POST GRADUATE GOVERNMENT COLLEGE SECTOR – 11 CHANDIGARH

PRACTICAL FILE

FUNDAMENTALS OF WEB PROGRAMMING

BCA – 16 -203

SUBMITTED TO: -

MRS. RENU

SUBMITTED BY: -

Akhil Semwal

4033/23

SECTION – A

BCA - I st

Contents

REMARKS

INTRODUCTION

Staying organized can be tough. Introducing a web-based to-do list application built with the fundamental web development languages – HTML, CSS, and JavaScript. This project goes beyond a simple checklist, offering a user-friendly platform to manage and prioritize your tasks. Effortlessly add new items and track your progress as you tick things off. No more scrambling for sticky notes or lost lists – this application keeps your tasks organized and readily accessible from any device with an internet connection. This empowers you to stay focused on achieving your goals and maximize your efficiency, ensuring you never miss a beat in your busy schedule.

This sites heavily depends on the concept of local storage and session storage. Understanding these topics will help in understanding working of the site:

LOCAL STORAGE : Local storage is a feature in JavaScript that allows web applications to store data locally on the user's browser. This data persists even after the browser window or tab is closed. Things to consider :

- Local storage data has no expiration date by default. It will persist until the user clears their browsing data or the website uses JavaScript to remove it.
- Local storage is specific to the origin (domain) of the website. Data stored on "example.com" cannot be accessed by "example2.com".
- Local storage data is accessible in both private browsing/incognito mode and normal browsing mode. However, data stored in private mode gets cleared when the user closes all private windows/tabs.

SESSION STORAGE : Session storage allows you to store data on the client-side during a user's browsing session. This data is accessible throughout the session, even if the user reloads the page. However, unlike local storage, session storage gets cleared as soon as the user closes the browser tab or window. Things to consider :

- Data persistence: Exists only within a browsing session and gets cleared when the user closes the tab or window.
- Scope: Specific to the current browser tab or window. Data stored in one tab cannot be accessed by other tabs.
-

- Accessibility: Accessible in both normal and private browsing modes. However, data stored in private mode is cleared when the user closes all private windows/tabs.

This website consists of a **LOGIN PAGE** where you can enter your name and password. Your name and password are then stored in local storage of the browser in form of key value pair of usernames and an id's which is combination of your username and password. In case your username is found in existing local storage then your password is verified from local storage. In case the login is successful you will be redirected to the MAIN PAGE and your username will be saved in session storage for current storage.

In all the further pages a navbar is displayed on the top of the site with a log out option, site name and link to another page in the site. The log out option clears the session storage which contains details of current login, after this user will be redirected to index page from where he can login again to the same or different account.

The **MAIN PAGE** is the core of this site where you can add, remove, and modify your tasks. On loading of this page task list is fetched from local storage corresponding to the username in session storage. This task list is then iterated through to create elements through java script. In any case an add button is displayed which can be used adding tasks. In case tasks are displayed, each task shows the task time, task details, modify button, checkbox for time completion, and a remove button to remove completed tasks. Each task is linked with java script and CSS to help user easily identify the task status. Clicking on modify or add task takes you to the TASK ADDING PAGE.

The **TASK ADDING PAGE** contains a table inside a form with two columns one to enter task time and other to enter the task details. Task is checked so that it is not empty. The local storage is using comma (,) and forward slash (/) for splitting tasks and times so to avoid error in functioning user is prompted to avoid using these characters in case they are being used in task. In case this page is opened to modify a task, the task is fetched from local storage and the input fields are given corresponding placeholders. In case task is successfully added user is redirected to MAIN PAGE.

Other than the above pages which are crucial for the site functioning there is another page – the **PROFILE PAGE**. This page displays the username and number of tasks pending. Other than this,

there is a button to delete account. On selecting this option, the user's details would be removed from the local storage and session storage.

TECHINQUES

This project uses fundamentals of web development technologies – HTML, CSS and JavaScript. Basic knowledge of these technologies will clarify the working of the site.

HTML

HTML, which stands for Hyper Text Markup Language, is the fundamental building block of web pages. It defines the structure and content of web content like text, images, videos, and more. It works alongside other technologies like CSS (Cascading Style Sheets) for styling and JavaScript for interactivity to create dynamic and engaging web experiences.

Here's a breakdown of key concepts in HTML:

Structure and Tags:

HTML uses tags to define the different elements on a web page. Tags are written like <element> and </element> and placed around the content they represent. For example, <h1> and </h1> tags define a heading element.

There are numerous tags available in HTML, each serving a specific purpose. Some common examples include:

<h1> to **<h6>** for headings of different sizes.

<p> for paragraphs.

<a> for creating links to other web pages or sections of the same page.

<img> for embedding images.

<ul> and **<ol>** for unordered and ordered lists, respectively.

Attributes:

Tags can have attributes that provide additional information about the element. For instance, an **<img>** tag can have a **src** attribute that specifies the image source URL.

No Direct Page Styling:

HTML itself doesn't provide a way to style the appearance of web pages. That's the realm of CSS. However, HTML elements can have CSS classes assigned to them, which CSS can then use to style those elements.

Comments:

HTML comments are used to add notes within the code that are ignored by web browsers. These are helpful for documenting your code and explaining its purpose. Comments are written like `comment` or '<!-- comment-->'.

CSS

Cascading Style Sheets (CSS) is a cornerstone technology of the web, alongside HTML and JavaScript. It's a language specifically designed to style the presentation of a web page. Here's a closer look at what CSS does:

Separation of Concerns:

CSS separates the presentation (how a web page looks) from the content (what the web page is about). This separation improves code maintainability and allows for easier control over the visual appearance of your web pages.

Styling with Selectors and Rules:

CSS uses selectors to target specific elements in an HTML document and then applies rules that define how those elements should be styled. Selectors can target elements by their ID, class, tag name, attributes, or even a combination of these.

Here's a basic example:

CSS

```
h1 {  /* targets all h1 elements */

  color: red;

  font-size: 2em;

}

.special {  /* targets elements with the class "special" */

  background-color: blue;

  padding: 10px;

}
```

Use code with caution.

In this example, the first rule targets all <h1> elements and sets their color to red and font size to 2 times the default size. The second rule targets any element with the class "special" and applies a blue background color with padding.

CSS Properties and Values:

CSS offers a wide range of properties that can be used to style various aspects of an element, like:

- Font properties (color, size, family)
- Background properties (color, image)
- Borders and margins

- Padding
- Layout properties (display, float, positioning)
- Visual effects (opacity, shadows, transitions)

Each property has a corresponding set of values that can be applied.

Specificity:

When multiple CSS rules apply to the same element, the rule with the highest specificity wins. Specificity is determined by a combination of factors like element type, class, and ID selectors.

JAVASCRIPT

JavaScript (JS) is a versatile programming language that reigns supreme in the world of web development. It's a high-level, interpreted language, meaning code is executed line by line without prior compilation. Let's explore what JavaScript brings to the table:

Making Web Pages Dynamic:

Unlike static HTML pages, JavaScript injects interactivity into web experiences. It allows you to:

- Update content and styles based on user actions (e.g., clicking buttons, form submissions).
- Create animations and other visual effects.
- Validate user input in forms.
- Communicate with web servers to retrieve and display data.

Essentially, JavaScript bridges the gap between static web pages and dynamic applications.

Core JavaScript Concepts:

Variables and Data Types: Like other programming languages, JavaScript uses variables to store data. It has various data types like **numbers**, **strings**, **booleans**, and **objects** to represent different kinds of information.

Functions: Reusable blocks of code that perform specific tasks. Functions can take parameters and return values.

Control Flow: Control flow statements **(if/else, loops)** allow you to control the execution flow of your code based on conditions and loops.

Objects: JavaScript is object-oriented, meaning it uses objects to encapsulate data (properties) and related functionality (methods).

DOM Manipulation: The Document Object Model (DOM) represents the structure of an HTML document. JavaScript can manipulate the DOM to change the content and structure of a web page dynamically.

CONCEPTS USED

ALL PAGES

- ➔ External and inline CSS.

- External java script.
- Local Storage and Session Storage.
- Onload function.
- Unordered list (navbar).
- Background animation.
- Alerts.

LOGIN PAGE

- Form.
- User input.
- Form submission.
- String manipulation.
- Event listener.
- User input type manipulation.

MAIN PAGE

- Java script loops.
- List and list manipulation.
- Nesting of functions.
- Using java script to create elements and modify attributes.
- Parent-child elements.
- Retrieving elements by class and id.
- Intervals.
- Class toggling using java script.
- Use and manipulation of date objects.
- Getting local time.

TASK ADDING PAGE

- Form.
- Table.
- Getting local time.

PROFILE PAGE

- Image.
- Table.
- Clearing session and local storage.

