Complete Technical Working Flow of Smart Tourist Safety Monitoring System

Phase 1: Account Creation & Digital Identity Setup

Step 1: Mobile App Registration

- **Technology:** React Native mobile app (iOS/Android)
- **Backend:** Node.js with Express.js API Gateway
- **Database:** MongoDB for user accounts
- **Process:** Tourist downloads app, creates account with email/password, Firebase Authentication handles login

Step 2: Asymmetric Key Generation

- **Technology:** Elliptic Curve Digital Signature Algorithm (ECDSA) with secp256k1 curve
- **Storage:** Private key encrypted with AES-256 and stored in device's secure keystore (iOS Keychain/Android KeyStore)
- **Process:** App generates key pair locally, public key stored on blockchain, private key never leaves device

Step 3: Document Verification & Upload

- **AI Model:** TensorFlow Lite with MobileNetV2 backbone for face recognition
- **Process:**
  - Tourist uploads ID document (Aadhaar/passport) + selfie
  - On-device AI compares selfie to document photo using facial feature vectors
  - Document authenticity checked using OCR and template matching
  - All processing happens locally - no images sent off-device

Step 4: Document Encryption & Storage

- **Encryption:** AES-256 symmetric encryption with randomly generated key
- **Off-Chain Storage:** IPFS (InterPlanetary File System) for encrypted document storage
- **Process:**
  - App encrypts document with AES-256 symmetric key

  - Uploads encrypted file to IPFS, receives CID (Content Identifier)
  - Computes SHA-256 hash of original (unencrypted) document

Step 5: Blockchain Digital ID Creation

- **Blockchain Platform:** Hyperledger Fabric v2.x (permissioned network)
- **Layer 2 Scaling:** Polygon PoS for high-throughput, low-cost transactions
- **Smart Contract:** Solidity-based contract on Ethereum-PoA
- **Process:**
  - Smart contract `createDigitalID()` function called
  - Stores: document hash, public key, IPFS CID, timestamp
  - Mints unique Digital ID token (ERC-721 NFT standard)
  - Transaction cost: ~$0.01 on Polygon vs $50 on Ethereum mainnet

Phase 2: Privacy Configuration & Risk Zone Setup

Step 6: Privacy Settings Configuration

- **UI Framework:** React Native with native device controls
- **Storage:** Encrypted local preferences using react-native-keychain
- **Options:**
  - Continuous Tracking: On/Off (default Off)
  - Emergency-Only Location: On (default On)
  - Medical Data: Blood type, allergies, medications (all optional)
  - Family Notifications: Emergency/Daily/Off

Step 7: Public Risk Zone Data Download

- **Data Sources:**
  - Geological Survey of India (GSI) - landslide zones
  - India Meteorological Department (IMD) - weather alerts
  - National Remote Sensing Centre (NRSC) - satellite imagery
  - State Tourism Departments - local hazard data
- **Storage:** SQLite database on device for offline access
- **Format:** GeoJSON polygons with risk attributes

Step 8: Dynamic Risk Zone Classification

- **Cloud Infrastructure:** AWS EC2 with auto-scaling groups
- **AI Model:** Random Forest Classifier (scikit-learn)
- **Data Pipeline:** Apache Kafka for real-time data streaming
- **Process:**
  - ETL pipeline ingests: weather APIs, incident reports, crowd-sourced data
  - Random Forest model retrains every 12 hours on new data
  - Outputs risk probability scores (0.0-1.0) for each map grid cell
  - Risk zones classified: Green (0.0-0.3), Yellow (0.3-0.7), Red (0.7-1.0)

Phase 3: Journey Monitoring & Local Processing

Step 9: Local Geo-Fencing

- **Technology:** Core Location (iOS) / Google Play Services Location API (Android)
- **Processing:** On-device geospatial calculations using Turf.js library
- **Process:**
  - GPS coordinates checked against local risk zone polygons
  - Point-in-polygon calculations performed on-device
  - No location data transmitted during normal operation
  - Local alerts generated when entering risk zones

Step 10: AI Anomaly Detection (Optional Enhanced Mode)

- **Edge AI Model:** LSTM-based sequence model optimized with TensorFlow Lite
- **Hardware:** Coral Edge TPU (optional) for faster on-device inference
- **Process:**
  - Monitors GPS track: speed, direction, acceleration patterns
  - LSTM learns normal behavior baseline for each user
  - Detects anomalies: sudden stops, erratic movement, departure from normal routes
  - All processing happens locally on device

Phase 4: Emergency Detection & Response

Step 11: Emergency Mode Activation

- **Triggers:**
  - Risk zone entry (automatic)
  - Panic button press (manual)
  - AI anomaly detection (if enabled)
- **Process:** Immediately switches to Emergency Mode and activates location sharing

