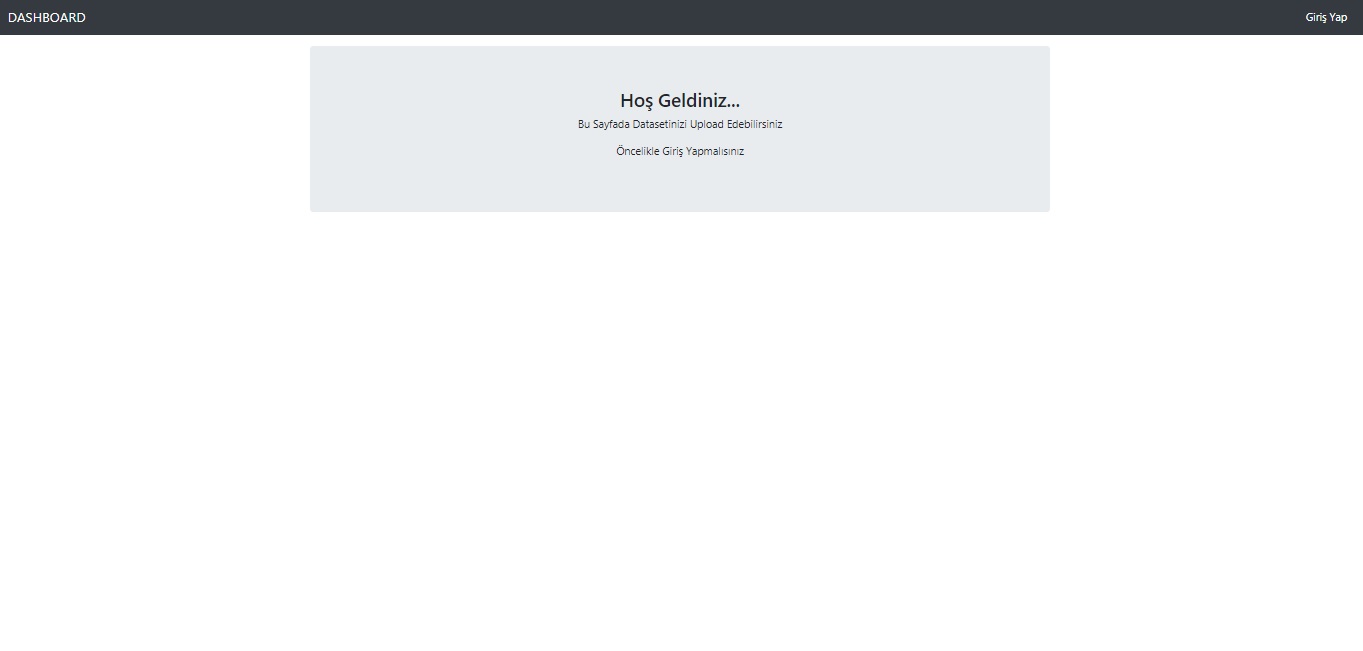
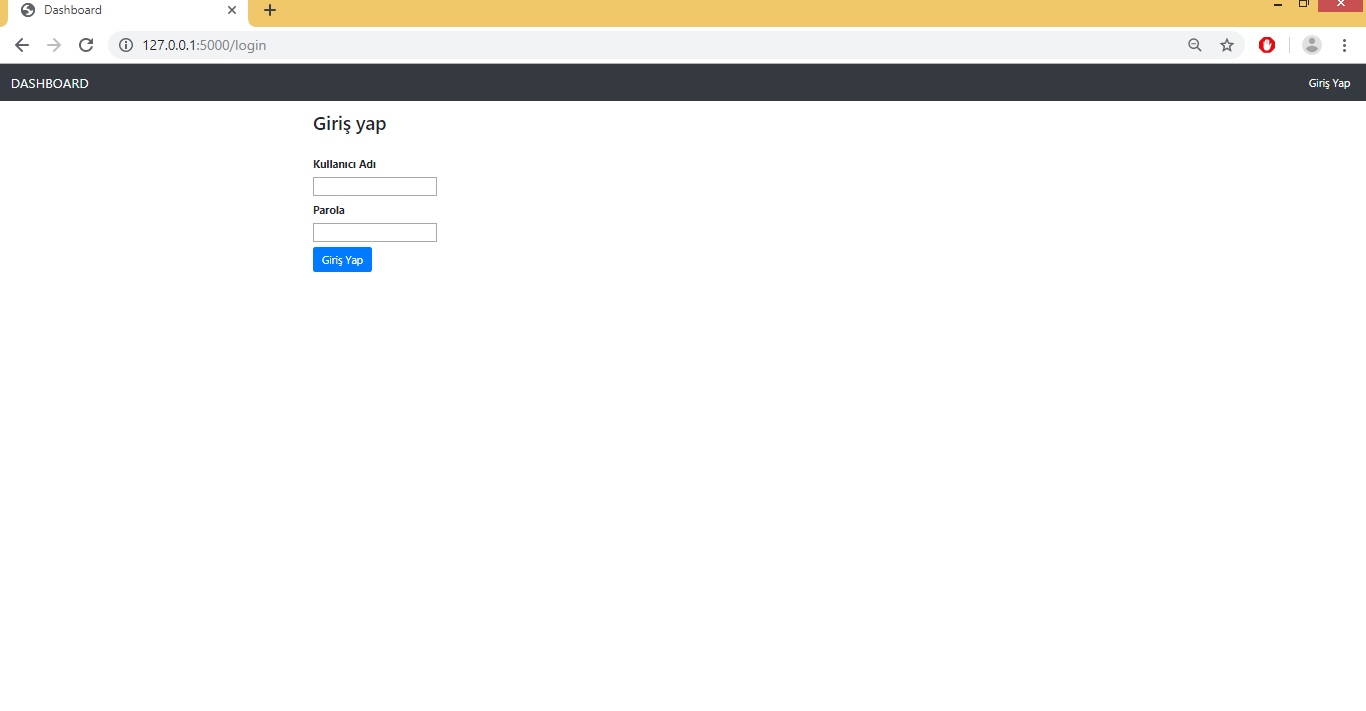