CSS

From index.css

```
/* index.html */
/* FORM CONTAINER */
.container{
  background-color: #90ee9052;
  width:50%;height: max-content;
  margin: 12% 20%;
  padding: 5%;
  box-shadow: -4px 4px 10px #00000085;
}
/* ID SUBBUTTON 1 */
#subButton1{
  border-radius: 20px;
  box-shadow: -1px 1px 5px #00000085;
  background-color: #ffffffbb;
  color: rgb(40, 39, 39);
  cursor: pointer;
}
/* SUBBUTTON HOVER */
#subButton1:hover{
  border: 3px solid rgb(40, 39, 39);
  background-color: #aaccffa6;
  color: rgb(40, 39, 39);
}

/* main.html */
/* NAVBAR */
.navList{
  height: max-content;
  background-color: rgba(125, 80, 80, 0.75);
  border-radius: 30px;
  list-style: none;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 0%;
  box-shadow: -2px 2px 10px #00000085;
}
/* NAVBAR ELEMENT */
.navList li{
  width: 33%;
}
/* SITE NAME */
.siteName{
  font-size: 300%;
  font-weight: bolder;
  font-family:cursive;
  text-shadow: 2px 2px 5px #ce4d44;
```

```
}
/* LOG OUT AND HOME TEXT */
.logOut,.home{
  font-size: 150%;
  color: #ffc2c7e6;
  text-decoration: none;
}
/* LOG OUT AND HOME HOVER */
.logOut:hover,.home:hover{
  color: #ffc2c7ac;
  text-decoration: underline;
  cursor: pointer;
}
/* PROFILE PIC */
.profilePic{
  float: right;
  width: 40px;
  border-radius: 50px;
  border: 2px solid white;
  background-color: antiquewhite;
  box-shadow: -2px 2px 10px #00000085;

}
/* PROFILE PIC HOVER */
.profilePic:hover{
  border-color: black;
  cursor: pointer;
  box-shadow: -2px 2px 10px #ffffff85;
}
.profilePic img{
  width: 40px;
}
/* CONTAINER TO ADD TASK */
.containerAdd{
  position: relative;
  width:80%;
  margin: 4% 10% 4% 10% ;
  background-color: #ffc2c799;
  font-size: 100px;
  color:#B6E5D8;
  border: 0px;
  border-radius: 40px;
  box-shadow: -1px 1px 10px #00000085;
}
/* CONTAINER HOVER */
.containerAdd:hover{
  background-color: #b6e5d88c;
  color:#ffc2c7e6;
  cursor: pointer;
}
```

```
/* TASK CLASS */
.task{
  position:relative;
  width:80%;
  height:max-content;
  margin: 3% 8%;
  background-color: #ffc2c7e6;
  font-size: 60%;
  color:#fd7F20;
  border: 0px;
  border-radius: 40px;
  padding: 1%;
  box-shadow: -1px 1px 10px #00000085;
}
/* TASK ON HOVER */
.task:hover{
  background-color:#ffc2c7ac;
  border: 0px;
  cursor: pointer;
}
/* TASKTEXT CLASS */
.taskText{
  margin-left: 10px;
}
/* TASK CHECKBOX */
.taskCheck{
  float: right;
  margin: 0.3% 1% 0% 1%;
}
/* TASKTEXT CHECKED */
.taskTextChecked{
  margin-left: 10px;
  text-decoration: line-through;
  color:#FC2E20;
}
/* TASKBUTTON */
.taskButton{
  float:right;
  background-color: #ffc2c7e6;
  font-size: 10px;
  color:#B6E5D8;
  border: 2px solid;
  margin-right: 10px;
  border-radius: 10px;
  text-align: center;
}
/* TASKBUTTON HOVER */
.taskButton:hover{
  background-color:#FC2E20;
  cursor: pointer;
}
```

```
/* TASK NEAR DEADLINE */
.deadline{
  animation: breathe 3s ease-in-out infinite alternate;
}
/* TASK DEAD */
.dead{
  background-color:#fc2f2089;
  color:white;
}
/* TASK RESET BUTTON */
.taskReset{
  width: 20px;
  height: 20px;
  float: right;
  margin-right: 5px;
}
/* BODY BACKGROUND */
body{
  background: repeating-linear-gradient(45deg,#ffc2c77b,#b6e5d846,#fbe5c858,#8fdde74e);
  background-color: #FFFFFF;
  color: #AACCFF;
  animation-name: diagonal_move;
  animation-duration: 50s;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
}
/* BACKGROUND ANIMATION */
@keyframes diagonal_move {
  /* BACKGROUND POSITION AT VARIOUS PERCENTAGE OF ANIMATION */
  0% {
    background-position: 0rem 0rem;
  }
  25% {
    background-position: 0rem 50rem;
  }
  50% {
    background-position: 50rem 50rem;
  }
  75% {
    background-position: 50rem 0rem;
  }
  100% {
    background-position: 0rem 0rem;
   }
 }
/* DEADLINE ANIMATION */
@keyframes breathe {
  0% {
   background-color:#ffc2c7e6;
  }
  50% {
```

```
    background-color:#fc2f2089;
    color:white;
  }
  100% {
    background-color:#ffc2c7e6;
  }
}

/* list.html */
/* ADD LIST BUTTON CLASS */
.addListBut{
  background-color: #ffc2c7e6;
  color:#fd7F20;
  border: 0px;
  width: 30%;
  height: 40px;
  margin: auto 35%;
  border-radius: 50px;
  font-size: 120%;
  box-shadow: -1px 1px 10px #00000085;
}
/* ADD LIST HOVER */
.addListBut:hover{
  background-color:#fd7F20;
  color:#ffc2c7e6;
  border: 0px;
  border-radius: 40px;
  cursor: pointer;
  box-shadow: -1px 1px 10px #00000085;
}

/* profile.html */
/* REMBUTTON CLASS */
.remButton{
  width: 50%;
  background-color: #AACCFF;
  color:#FC2E20;
  border: 0px;
  border-radius: 20px;
}
/* REMBUTTON HOVER */
.remButton:hover{
  background-color:#FC2E20;
  color:#AACCFF;
  cursor: pointer;
}
```