Step 12: Multi-Channel Emergency Communication

- **Primary:** WebSocket connection to emergency services dashboard
- **Secondary:** SMS via Twilio/TextLocal API
- **Tertiary:** Bluetooth Low Energy (BLE) mesh networking
- **Backup:** Iridium satellite communication (optional hardware)
- **Technology Stack:**
  - WebSocket: Socket.io for real-time bidirectional communication
  - SMS Gateway: Twilio API with SMS fallback every 5 minutes
  - BLE Mesh: React Native BLE Manager for device-to-device relay
  - Satellite: Iridium Short Burst Data (SBD) protocol

Step 13: Location Data Encryption & Transmission

- **Encryption:** End-to-end encryption using Curve25519 key exchange
- **Protocol:** HTTPS with TLS 1.3 for all API communications
- **Data Format:** JSON with GPS coordinates, timestamp, emergency type, user ID
- **Frequency:** Location updates every 30 seconds during emergency

Phase 5: Authority Response & Identity Verification

Step 14: Emergency Services Dashboard

- **Frontend:** React.js dashboard for rescue coordinators
- **Backend:** Node.js with Express.js, MongoDB for incident tracking
- **Real-time Updates:** Socket.io for live location streaming
- **Map Integration:** Mapbox GL JS for interactive rescue coordination maps

Step 15: Blockchain Identity Verification

- **Smart Contract Query:** Authorities call `getDigitalID(publicKey)` function
- **Returns:** Document hash, IPFS CID, verification status
- **Key Sharing:** Emergency protocol releases symmetric key encrypted under authority's public key
- **Process:**
    - Authority decrypts symmetric key using their private key
    - Downloads encrypted document from IPFS using CID
    - Decrypts document with symmetric key
    - Recomputes SHA-256 hash and compares to on-chain hash for verification

Step 16: AI-Powered Search Coordination

- **Predictive Model:** Gradient Boosted Trees (XGBoost) for location prediction
- **Input Features:** Last known GPS, speed, direction, terrain data, time elapsed
- **Output:** Probability heat map of current location
- **Integration:** Coordinates drone deployment, ground teams, satellite imagery

Phase 6: No-Network Contingency Protocols

Step 17: Offline Emergency Protocols

- **Local Storage:** SQLite buffer for GPS coordinates and emergency status
- **Auto-Sync:** Immediate upload when any connectivity returns
- **SMS Fallback:** Basic GSM text with latitude/longitude coordinates
- **BLE Mesh Relay:** Multi-hop message passing through nearby app users
- **Technology:** Core Bluetooth (iOS) / Android Bluetooth API

Step 18: Maximum Isolation SOS Mode

- **Visual Signals:** Phone flashlight flashing SOS pattern (3 short, 3 long, 3 short)
- **Audio Signals:** Speaker broadcasts GPS coordinates and SOS tone
- **Display:** Full-screen SOS message with user name and coordinates
- **Navigation:** Offline maps guide user to nearest known safe points
- **Technology:** Native device APIs for flashlight, audio, and offline mapping

Phase 7: Incident Resolution & Data Management

Step 19: Rescue Completion & Data Retention

- **Resolution Trigger:** Manual confirmation by rescue team or user
- **Auto-Stop:** Emergency mode ends, location sharing ceases
- **Data Retention:** Location data retained for 48 hours, then auto-deleted
- **Audit Logging:** All emergency actions logged on blockchain with timestamps

Step 20: System Learning & Improvement

- **Data Pipeline:** Apache Spark for big data processing
- **ML Pipeline:** MLflow for model versioning and deployment
- **Feedback Loop:** Incident outcomes feed back to improve:
    - Risk zone boundaries (Random Forest retraining)
    - Anomaly detection thresholds (LSTM fine-tuning)
    - Search prediction accuracy (XGBoost parameter optimization)

Complete Technology Stack Summary

Component	Technology	Purpose
Mobile App	React Native	Cross-platform mobile development
Backend API	Node.js + Express.js	RESTful API and business logic
Database	MongoDB	User data and incident storage
Authentication	Firebase Auth	User login and session management
Blockchain	Hyperledger Fabric + Polygon	Digital identity and audit trails
Smart Contracts	Solidity	Identity verification and access control
Off-Chain Storage	IPFS	Encrypted document storage
AI/ML Framework	TensorFlow Lite	On-device AI processing
Cloud ML	scikit-learn + XGBoost	Risk classification and prediction
Real-time Communication	Socket.io + WebSocket	Emergency alerts and location streaming
SMS Gateway	Twilio API	SMS fallback communication
Maps & GPS	Mapbox + Core Location	Mapping and location services
Encryption	AES-256 + Curve25519	Data encryption and secure communication
Container Platform	Docker + Kubernetes	Scalable cloud deployment
Cloud Infrastructure	AWS EC2 + Auto Scaling	Hosting and infrastructure
Monitoring	CloudWatch + Grafana	System monitoring and alerts

This comprehensive technical flow ensures **privacy-first design**, **robust emergency response**, and **scalable architecture** capable of supporting millions of tourists while maintaining sub-30-second emergency response times.