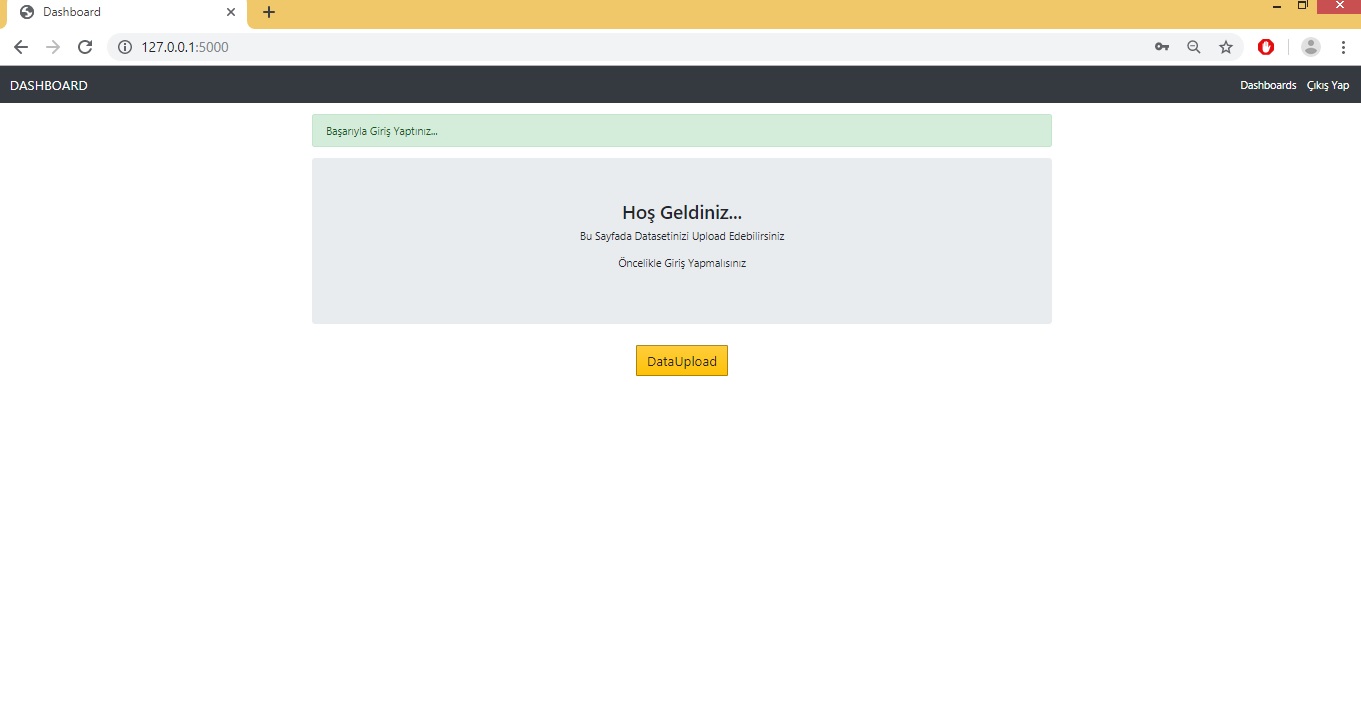
**Flask Plotly Projesi**

