Product Specification Document

1. Product Overview

Product Name: Centralized Donor-Recipient-Hospital Management Platform
Technology Stack: MERN (MongoDB, Express.js, React.js, Node.js)
Objective: Streamline the process of matching organ and blood donors with recipients while maintaining transparency, security, and efficiency for hospitals.

---

2. Product Goals

- Simplify donor, recipient, and hospital registration and management.
- Automate matching based on medical compatibility and distance.
- Provide real-time updates and notifications to all users.
- Ensure security and verification of medical documents.
- Enable hospitals to monitor and control donation queues effectively.

---

3. Key Features

3.1 User Registration & Authentication

- Roles: Donor, Recipient, Hospital Admin.
- Inputs: Name, Email, Aadhaar ID, Organ/Blood Type, Location, Urgency.
- Secure authentication using JWT and bcrypt.
- Role-based access control via middleware.

3.2 Document Verification

- Upload medical reports in PDF/image format.
- Store securely using Cloudinary API.
- Verification by hospital admins.
- Status (Pending, Approved, Rejected) reflected on dashboards.

3.3 Donor–Recipient Matching

- Algorithm uses organ/blood type compatibility, urgency, and distance.
- Distance calculated using Google Maps API.
- AI model (trained on historical data) predicts the best match.
- Matches queued based on urgency and availability.

3.4 Notifications & Alerts

- Email alerts via Nodemailer (Gmail SMTP or SendGrid).

- SMS alerts via Twilio or Fast2SMS.
- Real-time updates using Socket.IO.
- Background job handling using Bull and Redis.

3.5 Hospital Dashboard

- Real-time view of registered donors, recipients, and match status.
- Document verification and donor queue management.
- Charts showing donor-recipient trends and success rates.

3.6 Security and Compliance

- HTTPS-enabled endpoints.
- AES encryption for sensitive data.
- Helmet and rate limiting for Express.js.
- Blockchain (optional) for immutable record-keeping.

---

4. System Architecture

Frontend (React.js)

- Components: Registration, Dashboard, UploadForm, MatchQueue.
- State Management: Redux Toolkit.
- Google Maps integration for donor/recipient location display.

Backend (Node.js + Express.js)

- RESTful APIs for auth, document upload, match processing, and notifications.
- Middleware for authentication, authorization, and security.
- Real-time event handling with Socket.IO.

Database (MongoDB Atlas)

- Collections: users, donors, recipients, documents, matches.
- Schema validation using Mongoose.
- Indexes on location and urgency for faster queries.

---

5. Third-Party Integrations

Service	Purpose	NPM Package
Google Maps API	Distance & time calculation	@googlemaps/google-maps-services-js
Cloudinary	File storage	cloudinary
Email	Notifications	nodemailer

Service	Purpose	NPM Package
SMS	Notifications	twilio / fast2sms
Job Queue	Background tasks	bull / redis
Security	Protection	helmet, express-rate-limit, crypto-js

---

6. Workflow

1. Donor/Recipient registers → data stored in MongoDB.
2. Medical documents uploaded and verified by hospital admin.
3. Matching algorithm identifies best-fit donors.
4. Google Maps API computes proximity.
5. AI scoring prioritizes queue.
6. Notifications triggered for matched users.
7. Hospital dashboard updates in real time.

---

7. Scalability and Deployment

- **Frontend:** Vercel/Netlify.
- **Backend:** Render/Railway.
- **Database:** MongoDB Atlas.
- **Cache:** Redis (Upstash).
- Modular microservices for scalability.

---

8. Security Protocols

- JWT-based session management.
- Data encryption using crypto-js.
- CORS enabled for trusted origins.
- Role validation middleware.
- Cloudinary secured uploads.

---

9. Success Metrics

- Reduce donor–recipient match time by 50%.
- 100% document verification accuracy.
- 0 data breaches.
- Notification delivery success rate > 95%.
- System uptime > 99%.

10. Future Enhancements

- AI-driven match prediction improvements.
- Blockchain-based verification.
- Integration with government donor databases.
- Multi-language support.
- Predictive analytics for donor supply-demand forecasting.

11. Tools Summary

Function	Package
Authentication	bcrypt, jsonwebtoken
File Upload	multer, cloudinary
Notifications	nodemailer, twilio
Maps	@googlemaps/google-maps-services-js
Queue	bull, redis
Security	helmet, crypto-js
Deployment	vercel, render, mongodb atlas

12. References

- Google Maps API Documentation.
- Cloudinary SDK Reference.
- Twilio/Fast2SMS APIs.
- MongoDB Atlas documentation.
- Express.js security best practices.