Blood Donation App — Complete Documentation (Expo + StyleSheet)

Purpose: Complete A→Z guide to build a modern, fully functional Blood Donation mobile app using **Expo (React Native)** and **StyleSheet** for styling. This document includes architecture, technical stack, screens & UI details, backend API design, database schema, authentication, map & geolocation, push notifications, chat, deployment, testing, and sample code snippets.

Table of Contents

---

1) Project Overview & Goals

Goal: Build a mobile app to connect blood donors and recipients quickly and reliably. Main value: fast discovery of nearby donors, verified profiles, safe communication, and donation tracking.

MVP objectives:

- Register/login users
- Donor profile setup with blood group and location
- Search donors by blood group & nearby radius
- Map view with donor markers
- Request blood functionality
- Basic in-app chat/contact
- Push notifications for nearby requests

Non-functional goals:

- Privacy-first (data minimization)
- Responsive & fast UI
- Maintainable codebase with TypeScript
- Secure APIs and proper validation

---

2) User Roles & Permissions

- **Guest / Visitor**
- Can view public information and search limited results (optional)
- **Donor**
- Create/edit profile, set availability, share location, receive requests
- **Recipient / Seeker**
- Create requests, contact donors
- **Admin**
- Manage users, moderate reports, approve verification

Permissions matrix: donors can be contacted; seekers can create requests; admin has full access to moderation endpoints.

---

3) Feature List

MVP features

- Auth (email/password + phone OTP optional)
- Profile (donor details, last donation date, availability)
- Search & Filters (blood group, city, distance radius)
- Map screen (markers, tapping shows profile card)
- Request blood flow
- In-app chat (basic)
- Push notifications (FCM)
- Settings & privacy

v2 / Advanced

- Donor verification badge
- Real-time donor location sharing (opt-in)
- Hospital & blood bank listings
- Donation history & badges
- Admin dashboard (web)
- Multi-language (Urdu + English)

---

4) Tech Stack & Tools

Frontend (mobile):

- Expo (managed workflow)
- React Native + TypeScript
- React Navigation
- react-native-maps
- react-native-geolocation-service
- Firebase (Auth optional, Firestore for chat) or custom backend auth
- Axios / react-query (optional) for API calls
- StyleSheet for styling

Backend: (choose one primary option)

- **Option A (Node.js):** Node.js + Express + TypeScript, PostgreSQL (or Postgres + PostGIS) for geospatial queries, Prisma or TypeORM as ORM
- **Option B (Django):** Django + Django REST Framework, PostgreSQL + PostGIS

Realtime & Notifications:

- Firebase Cloud Messaging (FCM) for push notifications
- Firebase Realtime DB / Firestore or Socket.io for chat

Storage:

- Cloudinary or AWS S3 for images

Dev & Deployment:

- GitHub, GitHub Actions
- Backend deploy: Railway / Render / DigitalOcean / Heroku
- Database: Supabase (Postgres) or managed RDS

---

5) Project Structure

Frontend (Expo + TypeScript) - suggested folder layout

```
app/ (expo)
├─ assets/
├─ src/
│  ├─ api/               # axios wrappers, endpoints
│  ├─ components/        # reusable UI components
│  ├─ navigation/        # stacks & tab navigators
│  ├─ screens/           # screen components
│  ├─ services/          # auth, notifications, location
│  ├─ store/             # optional redux / react-query config
│  ├─ utils/
│  └─ theme/             # StyleSheet constants
```

```
├─ App.tsx
└─ app.json
```

Backend (Node.js + Express + TypeScript)

```
backend/
├─ src/
│  ├─ controllers/
│  ├─ models/
│  ├─ routes/
│  ├─ services/
│  ├─ middlewares/
│  ├─ utils/
│  └─ app.ts
├─ prisma/ or migrations/
└─ package.json
```

---

6) Data Models (DB Schema)

Below example uses PostgreSQL but can adapt to MongoDB.