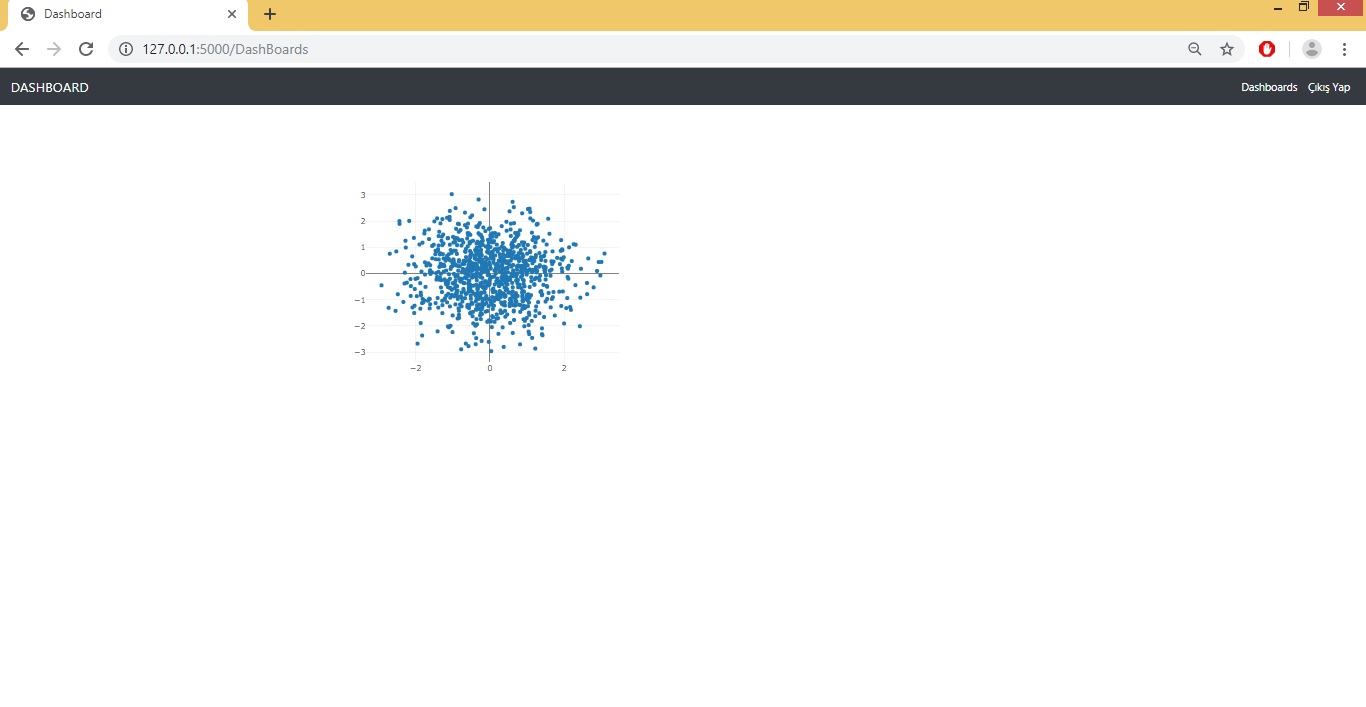
Öncelikle sayfanın genel görüntüsünü paylaşıp sonrasında programımızın arkaplanında kodlama nasıl işliyor detaylı bir şekilde anlatacağım