JAVASCRIPT

```
// GETTING USER DETAILS
function getUser() {
  user = document.getElementById("userName1");
  pass = document.getElementById("passWord1");
  // IF USERNAME AND PASSWORD NOT EMPTY
  if (user.value !='' & pass.value != ''){
    // IF NEW USER
    if (localStorage.getItem(user.value) == null){
      // STORING USERNAME AND PASSWORD TO LOCAL STORAGE
      localStorage.setItem(user.value,pass.value)
      // CREATING ID FROM USERNAME AND PASSWORD
      id = user.value + pass.value
      // APPENDING ID TO LOCAL STORAGE
      localStorage[user.value] += "/"+id
      // SETTING LOCATION TO MAIN PAGE
      window.location.href = "./main.html"
      // STORING USERNAME TO SESSION STORAGE
      sessionStorage.setItem(0,user.value)
    }
    // IF PASSWORD MISMATCH
    else if(pass.value != ((localStorage.getItem(user.value)).split("/"))[0]){
      alert("INCORRECT PASSWORD")
    }
    // IF OLD USER
    else{
      // GETTING ID
      id = ((localStorage.getItem(user.value)).split("/"))[1]
      // SETTING LOCATION TO MAIN PAGE
      window.location.href = "./main.html"
      // STORING USERNAME TO SESSION STORAGE
      sessionStorage.setItem(0,user.value)
    }
  }
  // IS USERNAME/PASSWORD EMPTY
  else{
    alert("USERNAME/PASSWORD CANNOT BE EMPTY")
  }
}
// ADDING TASKS
function addList() {
  taskTime = document.getElementById("taskTime1")
  taskArea = document.getElementById("taskArea1")
  // IF TASK ENTERED
  if (taskArea.value != ""){
    if (taskArea.value.indexOf(",") != -1 | taskArea.value.indexOf("/") != -1){
      alert("( , ) AND ( / ) CANNOT BE USED IN TASKS")
      return
    }
    task = (taskTime.value+","+taskArea.value)
```

```
        // STORING TASK TIME AND DESCRIPTION TO LOCAL STORAGE
        localStorage[sessionStorage[0]] += "/"+task
        // RESETING TIME AND TASK
        taskTime.value = getTime()
        taskArea.value = ""
    // IF TASK NOT ENTERED
    } else{
        alert("PLEASE ENTER A TASK")
    }
}
// DISPLAYING LISTS
function displayList(){
    // GETTING TASKS FROM '/' SEPARATED STRING IN LOCAL STORAGE
    taskList = localStorage[sessionStorage[0]].split("/")
    // 0 -> PASSWORD 1-> ID
    taskList = taskList.slice(2)
    // ITERATING THROUGH EACH ELEMENT OF TASKLIST
    taskList.forEach(element => {
        createTask(element);
    });
}
// FUNCTIONS TO BE LOADED ON DOCUMENT LOAD
window.onload = function() {
    // CHECKING LOCATION OF WINDOW
    if (window.location.pathname === '/FWPproject/main.html') {
        displayList();
        anchors = document.getElementsByClassName("logOut")
        for (let i = 0; i < anchors.length; i++) {
            // IF LOGOUT IS CLICKED CLEAR SESSION STORAGE REMOVING CURRENT STORAGE
            anchors[i].addEventListener('click', function() {
                sessionStorage.clear();
            });
        }
        // CHECKING TASK DEADLINE
        const intervalId = setInterval(checkTasks(), 1000);
    }
    if (window.location.pathname === '/FWPproject/profile.html') {
        // DISPLAYING PENDING TASKS
        user = document.getElementById("userName")
        user.textContent = sessionStorage[0]
        taskList = localStorage[sessionStorage[0]].split("/")
        taskList = taskList.slice(2)
        pTask =  document.getElementById("pTasks")
        if (taskList.length > 1) {
            pTask.textContent = taskList.length + " TASKS PENDING"
        }
        else{
            pTask.textContent = taskList.length + " TASK PENDING"
        }
        anchors = document.getElementsByClassName("logOut")
        for (let i = 0; i < anchors.length; i++) {
```

```
      // IF LOGOUT IS CLICKED CLEAR SESSION STORAGE REMOVING CURRENT STORAGE
      anchors[i].addEventListener('click', function() {
        sessionStorage.clear();
      });
    }
  }
  if (window.location.pathname === '/FWPproject/index.html') {
    passVis = document.getElementById("passVisibility")
    pass = document.getElementById("passWord1")
    // IF PASSWORD VISIBILTY IS TOGGLED CHANGE PASSWORD VISIBILITY
    passVis.addEventListener('change',function () {
      if (pass.type === "password") {
        pass.type = "text"
      } else{
        pass.type = "password"
      }
    })
  }
  if (window.location.pathname === '/FWPproject/list.html') {
    // SETTING DEFAULT INPUT TIME TO CURRENT SYSTEM TIME
    defaultTime = document.getElementById("taskTime1")
    defaultTime.value = getTime();
    if (sessionStorage.getItem("myText") && typeof sessionStorage.getItem("myText") ===
'string'){
      const parts = sessionStorage.getItem("myText").split("/");
 document.getElementById('taskArea1').value = parts[1];
      (document.getElementById("addListBut")).addEventListener('click', function() {
      localStorage[sessionStorage[0]] = localStorage[sessionStorage[0]].replace(new
RegExp("/"+(parts[0]), 'g'), '')
  sessionStorage.removeItem("myText")
      })
    }
  }
};
// DELETING ACCOUNT
function removeAcc(){
  let result = window.confirm('ARE YOU SURE YOU WANT TO DELETE ACCOUNT?');
  // CONFIRMATION
  if (result) {
    // REMOVE FROM LOCAL STORAGE
    localStorage.removeItem(sessionStorage[0])
    // CLEAR SESSION STORAGE
    sessionStorage.clear()
    alert('ACCOUNT DELETED!');
    // RETURNING TO LOGIN PAGE
    window.location.href = "./index.html"
  } else {
    alert('DELETION CANCELLED.');
  }
}
// FUNCTION TO GET CURRENT TIME
```

```
function getTime(){
  let currentTime = new Date().toLocaleTimeString('en-US', {hour12: false})
  return currentTime
}
// FUNCTION TO CALCULATE THE DIFFERENCE BETWEEN TWO TIMES IN SECONDS
function getTimeDifference(time1, time2) {
  // GET TODAY'S DATE
  const today = new Date();
  // CONVERT TIME STRINGS TO DATE OBJECTS
  const date1 = new Date(today.getFullYear(), today.getMonth(), today.getDate(),
time1.split(':')[0], time1.split(':')[1], time1.split(':')[2]);
  const date2 = new Date(today.getFullYear(), today.getMonth(), today.getDate(),
time2.split(':')[0], time2.split(':')[1], time2.split(':')[2]);

  // CALCULATE THE DIFFERENCE IN MILLISECONDS
  const differenceInMs = date2.getTime() - date1.getTime();

  // CONVERT MILLISECONDS TO MINUTES AND RETURN
  return differenceInMs / (1000*60);
 }
// FUNCTION TO CHECK IF TASK DEADLINE IS MET
function checkTasks() {
  taskList = localStorage[sessionStorage[0]].split("/")
    taskList = taskList.slice(2)
    taskList.forEach(element => {
      taskTime = document.getElementById(element+"textTime")
      taskTime.textContent = element.split(",")[0]
      containerDiv = document.getElementById(element)
      if (getTimeDifference(getTime(),taskTime.textContent) < 10) {
        containerDiv.classList.add("deadline")
      }
      if (getTimeDifference(getTime(),taskTime.textContent) < 0) {
        containerDiv.classList.add("dead")
        containerDiv.classList.remove("deadline")
      }

    })
}
// FUNCTION TO CREATE TASK LIST ON MAIN PAGE
function createTask(element){
  // CONTAINER DIV
  let containerDiv = document.createElement('div')
  containerDiv.classList.add("task")
  containerDiv.id = element
  // TASK DIV WITH H2 TAG
  let taskDiv = document.createElement('h2')
  // TASK TIME WITH SPAN TAG
  let taskTime = document.createElement('span')
  taskTime.classList.add("taskText")
  taskTime.id = element+"textTime"
  taskTime.textContent = element.split(",")[0]
```

```
  // TASK WITH SPAN TAG
  let task = document.createElement('span')
  task.classList.add("taskText")
  task.id = element+"text"
  // GETTING TIME FROM TASKLIST ELEMENT
  task.textContent = element.split(",")[1]
  // TASKSET WITH IMAGE TAG
  let taskSet = document.createElement('img')
  taskSet.src = "./assets/setting.png"
  taskSet.classList.add("taskReset")
  taskSet.addEventListener('click', function(){
    sessionStorage["myText"] = element+"/"+task.textContent
    window.location.href = '/FWPproject/list.html'
  })
  // TASKSTAT WITH INPUT TAG
  let taskStat = document.createElement('input')
  taskStat.classList.add("taskCheck")
  taskStat.id = element+"check"
  taskStat.type = "checkbox"
  // EVENTLISTENER TO CHECK IF THE TASKSTAT HAS BEEN
CHANGED(CHECKED/UNCHECKED)
  taskStat.addEventListener('change', function() {
    // IF TASKSTAT HAS CHANGED THAN CORRESPONDINGLY CHANGE TASKTEXT AND
TASKTIME
    document.getElementById(element+"text").classList.toggle("taskTextChecked")
    document.getElementById(element+"textTime").classList.toggle("taskTextChecked")
  });
  // REMOVETASK WITH BUTTON TAG
  let removeTask = document.createElement('button')
  removeTask.classList.add("taskButton")
  removeTask.id = element+"rem"
  removeTask.textContent = "X"
  // EVENTLISTENER TO CHECK IF THE REMOVETASK HAS BEEN CLICKED
  removeTask.addEventListener('click', function() {
    // REMOVE ELEMENT FROM HTML
    document.getElementById(element).remove();
    // REMOVE TASK FROM LOCAL STORAGE
    localStorage[sessionStorage[0]] = localStorage[sessionStorage[0]].replace(new
RegExp("/"+element, 'g'), '');
  });
  // APPENDING CHILD CLASSSES TO TASKDIV
  taskDiv.appendChild(taskTime)
  taskDiv.appendChild(task)
  taskDiv.appendChild(removeTask)
  taskDiv.appendChild(taskStat)
  taskDiv.appendChild(taskSet)
  // APPENDING TASKDIV TO CONTAINERDIV
  containerDiv.appendChild(taskDiv)
  // APPENDING CONTAINERDIV TO DOCUMENT BODY
  document.body.appendChild(containerDiv)
}
```

```
// GO TO LIST HTML
function goList(){
   window.location.href = "./list.html"
}
```

LOGIN PAGE

HTML

From index.html

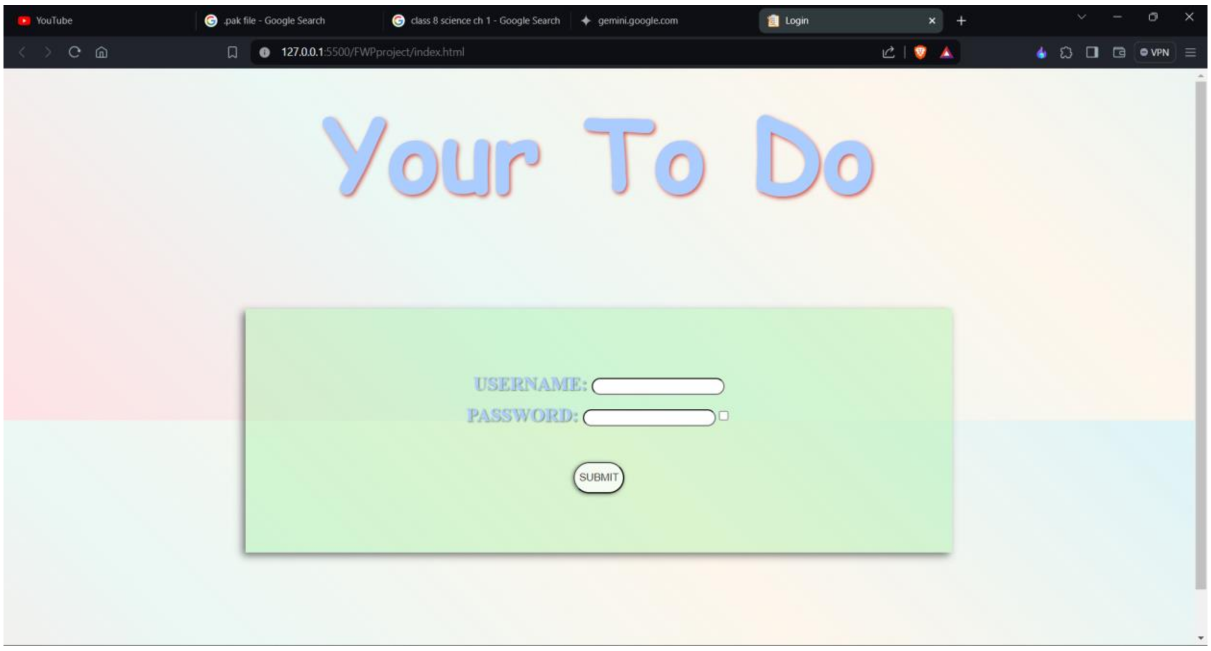
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="./index.css" type="text/css" rel="stylesheet">
  <script src="index.js"></script>
  <link rel="icon" type="image/png" href="./assets/siteLogo.png">
  <title>Login</title>
</head>
<body style="text-align: center;line-height: 250%;">
  <h1 style="font-family:cursive;font-size: 800%;text-shadow: 2px 2px 5px #ce4d44;">Your To
Do</h1>
  <div class="container">
    <form style="font-size:x-large;font-weight: bold; text-shadow: 1px 1px 2px #00000093">
      USERNAME: <input style="border-radius: 20px;" id="userName1" type="text"
placeholder=""><br>
      PASSWORD: <input style="border-radius: 20px;" id="passWord1" type="password"><input
style="margin-top: 2px;" id="passVisibility" type="checkbox">
    </form>
    <button id="subButton1" onclick="getUser()" style="height: 40px;margin-top:
40px;">SUBMIT</button>
  </div>
</body>
</html>
```

CSS

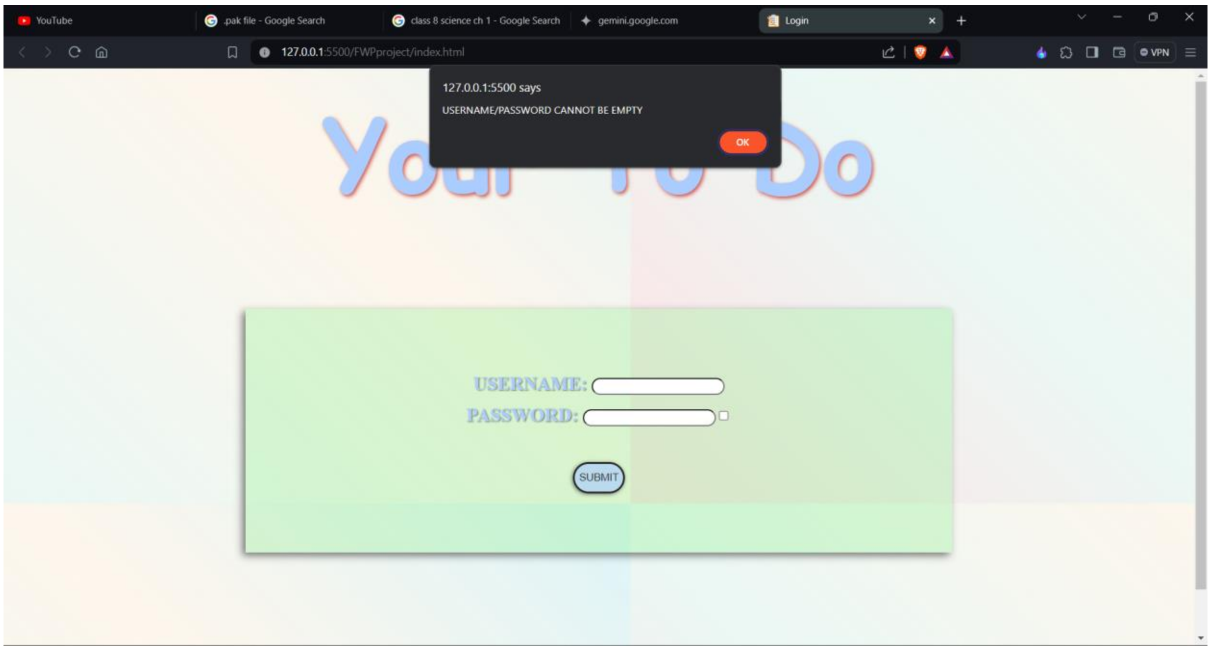
From index.css

