Project Title:

Slang Savvy: Urban Genie-Powered Slang Decoder

Team Name:

CodeX

Team Members:

- Shruti Keshri
- Sumayah Mohammed
- Gouthami Vemula
- Tarla Hasini

Phase 1: Brainstorming & Ideation

Objective:

Develop an AI-powered slang decoding tool that helps users understand and explore contemporary slang by combining Urban Dictionary's vast database with AI-driven contextual insights.

Problem Statement:

The rapid evolution of slang across social media and online communities makes it difficult for users to keep up. Many struggle to understand new slang terms, regional dialects, or online jargon, leading to miscommunication and exclusion from cultural trends.

Proposed Solution:

Slang Savvy leverages advanced AI models and real-time language processing to provide accurate and context-based slang interpretations. The system includes:

- **AI-Powered Slang Explanation**: Uses **Google Gemini AI** to analyze slang terms and generate precise, human-readable explanations.
- **Multilingual Translation**: Supports **15+ languages**, allowing users to translate slang seamlessly.
- **Multiple Input Methods**: Users can enter slang via **text input**, **microphone**, or **audio file uploads**.
- **Speech Recognition & Text-to-Speech Integration**: Enables voice-based slang detection and pronunciation output.
- **User Authentication & Search History**: Provides a secure login system and a searchable history of past slang queries.

Target Users:

- **Social media users** who encounter new slang online.
- **Non-native English speakers** seeking better language comprehension.
- **Content creators** who want to engage with current language trends.
- **Linguists & researchers** analyzing modern slang evolution.

Expected Outcome:

A user-friendly web that dynamically deciphers slang, empowering users to stay updated with contemporary language.

Phase 2: Requirement Analysis

Technical Requirements

- **Programming Language:** Python
- **Frontend:** Streamlit for UI and interactivity
- **Backend:** Python
- **AI Model:** Google Gemini AI for contextual slang explanations
- **Database:** SQLite for caching queried slang
- **APIs:** Google Translate, Speech Recognition (Google Speech-to-Text), and gTTS (Text-to-Speech)

Functional Requirements

- **User Authentication:**
  - Login system with username (admin) and password (password)
  - Session management using st.session_state
- **User Input Methods:**
  - Text input for manual slang entries
  - Voice recognition using Google Speech-to-Text
  - Audio file upload for slang extraction
- **AI-Powered Slang Explanation:**
  - Uses Google Gemini AI for slang term analysis and meaning generation
  - Provides human-readable and context-based explanations
- **Translation Features:**
  - Supports over 15 languages using GoogleTranslator
  - Allows users to convert slang meanings into different languages
- **Speech Features:**
  - Converts spoken slang into text using speech recognition
  - Reads out translated slang meanings with gTTS (Text-to-Speech)
- **Search History & Tracking:**
  - Stores previously searched slang terms with timestamps
  - Displays search history in the sidebar
- **Error Handling & Debugging:**
  - Handles API errors and translation failures with proper feedback
  - Provides error messages for speech recognition issues