[İndex.html]

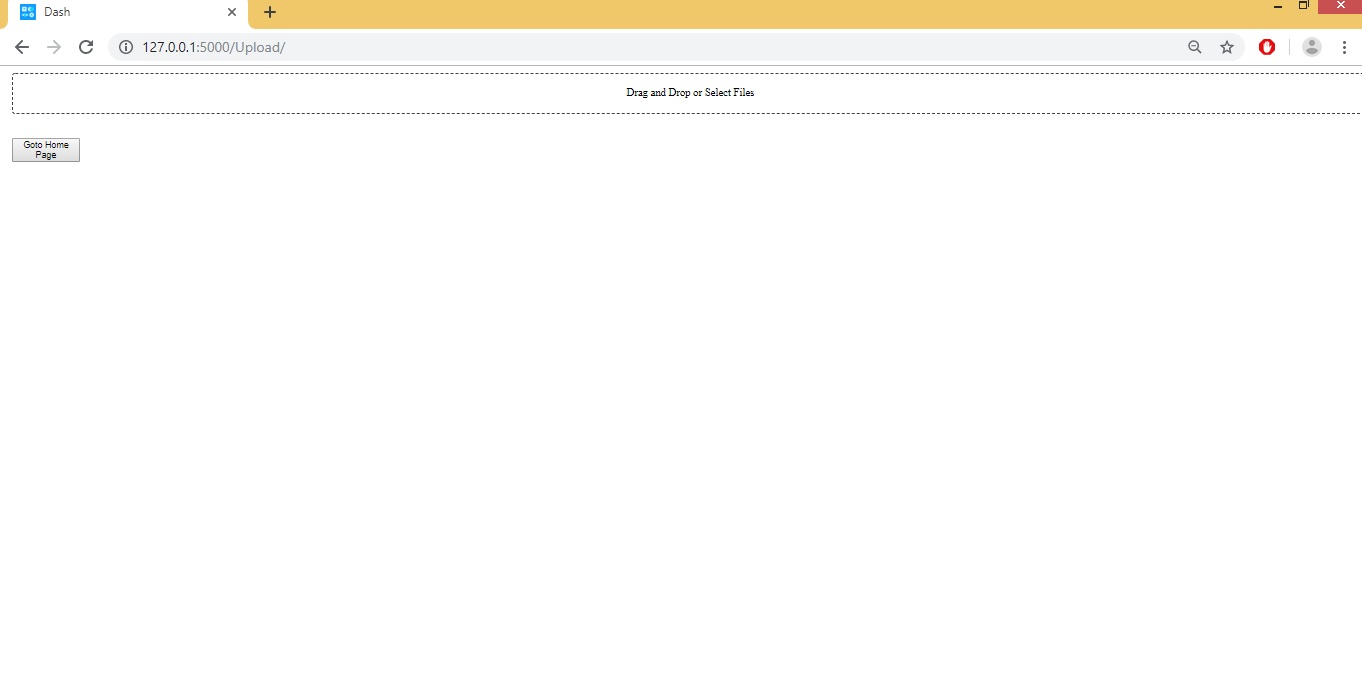
[Login.html]



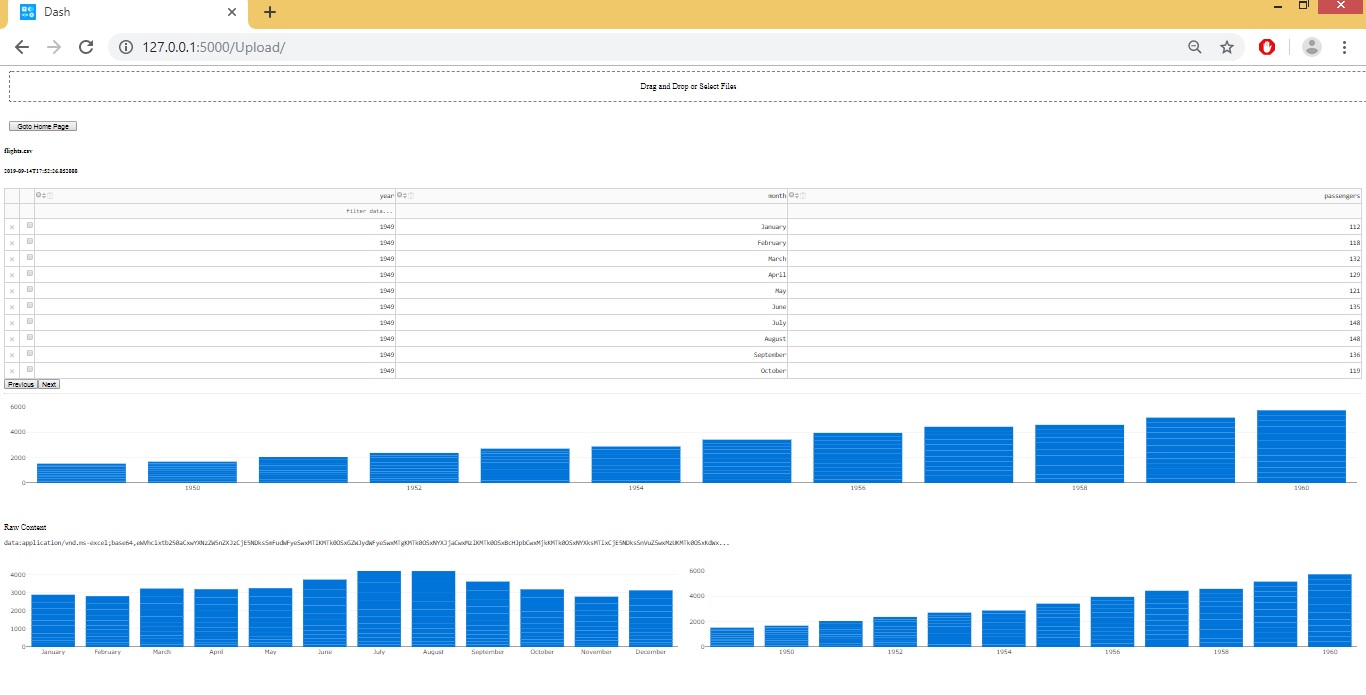
[Login işleminden sonra İndex.html görüntüsü]



