BIOCOMPUTING II

GROUP PROJECT

CODE DOCUMENTATION-FRONT END LAYER

Name- Shaikh Mohammad Faizan

Email:faizanshaikh1897@gmail.com

May 2020

Birkbeck, University of London,

United Kingdom

HTML Tags and their use

TAGS AND ATTRIBUTES	DESCRIPTION
<!DOCTYPE html>	All HTML documents start with this declaration. It tells the browser about what document type to expect.
<html lang="en">	The “lang” attribute sets the primary language for th document. In this case, “en” means English.
<head>	This element contains metadata and it is placed between <html> tag and <body> tag.
<meta charset="UTF-8">	It specifies character encoding for the document. "UTF-8"- character encoding for Unicode.
<meta name="viewport" content="width=device-width, initial-scale=1">	Viewport is the user’s visible area of the web page. The dimensions and scaling of the page are controlled through instructions provided by this.
<meta http-equiv="X-UA-Compatible" content="ie=edge">	It tells the internet to display content in the highest available mode.
<style>	This tag defines style (CSS) information for the document.
<title>	Specifies the title of a web page.
</script>	Used to embed client-side script (JavaScript). This script usually contains scripting statements.
<body>	Defines the documents body and contains all contents of HTML document.
<div>	Block level HTML element used to divide or section other HTML tags.
<id>	Used to label sections as part of HTML documents.
<span>	An inline HTML element used to group a set of inline elements
<class>	Making a class allows us to style each chunk with one rule, considering a page has multiple sidebar chunks.
<ul class="tabs">	Unordered list of tabs.
<li>	Used to represent item in a list

<a href='glossary.html'>Glossary</a>	Tells the browser where to go and take data using href attribute. In this case”glossary.html” file.
<h1>-<h6>	These elements represent six levels of section headings.
<form method = "get" action = "/cgi-bin/cgiwrap/sm004/searchdb.py" >	Creates form for user input with get method with a path to a script. In this case, a cgi script “searchdb.py” which executes a function.
<input name="data" type="text" placeholder="E.g: AB002059" size="52" required />	Element used to create interactive controls for forms. In this case it specifies the text type, the placeholder attribute provides a brief hint to hat type of data it expects and the size describes the size of the checkbox.
<select name= 'pulldown'>	Gives a dropdown list. It has a name attribute and it specifies the type of the menu.
<optgroup label = Identifiers>	Specifies label for an option group. In this case “identifiers”.
<option id = 'gi'> Gene ID </option>	Defines an option in a select list. In this case, “Gene ID”

CGI Script

Import CGI	Imports module for CGI handling
cgi.fieldStorage()	This instance can be usually indexed like a python dictionary.
Form= cgi.FieldStorage()	Fields accessed through “form[key]” are themselves instances of FieldStorage.
Value=form.getvalue()	This method returns string value. If the requested key is absent, it also accepts an optional argument.

For Instance, consider the following code. This CGI script allows the browser to take up a user input and search for a gene in the chromosome by “Gene Identifiers” and returns that gene’s data in a table. All necessary python functions are stored in a module called “business” which is imported as bl.

```
#!/usr/bin/env python3

import cgi

import business as bl

form = cgi.FieldStorage() #set instance of field storage

value = form.getvalue("x")

print ("Content-Type: text/html\n")

#value = 'AB002059'

table = bl.search_db(value)

if table == False:

  html = "<html>"

  html += "<html lang= 'en'>\n"

  html += "<head>\n"

  html += "<meta charset = 'utf-8'/>"

  html += "<title> Biocomputing II - Human Genome Browser </title>\n"

  html += "</head>\n"

  html += "<body>\n"
  html += "<h1>Please enter a valid search ID.</h1>\n"
  html += "</body>\n"
  html += "</html>\n"
  print (html)
else:
  html = "<html>"
  html += "<html lang= 'en'>\n"
  html += "<head>\n"
  html += "<meta charset = 'utf-8'/>"
  html += "<title> Biocomputing II - Human Genome Browser </title>\n"
  html += "</head>\n"
  html += "<body>\n"
  html += "<h1> Result</h1>\n"
  #Print filtered results as table
  html += "<div class="'col-md-s'"."
  html += "<table class="'table table-bordered'">"
  html += "<thead>"
  html +=     "<tr>"
  html +=         "<th> Accession           </th>"
  html +=         "<th> Gene                </th>"
  html +=         "<th> Protein Product     </th>"
  html +=         "<th> Chromosomal Location </th>"
  html +=     "</tr>"
  html += "</thead>"
  html += "<tbody>"
  html += "<ul>\n"
  gene_id = table['gene_id']
  location = table['location']
```

```
protein_id = table['protein_id']
accession = table['accession']
html += "<tr>\n"
html += "<td>\n"+ str(accession) + "</td>\n"
html += "<td>\n"+ str(gene_id) + "</td>\n"
html += "<td>\n"+ str(protein_id) + "</td>\n"
html += "<td>\n"+ str(location) + "</td>\n"
html += "</tr>\n"
html += "</tbody>"
html += "</table>"
html += "</ul>\n"
html += "</body>\n"
html += "</html>\n"
print(table)
```

All other CGI scripts are written in a similar format and function similarly.

JAVASCRIPT

The following Javascript is used for the browser. The comments in blue provide necessary description.

```
const tabs = document.querySelectorAll('[data-tab-target]')

//selects all the different tabs in html

const tabContents = document.querySelectorAll('[data-tab-content]')

//Tab Contents contain all the different tab contents in the html

//loop through each tab, add an event listener for each tab when clicked

tabs.forEach(tab => {

  tab.addEventListener("click", () => {

    const target = document.querySelector(tab.dataset.tabTarget)

    //grab the relevant tab target in html based on what tab is clicked on the page.

    //loop through each tab content, remove 'active' class- make them disappear

    tabContents.forEach(tabContent => {

      tabContent.classList.remove("active") //make only the tab clicked active

    })

    tabs.forEach(tab => {

      tab.classList.remove("active")

    })//create class for css styling

    tab.classList.add("active")

    target.classList.add("active")

  })

})
```