
- Admin approval workflow
- Dashboard: patient list, monthly stress ranking, flagged/high-risk patients, tele-consultation queue, appointment calendar
- Patient history: wellness logs, emotion analysis logs, chat transcripts (with consent), prescriptions & recommendations
- Recommend resources to patient (request admin to upload or push directly from doctor toolbox)
- Mark patient as high-risk and trigger follow-up actions

Admin

- Approve doctor registrations (verify identity proof)
- Manage languages, add/remove doctors, manage resources (upload videos/audio/articles), moderate content
- Analytics: user counts, engagement, resource usage, stress metrics
- Compliance & audit logs

---

3. Core workflows (step-by-step)

A. Registration

1. Patient fills form + uploads photo → creates user but status `active` immediately (email optional verification).
2. Doctor fills application + uploads identity proof → status `pending` until admin approves. System emails doctor on status change.
3. Admin registers using secret password and email (special admin seed). Admin can create additional admin accounts.

B. Login + Face-based emotion snapshot

1. User enters email/phone + password.
2. Server validates credentials (bcrypt + rate limit). If doctor, optional extra check for 2FA.
3. After successful login, client may prompt for a short webcam capture (1–3 sec) to run emotion recognition (consent required).
4. Emotion result saved as a timestamped record and shown on dashboard summary.

   Note: Face recognition should prioritize privacy — store embeddings only if user consents and keep them encrypted. Provide opt-out.

C. Book & start teleconsultation (patient → doctor)

1. Patient selects doctor & available timeslot (or requests immediate call).
2. System creates appointment record; if online, creates a WebRTC room and a signaling channel (Socket.io).
3. At call time, both join the room. If bandwidth constraints/large sessions, use an SFU (mediasoup/Janus) for relaying.
4. Post-consultation: doctor can add notes, risk flags, resource recommendations.

D. Resource recommendation lifecycle

1. Doctor recommends resource(s) to patient (tagged to patient record).
2. If resource not present, doctor sends request to admin to upload.
3. Admin uploads & metadata stored; notifications sent to patient.

---

4. Tech stack (recommended)

Frontend

- Framework: **React** (Next.js if you want SSR/SEO) or plain React for SPA
- Styling: **Tailwind CSS** (easy, modern) — aligns with canvas guidelines
- State: **Redux Toolkit** or **React Query** for server state
- i18n: **react-i18next** for language switch and localization
- Real-time: **Socket.io-client** for signaling and instant chat
- WebRTC integration: adapter + simple wrappers; use **mediasoup-client** if SFU used
- Forms/Validation: **React Hook Form + Zod**

Backend

- Language: **Node.js (TypeScript)** with **NestJS** or **Express**; or **Python** with **FastAPI**/Django
- Real-time server: **Socket.io** for signaling & live chat
- Media: Use an **SFU** (mediasoup or Janus) for scaling multi-party audio/video; or hosted option (Twilio, Daily.co) if you prefer managed service
- Emotion detection & face embeddings: Python microservice (Flask/FastAPI) using pretrained models (OpenCV + DeepFace / FaceNet / a lightweight expression-recognition CNN). Run as separate container.
- Background jobs: **Redis + BullMQ** or **Celery (Python)** for notifications, emails, heavy ML tasks

Database & storage

- Primary DB: **PostgreSQL** for relational data (users, appointments, logs)
- NoSQL: **MongoDB** (optional) for chat transcripts / flexible documents
- Cache: **Redis** for sessions, rate limits, and queue backend
- Search & analytics: **Elasticsearch** or use PostgreSQL + materialized views for simpler analytics
- File/media storage: **S3-compatible** (AWS S3 / DigitalOcean Spaces) for videos/audio/images
- CDN: front-edge CDN for resource streaming (e.g., CloudFront)

DevOps & infra

- Containerize with **Docker**; orchestrate with **Kubernetes** (optional)
- CI/CD: GitHub Actions / GitLab CI
- Monitoring: Prometheus + Grafana; Log aggregation: ELK or Loki
- SSL: Let’s Encrypt / managed certs

Security & compliance

- Hash passwords with **bcrypt/argon2**; use **JWT** for auth tokens with refresh tokens stored securely
- Rate-limiting, brute-force protection, and CAPTCHA for signups
- Encrypt sensitive data at rest (e.g., identity docs) and in transit (TLS)
- Role-based access control (RBAC)
- For medical data, consider regional regulations (e.g., HIPAA if US; India: follow local privacy laws). Add audit logging and data retention policies.

