Virtual Internship – 6.0

Group - 4

Vaibhavi Nimba Patil

Preetham Aditya

Vardhan Momula

Ravi Kumar Rathlavat

Mental Health Detection Dashboard

- The Mental Health Detection Dashboard is a comprehensive system designed to detect, analyze, and visualize mental health disorders from textual data. It integrates multiple functionalities into a single platform, enabling users to upload datasets, train predictive models, extract relationships, and explore insights interactively.

Features that make it a Cross-Domain Knowledge Mapping Tool

1. Dataset Loading & Preprocessing: Handles CSV datasets containing mental health-related text and labels. Cleans and standardizes textual data for consistent analysis.
2. Model Training & Prediction: Uses machine learning (TF-IDF + Logistic Regression) to classify disorders. Includes a voice assistant for real-time audio input and prediction.
3. Graph & Knowledge Mapping: Builds a concept graph connecting disorders, symptoms, and user input. Enables visualization of relationships between different mental health conditions.
4. Triplet Extraction: Extracts subject–relation–object triplets from user text to understand connections.
5. Semantic Search: Searches and retrieves similar cases across the dataset, linking concepts across domains.
6. Admin & Feedback System: Collects user feedback and tracks prediction history, contributing to dataset enrichment and better insights.

Milestone-1 Data loading and preprocessing

```
import streamlit as st
import pandas as pd
import spacy

# Load spaCy
nlp = spacy.load("en_core_web_sm")

# Load dataset
def load_df_from_path(path):
    df = pd.read_csv(path)
    df.columns = ["text", "label"]
    df["text"] = df["text"].astype(str)
    return df
```

```
# Triplet extraction
def extract_triplets(text):
    doc = nlp(text)
    triplets = []
    for sent in doc.sents:
        subj, verb, obj = "", "", ""
        for token in sent:
            if "subj" in token.dep_: subj = token.text
            if "obj" in token.dep_: obj = token.text
            if token.pos_ == "VERB": verb = token.lemma_
        if subj and verb and obj:
            triplets.append((subj, verb, obj))
    return triplets
# Example usage
df = load_df_from_path("dataset.csv")
for text in df["text"].head(3):
    st.write(f"Text: {text}")
    st.write("Triplets:", extract_triplets(text))
```

Milestone – 2 POStagging and Feature extraction

```
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer

# Load spaCy
nlp = spacy.load("en_core_web_sm")

# POS Tagging
def pos_tag_text(text):
    doc = nlp(text)
    return [(token.text, token.pos_) for token in doc]
```

```
# Feature extraction using TF-IDF
def extract_features(corpus):
    vect = TfidfVectorizer(stop_words="english", max_features=5000)
    X = vect.fit_transform(corpus)
    return X, vect

# Example usage
sample_text = [
    "I feel sad and anxious",
    "I have trouble sleeping at night"
]

# POS tagging
for t in sample_text:
    print("Text:", t)
    print("POS Tags:", pos_tag_text(t))

# TF-IDF features
X, vect = extract_features(sample_text)
print("Feature shape:", X.shape)
```

Model Training & Prediction

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Train model
def train_and_store(df):
    X, y = df["text"], df["label"]
    vect = TfidfVectorizer(stop_words="english", max_features=5000)
    X_vec = vect.fit_transform(X)
    clf = LogisticRegression(max_iter=1000)
    clf.fit(X_vec, y)
    st.session_state.vectorizer = vect
    st.session_state.model = clf

# Predict disorder
def predict_disorder(text):
    vec = st.session_state.vectorizer.transform([text])
    return st.session_state.model.predict(vec)[0]
```


```
# Predict disorder
def predict_disorder(text):
    vec = st.session_state.vectorizer.transform([text])
    return st.session_state.model.predict(vec)[0]
```

Milestone-3 Knowledge graph and Semantic Search

```
import networkx as nx
from pyvis.network import Network
import streamlit as st
import tempfile

# Build graph from sample data
G = nx.DiGraph()
G.add_node("depression", type="label")
G.add_node("sadness", type="concept")
G.add_edge("depression", "sadness", relation="related")

# Display graph in Streamlit
def show_pyvis(G, height=650):
    net = Network(height=f"{height}px", width="100%", directed=True)
    net.from_nx(G)
    tmp = tempfile.NamedTemporaryFile(delete=False, suffix=".html")
    net.save_graph(tmp.name)
    return tmp.name

html_file = show_pyvis(G)
st.components.v1.html(open(html_file, "r", encoding="utf-8").read(), height=650)
```

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample dataset
texts = ["I feel very sad", "I have trouble sleeping", "I feel anxious"]
labels = ["depression", "insomnia", "anxiety"]

# TF-IDF Vectorization
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)

# Semantic search function
def semantic_search(query, X, topk=2):
    q_vec = vectorizer.transform([query])
    sims = (q_vec @ X.T).toarray().flatten()
    idxs = np.argsort(sims)[-topk:][::-1]
    return [(texts[i], labels[i], sims[i]) for i in idxs]

# Example search
results = semantic_search("I feel sad")
st.write("Semantic search results:", results)
```