```
/* index.html */
/* FORM CONTAINER */
.container{
  background-color: #90ee9052;
  width:50%;height: max-content;
  margin: 12% 20%;
  padding: 5%;
  box-shadow: -4px 4px 10px #00000085;
}
```

DEFAULT


YouTube
.pak file - Google Search
class 8 science ch 1 - Google Search
gemini.google.com
Login
127.0.0.1:5500/FWPproject/index.html
VPN
Your To Do
USERNAME:
PASSWORD:
SUBMIT

EMPTY PASSWORD SUBMIT


YouTube
.pak file - Google Search
class 8 science ch 1 - Google Search
gemini.google.com
Login
127.0.0.1:5500/FWPproject/index.html
VPN
USERNAME:
PASSWORD:
SUBMIT
OK

127.0.0.1:5500 says

USERNAME/PASSWORD CANNOT BE EMPTY

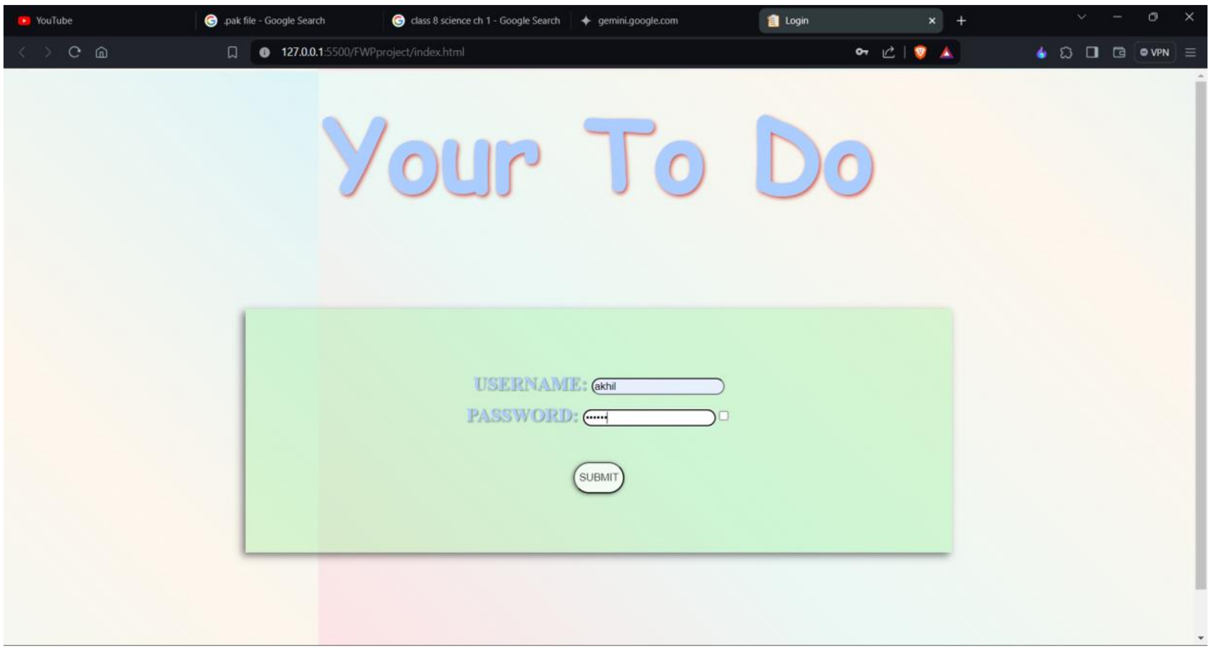
```
/* ID SUBBUTTON 1 */
#subButton1{
  border-radius: 20px;
  box-shadow: -1px 1px 5px #00000085;
  background-color: #ffffffbb;
  color: rgb(40, 39, 39);
  cursor: pointer;
}
/* SUBBUTTON HOVER */
#subButton1:hover{
  border: 3px solid rgb(40, 39, 39);
  background-color: #aaccffa6;
  color: rgb(40, 39, 39);
}
```

JAVASCRIPT

From index.js

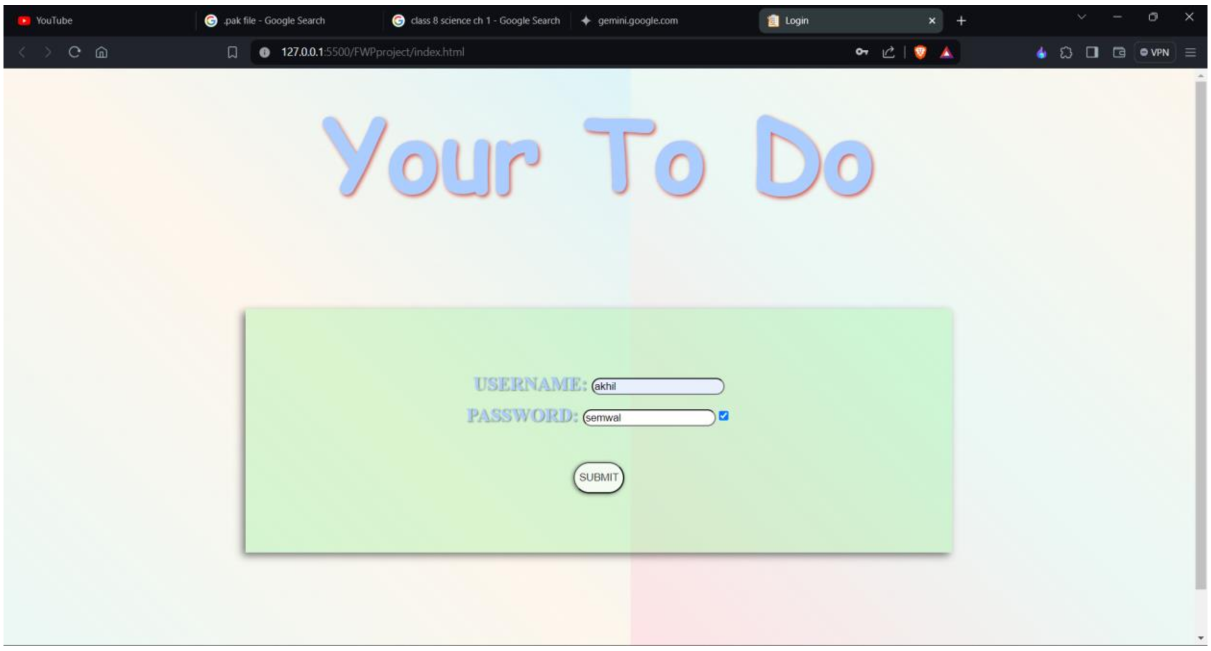
```
// GETTING USER DETAILS
function getUser() {
  user = document.getElementById("userName1");
  pass = document.getElementById("passWord1");
  // IF USERNAME AND PASSWORD NOT EMPTY
  if (user.value !='' & pass.value != ''){
    // IF NEW USER
    if (localStorage.getItem(user.value) == null){
      // STORING USERNAME AND PASSWORD TO LOCAL STORAGE
      localStorage.setItem(user.value,pass.value)
      // CREATING ID FROM USERNAME AND PASSWORD
      id = user.value + pass.value
      // APPENDING ID TO LOCAL STORAGE
      localStorage[user.value] += "/"+id
      // SETTING LOCATION TO MAIN PAGE
      window.location.href = "./main.html"
      // STORING USERNAME TO SESSION STORAGE
      sessionStorage.setItem(0,user.value)
    }
    // IF PASSWORD MISMATCH
    else if(pass.value != ((localStorage.getItem(user.value)).split("/"))[0]){
      alert("INCORRECT PASSWORD")
    }
    // IF OLD USER
    else{
      // GETTING ID
      id = ((localStorage.getItem(user.value)).split("/"))[1]
      // SETTING LOCATION TO MAIN PAGE
      window.location.href = "./main.html"
      // STORING USERNAME TO SESSION STORAGE
      sessionStorage.setItem(0,user.value)
```

PASSWORD TOGGLING


YouTube
.pak file - Google Search
class 8 science ch 1 - Google Search
gemini.google.com
Login
127.0.0.1:5500/FWPproject/index.html
VPN

Your To Do

USERNAME: akhil
PASSWORD: ••••••
SUBMIT


YouTube
.pak file - Google Search
class 8 science ch 1 - Google Search
gemini.google.com
Login
127.0.0.1:5500/FWPproject/index.html
VPN

Your To Do

USERNAME: akhil
PASSWORD: semwal
SUBMIT

