**CyberGuard AI - National Cyber Crime Reporting Portal**

**🌟 Overview**

**CyberGuard AI** is a cutting-edge platform designed to revolutionize the way cybercrime complaints are reported, tracked, and managed. By harnessing the power of artificial intelligence—specifically Google’s Gemini AI—this platform offers an intuitive and secure interface for users to submit and monitor cybercrime complaints. Built with **Streamlit** for a user-friendly frontend and **Supabase** for secure data management, CyberGuard AI ensures that every complaint is processed efficiently, categorized accurately, and handled with the utmost care for user privacy and security.

**Key Benefits**

* **AI-Driven Categorization**: Automatically classifies complaints using advanced AI, ensuring swift and accurate processing.
* **User-Friendly Interface**: Simplifies complaint submission with an intuitive design and support for multiple input methods (text, voice, files).
* **Secure and Private**: Protects sensitive user data with end-to-end encryption and compliance with privacy regulations.
* **Real-Time Tracking**: Provides transparency with real-time updates on complaint status.

CyberGuard AI is more than just a reporting tool—it's a comprehensive solution empowering individuals and authorities to combat cybercrime effectively.

**🔑 Features**

CyberGuard AI offers a robust set of features to streamline the cybercrime reporting process:

* **Secure User Authentication**: Protects access with Supabase Auth, supporting multi-factor authentication (MFA).
* **Flexible Complaint Submission**: Submit complaints via text, voice recordings, or file uploads, with multi-language support.
* **AI-Powered Categorization**: Leverages Google Gemini AI to categorize complaints (e.g., Cyber Harassment, Financial Fraud) accurately.
* **Real-Time Complaint Tracking**: Monitor complaint status updates instantly using a unique ticket ID.
* **Secure Data Management**: Stores data in Supabase with encryption, ensuring compliance with GDPR and CCPA.
* **User Dashboard**: Centralized interface to manage complaints, view statuses, and access support resources.
* **Multi-Language Support**: Enhances accessibility for users worldwide.

These features combine to deliver a powerful, accessible, and secure platform for addressing cybercrime.

**🚀 Workflows**

CyberGuard AI revolves around two core workflows: **Complaint Submission** and **Complaint Tracking**, designed for simplicity and efficiency.

**Complaint Submission Workflow**

1. **Log In**: Access the platform with secure credentials.
2. **Select Language**: Choose your preferred language for the interface.
3. **Choose Input Method**:
   * **AI Chatbot**: Answer guided questions via a conversational interface powered by NLP.
   * **Manual Form**: Fill out a structured form with complaint details.
4. **Provide Details**: Submit text, voice recordings, or files (e.g., screenshots).
5. **Review and Submit**: Confirm accuracy and send the complaint.
6. **Receive Ticket ID**: Get a unique ID for tracking.

**Complaint Tracking Workflow**

1. **Log In**: Sign into your account.
2. **Navigate to Track Complaint**: Access the tracking section from the dashboard.
3. **Enter Ticket ID**: Input your unique ID.
4. **View Status**: Check real-time updates (e.g., "Under Review," "Resolved").

**Text-Based Diagram**:

text

CollapseWrapCopy

[User] --> [Login] --> [Select Language] --> [Input Method: Chatbot/Form] --> [Submit Details] --> [Ticket ID] --> [Track Status]

**💻 Technology Stack**

CyberGuard AI is built with a modern, reliable tech stack:

* **Frontend**: **Streamlit** - Delivers an interactive, responsive user interface.
* **Backend**: **Python (Flask/FastAPI)** - Manages requests, data processing, and integrations.
* **AI**: **Google Gemini AI** - Powers complaint categorization and NLP features.
* **Database**: **Supabase** - Securely stores user data and complaints with PostgreSQL.
* **Authentication**: **Supabase Auth** - Provides OAuth and MFA support.
* **Additional Libraries**:
  + **SpeechRecognition**: Converts voice inputs to text.
  + **gTTS**: Enables text-to-speech (if applicable).
  + **ReportLab**: Generates PDF reports.

This stack ensures performance, scalability, and a seamless user experience.

**🏗️ Architecture**

CyberGuard AI’s architecture is modular and scalable, with distinct layers:

* **Client Layer**: Streamlit UI for user interaction.
* **API Layer**: Python-based backend (Flask/FastAPI) handling requests and logic.
* **Processing Layer**: Google Gemini AI for categorization and data processing.
* **Data Storage Layer**: Supabase for encrypted storage and retrieval.
* **External Services**: Integrations with AI and other APIs.

**Text-Based Diagram**:

text

CollapseWrapCopy

+--------------------+

| Client Layer |

| (Streamlit UI) |

+--------------------+

|

v

+--------------------+

| API Layer |

| (Python/Flask/API) |

+--------------------+

|

v

+--------------------+

| Processing Layer |

| (Google Gemini AI) |

+--------------------+

|

v

+--------------------+

| Data Storage Layer |

| (Supabase) |

+--------------------+

This design ensures maintainability and adaptability to growing demands.

