*PROJECT NAM*E – **“SOLUTIONS ON NEUROLOGICAL DISORDERS CAUSED DUE TO EXCESSIVE USAGE OF ELECTRONIC DEVICES”**

Code:-

```
import flask
import dash
from dash import dcc, html
from dash.dependencies import Input, Output,State
import pandas as pd
app = dash.Dash(__name__)

data = pd.read_csv("C:\\Users\\harip\\.spyder-
py3\\NEW_NEUROLOGICAL_UPDATED_SLEEP.csv")
data['Participant_ID'] = data['Participant_ID'].astype(str)
print(data.head())

try:
    data = pd.read_csv(file_path)
    print("Data loaded successfully!")
except FileNotFoundError as e:
    print(f"File not found: {e}")
except pd.errors.EmptyDataError as e:
    print(f"No data found in the file: {e}")
except pd.errors.ParserError as e:
    print(f"Error while parsing the file: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

# Display column names to verify 'Age_Group' exists
if 'Age_Group' in data.columns:
    print("Column 'Age_Group' found!")
else:
    print("Column 'Age_Group' not found. The available columns are:")
    print(data.columns)

# Define the login layout
login_layout = html.Div([
    html.H1("Login"),
    dcc.Input(id="login-id", type="text", placeholder="Participant ID or Username"),
    html.Button("Login", id="login-button"),
```

```
    html.Div(id="login-output")
])

signup_layout = html.Div([
    html.H1("Signup"),
    dcc.Input(id="signup-id", type="text", placeholder="Participant ID or Username"),
    dcc.Input(id="age-input", type="number", placeholder="Age"),
    dcc.Dropdown(id="gender-input", placeholder="Gender", options=[
        {'label': 'Male', 'value': 'Male'},
        {'label': 'Female', 'value': 'Female'},
        {'label': 'Other', 'value': 'Other'},
    ]),
    dcc.Dropdown(id="neurological-disorder-input", placeholder="Neurological Disorder",
options=[
        {'label': 'Alzheimer\'s', 'value': 'Alzheimer\'s'},
        {'label': 'Parkinson\'s', 'value': 'Parkinson\'s'},
        {'label': 'Multiple Sclerosis', 'value': 'Multiple Sclerosis'},
        {'label': 'Epilepsy', 'value': 'Epilepsy'},
        {'label': 'Migraine', 'value': 'Migraine'},
    ]),
    dcc.Dropdown(id="severity-input", placeholder="Severity", options=[
        {'label': 'Mild', 'value': 'Mild'},
        {'label': 'Moderate', 'value': 'Moderate'},
        {'label': 'Severe', 'value': 'Severe'},
    ]),
    dcc.Dropdown(id="device-input", placeholder="Electronic Device", options=[
        {'label': 'Device A', 'value': 'Device A'},
        {'label': 'Device B', 'value': 'Device B'},
        {'label': 'Device C', 'value': 'Device C'},
    ]),
    dcc.Input(id="hours-of-use-input", type="number", placeholder="Hours of Use"),
    dcc.Input(id="frequency-input", type="number", placeholder="Frequency"),
    dcc.Dropdown(id="impact-input", placeholder="Impact on Symptoms", options=[
        {'label': 'Low', 'value': 'Low'},
        {'label': 'Moderate', 'value': 'Moderate'},
        {'label': 'High', 'value': 'High'},
    ]),
    dcc.Input(id="sleep-timings-input", type="text", placeholder="Sleep Timings"),
    dcc.Dropdown(id="sleep-regularity-input", placeholder="Sleep Regularity", options=[
        {'label': 'Regular', 'value': 'Regular'},
        {'label': 'Irregular', 'value': 'Irregular'},
    ]),
    dcc.Dropdown(id="age-group-input", placeholder="Age Group", options=[
        {'label': 'Teen', 'value': 'Teen'},
```

```
        {'label': '20-30', 'value': '20-30'},
        {'label': '30-50', 'value': '30-50'},
        {'label': '50+', 'value': '50+'},
    ]),
    html.Button("Signup", id="signup-button"),
    html.Div(id="signup-output")
])

# Function to assign age group based on age
def assign_age_group(age):
    if age <= 19:
        return "Teen"
    if age <= 30:
        return "20-30"
    if age <= 50:
        return "30-50"
    return "50+"

if 'Age' in data.columns:
    data['Age_Group'] = data['Age'].apply(assign_age_group)

app.layout = html.Div([
    html.H1("Neurological Recommendations"),
    html.Div([
        dcc.Input(id="participant-id", type="text", placeholder="Participant ID or Username"),
        html.Button("Show Details", id="show-details-button"),
        html.Div(id="details-output")
    ]),
    # Include login and signup layouts
    login_layout,
    signup_layout
])

def assign_recommendation(neurological_disorder, severity):
    if neurological_disorder == "Alzheimer's":
        if severity == "Mild":
            return "Encourage cognitive stimulation and mental exercises. Maintain a regular routine and structure in daily life. Ensure safety measures, including locking away potentially dangerous items."
        elif severity == "Moderate":
            return "Consider structured memory and cognitive rehabilitation programs. Ensure constant supervision to prevent wandering or accidents. Explore options for respite care or adult day programs. Maintain a calm and supportive environment, adjusting to the changing needs."
```

```
    elif severity == "Severe":
        return "In severe cases, seek specialized memory care facilities. Provide comprehensive daily assistance, including bathing, feeding, and mobility. Focus on maintaining comfort and quality of life. Offer emotional support to both the patient and caregivers."