```
    }
  }
  // IS USERNAME/PASSWORD EMPTY
  else{
    alert("USERNAME/PASSWORD CANNOT BE EMPTY")
  }
}
// FUNCTIONS TO BE LOADED ON DOCUMENT LOAD
window.onload = function() {
  // CHECKING LOCATION OF WINDOW
  if (window.location.pathname === '/FWPproject/main.html') {
        ...
  }
  if (window.location.pathname === '/FWPproject/profile.html') {
        ...
  }
  if (window.location.pathname === '/FWPproject/index.html') {
    passVis = document.getElementById("passVisibility")
    pass = document.getElementById("passWord1")
    // IF PASSWORD VISIBILTY IS TOGGLED CHANGE PASSWORD VISIBILITY
    passVis.addEventListener('change',function () {
      if (pass.type === "password") {
        pass.type = "text"
      } else{
        pass.type = "password"
      }
    })
  }
  if (window.location.pathname === '/FWPproject/list.html') {
        ...
  }
};
```

MAIN PAGE

HTML

From main.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="./index.css" type="text/css" rel="stylesheet">
  <script src="index.js"></script>
  <link rel="icon" type="image/png" href="./assets/siteLogo.png">
  <title>Your To Do</title>
</head>
<body>
  <nav>
    <ul class="navList">
      <li><a class="logOut" href="./index.html">LOG OUT</a></li>
      <li style="text-align: center;"><span class="siteName">Your To Do</span></li>
      <li><a href="./profile.html" class="profilePic">
        <img src="./assets/profile.png" alt="PROFILE PIC">
      </a></li>
    </ul>
  </nav>
  <div>
    <button onclick="goList()" class="containerAdd">
      +
    </button>
  </div>
</body>
</html>
```

CSS

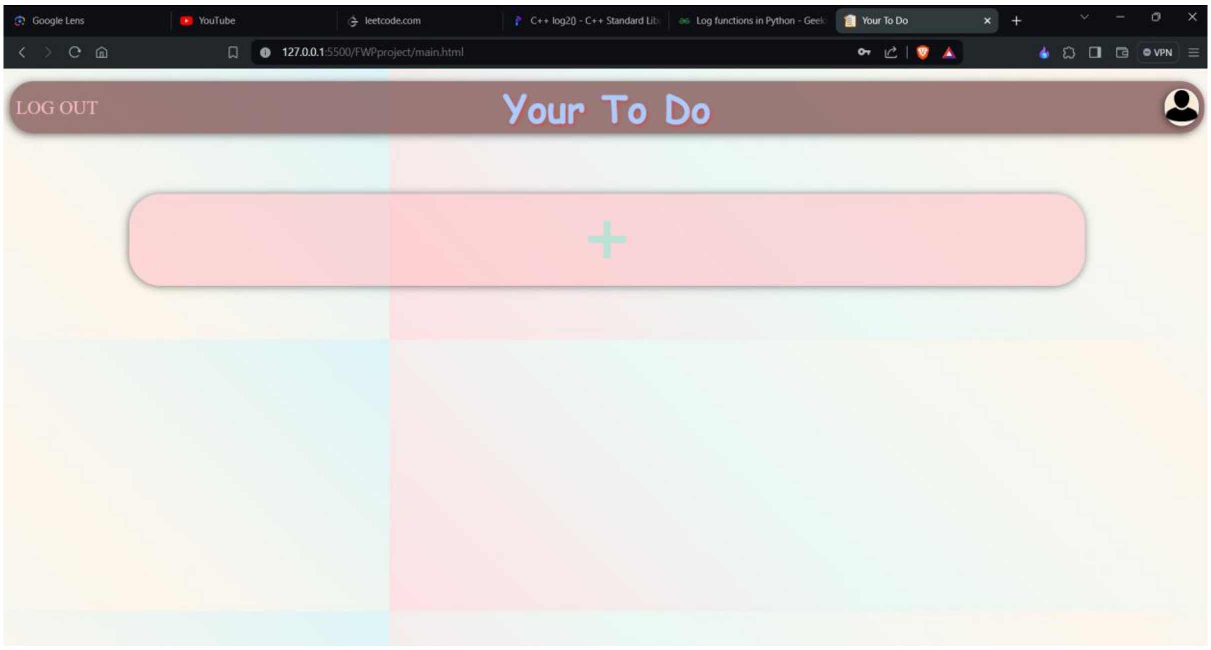
From index.css

```css
/* main.html */
/* NAVBAR */
.navList{
  height: max-content;
  background-color: rgba(125, 80, 80, 0.75);
  border-radius: 30px;
  list-style: none;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 0%;
  box-shadow: -2px 2px 10px #00000085;
```

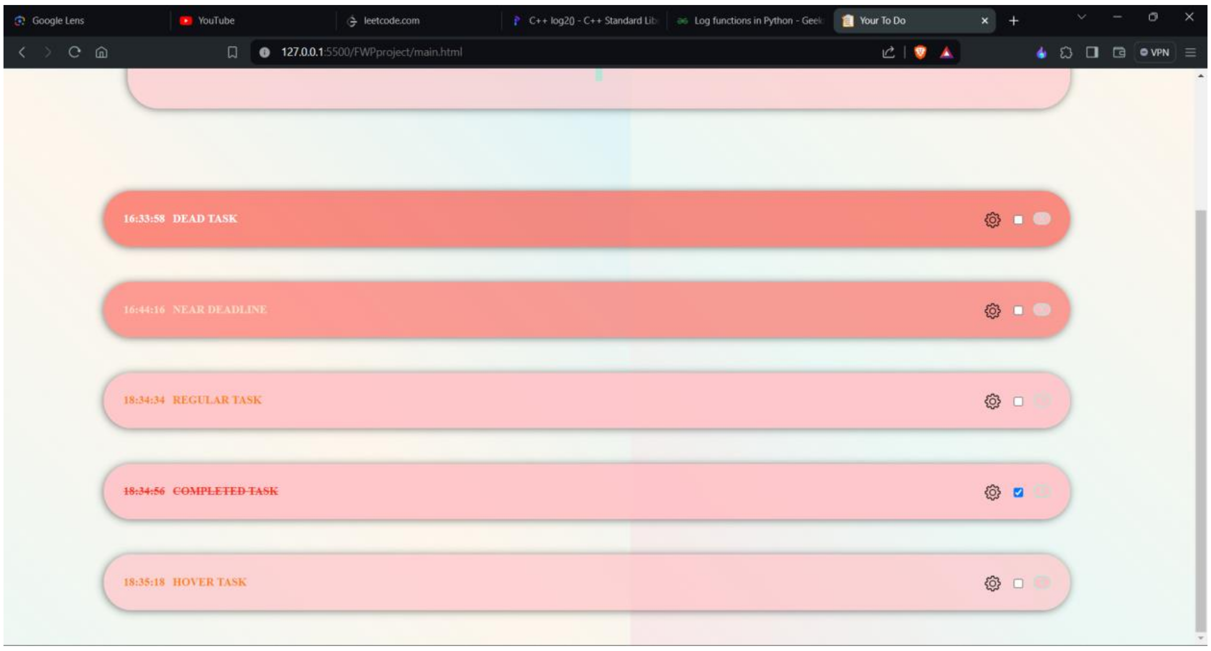
```
/* NAVBAR ELEMENT */
.navList li{
  width: 33%;
}
/* SITE NAME */
.siteName{
  font-size: 300%;
  font-weight: bolder;
  font-family:cursive;
  text-shadow: 2px 2px 5px #ce4d44;
}
/* LOG OUT AND HOME TEXT */
.logOut,.home{
  font-size: 150%;
  color: #ffc2c7e6;
  text-decoration: none;
}
/* LOG OUT AND HOME HOVER */
.logOut:hover,.home:hover{
  color: #ffc2c7ac;
  text-decoration: underline;
  cursor: pointer;
}
/* PROFILE PIC */
.profilePic{
  float: right;
  width: 40px;
  border-radius: 50px;
  border: 2px solid white;
  background-color: antiquewhite;
  box-shadow: -2px 2px 10px #00000085;

}
/* PROFILE PIC HOVER */
.profilePic:hover{
  border-color: black;
  cursor: pointer;
  box-shadow: -2px 2px 10px #ffffff85;
}
.profilePic img{
  width: 40px;
}
/* CONTAINER TO ADD TASK */
.containerAdd{
  position: relative;
  width:80%;
  margin: 4% 10% 4% 10% ;
  background-color: #ffc2c799;
  font-size: 100px;
  color:#B6E5D8;
  border: 0px;
```