[Dashboard.html]



[index.html sayfasındaki Data upload butonuna basıldığında Dash plotly Upload sayfasına yönlendiriyoruz]



[dosyamızı seçtikden sonra dosyamızın dökümanı ve grafikleri]

Flask ve Dash Plotly projemizde öncelikle Flask.py adında python dosyası oluşturdum

ardından Templates Klasörü oluşturup buraya web sayfamızın içerisinde bulunan sayfaları ekledim(DashBoards.html,index.html,layout.html,login html)

Templates Klasörünün içerisine includes diye bir klasör daha oluşturdum

includes klasörünü içerisine (formhelpers.html,messages.html,navbar.html)

dosyalarını oluşturduk sebebi ise navbar için tüm web sayfalarımızda navbar tanımlamaktansa tek bir kez tanımlayıp web sayfalarımıza aktardık

Şimdi Html sayfalarımızın içerisinde hangi kodlar var onlara bakalım.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">        <title>Dashboard</title>      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>      <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.6/d3.min.js"></script>  </head>  <body>      {% include "includes/navbar.html" %}      <br>      <br>      <br>  <div class="container">      {% include "includes/messages.html" %}      {% block body %}        {% endblock %}  </div>      <script src="{{ url\_for('static', filename='js/jquery-1.11.1.min.js') }}"></script>      <script src="{{ url\_for('static', filename='js/plots.js') }}"></script>      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>      <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>      <script src="https://cdn.ckeditor.com/4.12.1/standard/ckeditor.js"></script>      <script src="https://cdn.jsdelivr.net/gh/google/code-prettify@master/loader/run\_prettify.js"></script>      <script>              CKEDITOR.replace( 'content',{allowedContent :true} );      </script>    </body>  </html> |

Layout.html

Sayfalarımız Layout.html üzerinden çalışacaktır.     {% include "includes/navbar.html" %} bu işlem navbar.html sayfasının içerisindeki kodların Layout.html sayfasında çalışmasını sağlıyor.

|  |
| --- |
| {% extends "layout.html" %}    {% block body %}  <div class="jumbotron text-center">      <h3>Hoş Geldiniz...</h3>      <p>Bu Sayfada Datasetinizi Upload Edebilirsiniz</p>      <p>Öncelikle Giriş Yapmalısınız</p>  </div>  {% if session["logged\_in"] %}  <div class="container text-center">          <a type="button" class="btn btn-warning btn-lg" href="/Upload">DataUpload</button></a>  </div>          {% else %}          {% endif %}    {% endblock %} |

Index.html

{% extends "layout.html" %} bu işlem bizim layout.html içerisinde çalışmamızı sağlıyor.

|  |
| --- |
| {% extends "layout.html" %}  {% block body %}  <h3>Giriş yap</h3>  <hr>  {% from "includes/formhelpers.html" import render\_field %}  <form method = "post">          {{ render\_field(form.username) }}          {{ render\_field(form.password) }}          <button type="submit" class="btn btn-primary">Giriş Yap</button>        </form>      {% endblock body %} |

Login.html

Form içerisinde Post olarak kullanıcı giriş işlemini tanımlıyoruz.

|  |
| --- |
| {% extends "layout.html" %}    {% block body %}  {% if session["logged\_in"] %}      <div class="row">          <div class="col-md-6">              <div class="chart" id="bargraph">                  <script>                      var graphs = {{plot | safe}};                      Plotly.plot('bargraph',graphs,{});                  </script>              </div>          </div>      </div>          {% else %}          <div class="row">              <div class="col-md-6">          <p>Dataseti görebilmek için giriş yapmalısınız</p>              </div>          </div>          {% endif %}    {% endblock %} |

DashBoards.html

                <script>

                    var graphs = {{plot | safe}};

                    Plotly.plot('bargraph',graphs,{});