**📝 Usage Guide**

**Submitting a Complaint**

1. **Log In**: Sign in with your credentials.
2. **Select Language**: Pick your preferred language.
3. **Choose Input Method**:
   * **AI Chatbot**: Follow prompts to provide details conversationally.
   * **Manual Form**: Enter details in a structured form.
4. **Provide Details**: Add text, voice, or files.
5. **Review and Submit**: Verify and send your complaint.
6. **Receive Ticket ID**: Note the ID for tracking.

**Tracking a Complaint**

1. **Log In**: Access your account.
2. **Navigate to Track Complaint**: Go to the tracking section.
3. **Enter Ticket ID**: Input your ID.
4. **View Status**: See updates and details.

**Tips**: Save your ticket ID securely and check status regularly for updates.

**🛡️ Security & Privacy**

CyberGuard AI prioritizes user trust with robust security measures:

* **End-to-End Encryption**: Data is encrypted in transit (TLS 1.3) and at rest (AES-256).
* **Secure Authentication**: MFA and OAuth options via Supabase Auth.
* **Data Minimization**: Collects only essential data, deleted after processing if requested.
* **Compliance**: Adheres to GDPR, CCPA, and other privacy standards.
* **User Control**: Delete or anonymize your data anytime via the dashboard.

These safeguards ensure sensitive complaint data remains confidential.

**🧠 AI-Powered Features**

AI enhances CyberGuard AI’s functionality:

* **Complaint Categorization**: Google Gemini AI analyzes inputs to classify complaints (e.g., "Phishing" or "Identity Theft").
* **Natural Language Processing**: Understands text and voice inputs for seamless interaction.
* **Detail Extraction**: Pulls key info (e.g., dates, names) to aid processing.

**Example**: A user submits "I got a suspicious email asking for my bank details." The AI categorizes it as "Phishing" and extracts relevant details for authorities.

**📁 Project Structure**

**Folder Structure Explanation**

Unlike traditional multi-tiered applications with separate backend, frontend, and database directories, CyberGuard AI is implemented as a single-file Python application using Streamlit. This monolithic structure consolidates all functionalities—UI, business logic, AI integration, and database interactions—into one file (app.py). Below is a detailed breakdown of the logical structure within this single file and any supporting elements:

**1. Main Application File (app.py)**

* **Purpose**: Central entry point containing all code for the web interface, AI processing, database operations, and utility functions.
* **Structure Within the File**:
  + **Imports**: Includes essential Python libraries (streamlit, google.generativeai, speech\_recognition, supabase, etc.).
  + **Configuration**: Defines hardcoded configuration variables for Supabase (URL and key), SMTP (email settings), and Google Gemini API key.
  + **Function Definitions**:
    - generate\_cybercrime\_tips: Generates safety tips using Gemini AI.
    - send\_confirmation\_email: Sends email notifications with complaint details and tips.
    - recognize\_speech: Handles speech-to-text conversion with retry logic.
    - get\_gemini\_response: Interacts with the Gemini API for text and file processing.
    - categorize\_complaint: Performs AI-driven complaint categorization with detailed prompting.
    - process\_chatbot\_input: Manages chatbot interactions and form data collection.
    - dashboard: Defines the main UI logic and page navigation.
  + **Session State Initialization**: Uses init\_session\_state to manage application state (e.g., chat\_history, form\_data).
  + **UI and Logic**: Handles authentication, page rendering, and user interactions via Streamlit components.
* **Role**: Acts as both backend and frontend, leveraging Streamlit’s server-side rendering.

**2. Supporting Files (Optional)**

CyberGuard AI does not use separate directories, but additional files may be useful:

* **requirements.txt**: Lists dependencies for easy installation.
* streamlit==1.27.0
* google-generativeai==0.3.0
* supabase==1.0.3
* speechrecognition==3.10.0
* pyaudio==0.2.13
* reportlab==4.0.4
* pydub==0.25.1
* **.env (Optional)**: Stores sensitive credentials instead of hardcoding them.
* SUPABASE\_URL=https://example.supabase.co
* SUPABASE\_KEY=your\_supabase\_key
* SMTP\_EMAIL=your\_email@example.com
* SMTP\_PASSWORD=your\_smtp\_password
* GEMINI\_API\_KEY=your\_gemini\_api\_key
* **.gitignore**: Prevents unnecessary files from being committed.
* .env
* \_\_pycache\_\_/
* \*.pyc
* \*.wav
* **README.md**: Provides an overview, setup instructions, and usage guide.