DEFAULT


Google Lens
YouTube
leetcode.com
C++ log2() - C++ Standard Lib
Log functions in Python - Geek
Your To Do
127.0.0.1:5500/FWPproject/main.html
VPN
LOG OUT
Your To Do
+

TASKS



```
  border-radius: 40px;
  box-shadow: -1px 1px 10px #00000085;
}
/* CONTAINER HOVER */
.containerAdd:hover{
  background-color: #b6e5d88c;
  color:#ffc2c7e6;
  cursor: pointer;
}

/* TASK CLASS */
.task{
  position:relative;
  width:80%;
  height:max-content;
  margin: 3% 8%;
  background-color: #ffc2c7e6;
  font-size: 60%;
  color:#fd7F20;
  border: 0px;
  border-radius: 40px;
  padding: 1%;
  box-shadow: -1px 1px 10px #00000085;
}
/* TASK ON HOVER */
.task:hover{
  background-color:#ffc2c7ac;
  border: 0px;
  cursor: pointer;
}
/* TASKTEXT CLASS */
.taskText{
  margin-left: 10px;
}
/* TASK CHECKBOX */
.taskCheck{
  float: right;
  margin: 0.3% 1% 0% 1%;
}
/* TASKTEXT CHECKED */
.taskTextChecked{
  margin-left: 10px;
  text-decoration: line-through;
  color:#FC2E20;
}
/* TASKBUTTON */
.taskButton{
  float:right;
  background-color: #ffc2c7e6;
  font-size: 10px;
```

```
  color:#B6E5D8;
  border: 2px solid;
  margin-right: 10px;
  border-radius: 10px;
  text-align: center;
}
/* TASKBUTTON HOVER */
.taskButton:hover{
  background-color:#FC2E20;
  cursor: pointer;
}
/* TASK NEAR DEADLINE */
.deadline{
  animation: breathe 3s ease-in-out infinite alternate;
}
/* TASK DEAD */
.dead{
  background-color:#fc2f2089;
  color:white;
}
/* TASK RESET BUTTON */
.taskReset{
  width: 20px;
  height: 20px;
  float: right;
  margin-right: 5px;
}
```

JAVASCRIPT

From index.js

```
// FUNCTIONS TO BE LOADED ON DOCUMENT LOAD
window.onload = function() {
  // CHECKING LOCATION OF WINDOW
  if (window.location.pathname === '/FWPproject/main.html') {
    displayList();
    anchors = document.getElementsByClassName("logOut")
    for (let i = 0; i < anchors.length; i++) {
      // IF LOGOUT IS CLICKED CLEAR SESSION STORAGE REMOVING CURRENT STORAGE
      anchors[i].addEventListener('click', function() {
        sessionStorage.clear();
      });
    }
    // CHECKING TASK DEADLINE
    const intervalId = setInterval(checkTasks(), 1000);
  }
  if (window.location.pathname === '/FWPproject/profile.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/index.html') {
```

```
        ...
    }
    if (window.location.pathname === '/FWPproject/list.html') {
        ...
};

// DISPLAYING LISTS
function displayList(){
    // GETTING TASKS FROM '/' SEPARATED STRING IN LOCAL STORAGE
    taskList = localStorage[sessionStorage[0]].split("/")
    // 0 -> PASSWORD 1-> ID
    taskList = taskList.slice(2)
    // ITERATING THROUGH EACH ELEMENT OF TASKLIST
    taskList.forEach(element => {
        createTask(element);
    });
}
// FUNCTION TO CALCULATE THE DIFFERENCE BETWEEN TWO TIMES IN SECONDS
function getTimeDifference(time1, time2) {
    // GET TODAY'S DATE
    const today = new Date();
    // CONVERT TIME STRINGS TO DATE OBJECTS
    const date1 = new Date(today.getFullYear(), today.getMonth(), today.getDate(),
time1.split(':')[0], time1.split(':')[1], time1.split(':')[2]);
    const date2 = new Date(today.getFullYear(), today.getMonth(), today.getDate(),
time2.split(':')[0], time2.split(':')[1], time2.split(':')[2]);

    // CALCULATE THE DIFFERENCE IN MILLISECONDS
    const differenceInMs = date2.getTime() - date1.getTime();

    // CONVERT MILLISECONDS TO MINUTES AND RETURN
    return differenceInMs / (1000*60);
  }
// FUNCTION TO CHECK IF TASK DEADLINE IS MET
function checkTasks() {
    taskList = localStorage[sessionStorage[0]].split("/")
        taskList = taskList.slice(2)
        taskList.forEach(element => {
            taskTime = document.getElementById(element+"textTime")
            taskTime.textContent = element.split(",")[0]
            containerDiv = document.getElementById(element)
            if (getTimeDifference(getTime(),taskTime.textContent) < 10) {
                containerDiv.classList.add("deadline")
            }
            if (getTimeDifference(getTime(),taskTime.textContent) < 0) {
                containerDiv.classList.add("dead")
                containerDiv.classList.remove("deadline")
            }

        })
}
```

```
// FUNCTION TO CREATE TASK LIST ON MAIN PAGE
function createTask(element){
  // CONTAINER DIV
  let containerDiv = document.createElement('div')
  containerDiv.classList.add("task")
  containerDiv.id = element
  // TASK DIV WITH H2 TAG
  let taskDiv = document.createElement('h2')
  // TASK TIME WITH SPAN TAG
  let taskTime = document.createElement('span')
  taskTime.classList.add("taskText")
  taskTime.id = element+"textTime"
  taskTime.textContent = element.split(",")[0]
  // TASK WITH SPAN TAG
  let task = document.createElement('span')
  task.classList.add("taskText")
  task.id = element+"text"
  // GETTING TIME FROM TASKLIST ELEMENT
  task.textContent = element.split(",")[1]
  // TASKSET WITH IMAGE TAG
  let taskSet = document.createElement('img')
  taskSet.src = "./assets/setting.png"
  taskSet.classList.add("taskReset")
  taskSet.addEventListener('click', function(){
    sessionStorage["myText"] = element+"/"+task.textContent
    window.location.href = '/FWPproject/list.html'
  })
  // TASKSTAT WITH INPUT TAG
  let taskStat = document.createElement('input')
  taskStat.classList.add("taskCheck")
  taskStat.id = element+"check"
  taskStat.type = "checkbox"
  // EVENTLISTENER TO CHECK IF THE TASKSTAT HAS BEEN
CHANGED(CHECKED/UNCHECKED)
  taskStat.addEventListener('change', function() {
    // IF TASKSTAT HAS CHANGED THAN CORRESPONDINGLY CHANGE TASKTEXT AND
TASKTIME
    document.getElementById(element+"text").classList.toggle("taskTextChecked")
    document.getElementById(element+"textTime").classList.toggle("taskTextChecked")
  });
  // REMOVETASK WITH BUTTON TAG
  let removeTask = document.createElement('button')
  removeTask.classList.add("taskButton")
  removeTask.id = element+"rem"
  removeTask.textContent = "X"
  // EVENTLISTENER TO CHECK IF THE REMOVETASK HAS BEEN CLICKED
  removeTask.addEventListener('click', function() {
    // REMOVE ELEMENT FROM HTML
    document.getElementById(element).remove();
    // REMOVE TASK FROM LOCAL STORAGE
```

```
        localStorage[sessionStorage[0]] = localStorage[sessionStorage[0]].replace(new
RegExp("/"+element, 'g'), '');
    });
    // APPENDING CHILD CLASSSES TO TASKDIV
    taskDiv.appendChild(taskTime)
    taskDiv.appendChild(task)
    taskDiv.appendChild(removeTask)
    taskDiv.appendChild(taskStat)
    taskDiv.appendChild(taskSet)
    // APPENDING TASKDIV TO CONTAINERDIV
    containerDiv.appendChild(taskDiv)
    // APPENDING CONTAINERDIV TO DOCUMENT BODY
    document.body.appendChild(containerDiv)
}
// GO TO LIST HTML
function goList(){
    window.location.href = "./list.html"
}
```

TASK ADDING PAGE

HTML

From list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="./index.css" type="text/css" rel="stylesheet">
  <script src="index.js"></script>
  <link rel="icon" type="image/png" href="./assets/siteLogo.png">
  <title>New List</title>
</head>
<body>
  <nav>
    <ul class="navList">
      <li><a class="home" href="./main.html">HOME</a></li>
      <li style="text-align: center;"><span class="siteName">Your To Do</span></li>
      <li><a href="./profile.html" class="profilePic">
        <img src="./assets/profile.png" alt="PROFILE PIC">
      </a></li>
    </ul>
  <form>
    <table style="text-align: center; background-color: #ffc2c758; width: 80%; height:
300px;margin:10% 10%;border-collapse: collapse;border-color: #8FDDE7; font-size: 150%;"
border="4px" >
      <tr>
        <th>TIME</th>
        <th>TASK</th>
      </tr>
      <tr>
        <td><input id="taskTime1" type="time"></td>
        <td><textarea id="taskArea1" style="background-color: #FBE5C8;" rows="15"
cols="40"></textarea></td>
      </tr>
    </table>
  </form>
  <button class="addListBut" id="addListBut" onclick="addList()">ADD TO MY LIST</button>
</body>
</html>
```

CSS

From index.css