                </script>

Bu kod ile sayfa içerisine dash grfiğimizi gömmemizi sağlıyoruz.

|  |
| --- |
| {% macro render\_field(field) %}    <dt>{{ field.label }}    <dd>{{ field(\*\*kwargs)|safe }}    {% if field.errors %}      <ul class=errors>      {% for error in field.errors %}        <li>{{ error }}</li>      {% endfor %}      </ul>    {% endif %}    </dd>  {% endmacro %} |

İncludes/Formhelper.html

|  |
| --- |
| {% with messages = get\_flashed\_messages(with\_categories=true) %}    {% if messages %}      {% for category, message in messages %}      <div class="alert alert-{{category}}" role="alert">          {{ message }}      </div>      {% endfor %}      {% endif %}  {% endwith %} |

İncludes/messages.html

|  |
| --- |
| <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">      <a class="navbar-brand" href="/">DASHBOARD</a>      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">        <span class="navbar-toggler-icon"></span>      </button>      <div class="collapse navbar-collapse" id="navbarCollapse">        <ul class="navbar-nav ml-auto">          {% if session["logged\_in"] %}          <li class="nav-item active">            <a class="nav-link" href="/DashBoards">Dashboards<span class="sr-only">(current)</span></a>          </li>          <li class="nav-item active">              <a class="nav-link" href="/logout">Çıkış Yap<span class="sr-only">(current)</span></a>            </li>          {% else %}          <li class="nav-item active">              <a class="nav-link" href="/login">Giriş Yap<span class="sr-only">(current)</span></a>            </li>          {% endif %}          </ul>        </div>    </nav> |

İncludes/navbar.html

Flask.py işleyişi:

Kütüphanelerimizi tanımlıyoruz

server = Flask(\_\_name\_\_) ile Flaskımızın serverini server değişkenine atıyoruz

#Kullanıcı Login Decorator

def login\_required(f):

    @wraps(f)

    def decorated\_function(\*args, \*\*kwargs):

       if "logged\_in" in session:

           return f(\*args, \*\*kwargs)

       else:

            flash("Bu sayfayı görüntülemek için lütfen giriş yapın.","danger")

            return redirect(url\_for("login"))

    return decorated\_function

Giriş işlemi olmadan Dashboards sayfamıza girmesini engelliyoruz

class loginform(Form):

    username = StringField("Kullanıcı Adı")

    password = PasswordField("Parola")

Login işlemindeki kullanıcı adı ve parolar kısmını oluşturuyoruz bir class’da tanımlıyoruz.

#İndex Html process

@server.route('/')

def index():

    return render\_template('index.html')

Flask.py dosyazın çalışmaya başladığında bizi render\_template('index.html')

Komutu ile index.html sayfamıza yönlendiriyor

#login process

@server.route("/login",methods = ["GET","POST"])

def login():

    form=loginform(request.form)

    if request.method == "POST":

        username = form.username.data

        password\_entered = form.password.data

        if username == "kadir":

            if password\_entered == "1234":

                flash("Başarıyla Giriş Yaptınız...","success")

                session["logged\_in"]=True

                session["username"]=username

                return redirect(url\_for("index"))

            else:

                flash("Parolanızı Yanlış Girdiniz...","danger")

                return redirect(url\_for("login"))

        else:

            flash("Böyle bir kullanıcı bulunmuyor...","danger")

            return redirect(url\_for("login"))

    return render\_template("login.html",form = form)

Burada Form üzerinden gelen verilerimizi benim manuel olarak tanımladığım username ve password ile eşleştiriyoruz.

Eşleştirme başarılı ise Login işlemimiz gerçekleşiyor.

# Logout Process

@server.route("/logout")

def logout():

    session.clear()

    return redirect(url\_for("index"))

Logout işleminde Sessionumuzu temizleyerek login işlemi yaptıkdan sonra gözüken kısımları engellemiş oluyoruz.

#DashBoards Process

@server.route('/DashBoards')

def Dashboards():

    feature = 'Bar'

    bar = create\_plot(feature)

    return render\_template('Dashboards.html', plot=bar)

def create\_plot(feature):

    if feature == 'Scatter':

        N = 40

        x = np.linspace(0, 1, N)

        y = np.random.randn(N)

        df = pd.DataFrame({'x': x, 'y': y}) # creating a sample dataframe

        data = [

            go.Bar(

                x=df['x'], # assign x as the dataframe column 'x'

                y=df['y']

            )

        ]

    else:

        N = 1000

        random\_x = np.random.randn(N)

        random\_y = np.random.randn(N)

        # Create a trace

        data = [go.Scatter(

            x = random\_x,

            y = random\_y,

            mode = 'markers'

        )]

    graphJSON = json.dumps(data, cls=plotly.utils.PlotlyJSONEncoder)

    return graphJSON

Dashboards kısmında bir html sayfamızın içerisine gömülü bir şekilde grafiğimizi yerleştirdiğimizi görüyorsunuz.