Milestone-4 Admin & Feedback

```
from datetime import datetime

# Store feedback
def submit_feedback(user, text):
    if "feedback" not in st.session_state:
        st.session_state.feedback = []
    st.session_state.feedback.append({
        "user": user,
        "text": text,
        "timestamp":
datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    })
```

```
# Admin view
def admin_dashboard():
    st.subheader("Admin Dashboard")
    if st.session_state.get("pred_history"):

st.dataframe(pd.DataFrame(list(reversed(st.session_state.pred_
history))))
    else:
        st.info("No prediction history yet.")


    st.subheader("Feedback Entries")
    if st.session_state.get("feedback"):
        st.dataframe(pd.DataFrame(st.session_state.feedback))
    else:
        st.info("No feedback entries yet.")
```

Login Page

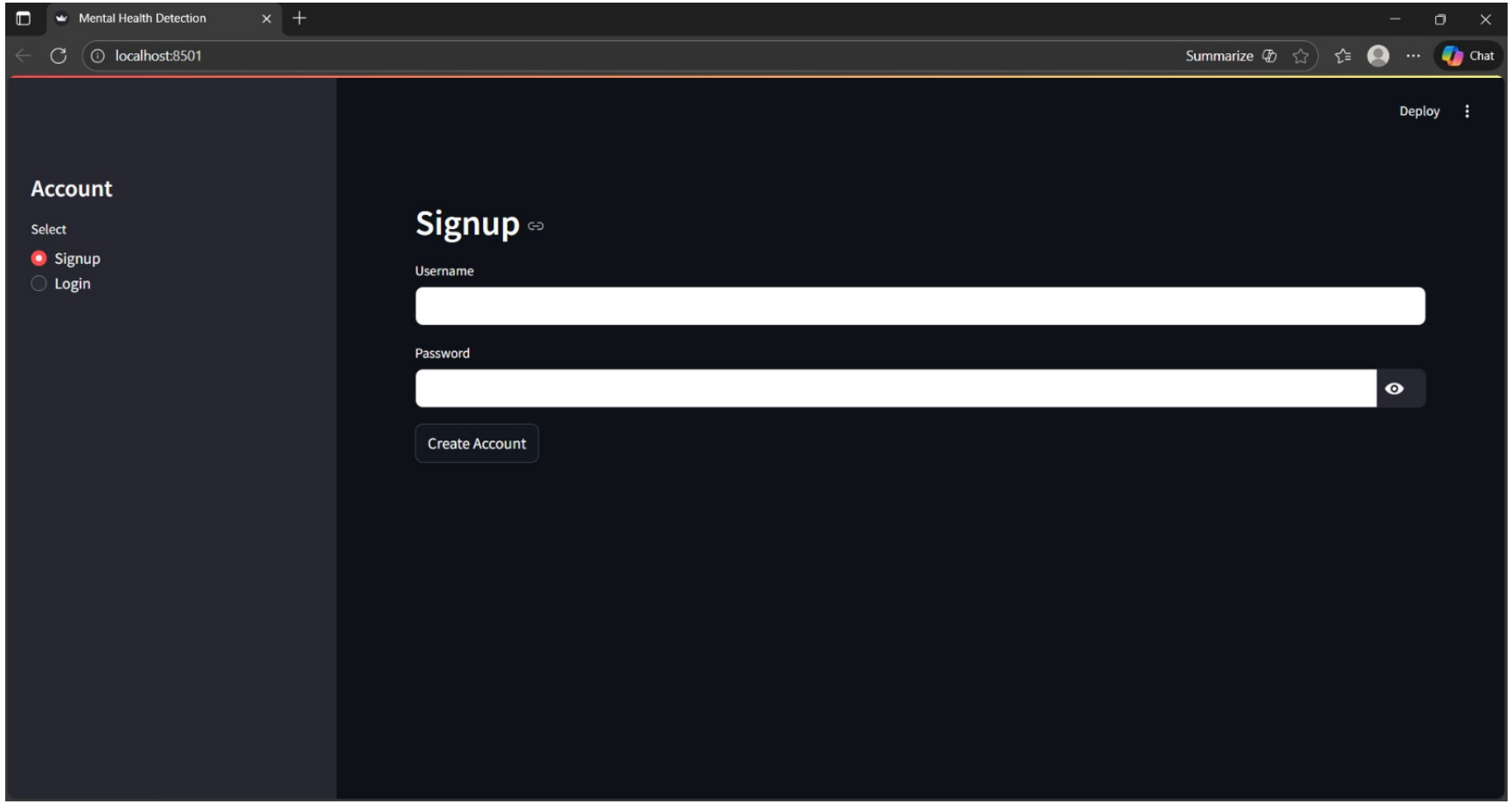
```
import streamlit as st

# Initialize session state
if "logged_in" not in st.session_state:
    st.session_state.logged_in = False

# Login function
def login(username, password):
    if username == "admin" and password == "1234":
        st.session_state.logged_in = True
        st.success("Login Successful!")
    else:
        st.error("Invalid username or password")

# Login UI
if not st.session_state.logged_in:
    st.title("Login Page")
    user = st.text_input("Username")
    pwd = st.text_input("Password", type="password")
    if st.button("Login"):
        login(user, pwd)
else:
    st.success("Welcome Admin!")
    st.write("You have access to the dashboard.")
```

Dashboard of Mental Health Detection Login Page



Dashboard of Mental Health Detection