ExpenseEase Documentation

ExpenseEase is a personal finance management application that provides users with a simple and intuitive interface for tracking and managing their financial transactions. This document outlines the purpose, functionality, and usage instructions for ExpenseEase.

Purpose

ExpenseEase is designed to assist users in managing their personal finances by offering features such as user authentication, transaction logging, and basic financial operations. The application aims to provide a straightforward and efficient way for users to keep track of their expenses, withdrawals, and transfers.

Functionality

ExpenseEase includes the following key features:

User Authentication:

Users can create accounts with unique usernames and passwords. The application ensures the uniqueness of usernames and provides error messages for existing usernames during account creation.

Transaction Logging:

All financial transactions are recorded in the SQLite database. Transactions include additions, withdrawals, and transfers between users, providing users with a comprehensive transaction history.

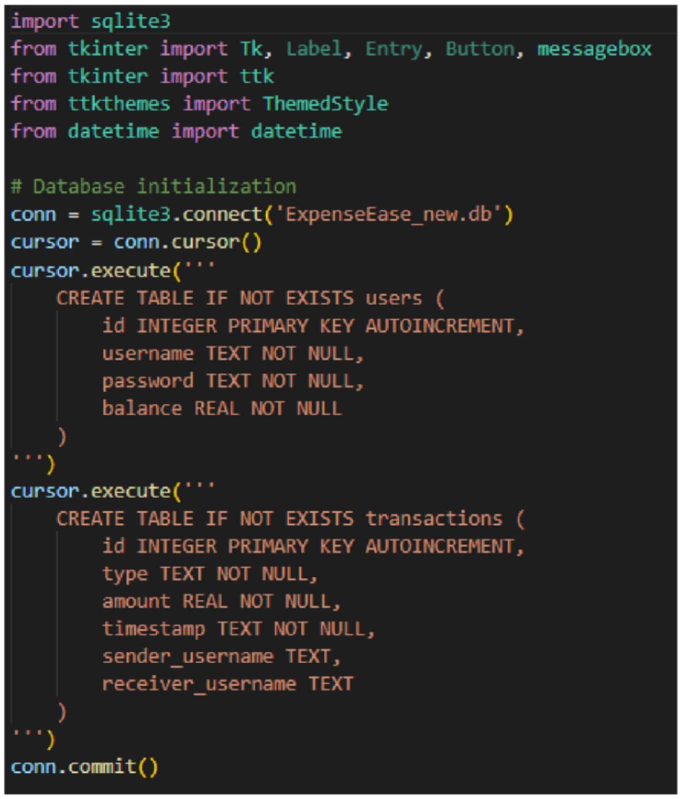
Dashboard Interface:

Upon successful login, users are presented with a dashboard that displays their username, current balance, and options to perform financial operations. The dashboard provides a convenient and centralized view of the user's financial status.

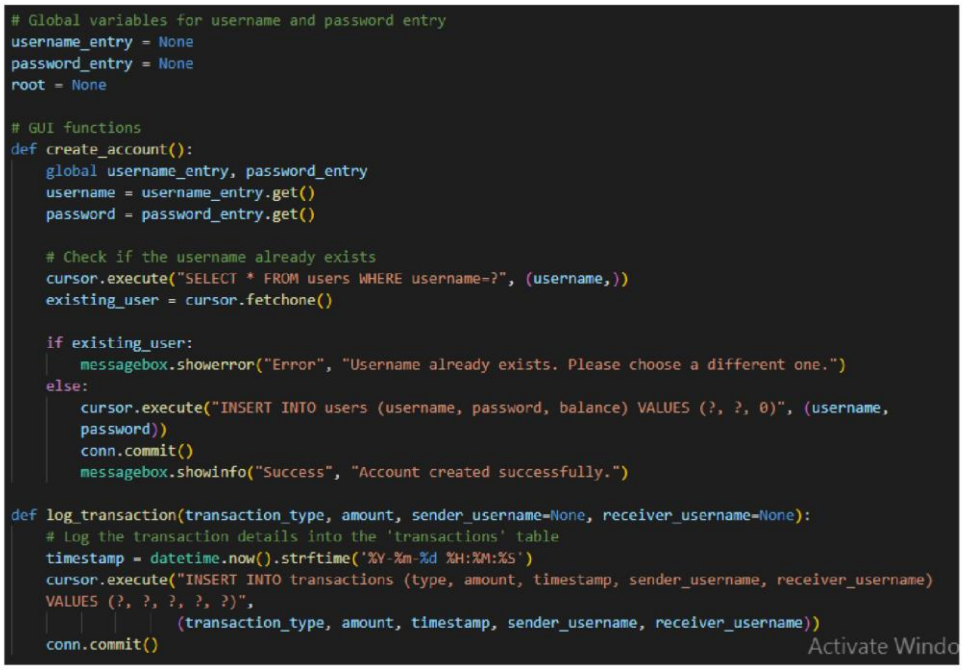
Themed GUI:

The graphical user interface is styled using the ttkthemes library, offering a visually appealing and consistent theme for the application.

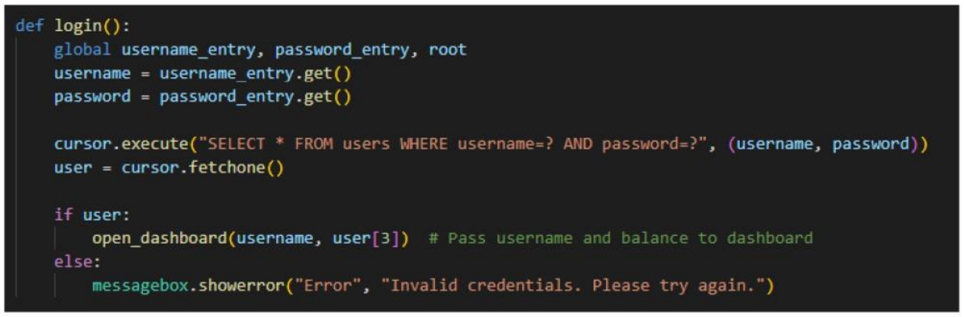
Screenshots



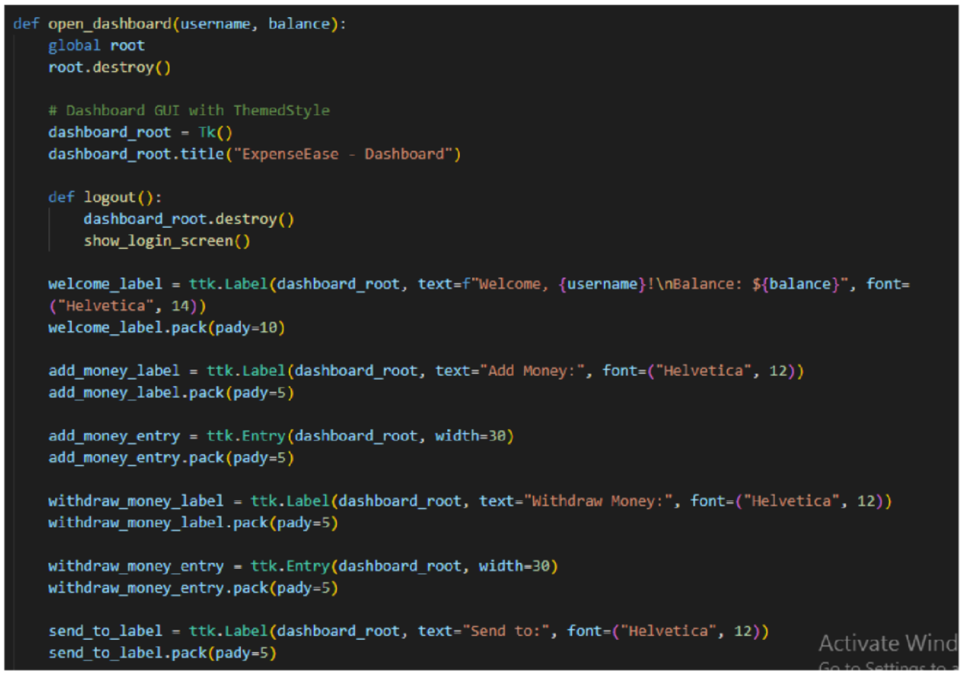
```python
import sqlite3
from tkinter import Tk, Label, Entry, Button, messagebox
from tkinter import ttk
from ttkthemes import ThemedStyle
from datetime import datetime

# Database initialization
conn = sqlite3.connect('ExpenseEase_new.db')
cursor = conn.cursor()
cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT NOT NULL,
        password TEXT NOT NULL,
        balance REAL NOT NULL
    )
''')
cursor.execute('''
    CREATE TABLE IF NOT EXISTS transactions (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        type TEXT NOT NULL,
        amount REAL NOT NULL,
        timestamp TEXT NOT NULL,
        sender_username TEXT,
        receiver_username TEXT
    )
''')
conn.commit()
```