"""DASH PAGE PART"""

Yukarıdaki işlemler Flaskın içerisinde olan işlemler biz şimdi dash içerisinde işlemleri yapacağız

Öncelikle Flask ile Dashplotly’i birbirine bağlamamız gerekiyor.

#Flask server connection to dash server

app = dash.Dash(

    \_\_name\_\_,

    server=server,

    routes\_pathname\_prefix='/Upload/',

)

Bu işlem ile Dash serverini Flask server ına bağladık.

app.css.append\_css({

    'external\_url': 'https://codepen.io/chriddyp/pen/bWLwgP.css'})

Dash için gerekli Css Kütühanesi

#For multi callback

app.config.suppress\_callback\_exceptions = True

Dash içerisinde bir çok CALLBACK fonksiyonu kulanacağımız için gerekli bir koddur

df='' df’yi Dash Başlangıcında boş veriyorum sonrasında dataset upload ettiğimizde Upload ettiğimiz dosya ile dolmuş olacak

app.layout = html.Div([

    dcc.Location(id='url', refresh=True),

    html.Div(id='page-content'),

    html.Div(id='datatable-interactivity-container')

])

App.layout sayfası DashPlotly saffamızın ana kısmı hazırlamış olacağımı diğer sayfaları burası üzerinden açıyoruz

#Dash İndex page

index\_page = html.Div([

    dcc.Upload(

        id='upload-data',

    children=html.Div([

            'Drag and Drop or ',

         html.A('Select Files')

        ]),

        style={

            'width': '100%',

            'height': '60px',

            'lineHeight': '60px',

            'borderWidth': '1px',

            'borderStyle': 'dashed',

            'borderRadius': '5px',

            'textAlign': 'center',

            'margin': '10px'

        },

        # Allow multiple files to be uploaded

        multiple=True

    ),

    html.Br(),

   html.A([html.Button(["Goto Home Page"],style={"width" : "5%","height":"20%",'margin': '10px'})], href='/'),

    html.Div(id='page-1-content')

])

İndex \_page sayfası Flaskdan Dash’e Geçiş yaparken /Upload linki ile geçiş yapıyoruz /upload linki geldiğinde index\_page sayfasını app.layout içerisindeki html.Div(id='page-content'), e aktarıyoruz.

#Return Documents and Graph Function

def parse\_contents(contents, filename, date):

    content\_type, content\_string = contents.split(',')

    decoded = base64.b64decode(content\_string)

    try:

        if 'csv' in filename:

            # Assume that the user uploaded a CSV file

            df = pd.read\_csv(

                io.StringIO(decoded.decode('utf-8')))

        elif 'xls' in filename:

            # Assume that the user uploaded an excel file

            df = pd.read\_excel(io.BytesIO(decoded))

    except Exception as e:

        print(e)

        return html.Div([

            'There was an error processing this file.'

        ])

    sayfa= html.Div([

        html.H5(filename),

        html.H6(datetime.datetime.fromtimestamp(date)),

        dash\_table.DataTable(

            id='datatable-interactivity',

            columns=[

            {"name": i, "id": i, "deletable": True, "selectable": True} for i in df.columns

        ],

        data=df.to\_dict('records'),

        editable=True,

        filter\_action="native",

        sort\_action="native",

        sort\_mode="multi",

        column\_selectable="single",

        row\_selectable="multi",

        row\_deletable=True,

        selected\_columns=[],

        selected\_rows=[],

        page\_action="native",

        page\_current= 0,

        page\_size= 10,

        ),

        html.Hr(),  # horizontal line

        html.Div(id='datatable-interactivity-container'),

        # For debugging, display the raw contents provided by the web browser

        html.Div('Raw Content'),

        html.Pre(contents[0:200] + '...', style={

            'whiteSpace': 'pre-wrap',

            'wordBreak': 'break-all'

        }),

        html.Br(),

        html.Div([

        html.Div([

            dcc.Graph(

                figure={

                    "data": [

                        {

                            "x": df[df.columns[1]],

                            "y": df[df.columns[2]],

                            "type": "bar",

                            "marker": {"color": "#0074D9"},

                        }

                    ],

                    "layout": {

                        "xaxis":{"automargin": True},

                        "yaxis": {"automargin": True},

                        "height": 250,

                        "margin": {"t": 10, "l": 10, "r": 10},

                    },

                },

            )