    else:
        return "Maintain a regular sleep schedule."
  if neurological_disorder == "Parkinson's":
    if severity == "Mild":
        return "Maintain a regular sleep schedule to manage symptoms. Continue the recommended treatment plan, including medication adjustments. Engage in regular, adapted physical activities to support mobility."
    elif severity == "Moderate":
        return "It's essential to maintain consistent medical follow-ups to monitor medication efficacy and adjust treatment plans as needed. Continue with physical activities and consider physical therapy to address mobility challenges. Focus on maintaining a balanced lifestyle and relationships to support emotional well-being."
    elif severity == "Severe":
        return "Maintain a regular sleep schedule to manage symptoms. Continue the recommended treatment plan, including medication adjustments. Engage in regular, adapted physical activities to support mobility. Specialized care may be necessary, potentially involving more intensive medical support, including the consideration of advanced treatment options."
    else:
        return "Maintain a regular sleep schedule."
  if neurological_disorder == "Multiple Sclerosis":
    if severity == "Mild":
        return "Consult with a neurologist for personalized treatment plans. Maintain a regular sleep schedule to manage fatigue and energy levels. Incorporate physical therapy, rehabilitation, and tailored exercise. Focus on stress management, mindfulness, and support from caregivers."
    elif severity == "Moderate":
        return "Focus on regular medical follow-ups, potential adjustments to the treatment plan, and continued physical therapy or rehabilitation. Emphasize maintaining a balanced work and personal life while managing the condition."
    elif severity == "Severe":
        return "Consult with a neurologist for personalized treatment plans. Maintain a regular sleep schedule to manage fatigue. Incorporate physical therapy, rehabilitation, and tailored exercise. Require more intensive medical and lifestyle support, including specialized care and potential adjustments to work or personal life to accommodate the condition."
    else:
        return "Maintain a regular sleep schedule."
  if neurological_disorder == "Epilepsy":
    if severity == "Mild":
```

```
            return "Maintain a consistent sleep schedule, take prescribed antiepileptic medications, and prioritize stress management to reduce seizures, ensuring a safe environment with parental support. Stay active for overall health."
        elif severity == "Moderate":
            return "Closely monitor medication effectiveness and consider potential adjustments in consultation with healthcare providers. Continue with stress management and regular medical check-ups. Consider sharing information with roommates or partners to ensure they understand the condition."
        elif severity == "Severe":
            return "Maintain a consistent sleep pattern to minimize seizure triggers. Adhere to prescribed antiepileptic medications and consult with a neurologist. Prioritize stress management and relaxation techniques. Require intensive medical and lifestyle support. Explore advanced treatment options and consider constant supervision to ensure safety."
        else:
            return "Maintain a regular sleep schedule."
    if neurological_disorder == "Migraine":
        if severity == "Mild":
            return "Identify and manage migraine triggers with the help of a neurologist. Maintain a consistent sleep schedule to reduce migraine attacks. Consult with healthcare professionals for medication and pain management. Implement stress management techniques, relaxation exercises, and mindfulness."
        elif severity == "Moderate":
            return "Consider more targeted migraine prevention medications or interventions. Focus on identifying and avoiding specific triggers. Continue with stress management techniques and seek support from healthcare professionals."
        elif severity == "Severe":
            return "Identify and manage migraine triggers with the help of a neurologist. Maintain a consistent sleep schedule to reduce migraine attacks. Consult with healthcare professionals for medication and pain management. Implement stress management techniques, relaxation exercises, and mindfulness. Require advanced treatments and interventions potentially including outpatient procedures or hospitalization. Focus on minimizing migraine attacks and maximizing pain management strategies."
        else:
            return "Maintain a regular sleep schedule."
    return "No recommendations available for this disorder and severity."

@app.callback(
    Output('details-output', 'children'),
    Input('show-details-button', 'n_clicks'),
    State('participant-id', 'value')
)
def show_participant_details(n_clicks, participant_id):
    if n_clicks is not None:
        participant_data = data[data['Participant_ID'] == participant_id]
```

