**TruthLens - Master Product Requirement Document (PRD)**

**📋 Basic Information**

**Project Name:** TruthLens  
**Theme:** National Security  
**Target:** Hackathon Submission + Internal Development  
**Timeline:** 6 Days (Hackathon Phase)  
**Team Size:** 5 Beginners + 1 GenAI Specialist  
**Platform:** Google Cloud Platform (Mandatory)

**🚨 Problem Statement**

**The Misinformation Crisis in India**

India faces a critical misinformation epidemic that threatens national security and social stability:

**1. Social Media Amplification**

* WhatsApp forwards spreading fake medical remedies during COVID-19
* Doctored images inciting communal tensions
* False election-related information affecting democratic processes

**2. Real-World Consequences**

* **2018 WhatsApp Lynchings:** Fake child-kidnapping videos led to mob violence
* **COVID-19 Misinformation:** False cures causing health emergencies
* **Election Interference:** Deepfakes and manipulated content affecting voter behavior
* **Financial Scams:** Fake investment schemes shared through messaging apps

**3. Scale of Impact**

* 700+ million internet users vulnerable to misinformation
* Multiple languages making detection complex
* Rapid viral spread across platforms
* Limited fact-checking resources in regional languages

**🎯 Project Objective**

**TruthLens aims to be India's first comprehensive AI-powered misinformation detection platform specifically designed for national security applications.**

**Core Capabilities:**

1. **Multi-Modal Analysis:** Text, URLs, images, and videos
2. **Intelligent Verdict System:** Clear classification with confidence scores
3. **Detailed Analysis Reports:** Evidence-based explanations
4. **Educational Component:** User awareness and media literacy
5. **Archive System:** Historical misinformation tracking
6. **Authority Escalation:** Direct reporting to relevant agencies
7. **Multi-Lingual Support:** Hindi, English, and regional languages

**⏰ Hackathon Constraints**

**Time Limitation: 6 Days**

**Team Composition: 5 Beginners + 1 GenAI Expert**

**Mandatory Technology Stack: Google Cloud Platform**

**Required GCP Services:**

* **Gemini AI:** Primary language processing
* **Vertex AI:** Custom ML model deployment
* **Fact Check API:** Cross-reference verification
* **Vision API:** Image and video analysis
* **Translate API:** Multi-language support
* **Firebase:** Real-time database and hosting
* **Cloud Run:** Serverless backend deployment

**Hackathon Deliverables:**

1. ✅ Complete PRD (This Document)
2. ✅ Working Prototype
3. ✅ Technical Documentation
4. ✅ 3-Minute Demo Video
5. ✅ Live Deployment on GCP

**🚀 MVP (Minimum Viable Product) - Hackathon Scope**

**Core Input Methods:**

* **Text Analysis:** Direct text input or copy-paste
* **URL Verification:** Website and social media links
* **Image Upload:** Photos, screenshots, memes

**Processing Pipeline:**

1. **Input Reception** → **Language Detection** → **Translation (if needed)**
2. **Multi-Engine Analysis** → **Gemini AI + Fact Check API + Vision API**
3. **Aggregation Engine** → **Confidence Scoring** → **Final Verdict**
4. **Result Generation** → **Storage** → **User Display**

**Output Features:**

* **Quick Verdict:** True/False/Inconclusive with confidence percentage
* **Detailed Analysis:** Evidence breakdown, source verification
* **Translation Support:** Results in user's preferred language
* **Archive Integration:** Automatic storage for future reference

**User Journey Pages:**

1. **Home:** Input interface
2. **Results:** Verification outcomes
3. **Archive:** Historical searches
4. **Learn:** Media literacy resources
5. **Authority:** Reporting mechanisms

**✨ Feature Requirements**

**Must-Have Features (MVP)**

* [ ] Text misinformation detection
* [ ] URL credibility analysis
* [ ] Image manipulation detection
* [ ] Multi-language support (Hindi + English)
* [ ] Confidence scoring system
* [ ] Basic reporting interface
* [ ] Archive functionality
* [ ] GCP deployment