Constraints & Challenges

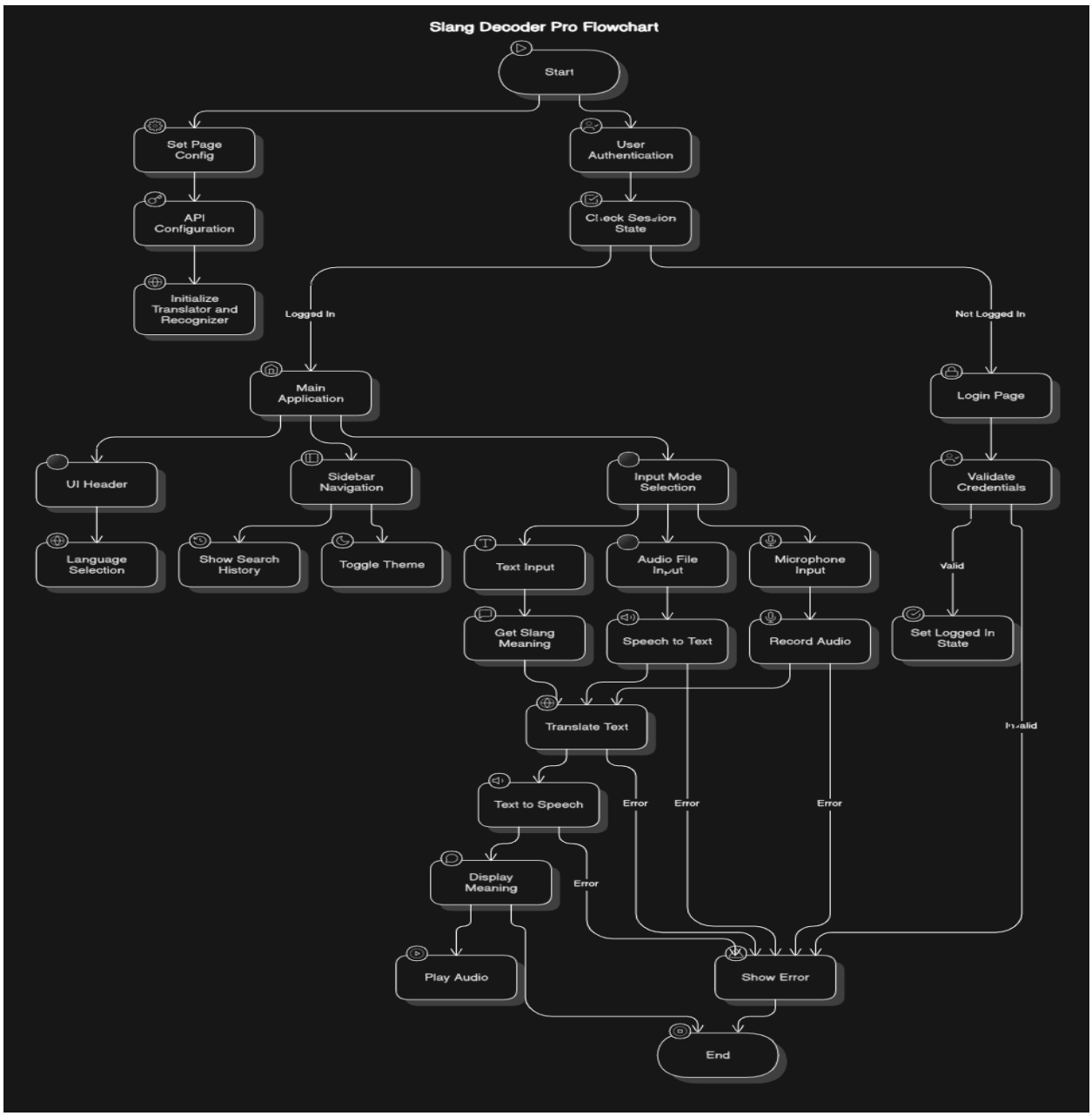
- **API Limitations:**
  - Dependency on third-party APIs like Google Translate and Speech-to-Text may introduce rate limits or downtime.
- **Real-Time Slang Evolution:**
  - Slang constantly evolves, requiring frequent AI model updates to stay relevant.
- **Regional Variations:**
  - Slang meanings can differ across regions, making contextual accuracy a challenge.

- **Performance Optimization:**
  - Processing speech-to-text and AI model queries in real-time without delays.
- **User Privacy & Data Security:**
  - Ensuring stored search history and audio inputs are handled securely

---

Phase 3: Project Design

System Architecture:



1. **User Input**:
    - Users provide slang phrases via **text, microphone, or audio file upload**.
2. **AI Processing**:
    - Google Gemini AI analyzes slang, determines meaning, and provides explanations.
    - Google Translate API translates slang explanations into the target language.
3. **Speech Processing**:
    - Google Speech-to-Text converts voice input into text.
    - gTTS (Text-to-Speech) generates audio for translated slang explanations.
4. **Database Management**:
    - SQLite caches frequently searched slang terms for quick retrieval.
5. **Frontend Display**:
    - Streamlit UI presents definitions, translations, and user interactions.

