**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**імені ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

# Лабораторна робота №3

**з курсу**

# «Технології обробки даних»

*Студента 5 курсу*
*групи ІС-11*
*спеціальності 122 «Комп'ютерні науки»*
*ОП «Інформаційні системи»*
Іванова Віктора Миколайовича

**Київ – 2023**

*Побудова інтерактивної візуалізації для web за допомогою dash*

1. Створити Dash-додаток, який буде відображати графік залежності ціни акцій (любих) від часу. Для цього потрібно використати бібліотеку pandas для завантаження даних та бібліотеку plotly для побудови графіку.
2. Додати до додатку інтерактивність за допомогою компонентів управління, таких як Dropdown, Slider або Date Picker.
3. Створити інші візуалізації, які відображають дані, використовуючи різні типи графіків та діаграм.
4. Зберегти та завантажити створений додаток на локальний сервер або на хмарну платформу, таку як Heroku.
5. Налаштувати автоматичне оновлення даних та візуалізації на основі даних з зовнішнього джерела, наприклад, з API.
6. Розробити власні ідеї та функціональність для покращення додатку та його візуалізації, такі як фільтри, статистика, експорт даних, тощо.
7. Додатково, ви можете розглянути можливість використання бібліотеки Dash Bootstrap Components для стилізації та поліпшення вигляду додатку.

# Код програми

```
import pandas as pd

import plotly.express as px

import dash

import dash_core_components as dcc

import dash_html_components as html

from dash.dependencies import Input, Output



# Завантаження даних про ціну акцій

df = pd.read_csv('stock_prices.csv')



# Ініціалізація Dash додатку

app = dash.Dash(__name__)



# Створення графіка залежності ціни акцій від часу

fig = px.line(df, x='Date', y='Price')



# Додавання компонентів управління
```

```
app.layout = html.Div([
    html.H1('Графік ціни акцій'),
    dcc.Graph(id='stock-chart', figure=fig),
    html.Label('Виберіть період часу:'),
    dcc.RangeSlider(
        id='date-slider',
        min=df['Date'].min(),
        max=df['Date'].max(),
        value=[df['Date'].min(), df['Date'].max()],
        marks={str(date): str(date) for date in df['Date'].unique()},
        step=None
    )
])
```

**# Визначення функції зворотного виклику для оновлення графіка на основі вибраного періоду часу**

```
@app.callback(
    Output('stock-chart', 'figure'),
```

```
    [Input('date-slider', 'value')]

)

def update_chart(selected_dates):

    filtered_df = df[df['Date'].between(selected_dates[0], selected_dates[1])]

    fig = px.line(filtered_df, x='Date', y='Price')

    return fig



# Запуск додатку

if __name__ == '__main__':

    app.run_server(debug=True)
```