**Nice-to-Have Features (Post-Hackathon)**

* [ ] Custom ML model training
* [ ] Reverse image search integration
* [ ] Video deepfake detection
* [ ] Real-time social media monitoring
* [ ] Advanced feedback loop system
* [ ] API for third-party integration
* [ ] Mobile application
* [ ] WhatsApp bot integration

**👥 User Flow (Step-by-Step Journey)**

**Primary User Journey:**

1. Landing Page  
 ↓  
2. Choose Input Type (Text/URL/Image)  
 ↓  
3. Submit Content for Analysis  
 ↓  
4. Processing Screen (Real-time updates)  
 ↓  
5. Results Display  
 ├── Quick Verdict  
 ├── Detailed Analysis  
 ├── Confidence Score  
 └── Source Evidence  
 ↓  
6. Action Options  
 ├── Save to Archive  
 ├── Learn More (Educational Content)  
 ├── Report to Authorities (if needed)  
 └── Share Results

**Secondary Flows:**

* **Archive Access:** View previous searches and patterns
* **Learning Center:** Media literacy and awareness content
* **Authority Dashboard:** Government/agency reporting interface

**📊 Non-Functional Requirements**

**Performance Standards:**

* **Response Time:** < 10 seconds for text analysis
* **Image Processing:** < 30 seconds for complex images
* **Concurrent Users:** Support 100+ simultaneous requests
* **Uptime:** 99.5% availability during hackathon demo

**Accuracy Targets:**

* **Text Analysis:** 85%+ accuracy rate
* **Image Detection:** 80%+ manipulation identification
* **False Positive Rate:** < 15%

**Security & Privacy:**

* **Data Encryption:** End-to-end encryption for user inputs
* **Privacy Compliance:** No personal data storage without consent
* **Audit Trail:** Complete logging for authority access

**Multi-Lingual Support:**

* **Primary Languages:** Hindi, English
* **Processing:** Automatic language detection
* **Translation:** Bi-directional translation capability

**📅 Development Roadmap**

**Hackathon Timeline (6 Days)**

**Day 1: Foundation Setup**

* [x] PRD completion
* [ ] GCP project setup and service activation
* [ ] Frontend React scaffold with routing
* [ ] Backend FastAPI project structure
* [ ] Firebase Firestore database initialization

**Day 2: Core Backend Development**

* [ ] FastAPI endpoints implementation
* [ ] Gemini AI integration
* [ ] Fact Check API connection
* [ ] Vision API setup
* [ ] Translation pipeline

**Day 3: Frontend Implementation**

* [ ] React component development
* [ ] Page layouts and navigation
* [ ] API integration layer
* [ ] State management setup

**Day 4: Integration & Testing**

* [ ] Frontend-backend connection
* [ ] End-to-end testing
* [ ] Error handling implementation
* [ ] Performance optimization

**Day 5: Deployment & Polish**

* [ ] GCP Cloud Run deployment
* [ ] Firebase Hosting setup
* [ ] UI/UX refinements
* [ ] Demo preparation

**Day 6: Final Testing & Submission**

* [ ] Final bug fixes
* [ ] Documentation completion
* [ ] Video demo recording
* [ ] Submission preparation

**Post-Hackathon Expansion (Months 1-6)**

* Advanced ML model integration
* Mobile app development
* Government partnership establishment
* Scale to regional languages
* API marketplace launch

**🏆 Pitch Framing for Judges**

**Why TruthLens Wins:**

**1. National Security Impact**

* Addresses critical threat to social harmony
* Protects democratic processes
* Supports law enforcement agencies
* Scalable to entire population

**2. Technical Innovation**

* Multi-modal AI analysis
* Real-time processing pipeline
* Integration of multiple GCP services
* Sophisticated aggregation algorithm

**3. Social Value**

* Educational component for media literacy
* Accessible to non-technical users
* Multi-language support for inclusion
* Evidence-based approach builds trust

**4. Commercial Viability**

* Government contract potential
* Enterprise API licensing
* International expansion opportunities
* Platform-as-a-Service model

**⚛️ Frontend Architecture (React)**

**Current Technology Stack:**

* **Framework:** React 18 + TypeScript
* **Build Tool:** Vite (Fast development)
* **Styling:** Tailwind CSS (Utility-first)
* **Routing:** React Router v6
* **State Management:** Context API + useReducer
* **HTTP Client:** Axios