```
/* list.html */
/* ADD LIST BUTTON CLASS */
```

```
  .addListBut{
  background-color: #ffc2c7e6;
  color:#fd7F20;
  border: 0px;
  width: 30%;
  height: 40px;
  margin: auto 35%;
  border-radius: 50px;
  font-size: 120%;
  box-shadow: -1px 1px 10px #00000085;
}
/* ADD LIST HOVER */
.addListBut:hover{
  background-color:#fd7F20;
  color:#ffc2c7e6;
  border: 0px;
  border-radius: 40px;
  cursor: pointer;
  box-shadow: -1px 1px 10px #00000085;
}
```

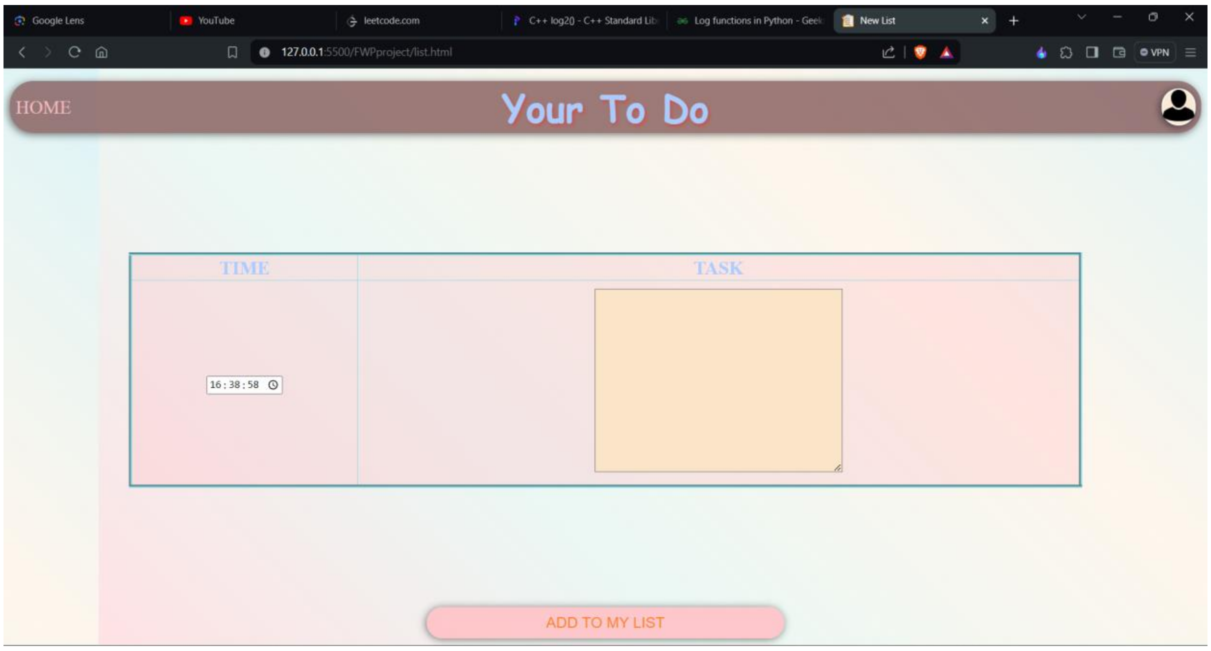
JAVASCRIPT

From index.js

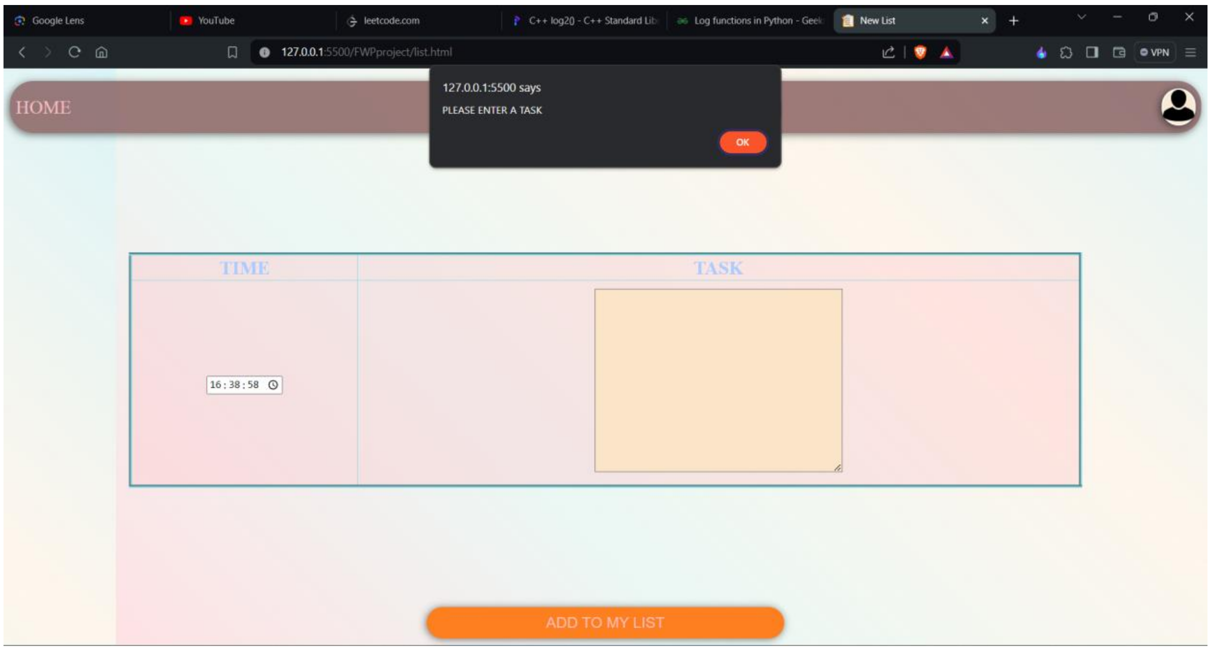
```
// FUNCTIONS TO BE LOADED ON DOCUMENT LOAD
window.onload = function() {
  // CHECKING LOCATION OF WINDOW
  if (window.location.pathname === '/FWPproject/main.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/profile.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/index.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/list.html') {
    // SETTING DEFAULT INPUT TIME TO CURRENT SYSTEM TIME
    defaultTime = document.getElementById("taskTime1")
    defaultTime.value = getTime();
    if (sessionStorage.getItem("myText") && typeof sessionStorage.getItem("myText") ===
'string'){
      const parts = sessionStorage.getItem("myText").split("/");
 document.getElementById('taskArea1').value = parts[1];
      (document.getElementById("addListBut")).addEventListener('click', function() {
```

```
        localStorage[sessionStorage[0]] = localStorage[sessionStorage[0]].replace(new
RegExp("/"+(parts[0]), 'g'), '')
  sessionStorage.removeItem("myText")
        })
      }
  }
```

DEFAULT



EMPTY TASK

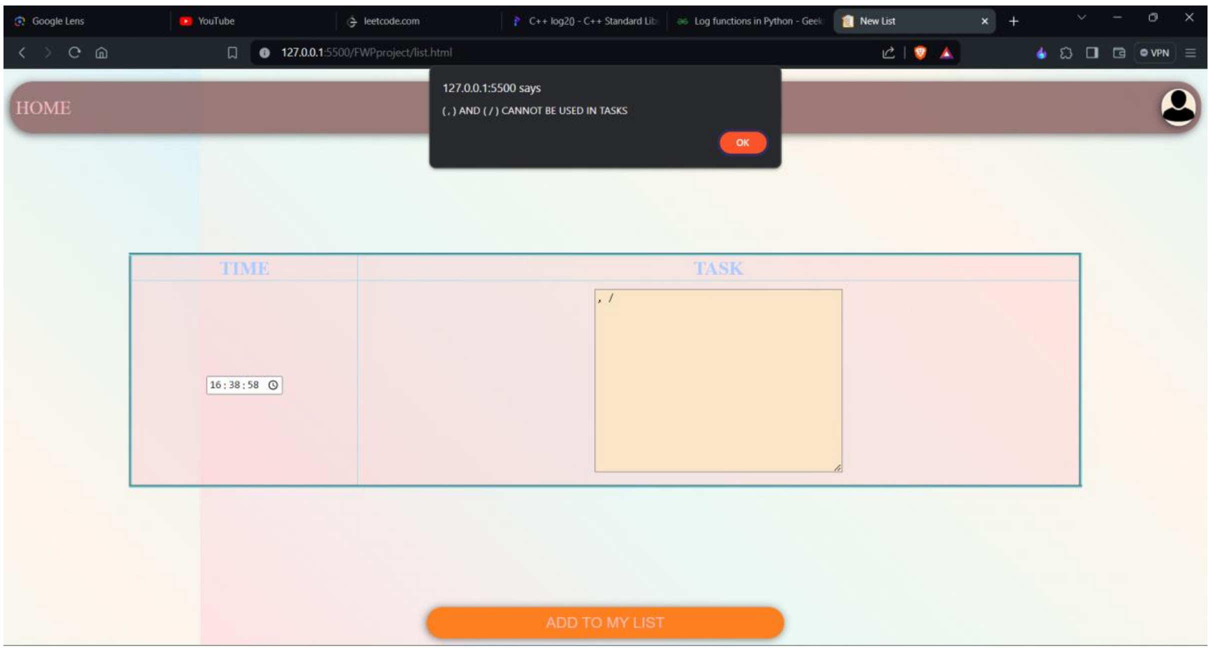


127.0.0.1:5500 says

PLEASE ENTER A TASK