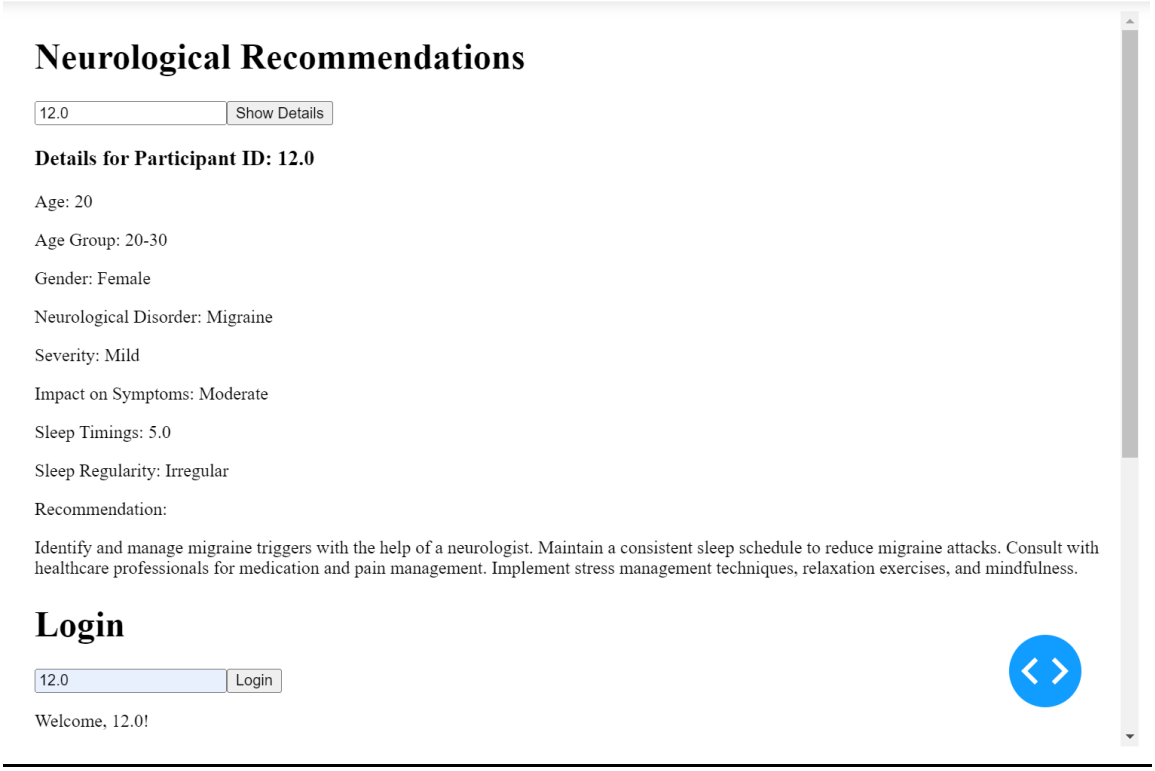
```
        if not participant_data.empty:
            details = participant_data.iloc[0]  # Get the first row if there are multiple matching rows
            age = details['Age']
            age_group = details['Age_Group']
            gender = details['Gender']
            disorder = details['Neurological_Disorder']
            severity = details['Severity']
            impact_on_symptoms = details['Impact_on_Symptoms']
            sleep_timings = details['Sleep.Timings']
            sleep_regularity = details['Sleep_Regularity']

            recommendation = assign_recommendation(disorder, severity)

            return html.Div([
                html.H3(f"Details for Participant ID: {participant_id}"),
                html.P(f"Age: {age}"),
                html.P(f"Age Group: {age_group}"),
                html.P(f"Gender: {gender}"),
                html.P(f"Neurological Disorder: {disorder}"),
                html.P(f"Severity: {severity}"),
                html.P(f"Impact on Symptoms: {impact_on_symptoms}"),
                html.P(f"Sleep Timings: {sleep_timings}"),
                html.P(f"Sleep Regularity: {sleep_regularity}"),
                html.P("Recommendation:"),
                html.P(recommendation)
            ])
        else:
            return html.P("Participant ID not found")

app.layout = html.Div([
    html.H1("Neurological Recommendations"),
    html.Div([
        dcc.Input(id="participant-id", type="text", placeholder="Participant ID or Username"),
        html.Button("Show Details", id="show-details-button"),
        html.Div(id="details-output")
    ]),
    # Include login and signup layouts
    login_layout,
    signup_layout
])

# Callback to handle login
@app.callback(
```

```
    Output('login-output', 'children'),
    Input('login-button', 'n_clicks'),
    State('login-id', 'value')
)
def login_user(n_clicks, participant_id):
    if n_clicks:
        # Remove leading/trailing whitespaces and make it case-insensitive
        participant_id = participant_id.strip().upper()

        # Check if the participant ID exists in the DataFrame
        if participant_id in data['Participant_ID'].str.strip().str.upper().values:
            return html.P(f"Welcome, {participant_id}!")
        else:
            return html.P("Login failed. Participant ID not found.")
    return None

@app.callback(
    Output('signup-output', 'children'),
    Input('signup-button', 'n_clicks'),
    State('signup-id', 'value'),
    State('age-input', 'value'),
    State('gender-input', 'value'),
    State('neurological-disorder-input', 'value'),
    State('severity-input', 'value'),