**Component Architecture:**

src/  
├── components/  
│ ├── common/  
│ │ ├── Navbar.tsx  
│ │ ├── Footer.tsx  
│ │ ├── Loader.tsx  
│ │ └── ErrorBoundary.tsx  
│ ├── home/  
│ │ ├── HeroSection.tsx  
│ │ ├── InputSection.tsx  
│ │ └── FeatureHighlights.tsx  
│ ├── results/  
│ │ ├── ResultsCard.tsx  
│ │ ├── ConfidenceScore.tsx  
│ │ ├── DetailedAnalysis.tsx  
│ │ └── ActionButtons.tsx  
│ ├── archive/  
│ │ ├── ArchiveList.tsx  
│ │ ├── SearchHistory.tsx  
│ │ └── FilterControls.tsx  
│ ├── learn/  
│ │ ├── EducationCards.tsx  
│ │ ├── TipOfTheDay.tsx  
│ │ └── QuizSection.tsx  
│ └── authority/  
│ ├── ReportForm.tsx  
│ ├── CaseTracker.tsx  
│ └── ContactInfo.tsx  
├── pages/  
│ ├── Home.tsx  
│ ├── Results.tsx  
│ ├── Archive.tsx  
│ ├── Learn.tsx  
│ └── Authority.tsx  
├── services/  
│ ├── api.ts  
│ ├── auth.ts  
│ └── storage.ts  
├── contexts/  
│ ├── AppContext.tsx  
│ └── AuthContext.tsx  
├── utils/  
│ ├── helpers.ts  
│ ├── constants.ts  
│ └── validators.ts  
└── types/  
 ├── api.ts  
 ├── user.ts  
 └── analysis.ts

**Page Specifications:**

**1. Home Page (Landing)**

* **HeroSection:** Value proposition and CTA
* **InputSection:** Text/URL/Image upload interface
* **FeatureHighlights:** Key capabilities showcase
* **Language Toggle:** Hindi/English switcher

**2. Results Page**

* **ResultsCard:** Verdict display with visual indicators
* **ConfidenceScore:** Percentage with progress bar
* **DetailedAnalysis:** Evidence breakdown and sources
* **ActionButtons:** Archive, Learn, Report options

**3. Archive Page**

* **SearchHistory:** Chronological list of past analyses
* **FilterControls:** Date, type, verdict filtering
* **ArchiveList:** Paginated results with quick preview

**4. Learn Page**

* **EducationCards:** Media literacy modules
* **TipOfTheDay:** Daily awareness content
* **QuizSection:** Interactive knowledge testing

**5. Authority Page**

* **ReportForm:** Escalation to law enforcement
* **CaseTracker:** Status updates on reported content
* **ContactInfo:** Direct agency connections

**🔧 Backend Architecture (FastAPI)**

**Technology Stack:**

* **Framework:** FastAPI (Modern Python web framework)
* **Database:** Google Firestore (NoSQL document database)
* **AI Services:** Gemini AI, Vertex AI, Vision API
* **External APIs:** Fact Check API, Translate API
* **Deployment:** Google Cloud Run (Serverless containers)

**Project Structure:**

backend/  
├── app/  
│ ├── \_\_init\_\_.py  
│ ├── main.py # FastAPI app initialization  
│ ├── config.py # Environment configuration  
│ ├── dependencies.py # Shared dependencies  
│ ├── api/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── endpoints/  
│ │ │ ├── verify.py # Content verification  
│ │ │ ├── results.py # Results retrieval  
│ │ │ ├── archive.py # Archive management  
│ │ │ ├── learn.py # Educational content  
│ │ │ └── authority.py # Authority reporting  
│ │ └── deps.py # API dependencies  
│ ├── core/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── security.py # Authentication & authorization  
│ │ ├── database.py # Firestore connection  
│ │ └── logging.py # Application logging  
│ ├── models/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── analysis.py # Analysis result models  
│ │ ├── user.py # User models  
│ │ └── authority.py # Authority report models  
│ ├── services/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── analysis\_engine.py # Core analysis logic  
│ │ ├── gemini\_service.py # Gemini AI integration  
│ │ ├── vision\_service.py # Google Vision API  
│ │ ├── factcheck\_service.py # Fact Check API  
│ │ ├── translate\_service.py # Translation service  
│ │ └── aggregator.py # Results aggregation  
│ └── utils/  
│ ├── \_\_init\_\_.py  
│ ├── helpers.py # Utility functions  
│ ├── validators.py # Input validation  
│ └── exceptions.py # Custom exceptions  
├── requirements.txt # Python dependencies  
├── Dockerfile # Container configuration  
└── cloudbuild.yaml # GCP build configuration