```
};
// ADDING TASKS
function addList() {
  taskTime = document.getElementById("taskTime1")
  taskArea = document.getElementById("taskArea1")
  // IF TASK ENTERED
  if (taskArea.value != ""){
    if (taskArea.value.indexOf(",") != -1 | taskArea.value.indexOf("/") != -1){
      alert("( , ) AND ( / ) CANNOT BE USED IN TASKS")
      return
    }
    task = (taskTime.value+","+taskArea.value)
    // STORING TASK TIME AND DESCRIPTION TO LOCAL STORAGE
    localStorage[sessionStorage[0]] += "/"+task
    // RESETING TIME AND TASK
    taskTime.value = getTime()
    taskArea.value = ""
  // IF TASK NOT ENTERED
  } else{
    alert("PLEASE ENTER A TASK")
  }
}
```

INVALID CHARACTERS



127.0.0.1:5500 says

(,) AND (/) CANNOT BE USED IN TASKS

PROFILE PAGE

HTML

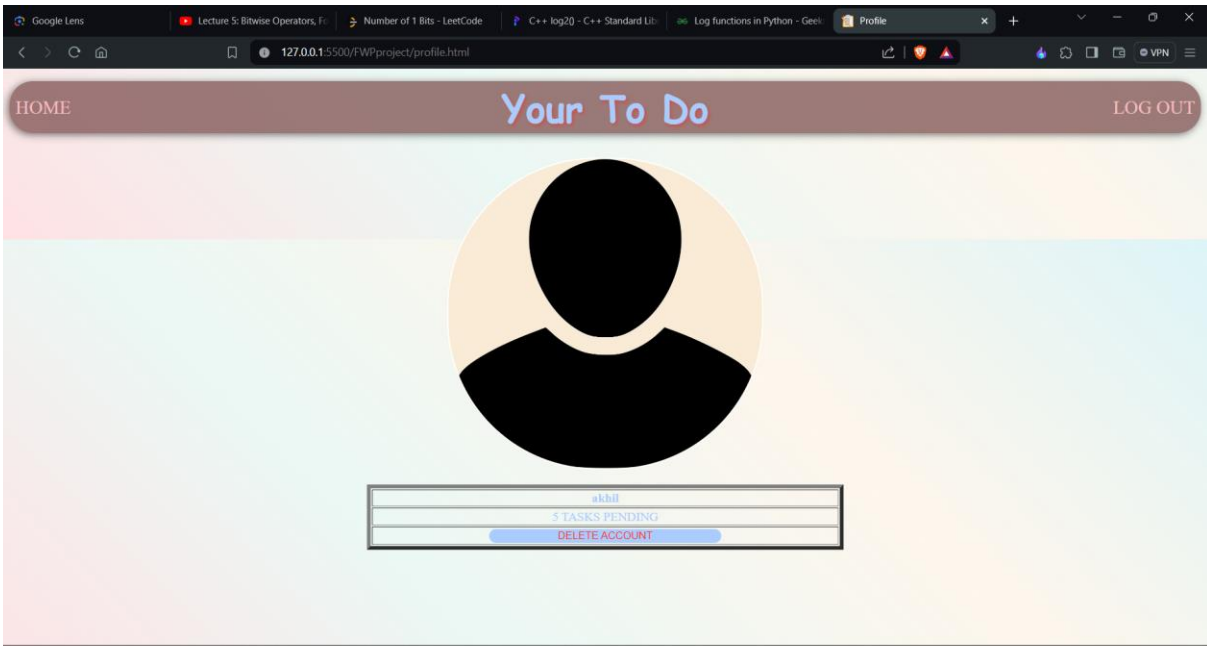
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="./index.css" type="text/css" rel="stylesheet">
  <script src="./index.js"></script>
  <link rel="icon" type="image/png" href="./assets/siteLogo.png">
  <title>Profile</title>
</head>
<body style="text-align: center;">
  <nav>
    <ul class="navList">
      <li><a class="home" style="float: left;" href="./main.html">HOME</a></li>
      <li style="text-align: center;"><span class="siteName">Your To Do</span></li>
      <li><a class="logOut" style="float: right;" href="./index.html">LOG OUT</a></li>
    </ul>
  </nav>
  <img width="400px" src="./assets/profile.png" alt="" style="border-radius: 190px;border: 2px
solid white;background-color: antiquewhite;margin-top: 1%;">
  <table border="5px" style="width: 40%; margin-left: 30%;margin-top: 1%;">
    <tr>
      <th id="userName">USER</th>
    </tr>
    <tr>
      <td id="pTasks">PENDING TASKS</td>
    </tr>
    <tr>
      <td id="rAcc"><button class="remButton" onclick="removeAcc()">DELETE
ACCOUNT</button></td>
    </tr>
  </table>
</body>
</html>
```

CSS

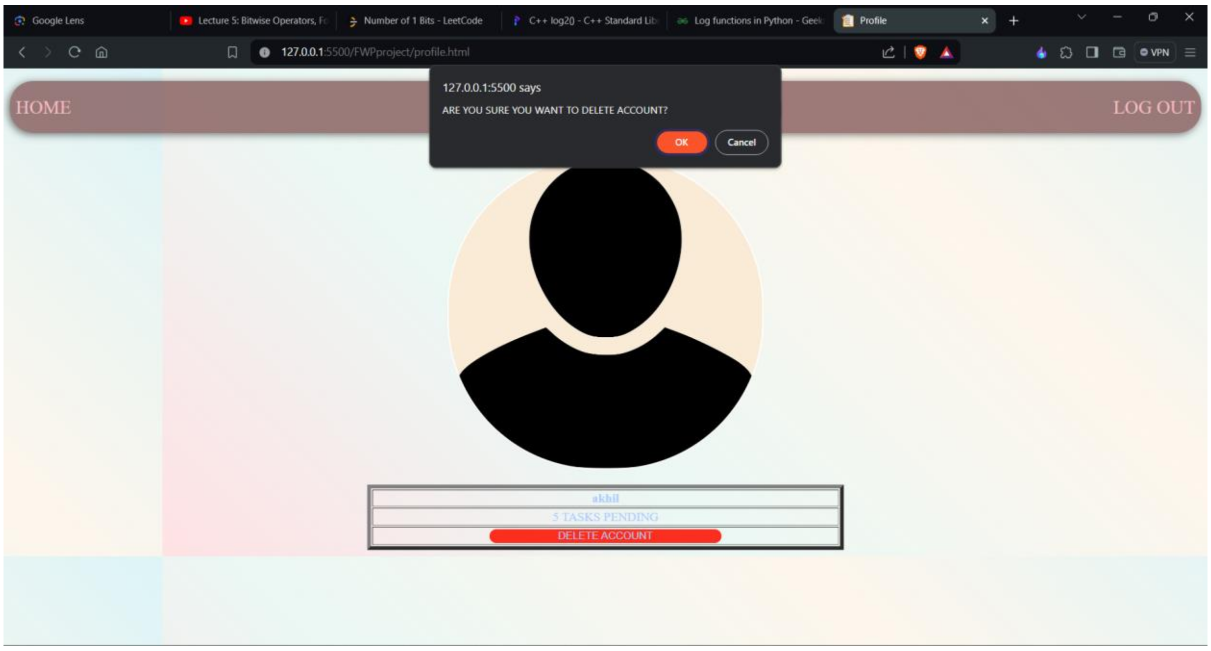
From index.css

```css
/* profile.html */
/* REMBUTTON CLASS */
.remButton{
  width: 50%;
  background-color: #AACCFF;
  color:#FC2E20;
  border: 0px;
```

DEFAULT


Google Lens
Lecture 5: Bitwise Operators, F
Number of 1 Bits - LeetCode
C++ log20 - C++ Standard Lib
Log functions in Python - Geek
Profile
127.0.0.1:5500/FWPproject/profile.html
HOME
Your To Do
LOG OUT
akhil
5 TASKS PENDING
DELETE ACCOUNT

DELETE ACCOUNT


Google Lens
Lecture 5: Bitwise Operators, F
Number of 1 Bits - LeetCode
C++ log20 - C++ Standard Lib
Log functions in Python - Geek
Profile
127.0.0.1:5500/FWPproject/profile.html
HOME
LOG OUT
OK
Cancel
akhil
5 TASKS PENDING
DELETE ACCOUNT

127.0.0.1:5500 says

ARE YOU SURE YOU WANT TO DELETE ACCOUNT?

```
  border-radius: 20px;
}
/* REMBUTTON HOVER */
.remButton:hover{
  background-color:#FC2E20;
  color:#AACCFF;
  cursor: pointer;
}
```

JAVASCRIPT

From index.js

```
// FUNCTIONS TO BE LOADED ON DOCUMENT LOAD
window.onload = function() {
  // CHECKING LOCATION OF WINDOW
  if (window.location.pathname === '/FWPproject/main.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/profile.html') {
    // DISPLAYING PENDING TASKS
    user = document.getElementById("userName")
    user.textContent = sessionStorage[0]
    taskList = localStorage[sessionStorage[0]].split("/")
    taskList = taskList.slice(2)
    pTask =  document.getElementById("pTasks")
    if (taskList.length > 1) {
      pTask.textContent = taskList.length + " TASKS PENDING"
    }
    else{
      pTask.textContent = taskList.length + " TASK PENDING"
    }
    anchors = document.getElementsByClassName("logOut")
    for (let i = 0; i < anchors.length; i++) {
      // IF LOGOUT IS CLICKED CLEAR SESSION STORAGE REMOVING CURRENT STORAGE
      anchors[i].addEventListener('click', function() {
        sessionStorage.clear();
      });
    }
  }
  if (window.location.pathname === '/FWPproject/index.html') {
    ...
  }
  if (window.location.pathname === '/FWPproject/list.html') {
    ...
  }
};
// DELETING ACCOUNT
function removeAcc(){
  let result = window.confirm('ARE YOU SURE YOU WANT TO DELETE ACCOUNT?');
```

```
  // CONFIRMATION
  if (result) {
    // REMOVE FROM LOCAL STORAGE
    localStorage.removeItem(sessionStorage[0])
    // CLEAR SESSION STORAGE
    sessionStorage.clear()
    alert('ACCOUNT DELETED!');
    // RETURNING TO LOGIN PAGE
    window.location.href = "./index.html"
  } else {
    alert('DELETION CANCELLED.');
  }
}
```

Thank You