# **Voice-Activated UPI Transaction System Using Streamlit and SQLite**

**BY**: Lohith R Gowda , Mohith G

### **1. Introduction**

This project implements a voice-activated UPI transaction system using Python, Streamlit for the frontend, and SQLite for database management. The system allows users to log in, view balances, make transactions through voice commands by uploading audio files, and manage accounts (add or delete users). This cloud-compatible system offers a user-friendly interface and voice-to-text processing using the Whisper ASR model in transformers.

### **2. Objectives**

* Implement a secure and interactive UPI transaction system using voice commands.
* Allow users to upload audio files for transcription and facilitate transactions between users.
* Display user balances before and after each transaction.

### **3. System Requirements**

* **Python**: Version 3.8 or higher
* **Libraries**:
  + **Streamlit**: Frontend interface
  + **SQLite**: Backend database management
  + **Transformers**: Whisper ASR model for transcription
  + **Torch**: Required for transformers model execution

### **4. Project Setup**

#### **4.1 Database Initialization**

The SQLite database is initialized with predefined users. Each user has an initial balance, a username, and a password. Usernames are simple and easily recognized names for reliable transcription.

#### **4.2 Code Files**

* **backend.py**:
  + Manages database operations such as user validation, transaction processing, and balance retrieval.
  + Key functions include:
    - create\_connection(): Establishes connection to SQLite database.
    - initialize\_db(): Initializes the database with predefined users.
    - validate\_login(): Verifies login credentials.
    - process\_transaction(): Processes transactions based on transcription and adjusts balances.
* **app.py**:
  + Provides an interactive Streamlit frontend where users can log in, view balances, upload audio files for transcription, and manage accounts.
  + Key sections include:
    - **Login Interface**: Users input credentials to access accounts.
    - **Transaction Interface**: Users can upload audio files, and the transcription initiates transactions.
    - **Add/Delete User**: Allows account management with database updates.
* **transcriber.py**:
  + Contains functions to transcribe audio files using the Whisper ASR model.

### **5. Code Overview**

#### **backend.py**

This file manages database interactions and transaction processing, including:

python

Copy code

import sqlite3

import re

def create\_connection():

conn = sqlite3.connect("transactions.db")

return conn

def initialize\_db():

# Initializes database with predefined users

conn = create\_connection()

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS accounts (name TEXT PRIMARY KEY, balance REAL, username TEXT UNIQUE, password TEXT)''')

# Predefined users with simple names for transcription reliability

users = [("Alice", 50000, "Alice", "123"), ("Bob", 50000, "Bob", "123"), ...]

for name, balance, username, password in users:

cursor.execute("INSERT OR IGNORE INTO accounts (name, balance, username, password) VALUES (?, ?, ?, ?)", (name, balance, username, password))

conn.commit()

conn.close()

Key functions include validate\_login, add\_user, delete\_user, update\_balance, get\_balance, and process\_transaction, which handles transcription processing and balance adjustments.

#### **transcriber.py**

This file transcribes uploaded audio files using the Whisper ASR model.

python

Copy code

from transformers import pipeline

# Whisper ASR pipeline initialization

pipe = pipeline("automatic-speech-recognition", model="openai/whisper-large-v3-turbo")

def transcribe\_audio(audio\_file):

try:

transcription = pipe(audio\_file)

return transcription['text']

except Exception as e:

return f"Error in transcription: {e}"

#### **app.py**

This file creates the Streamlit frontend interface.

python

Copy code

import streamlit as st

from transcriber import transcribe\_audio

from backend import initialize\_db, process\_transaction, validate\_login, add\_user, delete\_user, get\_balance

# Initialize the database

initialize\_db()

# Streamlit app UI

st.title("Voice-Activated UPI Transactions with Whisper")

uploaded\_file = st.file\_uploader("Upload an audio file for transcription", type=["wav", "mp3"])

if uploaded\_file:

transcription = transcribe\_audio(uploaded\_file)

result, balance\_before, balance\_after = process\_transaction(transcription, st.session\_state.user\_name)

st.write(result)

### **6. Functionality**

* **Login System**: Secure login with predefined credentials.
* **Transaction Processing**: Users upload an audio file with a transaction command, which is transcribed and processed.
* **Balance Display**: Shows balance before and after transactions.
* **Add/Delete User**: Allows users to manage accounts via the interface.

### **7. User Interface**

The Streamlit interface includes:

* **Home Page**: Login and account management.
* **Transaction Page**: Allows audio file uploads for transcription.
* **Account Management**: Displays balances and handles transactions.

### **8. Testing and Validation**

Testing focused on:

* **Audio Command Accuracy**: Ensures transcription correctly interprets commands.
* **Transaction Validation**: Verifies balance adjustments.
* **Database Integrity**: Tests add, delete, and update functions to maintain data consistency.

### **9. Limitations and Future Enhancements**

#### **Limitations**

* Limited to predefined users in the database.
* Only supports audio file upload for transcription.

#### **Future Enhancements**

* **Real-Time Voice API**: Integrate real-time voice-to-text for a more interactive experience.
* **Enhanced Security**: Add encryption for stored credentials.
* **Dynamic User Addition**: Extend support to new users with dynamic data input.

### **10. Conclusion**

The Voice-Activated UPI Transaction System provides an effective demonstration of managing transactions using voice commands. With future improvements, the system can be extended to offer real-time voice processing and enhanced security, making it a scalable solution for real-world applications.