**API Endpoints Specification:**

**1. Content Verification**

POST /api/v1/verify  
Request Body:  
{  
 "content\_type": "text|url|image",  
 "content": "string|url|base64\_image",  
 "language": "hi|en",  
 "user\_id": "optional\_string"  
}  
  
Response:  
{  
 "analysis\_id": "uuid",  
 "verdict": "true|false|inconclusive",  
 "confidence\_score": 0.85,  
 "summary": "Brief analysis summary",  
 "processing\_time": 8.2,  
 "language\_detected": "hi"  
}

**2. Detailed Results**

GET /api/v1/results/{analysis\_id}  
Response:  
{  
 "analysis\_id": "uuid",  
 "verdict": "true|false|inconclusive",  
 "confidence\_score": 0.85,  
 "detailed\_analysis": {  
 "gemini\_analysis": "AI-generated explanation",  
 "factcheck\_results": [...],  
 "vision\_analysis": "Image manipulation assessment",  
 "sources\_checked": [...],  
 "evidence": [...]  
 },  
 "metadata": {  
 "timestamp": "2025-09-14T11:03:00Z",  
 "processing\_time": 8.2,  
 "language": "hi"  
 }  
}

**3. Archive Management**

GET /api/v1/archive  
Query Params: ?user\_id=string&limit=20&offset=0&filter=verdict  
  
POST /api/v1/archive  
Request Body:  
{  
 "analysis\_id": "uuid",  
 "user\_notes": "optional\_string"  
}

**4. Educational Content**

GET /api/v1/learn/modules  
GET /api/v1/learn/tip-of-day  
GET /api/v1/learn/quiz/{module\_id}

**5. Authority Reporting**

POST /api/v1/authority/escalate  
Request Body:  
{  
 "analysis\_id": "uuid",  
 "reason": "string",  
 "urgency\_level": "low|medium|high|critical",  
 "additional\_context": "string"  
}  
  
GET /api/v1/authority/cases/{user\_id}

**Database Schema (Firestore):**

// Collection: analyses  
{  
 "analysis\_id": "uuid",  
 "user\_id": "string",  
 "content\_type": "text|url|image",  
 "original\_content": "string|url|base64",  
 "language\_detected": "hi|en",  
 "verdict": "true|false|inconclusive",  
 "confidence\_score": 0.85,  
 "gemini\_analysis": "string",  
 "factcheck\_results": [...],  
 "vision\_analysis": "string",  
 "sources": [...],  
 "timestamp": "2025-09-14T11:03:00Z",  
 "processing\_time": 8.2  
}  
  
// Collection: archive  
{  
 "user\_id": "string",  
 "analysis\_id": "uuid",  
 "saved\_timestamp": "2025-09-14T11:03:00Z",  
 "user\_notes": "string"  
}  
  
// Collection: authority\_reports  
{  
 "report\_id": "uuid",  
 "analysis\_id": "uuid",  
 "user\_id": "string",  
 "reason": "string",  
 "urgency\_level": "low|medium|high|critical",  
 "status": "pending|reviewed|escalated|resolved",  
 "created\_timestamp": "2025-09-14T11:03:00Z",  
 "updated\_timestamp": "2025-09-14T11:03:00Z"  
}

**🏗️ Complete Project Architecture**

**System Architecture Overview:**

┌─────────────────────────────────────────────────────────────┐  
│ FRONTEND LAYER │  
│ (React + TypeScript) │  
├─────────────────────────────────────────────────────────────┤  
│ API LAYER │  
│ (FastAPI + Python) │  
├─────────────────────────────────────────────────────────────┤  
│ GOOGLE CLOUD SERVICES │  
│ (Gemini AI + Vision API + Fact Check + etc.) │  
└─────────────────────────────────────────────────────────────┘