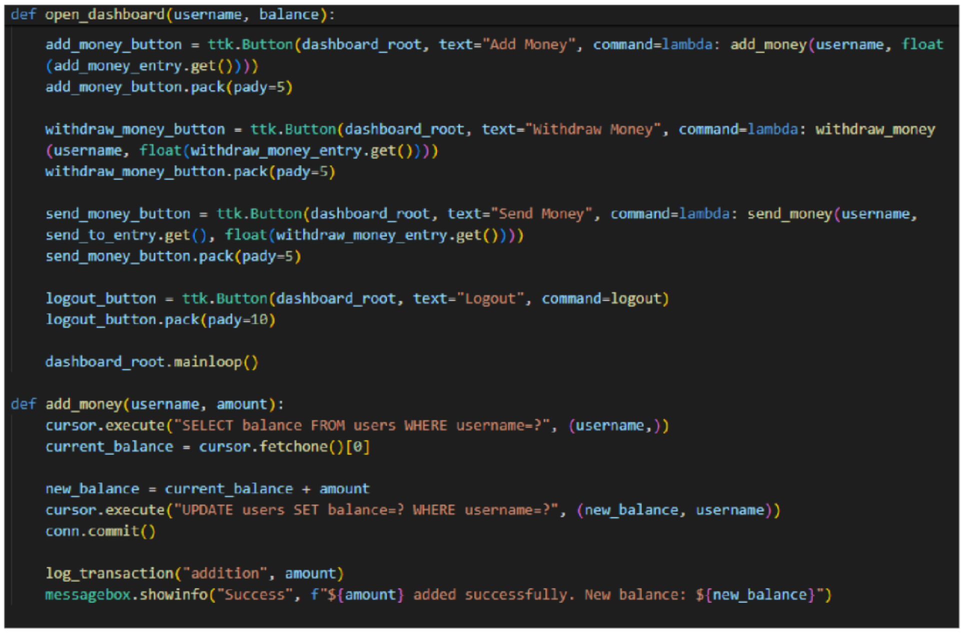
```python
# Global variables for username and password entry
username_entry = None
password_entry = None
root = None

# GUI functions
def create_account():
    global username_entry, password_entry
    username = username_entry.get()
    password = password_entry.get()

    # Check if the username already exists
    cursor.execute("SELECT * FROM users WHERE username=?", (username,))
    existing_user = cursor.fetchone()

    if existing_user:
        messagebox.showerror("Error", "Username already exists. Please choose a different one.")
    else:
        cursor.execute("INSERT INTO users (username, password, balance) VALUES (?, ?, 0)", (username,
        password))
        conn.commit()
        messagebox.showinfo("Success", "Account created successfully.")

def log_transaction(transaction_type, amount, sender_username=None, receiver_username=None):
    # Log the transaction details into the 'transactions' table
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    cursor.execute("INSERT INTO transactions (type, amount, timestamp, sender_username, receiver_username)
    VALUES (?, ?, ?, ?, ?)",
                   (transaction_type, amount, timestamp, sender_username, receiver_username))
    conn.commit()
```