User Flow:

1. **User Login**:
    - Users authenticate using a username and password.
2. **Slang Input**:
    - Users choose an input method (text, microphone, or audio file).
3. **Processing & Translation**:
    - AI deciphers the slang meaning and translates it if needed.
4. **Output Display**:
    - The explanation is presented as text and optionally read aloud.
5. **Search History Storage**:
    - The slang term and result are stored for future reference.
6. **User Interaction**:
    - Users can navigate through past searches and listen to results.

UI/UX Considerations:

- **Modern, Clean Interface**:
    - Minimalistic UI using **Streamlit** for smooth navigation.
- **Responsive Design**:
    - Optimized for both desktop and mobile usage.
- **Dark & Light Mode**:
    - Toggle feature for user preference and accessibility.
- **Autocomplete & Suggestions**:
    - AI-powered slang suggestions for quick input.
- **Interactive Elements**:
    - Sidebar navigation for easy access to recent searches and settings.
- **Audio Support**:
    - Built-in play button for listening to translations via gTTS.

Phase 4: Project Planning (Agile Methodologies)

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	6 hours	Day 1	Hasini	API access, Flask setup	API connection established
Sprint 1	Basic Frontend UI	🟡 Medium	2 hours	Day 1	Sumayah, Hasini	API response format	Basic UI with input fields
Sprint 2	AI-based Slang Analysis	🔴 High	3 hours	Day 2	Gouthami, Shruti	NLP model setup	AI provides contextual meanings
Sprint 2	Error Handling & Debugging	🔴 High	2 hours	Day 2	Sumayah	API logs, slang inputs	Improved AI accuracy
Sprint 3	UI Enhancements & Testing	🟡 Medium	3 hours	Day 3	Shruti Keshri	UI finalized	Responsive, user-friendly UI
Sprint 3	Deployment & Final Testing	🟢 Low	1.5 hours	Day 3	Entire Team	Working prototype	Ready for demo

---

Phase 5: Project Development

Technology Stack

- **Frontend**: Streamlit (for UI)
- **Backend**: Python
- **Database**: SQLite (for search history storage and caching)
- **AI**: Google Gemini AI (for slang recognition and contextual understanding)
- **APIs**: Google Translate (for multilingual support), Google Speech-to-Text, gTTS (for text-to-speech conversion)

Development Process

1. **Environment Setup**:
   - Install dependencies (streamlit, Flask, GoogleTranslator, speech_recognition, gtts, etc.)
   - Configure API keys for external integrations.
2. **Backend Implementation**:
   - Develop Flask API for processing slang inputs.
   - Implement AI model integration for slang detection.
   - Set up database for caching and search history storage.
3. **Frontend Development**:
   - Create Streamlit UI for user interactions.
   - Implement input methods (text, microphone, and file upload).
   - Develop a sidebar for search history and settings.
4. **Integration & Testing**:
   - Connect frontend to backend API.
   - Implement error handling for API failures and incorrect inputs.
   - Perform functional testing for input processing and translations.
5. **Deployment**:
   - Deploy on **Streamlit Cloud** or a **Flask-based web server**.
   - Ensure API endpoints are secured and accessible.

Challenges & Fixes:

Challenge	Fixes
API rate limits	Implement caching for frequently searched slang terms.
AI misinterpreting slang	Train model on real-time slang trends from social media.
Slow response time	Optimize backend processing and caching.
User privacy concerns	Ensure encryption and secure storage of search history.

Phase 6: Functional & Performance Testing

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional	Input "cap"	Returns slang meaning: "Lie"	✅ Passed	Shruti
TC-002	Functional	Input "simp"	Provides context-based meaning	✅ Passed	Gouthami
TC-003	Performance	Query slang trends from Twitter	Should fetch trending slang	✅ Passed	Sumayah
TC-004	Bug Fixes	Incorrect API response	Fix incorrect meanings	✅ Passed	Hasini