**Three-Layer Architecture Explanation:**

**Layer 1: Frontend (The Face)**

* **React Application:** User interface and experience
* **Firebase Hosting:** Static site deployment
* **State Management:** Real-time updates and caching
* **Responsive Design:** Mobile and desktop compatibility

**Layer 2: Backend (The Brain)**

* **FastAPI Server:** REST API and business logic
* **Cloud Run Deployment:** Serverless, scalable hosting
* **Firestore Database:** Document storage and retrieval
* **Authentication:** User session management

**Layer 3: GCP Services (The Muscles)**

* **Gemini AI:** Advanced language understanding
* **Vision API:** Image and video analysis
* **Fact Check API:** Third-party verification
* **Translate API:** Multi-language support
* **Cloud Storage:** File and media storage

**Data Flow Architecture:**

User Input  
 ↓  
React Frontend (Validation)  
 ↓  
FastAPI Backend (Routing)  
 ↓  
Analysis Engine (Orchestration)  
 ↓  
┌─────────────────────────────────────┐  
│ Parallel Processing Pipeline │  
├─────────────────────────────────────┤  
│ Gemini AI │ Vision API │ Fact │  
│ Analysis │ Processing │ Check │  
└─────────────────────────────────────┘  
 ↓  
Results Aggregator (Confidence Scoring)  
 ↓  
Firestore Database (Storage)  
 ↓  
Frontend Display (User Experience)

**☁️ Google Cloud Platform (GCP) Integration**

**What is Google Cloud Platform?**

Google Cloud Platform (GCP) is Google's suite of cloud computing services that runs on the same infrastructure that Google uses internally for products like Google Search, Gmail, and YouTube. It provides computing, storage, networking, big data, machine learning, and IoT services.

**Why GCP is Mandatory for This Hackathon:**

* **Sponsor Requirement:** Google is the hackathon sponsor
* **AI Services:** Access to cutting-edge Gemini AI and Vertex AI
* **Integration:** Seamless service connectivity
* **Scalability:** Enterprise-grade infrastructure
* **Cost Efficiency:** Free tier and hackathon credits

**TruthLens GCP Service Utilization:**

**1. Cloud Run (Backend Hosting)**

* **Purpose:** Deploy FastAPI backend as containerized service
* **Benefits:** Automatic scaling, pay-per-use, zero server management
* **Configuration:** Docker container with Python runtime

**2. Firebase Hosting (Frontend Hosting)**

* **Purpose:** Deploy React application as static website
* **Benefits:** Global CDN, SSL certificates, custom domains
* **Integration:** Seamless connection to other Firebase services

**3. Firestore (Database)**

* **Purpose:** NoSQL document database for analysis results
* **Benefits:** Real-time synchronization, offline support, automatic scaling
* **Schema:** Collections for analyses, archive, and authority reports

**4. Cloud Storage (File Storage)**

* **Purpose:** Store uploaded images, videos, and generated reports
* **Benefits:** Secure, durable, globally accessible storage
* **Integration:** Direct API access from backend services

**5. Gemini AI (Primary AI Engine)**

* **Purpose:** Advanced text analysis and misinformation detection
* **Capabilities:** Context understanding, fact verification, sentiment analysis
* **Integration:** REST API with authentication tokens

**6. Vertex AI (Custom Model Deployment)**

* **Purpose:** Deploy custom-trained ML models for specialized detection
* **Benefits:** AutoML capabilities, model versioning, A/B testing
* **Future Use:** Post-hackathon custom model deployment

**7. Fact Check API (Third-party Verification)**

* **Purpose:** Cross-reference claims with verified fact-checking sources
* **Benefits:** Access to ClaimReview structured data
* **Integration:** RESTful API calls with claim text

**8. Vision API (Image Analysis)**

* **Purpose:** Detect image manipulation, extract text, analyze content
* **Capabilities:** OCR, face detection, inappropriate content detection
* **Integration:** Direct image upload and analysis