Tables / Collections

users

- id (uuid)
- name
- email
- phone
- password_hash
- role (donor | seeker | admin)
- profile_photo_url
- created_at
- updated_at

donor_profiles

- id (uuid)
- user_id (fk users)
- blood_group (enum)
- gender
- dob
- weight
- last_donation_date
- availability (boolean)
- city
- latitude (decimal)

- longitude (decimal)
- verified (boolean)
- about
- created_at

requests

- id
- seeker_id (fk users)
- blood_group
- quantity
- hospital_name
- hospital_address
- location_lat
- location_lng
- status (open | accepted | fulfilled | cancelled)
- created_at

messages (if using DB-backed chat)

- id
- conversation_id
- sender_id
- recipient_id
- text
- attachments
- created_at

verifications

- id
- user_id
- status
- document_url
- reviewed_by
- reviewed_at

audit / logs

- To store activities for moderation and analytics

Geospatial indexing (Postgres)

- Add a `geography(Point, 4326)` column or store lat/lng and create index
- Use PostGIS functions or Haversine query for radius search

---

7) API Endpoints (Detailed)

Base URL: `https://api.yourdomain.com/v1`

Auth

- `POST /auth/register` — {name, email, phone, password}
- `POST /auth/login` — {email, password} → returns JWT
- `POST /auth/verify-phone` — OTP flow (optional)
- `POST /auth/refresh` — refresh token

User & Profile

- `GET /users/me` — get profile
- `PUT /users/me` — update profile
- `POST /users/me/photo` — upload profile photo

Donor Profile

- `POST /donors` — create donor profile
- `GET /donors/:id` — get donor profile
- `PUT /donors/:id` — update donor
- `GET /donors` — list donors (filters: blood_group, city, lat, lng, radius)

Search / Nearby

- `GET /search/donors?blood_group=A%2B&lat=...&lng=...&radius=10` — returns donors within radius (km)

Requests

- `POST /requests` — create blood request
- `GET /requests/:id`
- `GET /requests?user_id=...`
- `PUT /requests/:id/accept` — donor accepts
- `PUT /requests/:id/status` — update status

Chat (if DB-backed)

- `POST /conversations` — start conversation
- `GET /conversations?user_id=...`
- `POST /conversations/:id/messages`
- `GET /conversations/:id/messages`

Admin

- `GET /admin/users`
- `PUT /admin/users/:id/verify`
- `GET /admin/requests`

---

8) Authentication & Security

- Use **JWT** with short expiry + refresh tokens or Firebase Auth if preferred.
- Store sensitive tokens in secure storage (Expo SecureStore / Keychain)

- Always hash passwords (bcrypt)
- Validate & sanitize all inputs on backend
- Rate-limit endpoints (esp. auth & search APIs)
- Use HTTPS everywhere
- GDPR/Privacy: store minimal location detail; allow users to delete account & data

---

9) Screen-by-Screen Specification

Below every screen lists: purpose, inputs, outputs, interactions, and navigation.

1. SplashScreen

- Purpose: quick brand intro & decide auth state
- Behavior: check auth token; navigate to Auth flow or Home

2. Onboarding (optional)

- Simple 2–3 slides explaining app features

3. Auth Screens

SignupScreen

- Fields: name, email, phone, password, confirm password
- Validation: strong password, unique email
- After signup: navigate to ProfileSetup

LoginScreen

- Fields: email, password
- Actions: login, forgot password

4. ProfileSetupScreen (Donor)

- Fields:
- Name (prefilled)
- Blood Group (picker)
- DOB / Age
- Weight
- Last donation date (date picker)
- City (picker or auto-detect)
- Location: "Use current location" button (request permission)
- Availability toggle
- Upload photo
- Save → create `donor_profile` record

5. HomeScreen (Search + Feed)

- Search bar: blood group dropdown + city input + radius selector
- Quick filters: "My city", "Nearby"
- Cards: list of donors (name, group, distance, last donation)

- CTA: Map toggle / Request blood button

6. MapScreen

- Map fills screen (react-native-maps)
- Markers are donors; marker color by blood group or availability
- Tapping marker opens a bottom sheet with donor summary & actions (Chat / Call / Request)
- User can long-press to set their own location (or use "My location")

7. DonorProfileScreen

- Full donor details and contact buttons
- Report user / Verify user option (if admin)

8. RequestBloodScreen

- Fields: blood group, quantity, hospital, address, needed by date/time, notes
- Submit → creates request and notifies matching donors in radius

9. MyRequestsScreen

- Shows list of user-created requests and statuses

10. ChatScreen

- Real-time messages (incoming push notifications for new msg)
- Simple UI with messages, timestamps, and send input

11. SettingsScreen

- Manage account, notification preferences, privacy & logout

12. Admin Screens (web)

- User management, request moderation, analytics