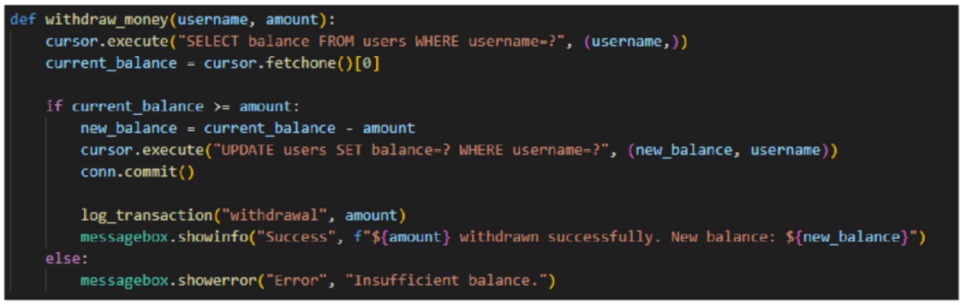
```python
def login():
    global username_entry, password_entry, root
    username = username_entry.get()
    password = password_entry.get()

    cursor.execute("SELECT * FROM users WHERE username=? AND password=?", (username, password))
    user = cursor.fetchone()

    if user:
        open_dashboard(username, user[3])  # Pass username and balance to dashboard
    else:
        messagebox.showerror("Error", "Invalid credentials. Please try again.")
```



```python
def open_dashboard(username, balance):
    global root
    root.destroy()

    # Dashboard GUI with ThemedStyle
    dashboard_root = Tk()
    dashboard_root.title("ExpenseEase - Dashboard")

    def logout():
        dashboard_root.destroy()
        show_login_screen()

    welcome_label = ttk.Label(dashboard_root, text=f"Welcome, {username}!\nBalance: ${balance}", font=
    ("Helvetica", 14))
    welcome_label.pack(pady=10)

    add_money_label = ttk.Label(dashboard_root, text="Add Money:", font=("Helvetica", 12))
    add_money_label.pack(pady=5)

    add_money_entry = ttk.Entry(dashboard_root, width=30)
    add_money_entry.pack(pady=5)

    withdraw_money_label = ttk.Label(dashboard_root, text="Withdraw Money:", font=("Helvetica", 12))
    withdraw_money_label.pack(pady=5)

    withdraw_money_entry = ttk.Entry(dashboard_root, width=30)
    withdraw_money_entry.pack(pady=5)

    send_to_label = ttk.Label(dashboard_root, text="Send to:", font=("Helvetica", 12))
    send_to_label.pack(pady=5)
```



```python
def open_dashboard(username, balance):
    add_money_button = ttk.Button(dashboard_root, text="Add Money", command=lambda: add_money(username, float
    (add_money_entry.get())))
    add_money_button.pack(pady=5)

    withdraw_money_button = ttk.Button(dashboard_root, text="Withdraw Money", command=lambda: withdraw_money
    (username, float(withdraw_money_entry.get())))
    withdraw_money_button.pack(pady=5)

    send_money_button = ttk.Button(dashboard_root, text="Send Money", command=lambda: send_money(username,
    send_to_entry.get(), float(withdraw_money_entry.get())))
    send_money_button.pack(pady=5)

    logout_button = ttk.Button(dashboard_root, text="Logout", command=logout)
    logout_button.pack(pady=10)

    dashboard_root.mainloop()

def add_money(username, amount):
    cursor.execute("SELECT balance FROM users WHERE username=?", (username,))
    current_balance = cursor.fetchone()[0]

    new_balance = current_balance + amount
    cursor.execute("UPDATE users SET balance=? WHERE username=?", (new_balance, username))
    conn.commit()

    log_transaction("addition", amount)
    messagebox.showinfo("Success", f"${amount} added successfully. New balance: ${new_balance}")
```



```python
def withdraw_money(username, amount):
    cursor.execute("SELECT balance FROM users WHERE username=?", (username,))
    current_balance = cursor.fetchone()[0]

    if current_balance >= amount:
        new_balance = current_balance - amount
        cursor.execute("UPDATE users SET balance=? WHERE username=?", (new_balance, username))
        conn.commit()

        log_transaction("withdrawal", amount)
        messagebox.showinfo("Success", f"${amount} withdrawn successfully. New balance: ${new_balance}")
    else:
        messagebox.showerror("Error", "Insufficient balance.")
```