**9. Translate API (Multi-language Support)**

* **Purpose:** Translate content and results between Hindi and English
* **Benefits:** High-accuracy translation, language detection
* **Integration:** Real-time translation pipeline

**10. Perspective API (Content Moderation)**

* **Purpose:** Detect toxic, spam, or inappropriate content
* **Benefits:** Toxicity scoring, threat detection
* **Integration:** Pre-processing filter for content analysis

**11. Safe Browsing API (URL Safety)**

* **Purpose:** Check URLs for malware, phishing, or unwanted software
* **Benefits:** Real-time threat detection, Google's security database
* **Integration:** URL verification pipeline

**GCP Deployment Roadmap:**

**Step 1: Project Setup (Day 1)**

# Create GCP project  
gcloud projects create truthlens-hackathon  
  
# Enable required APIs  
gcloud services enable run.googleapis.com  
gcloud services enable firestore.googleapis.com  
gcloud services enable storage.googleapis.com  
gcloud services enable translate.googleapis.com  
gcloud services enable vision.googleapis.com

**Step 2: Service Configuration (Day 1-2)**

# Initialize Firebase  
firebase init hosting  
firebase init firestore  
  
# Create Cloud Storage bucket  
gsutil mb gs://truthlens-uploads  
  
# Set up service account keys  
gcloud iam service-accounts create truthlens-service

**Step 3: Backend Deployment (Day 2-3)**

# Build and deploy to Cloud Run  
gcloud run deploy truthlens-api \  
 --source . \  
 --region asia-south1 \  
 --allow-unauthenticated

**Step 4: Frontend Deployment (Day 3-4)**

# Build React application  
npm run build  
  
# Deploy to Firebase Hosting  
firebase deploy --only hosting

**Step 5: Integration Testing (Day 4-5)**

* End-to-end API testing
* Frontend-backend connection verification
* GCP services integration validation
* Performance optimization

**Step 6: Production Readiness (Day 5-6)**

* SSL certificate setup
* Custom domain configuration
* Monitoring and logging setup
* Final security review

**🎯 MVP Recap & Deliverables**

**Core MVP Features Delivered:**

✅ **Multi-Modal Input Processing** (Text, URL, Image)  
✅ **AI-Powered Analysis Engine** (Gemini + Vision + Fact Check)  
✅ **Intelligent Verdict System** (True/False/Inconclusive + Confidence)  
✅ **Multi-Language Support** (Hindi + English with auto-detection)  
✅ **Archive System** (Personal search history)  
✅ **Educational Component** (Media literacy resources)  
✅ **Authority Reporting** (Government escalation pathway)  
✅ **Complete GCP Deployment** (Production-ready hosting)

**Technical Deliverables:**

1. **React Frontend:** 5 pages with responsive design
2. **FastAPI Backend:** RESTful API with 6 core endpoints
3. **Firestore Database:** Structured schema for all data types
4. **GCP Integration:** 11 cloud services properly configured
5. **Documentation:** Complete PRD, API docs, deployment guide
6. **Demo Video:** 3-minute presentation for judges

**Success Metrics:**

* **Functionality:** All user flows working end-to-end
* **Performance:** < 10 second response times
* **Accuracy:** 85%+ detection rate on test cases
* **Usability:** Non-technical users can operate independently
* **Scalability:** Platform ready for 100+ concurrent users

**🎬 Demo Script for Judges (3-Minute Video)**

**Minute 1: Problem & Solution (Hook)**

*"India faces a misinformation crisis. Fake news spreads faster than truth, threatening our national security. TruthLens is an AI-powered solution that detects misinformation in seconds, not hours."*

**Demo:** Show fake news example → Input to TruthLens → Instant verdict

**Minute 2: Technical Innovation (Proof)**

*"Powered by Google Cloud's most advanced AI services - Gemini, Vision API, and Fact Check API - TruthLens analyzes text, images, and URLs simultaneously."*

**Demo:** Upload suspicious image → Show real-time processing → Detailed analysis results

**Minute 3: Impact & Future (Vision)**

*"From personal fact-checking to government agencies, TruthLens scales to protect 700+ million internet users. This is just the beginning."*

