REQUIREMENTS SPECIFICATION

```
def authenticate_user(username, password):
if username in users and users[username] == password:
return True
return False
```

2. *Account Management*:

```
   python
class Account:
   def __init__(self, account_number, balance):
      self.account_number = account_number
      self.balance = balance
   def deposit(self, amount):
      self.balance += amount
   def withdraw(self, amount):
      if amount > self.balance:
         raise ValueError("Insufficient funds")
      self.balance -= amount
```

3. *Transaction History*:

```
class TransactionHistory:
def *init*(self):
self.transactions = []
def add_transaction(self, transaction):
   self.transactions.append(transaction)
def get_transactions(self):
   return self.transactions
```

INITIALIZATION OF ENVIRONMENT VARIABLES

```
import os
    my_api_key = os.environ['MY_API_KEY']
    print(f"My API Key (direct access): {my_api_key}")
except KeyError:
    print("Error: MY_API_KEY environment variable not set.")
database_url = os.environ.get('DATABASE_URL', 'sqlite:///default.db')
print(f"Database URL: {database_url}")
secret_key = os.environ.get('APP_SECRET', 'a_default_secret_for_development')
print(f"Application Secret Key: {secret_key}")
print("\nAll environment variables:")
for key, value in os.environ.items():
    print(f"{key}={value}")
```

AI INTEGRATION WITH IBM WATSONX

```
from dotenv import load_dotenv
import os
load_dotenv() # Load variables from .env
api_key = os.environ.get("IBM_CLOUD_API_KEY")
project_id = os.environ.get("WATSONX_PROJECT_ID")
ibm_cloud_region = os.environ.get("IBM_CLOUD_REGION", "us-south") # Default to us-south
```

GOOGLE CLASSROOM SYNC

```
import os.path
from google.auth.transport.requests import Request
```

```
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
.profile.emails' for user emails
def authenticate_classroom_api():
    """Shows basic usage of the Classroom API.
    Prints the names of the first 10 courses the user has access to.
    """
    creds = None
    if os.path.exists('token.json'):
        creds = Credentials.from_authorized_user_file('token.json', SCOPES)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'client_secret.json', SCOPES) # Path to your downloaded client_secret.json
            creds = flow.run_local_server(port=0)
        with open('token.json', 'w') as token:
            token.write(creds.to_json())
    return creds
def get_classroom_service():
    creds = authenticate_classroom_api()
    try:
        service = build('classroom', 'v1', credentials=creds)
```

```
        return service
    except HttpError as err:
        print(f"An API error occurred: {err}")
        return None
().list(courseId=course_id_to_sync).execute()
```

PINECONE VECTOR DB INTEGRATION

```
pincode_data = {
    "522001": {"city": "Guntur", "state": "Andhra Pradesh", "lat": 16.3067, "lon": 80.4385},
    "500001": {"city": "Hyderabad", "state": "Telangana", "lat": 17.3850, "lon": 78.4867},
    "530001": {"city": "Visakhapatnam", "state": "Andhra Pradesh", "lat": 17.6868, "lon": 83.2185},
}
```

STREAMLIT FRONTEND UI

```
import streamlit as st
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
st.set_page_config(
    page_title="My Awesome Streamlit UI",
    page_icon=" ✨ ",
    layout="wide", # Can be "centered" or "wide"
    initial_sidebar_state="expanded" # Can be "auto", "expanded", or "collapsed"
)
```

```
st.title("🚀 My Interactive Streamlit App")
st.header("Explore Data and Interact with Widgets")
st.markdown("---") # A horizontal line for separation
# --- 3. Sidebar for Inputs ---
st.sidebar.header("⚙ App Settings")
name = st.sidebar.text_input("Enter your name:", "Guest")
st.sidebar.write(f"Hello, *{name}*!")
selected_option = st.sidebar.selectbox(
    "Choose an option:",
    ("Overview", "Data Explorer", "About")
)
st.sidebar.markdown("---")
```