**High Level System Design – ProtecTalk**

1. **Introduction**

ProtecTalk is an application designed to enhance user security by analyzing calls for potential scams after they occur. It leverages AI-driven natural language processing (NLP) to detect suspicious patterns in audio retrieved from the device's call recordings, alerting users and their trusted contacts post-call.

1. **System Architecture Overview**

ProtecTalk employs a client-server model. The Android mobile application, built using Kotlin, manages user interactions, accesses call audio from the device's recordings folder, and handles notifications. The backend—developed in Java—receives call transcriptions to support the development and training of future AI models. The mobile app handles all live scam detection locally by sending transcriptions to an integrated AI model. MongoDB Atlas stores user data, call transcriptions, call logs, and alerts, while Firebase Authentication secures user access. Notifications are delivered through Firebase Cloud Messaging (FCM).

1. **Software Modules**
   1. **Frontend Modules (Mobile Application)**
      1. **User Interface Module**

**Purpose:** Managing user interface and interactions

**Components:**

* + - Intro Screen
    - Login/Registration screen via authentication
    - Home screen as a settings screen
    - Manage protection screen
    - Trusted Contant/Protegee addition/remove screen

**Technology:** Kotlin

* + 1. **Authentication Module**

**Purpose:** Manage user sign-in and authentication token handling.

**Components:**

* + - Firebase Auth Integration (Email/Phone login)
    - ID Token Retrieval and Refreshing
    - Auth State Observer

**Technology:** Firebase Authentication SDK (Kotlin)

* + 1. **Number Reputation Module**

**Purpose:** Query server-side or third-party data to check if a phone number is known or suspected to be a scam.

* + - **Number Lookup Client:** Sends number to reputation service (internal DB or external API).
    - **Reputation Evaluator:** Flags number based on response.
    - **Trigger Dispatcher:** May initiate early warnings or contribute to final scam score.

**Technology:** Reputation DB or third-party service (e.g., Truecaller, Hiya)

* + 1. **Audio Transcription Module**

**Purpose:** Access recorded voice calls, manage permissions, and transcribe audio using Google’s transcription service.

**Components:**

* + - **Permission Manager:** Handles OS-level storage access permissions.
    - **Call Metadata Manager:** Identifies and selects relevant audio recordings (e.g., based on number or timestamp).
    - **Recording Accessor:** Locates and retrieves audio files from device storage.
    - **Transcription Service:** Sends audio to Google Cloud Speech-to-Text and receives raw transcripts.
    1. **Transcription Filtering Module**

**Purpose:** Perform on-device filtering of sensitive information in transcribed text before any external or local analysis.

**Components:**

* + - **Transcription Receiver:** Accepts raw transcript from the transcription module.
    - **Sensitive Information Filter:** Sends text to Google Cloud Data Loss Prevention (DLP) API to detect and redact sensitive data (e.g., names, phone numbers, IDs, Addresses).
    1. **Scam Analysis Module**

**Purpose:** Analyze sanitized transcriptions using a dedicated AI model to detect potential scam patterns.

**Components:**

* + - **Scam Request Handler:** Sends redacted transcript to the remote scam detection model.
    - **Response Interpreter:** Parses the scam probability score or risk level.
    - **Decision Dispatcher:** Triggers downstream actions (alerts, logs, UI updates) based on analysis outcome.
    1. **Alerts Module**

**Purpose:** Manage alerts based on analysis results and system events.

**Components:**

* **Local Alert Manager:** Provides user feedback through visual (notification banners), haptic (vibration), and vocal cues.
* **Remote Alert Sender:** Sends alerts to remote users or systems via Firebase Cloud Messaging (FCM).

**Technology:** Firebase Cloud Messaging (FCM), Android Notification APIs, Text-to-Speech (TTS), Vibration API

* 1. **Backend Modules (Server)**
     1. **API Gateway Module**

**Purpose:** Managing external API requests and routing

**Components:**

* Authentication middleware
* Request validation

**Technology:** Sprint Boot / Security

* + 1. **User Management Module**

**Purpose:** Handling user accounts and relationships

**Components:**

* User registration
* Authentication
* Profile management
* Protector relationship management

**Technology:** Spring Security + JWT/OAuth2, Firebase Admin SDK for Java, Spring Data JPA

* + 1. **Alert Management Module**

**Purpose:** Processing and distributing alerts

**Components:**

* Alert generator
* Notification dispatcher
* Alert history manager

**Technology:** Node.js, FCM Admin SDK

* + 1. **Data Access Module**

**Purpose:** Managing database operations

**Components:**

* Database connection pool
* Query builder
* Data models
* Caching layer

**Technology:** Mongoose ODM, MongoDB drivers

* + 1. **Monitoring and Logging Module**

**Purpose:** System health monitoring and error tracking

**Components:**

* Application logs
* Error reporting
* Audit trails

**Technology:** Winston, Prometheus, ELK Stack

* + 1. **Fraud Detection & Knowledge Base Service Module**

**Purpose:** Maintains and updates a live database of known scam patterns, keywords, and fraud behavior.

**Components:**

* Fraud Database Updater – Gathers new scam data from external sources and internal detections from user reports.
* Pattern Recognition Engine – Stores and refines AI-detected fraud patterns.
* Keyword List Management – Maintains a dynamic list of scam-related keywords.