    State('device-input', 'value'),
    State('hours-of-use-input', 'value'),
    State('frequency-input', 'value'),
    State('impact-input', 'value'),
    State('sleep-timings-input', 'value'),
    State('sleep-regularity-input', 'value'),
    State('age-group-input', 'value')
)
def signup_user(n_clicks, participant_id, age, gender, neurological_disorder, severity, device,
hours_of_use, frequency, impact, sleep_timings, sleep_regularity, age_group):
    global data  # Declare 'data' as a global variable

    if n_clicks is not None:
        # Create a new DataFrame with the user's details
        new_entry = pd.DataFrame({
            'Participant_ID': [participant_id],
            'Age': [age],
            'Gender': [gender],
            'Neurological_Disorder': [neurological_disorder],
            'Severity': [severity],
```

```
            'Electronic_Device': [device],
            'Hours_of_Use': [hours_of_use],
            'Frequency': [frequency],
            'Impact_on_Symptoms': [impact],
            'Sleep.Timings': [sleep_timings],
            'Sleep_Regularity': [sleep_regularity],
            'Age_Group': [age_group]
        })

        # Append the new entry to the existing CSV data
        data = data.append(new_entry, ignore_index=True)
        data.to_csv("C:\\Users\\harip\\.spyder-py3\\NEW_NEUROLOGICAL_UPDATED_SLEEP.csv",
index=False)

        return html.P(f"New participant with ID {participant_id} has been successfully registered.")


app.layout = html.Div([
    html.H1("Neurological Recommendations"),
    html.Div([
        dcc.Input(id="participant-id", type="text", placeholder="Participant ID or Username"),
        html.Button("Show Details", id="show-details-button"),
        html.Div(id="details-output")
    ]),
    # Include login and signup layouts
    login_layout,
    signup_layout
])

if __name__ == '__main__':
    app.run_server(debug=True, port = 8052)
```

Output:-

Login of Existing User



Creating a new Account –

Neurological Recommendations

13.0 Show Details

Details for Participant ID: 13.0

Age: 16

Age Group: Teen

Gender: Male

Neurological Disorder: Parkinson's

Severity: Severe

Impact on Symptoms: Moderate

Sleep Timings: 4

Sleep Regularity: Irregular

Recommendation:

Maintain a regular sleep schedule to manage symptoms. Continue the recommended treatment plan, including medication adjustments. Engage in regular, adapted physical activities to support mobility. Specialized care may be necessary, potentially involving more intensive medical support, including the consideration of advanced treatment options.

Login

13.0 Login

Welcome, 13.0!