**3. Database Integration (Supabase)**

* **Implementation**: Managed within app.py using the supabase Python client.
* **Structure**: No separate /database/ folder; inline database interactions.
  + save\_to\_supabase: Inserts complaint data.
  + fetch\_complaint\_from\_supabase: Retrieves complaint details.
  + register\_user and sign\_in\_user: Handle user authentication.
* **Schema**:
* CREATE TABLE users (
* id SERIAL PRIMARY KEY,
* username TEXT NOT NULL UNIQUE,
* password TEXT NOT NULL,
* email TEXT NOT NULL UNIQUE
* );
* CREATE TABLE complaints (
* id SERIAL PRIMARY KEY,
* ticket\_id TEXT NOT NULL UNIQUE,
* data JSONB NOT NULL,
* translated\_data JSONB NOT NULL,
* status TEXT NOT NULL,
* date\_filed TEXT NOT NULL,
* last\_updated TEXT NOT NULL
* );

**4. Documentation (Optional)**

* **No /docs/ Folder**, but files like:
  + SETUP\_GUIDE.md: Installation and running instructions.
  + API\_REFERENCE.md: Function documentation (if APIs are exposed).

**5. Scripts (Optional)**

* **No /scripts/ Folder**, but useful scripts include:
  + **start.sh**: Launches the Streamlit app.
  + #!/bin/bash
  + streamlit run app.py --server.port 8501
  + **deploy.sh**: Deploys to cloud platforms.
  + #!/bin/bash
  + git push heroku main
  + heroku ps:scale web=1

**6. Key Characteristics**

* **Monolithic Design**: Combines frontend (Streamlit UI), backend (Python logic), and database interactions.
* **Dependency-Driven**: Requires requirements.txt for setup.
* **Minimal File System Overhead**: No complex folder hierarchy.

**Steps to Set Up and Run the Code**

**1. Install Required Dependencies**

* **Prerequisites**:
* sudo apt-get install portaudio19-dev # Ubuntu
* brew install portaudio # macOS
* pip install pyaudio # Windows
* **Clone the Repository**:
* git clone https://github.com/your-repository-name/CyberGuardAI.git
* cd CyberGuardAI
* **Install Dependencies**:
* pip install -r requirements.txt

**2. Set Up Environment**

* **Create .env File**:
* touch .env
* SUPABASE\_URL=https://example.supabase.co
* SUPABASE\_KEY=your\_supabase\_key
* SMTP\_EMAIL=your\_email@example.com
* SMTP\_PASSWORD=your\_smtp\_password
* GEMINI\_API\_KEY=your\_gemini\_api\_key
* **Modify app.py**:
* import os
* supabase\_url = os.environ.get("SUPABASE\_URL")
* supabase\_key = os.environ.get("SUPABASE\_KEY")
* SMTP\_EMAIL = os.environ.get("SMTP\_EMAIL")
* SMTP\_PASSWORD = os.environ.get("SMTP\_PASSWORD")
* genai.configure(api\_key=os.environ.get("GEMINI\_API\_KEY"))

**3. Set Up Supabase Database**

* **Create Tables in Supabase SQL Editor**:
* CREATE TABLE users (...);
* CREATE TABLE complaints (...);
* **Ensure Credentials Match app.py**.

**4. Run the Application**

* **Navigate to Project Directory**:
* cd CyberGuardAI
* **Launch Streamlit**:
* streamlit run app.py

**5. Running Tests (Optional)**

**Create test\_app.py**:

import unittest

* from app import recognize\_speech
* class TestCyberGuardAI(unittest.TestCase):
* def test\_speech\_recognition(self):
* result = recognize\_speech("en-IN")
* self.assertIsNotNone(result)
* if \_\_name\_\_ == "\_\_main\_\_":
* unittest.main()
* **Run Tests**:
* python -m unittest test\_app.py

**6. Troubleshooting**

* **Common Issues**:
* pip uninstall pyaudio && pip install pyaudio # Fix PyAudio issues

**🔌 API Integrations**

* **Google Gemini AI**: Categorization and NLP.
* **Supabase**: Database, authentication, and storage.
* **SpeechRecognition**: Voice-to-text conversion (optional).
* **gTTS**: Text-to-speech (optional).
* **ReportLab**: PDF report generation.

These integrations drive the platform’s core capabilities.

**🌐 Deployment Options**

**Streamlit Cloud**

1. **Push to GitHub**: Upload code to a GitHub repository.
2. **Connect to Streamlit Cloud**: Link your repo in the Streamlit Cloud dashboard.
3. **Configure Secrets**: Add API keys (e.g., Supabase, Gemini) securely.
4. **Deploy**: Launch the app with one click.

Alternative options include Docker or VPS deployment for custom setups.

**📄 License**

This project is licensed under the **MIT License**. See the LICENSE file for details.

**🙏 Acknowledgements**

* **Streamlit**: For the intuitive frontend framework.
* **Supabase**: For secure database and auth services.
* **Google Gemini**: For advanced AI capabilities.
* **Python Community**: For essential libraries and tools.
* **Contributors**: [Add team members or contributors here].

Thank you to all who’ve made CyberGuard AI possible!