**Demo:** Authority reporting feature → Archive system → Educational content

**📋 Important Development Notes**

**For AI Agents & New Developers:**

**Backend-Frontend Connection:**

// Frontend API Service (services/api.ts)  
const API\_BASE\_URL = process.env.REACT\_APP\_API\_URL || 'http://localhost:8000'  
  
export const analyzeContent = async (data: AnalysisRequest): Promise<AnalysisResponse> => {  
 const response = await axios.post(`${API\_BASE\_URL}/api/v1/verify`, data)  
 return response.data  
}

# Backend CORS Configuration (main.py)  
app.add\_middleware(  
 CORSMiddleware,  
 allow\_origins=["http://localhost:3000", "https://truthlens-app.web.app"],  
 allow\_credentials=True,  
 allow\_methods=["\*"],  
 allow\_headers=["\*"],  
)

**State Management Pattern:**

// Context for global state management  
interface AppState {  
 currentAnalysis: AnalysisResult | null  
 archive: AnalysisResult[]  
 user: User | null  
 loading: boolean  
}  
  
// Actions for state updates  
type AppAction =   
 | { type: 'SET\_ANALYSIS'; payload: AnalysisResult }  
 | { type: 'ADD\_TO\_ARCHIVE'; payload: AnalysisResult }  
 | { type: 'SET\_LOADING'; payload: boolean }

**Error Handling Strategy:**

# Backend error handling  
@app.exception\_handler(ValidationError)  
async def validation\_exception\_handler(request: Request, exc: ValidationError):  
 return JSONResponse(  
 status\_code=422,  
 content={"detail": exc.errors(), "type": "validation\_error"}  
 )

// Frontend error boundaries  
class ErrorBoundary extends Component<Props, State> {  
 componentDidCatch(error: Error, errorInfo: ErrorInfo) {  
 console.error('Application error:', error, errorInfo)  
 // Send to logging service  
 }  
}

**Integration Checklist:**

* [ ] Environment variables configured for all services
* [ ] CORS properly set up for cross-origin requests
* [ ] Authentication tokens securely managed
* [ ] Error boundaries implemented on frontend
* [ ] Loading states handled for all async operations
* [ ] Response caching implemented for performance
* [ ] Mobile responsive design tested
* [ ] Accessibility features included

**🔒 Security & Compliance Considerations**

**Data Protection:**

* **No Personal Data Storage:** User inputs are processed but not permanently stored without consent
* **Encryption in Transit:** All API calls use HTTPS/TLS
* **Temporary Processing:** Images and content processed in memory, not stored permanently
* **Audit Trails:** All authority reports logged for accountability

**API Security:**

* **Rate Limiting:** Prevent abuse with request throttling
* **Input Validation:** Sanitize all user inputs
* **Authentication:** Optional user accounts for archive features
* **CORS Configuration:** Restricted to authorized domains

**Compliance Framework:**

* **IT Act 2000:** Compliance with Indian cyber law
* **Data Protection:** GDPR-inspired privacy practices
* **Government Reporting:** Secure channels for authority escalation
* **Content Moderation:** Automatic filtering of inappropriate content

**🌟 Conclusion**

TruthLens represents a comprehensive solution to India's misinformation challenge, leveraging cutting-edge AI technology to protect national security and promote media literacy. With its robust architecture, user-friendly interface, and scalable cloud infrastructure, TruthLens is positioned to become an essential tool in the fight against misinformation.

The 6-day hackathon development timeline is aggressive but achievable with the detailed architecture and clear development roadmap outlined in this PRD. The combination of React frontend, FastAPI backend, and Google Cloud services creates a modern, scalable, and maintainable platform that can evolve beyond the hackathon into a production-ready service.

Success in this hackathon requires focused execution, clear communication, and adherence to the technical specifications outlined in this document. The modular architecture ensures that team members can work in parallel while maintaining system integration integrity.

**Document Version:** 1.0  
**Last Updated:** September 14, 2025  
**Next Review:** Post-Hackathon (September 21, 2025)  
**Status:** Ready for Development 🚀

*This PRD serves as the single source of truth for TruthLens development. All team members, AI agents, and stakeholders should refer to this document for project requirements, technical specifications, and implementation guidelines.*