---

5. Data models (brief)

users

- id, role (`patient` | `doctor` | `admin`), name, email, phone, password_hash, preferred_language, location, profile_photo_url, created_at

doctor_applications

- id, user_id, identity_proof_url, experience_text, timeslots (JSON), status (`pending` | `approved` | `rejected`), admin_id, reviewed_at

appointments

- id, patient_id, doctor_id, start_time, end_time, mode (`online` | `offline`), status, room_id, notes

wellness_entries

- id, patient_id, date, sleep_hours, mood_rating, meditation_minutes, notes, attachments

emotion_logs

- id, user_id, timestamp, emotion_vector (encrypted), dominant_emotion, confidence

resources

- id, title, type (`video` | `audio` | `article` | `book`), url, uploaded_by_admin_id, recommended_by_doctor_id (nullable), tags

analytics / usage

- event_id, user_id, event_type, metadata, timestamp

---

6. API surface (examples)

- `POST /api/auth/register-patient` — create patient
- `POST /api/auth/apply-doctor` — create doctor request + upload proof
- `POST /api/auth/login` — returns JWT + refresh token
- `POST /api/auth/face-capture` — send face snapshot for emotion analysis
- `GET /api/patients/:id/wellness` — wellness logs
- `POST /api/appointments` — book appointment
- `POST /api/call/signal` — socket signaling events handled via WebSocket
- `POST /api/resources` — admin upload
- `POST /api/doctor/recommend` — doctor recommends resource to patient

---

7. AI / ML integration & Efficient prompts

Below are ready-to-use prompts you can feed to an LLM (or store as templates) for features: chatbot, triage, resource summary, and content generation. Adjust `{{...}}` placeholders at runtime.

A. Patient emotional-support chatbot (short empathetic reply + coping tip)

Prompt template

```
You are a concise, empathetic mental health support assistant. The patient says:
"{{user_message}}". Their recent mood rating: {{mood_rating}} (1-10). Provide:
1) A one-sentence empathetic reflection.
2) One simple breathing or grounding exercise they can try now (2–3 steps).
3) A short reassurance sentence and a gentle suggestion to contact their doctor
if they feel at risk.
Keep language simple and aligned with the patient's preferred language:
{{language}}.
```

B. Triage: is patient high risk? (for doctor review)

Prompt template

```
You are a clinical triage assistant. Patient details: age {{age}}, recent
messages: "{{last_messages}}", mood ratings (last 7 days): {{ratings_array}},
emotion detections: {{emotion_summary}}. Answer:
- Risk level: low / moderate / high (one word)
- Brief justification (1-2 lines)
- Suggested immediate action (e.g., call patient, schedule emergency consult,
send resources)
```

C. Summarize long consultation (doctor -> concise notes)

Prompt template

```
Summarize the following consultation transcript into: 1) Problem summary (2
sentences), 2) Key observations, 3) Recommended next steps (4 bullets), 4) Any
safety concerns. Transcript:
"""
{{transcript}}
"""
```

D. Recommend resources (based on patient history)

Prompt template

```
Patient profile: age {{age}}, main concerns: {{concerns}}, recent stressors:
{{stressors}}. Based on evidence-based self-help and psychoeducation, recommend
up to 4 suitable resources (video/article/audio/book) with 1-line rationale each
and estimated time to complete.
```

E. Emotion detection service (internal ML prompt)

Spec: For every webcam capture (consented), run a lightweight face expression classifier to return: `{dominant_emotion, confidence, emotion_vector}`. Store only aggregated results unless user consents to saving images.

---

8. Implementation roadmap (MVP → v1 → v2)

MVP (4–8 weeks)

- Auth (patient + doctor application flow + admin approve)
- Patient dashboard: basic wellness tracker + to-do list
- Simple chat (text) + appointment booking (online/offline)
- Admin resource upload (manual)
- Basic emotion detection service (local; low-cost model) — optional opt-in

v1 (2–3 months)

- Add WebRTC audio/video calls with signaling and SFU for scale
- AI chatbot & triage integrations
- Resource recommendations & doctor-to-admin upload requests
- Multilingual UI
- Analytics dashboard for admin

v2 (3+ months)

- Improve ML model accuracy, privacy-preserving face embeddings, model on-device inference where possible
- Full audit/compliance features, advanced reporting
- Mobile apps (React Native) and offline sync

---

9. Privacy & Ethics checklist

- Explicit consent flows for face capture & storage

- Option to opt-out of face/emotion features
- Minimal data retention and secure deletion process
- Human oversight for critical triage decisions — LLM should assist, not replace clinician judgment
- Regular security audits

---

10. Example dev tasks (first sprint)

1. Scaffold repos: `frontend/`, `backend/`, `ml-emotion/`, infra config
2. Implement user table + patient registration + login + JWT
3. Implement doctor application endpoint + admin approval UI
4. Implement basic wellness entry endpoints and patient dashboard UI
5. Setup Socket.io skeleton for chat & signaling

---

11. Helpful libraries & tools list

- Frontend: React, Next.js, Tailwind, react-i18next, React Hook Form
- Backend: NestJS or FastAPI, Socket.io, TypeORM/Prisma
- ML: OpenCV, DeepFace, face-api.js (for light client inference), PyTorch/TensorFlow
- Media: mediasoup, Janus, Jitsi (self-hosted), Daily/Twilio (managed)
- Queue/Background: Redis + BullMQ / Celery
- Storage & infra: AWS S3, CloudFront, Docker, Kubernetes

---

12. Final notes & suggestions

- Start small: MVP without video + basic chat + wellness tracker. Add WebRTC when you have stable auth + networking.
- Keep emotion analysis opt-in and transparent.
- Use hosted WebRTC services (Daily/Twilio) if you want faster launch; switch to self-hosted SFU for cost control at scale.

If you want, I can now: - generate the database schema SQL & Prisma models, - produce a React + Node project scaffold (file list + key code snippets), or - create UI wireframes & component list for each dashboard.

Tell me which of these you'd like me to generate next and I’ll produce it right away.