        ],className="six columns",style={"width" : "50%",'display': 'inline-block'}),

        html.Div([

            dcc.Graph(

                figure={

                    "data": [

                        {

                            "x": df[df.columns[0]],

                            "y": df[df.columns[2]],

                            "type": "bar",

                            "marker": {"color": "#0074D9"},

                        }

                    ],

                    "layout": {

                        "xaxis":{"automargin": True},

                        "yaxis": {"automargin": True},

                        "height": 250,

                        "margin": {"t": 10, "l": 10, "r": 10},

                    },

                },

            )

        ],className="six columns",style={ "width" : "50%",'display': 'inline-block'})

        ],className="row"),

    ])

    return sayfa

Burada datasetimizi Upload Ettiğimizde Callback fonksiyonundan Bu fonksiyonu çağırıyoruz

Ve datasetimizi yüklemiş ve grafiklerimizi döndürmüş oluyoruz oluyoruz.

@app.callback(Output('page-1-content', 'children'),

            [Input('upload-data', 'contents')],

              [State('upload-data', 'filename'),

               State('upload-data', 'last\_modified')])

def update\_output1(list\_of\_contents, list\_of\_names, list\_of\_dates):

        if list\_of\_contents is not None:

            children = [

            parse\_contents(c, n, d) for c, n, d in

            zip(list\_of\_contents, list\_of\_names, list\_of\_dates)

            ]

        return children

Datasetimizi yüklediğimizde @app.callback fonksiyonumuz çalışıyor burada Input kısmı gelen veriği tutuyor State kısmıda bastığımız butonun içersinde yapılan iş sonucu filenme ve last\_modified döndürüyor

Output kısmı ise callback fonksiyonu çalışıp bittikden sonra çıkan sonucun yükleneceği kısmı belirliyoruz ‘page-1-content’e yükle neresine ‘children’ yai içerisine yükle nlamına geliyor.

#Selected Colum Callback

@app.callback(

    Output('datatable-interactivity', 'style\_data\_conditional'),

    [Input('datatable-interactivity', 'selected\_columns')]

)

def update\_styles(selected\_columns):

    return [{

        'if': { 'column\_id': i },

        'background\_color': '#D2F3FF'

    } for i in selected\_columns]

Bu kısımda tablomuzdan seçilen sütunu grafikde göstermemizi sağlıyor.

Ancak Bu projede çalışması için düzenlenmesi gerek

#selected Rows

@app.callback(

    Output('datatable-interactivity-container', "children"),

    [Input('datatable-interactivity', "derived\_virtual\_data"),

     Input('datatable-interactivity', "derived\_virtual\_selected\_rows")])

def update\_graphs(rows, derived\_virtual\_selected\_rows):

    if derived\_virtual\_selected\_rows is None:

        derived\_virtual\_selected\_rows = []

    dff = df if rows is None else pd.DataFrame(rows)

    colors = ['#7FDBFF' if i in derived\_virtual\_selected\_rows else '#0074D9'

              for i in range(len(dff))]

    return [

        dcc.Graph(

            figure={

                "data": [

                    {

                       "x": dff[dff.columns[2]],

                        "y": dff[dff.columns[1]],

                        "type": "bar",

                        "marker": {"color": colors},

                    }

                ],

                "layout": {

                    "xaxis": {"automargin": True},

                    "yaxis": {

                        "automargin": True,

                    },

                    "height": 250,

                    "margin": {"t": 10, "l": 10, "r": 10},

                },

            },

        )

    ]

Burada tablomuzdan seçmiş olduğumuz Satırlardaki verileri grafikde belirtiyor.

# Update the Url

@app.callback(dash.dependencies.Output('page-content', 'children'),

              [dash.dependencies.Input('url', 'pathname')])

def display\_page(pathname):

    if pathname == '/':

        return server[render\_template("index.html")]

    else:

        return index\_page

    # You could also return a 404 "URL not found" page here

Burada Dashımıza gelen linki kontrol edip yönlendirme yapıyoruz.

    if pathname == '/':

        return server[render\_template("index.html")]

Bu kısım Dash içerisinde GoHome butonuna basıldığında tekrardan Flask index sayfasına gideceğimiz için render\_templates’i Flask serverinda yani server içerisinde yapıyoruz.

if \_\_name\_\_ == '\_\_main\_\_':

    server.run(debug=True)

Burada Serverımızı ayağa kaldırıyoruz.