**Technology:** Python (Flask), MongoDB, External APIs (for scam updates).

1. **Database Design**
   1. **User Data Database (MongoDB)**

**Purpose:** Stores user-related information, including profiles, relationships, devices, and activity logs.

**Key Collections:**

* Users – Stores user details and authentication data.
* Protectors – Manages user-protector relationships and notifications.
* Devices – Tracks user devices and messaging tokens.
* Call Logs – Records call history with timestamps and alert statuses.
* Call Transcriptions – Transcriptions of potential scam calls approved and sent by the user.
* Alerts – Stores scam alerts, confidence scores, and acknowledgment statuses.
  1. **Scam Pattern Database (MongoDB)**

**Purpose:** Maintains scam detection patterns and AI models for fraud analysis.

**Key Collections:**

* Scam Patterns – Contains known fraud patterns and severity levels.
* Fraud Scripts – Stores scam script samples and common phrases.
* Keywords – Lists important words and contextual weights for detection.
* ML Models – Manages trained machine learning models and performance metrics.

1. **Data Flow & Communication**

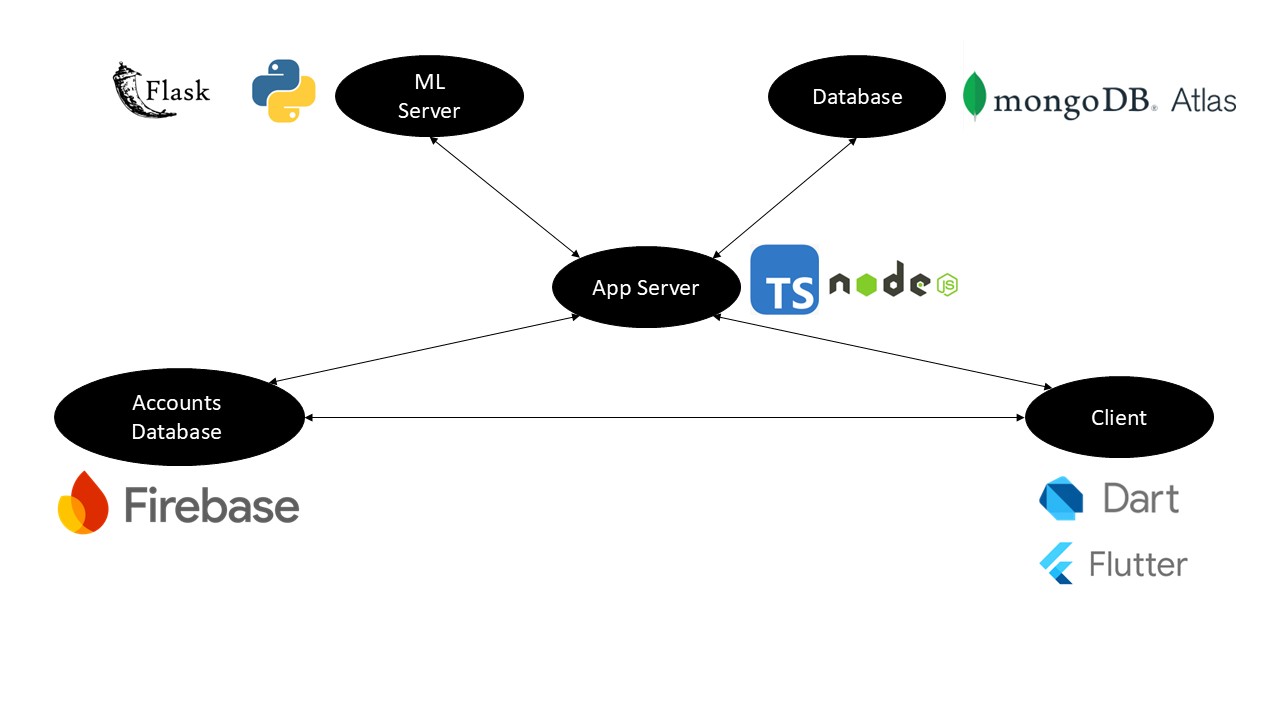
* **Call Detection & Notification:** The app detects an incoming call and begins monitoring for suspicious patterns.
* **Local Transcription & Scam Analysis:** The conversation is transcribed and analyzed in real-time using on-device AI.
* **Alert Generation & Logging:** If a scam is detected, an alert is generated locally and sent to the server for record-keeping (if user permits).
* **Server Processing & Notifications**: The backend stores metadata and triggers alerts to users and protectors via FCM.
* **Scam Intelligence Updates** – User reports and external threat feeds continuously refine scam detection models and patterns.

1. **Integration Points**

* Firebase Authentication: User authentication and authorization
* Firebase Cloud Messaging: Real-time notifications
* Google Cloud Speech-to-Text: Speech recognition and transcription
* MongoDB Atlas: Cloud-hosted database service
* External AI Services: Optional integration with third-party AI services for enhanced detection

1. **Security Considerations**

* End-to-end encryption for all communications
* Secure storage of sensitive user data
* Regular security audits and penetration testing
* Compliance with relevant data protection regulations
* User consent management for audio recording and processing

**Architecture Diagram**