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

BY: Lohith R Gowda - 1RVU22CSE093

G Mohith - 1RVU22CSE057

## **1. Introduction**

This project implements a voice-activated transaction system for UPI payments using Python, Streamlit for the frontend interface, and SQLite for database management. The system allows users to log in, view their balance, make transactions via voice commands, and add or delete accounts. This system is particularly designed for localized usage and offers an interactive interface where users can record voice commands to perform transactions.

## **2. Objectives**

The objectives of this project are:

* To implement a secure and user-friendly UPI transaction system.
* To use voice commands to facilitate transactions between users.
* To dynamically display the balance of the user before and after each transaction.

## **3. System Requirements**

* **Python**: Version 3.8 or higher
* **Libraries**:
  + Streamlit for frontend interface.
  + SQLite for backend database management.
  + SpeechRecognition for voice-to-text functionality (external setup required).

## 

## 

## **4. Project Setup**

### **4.1 Database Initialization**

The database, managed by SQLite, is initialized with predefined users, each with a starting balance and a username-password combination.

### **4.2 Code Files**

The project includes two main code files:

#### **backend.py**

This file manages all database operations including user validation, transaction processing, and balance retrieval. Key functions include:

* **create\_connection()**: Establishes a connection to the SQLite database.
* **initialize\_db()**: Initializes the database with predefined users.
* **validate\_login()**: Verifies login credentials.
* **process\_transaction()**: Processes transactions based on the voice transcription and adjusts balances.

#### **app.py**

This file uses Streamlit to create an interactive frontend where users can log in, view balances, initiate voice-based transactions, and perform other account management actions.

Key sections include:

* **Login Interface**: Users enter credentials to access their accounts.
* **Transaction Interface**: Users can record and transcribe voice commands to execute transactions.
* **Add/Delete User**: Allows adding or removing users from the database.

## 

## 

## **5. Code Overview**

**backend.py**

import sqlite3

# Database connection

def create\_connection():

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

return conn

# Initialize database with predefined users

def initialize\_db():

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 usernames matching their names and a default password "123"

users = [

("Aarav", 50000, "Aarav", "123"),

("Vihaan", 50000, "Vihaan", "123"),

("Arjun", 50000, "Arjun", "123"),

("Vivaan", 50000, "Vivaan", "123"),

("Ayaan", 50000, "Ayaan", "123"),

("Reyansh", 50000, "Reyansh", "123"),

("Sai", 50000, "Sai", "123"),

("Karan", 50000, "Karan", "123"),

("Rohan", 50000, "Rohan", "123"),

("Dev", 50000, "Dev", "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()

# Function definitions continue with validate\_login, add\_user, delete\_user, update\_balance, get\_balance, user\_exists, process\_transaction

**App.py**

import streamlit as st

from transcriber import record\_audio, 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")

st.write("Voice commands to make payments and clear bills.")

# Login, transaction, add/delete user functionality managed here

# Interface buttons for 'Start Recording', 'Stop Recording and Transcribe', and 'Logout'

## **6. Functionality**

1. **Login System**: Secure login with predefined credentials.
2. **Transaction Processing**: Allows users to record a transaction command, verifies the command, checks receiver validity, and updates balances.
3. **Balance Display**: Displays balance before and after each transaction, with clear separation in the interface.
4. **Add/Delete User**: Users can add or delete accounts through the interface, with instant database updates.

## **7. User Interface**

The user interface, built with Streamlit, includes:

* **Home Page**: Login and user management options.
* **Transaction Page**: Allows recording and transcribing of voice commands.
* **Account Management**: Displays current balance and logs transactions.

## **8. Testing and Validation**

Testing focused on:

* **Voice Command Accuracy**: Ensuring the transcription correctly interprets commands.
* **Transaction Validity**: Ensuring accurate balance adjustments.
* **Database Integrity**: Testing add, delete, and update functions to prevent data loss or duplication.

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

### **Limitations**

* Limited to predefined users in the database.
* Requires manual entry of voice commands for testing in absence of real voice-to-text API integration.

### **Future Enhancements**

* **Dynamic User Addition**: Extend to support new users.
* **Enhanced Security**: Include encryption for stored credentials.
* **Real-Time Voice API**: Integrate with a live voice-to-text API for better transcription accuracy.

## **10. Conclusion**

The Voice-Activated UPI Transaction System demonstrates a functional, interactive approach to managing transactions via voice commands. Future improvements could extend usability and security, making this a scalable solution for real-world applications.