AI Assisted Coding

Program :B.tech(CSE)

Name of Student : ANANTHA MANIDEEP

Enrollment No. :2403A52078

Batch No. :02

#LAB ASSIGNMENT-14.2:

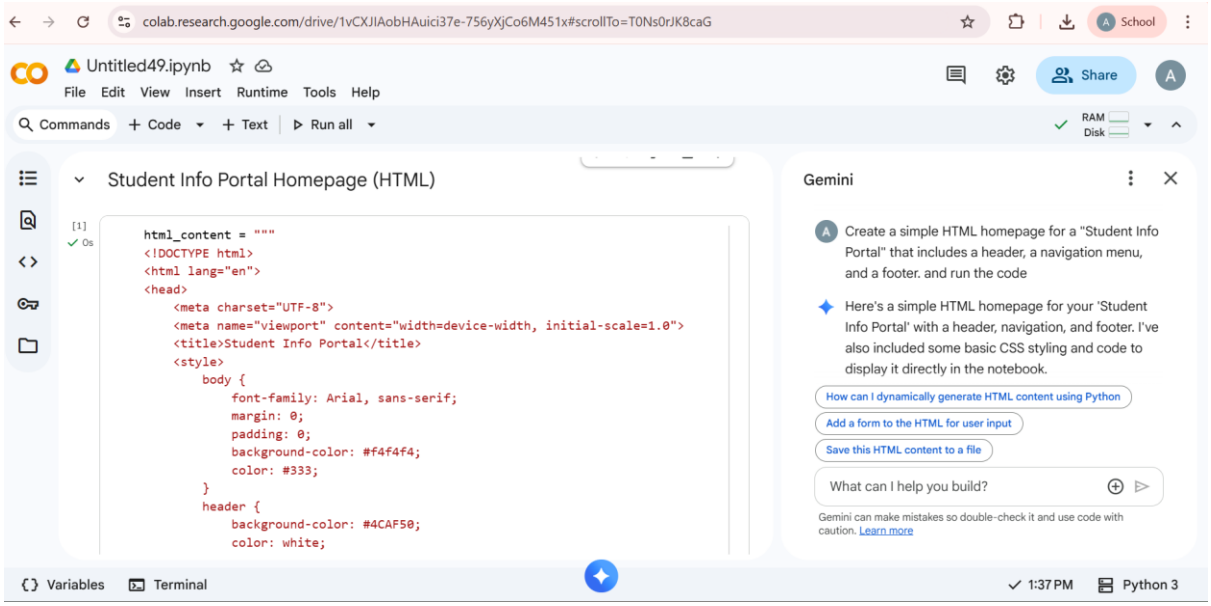
➢ TASK DESCRIPTION-1:

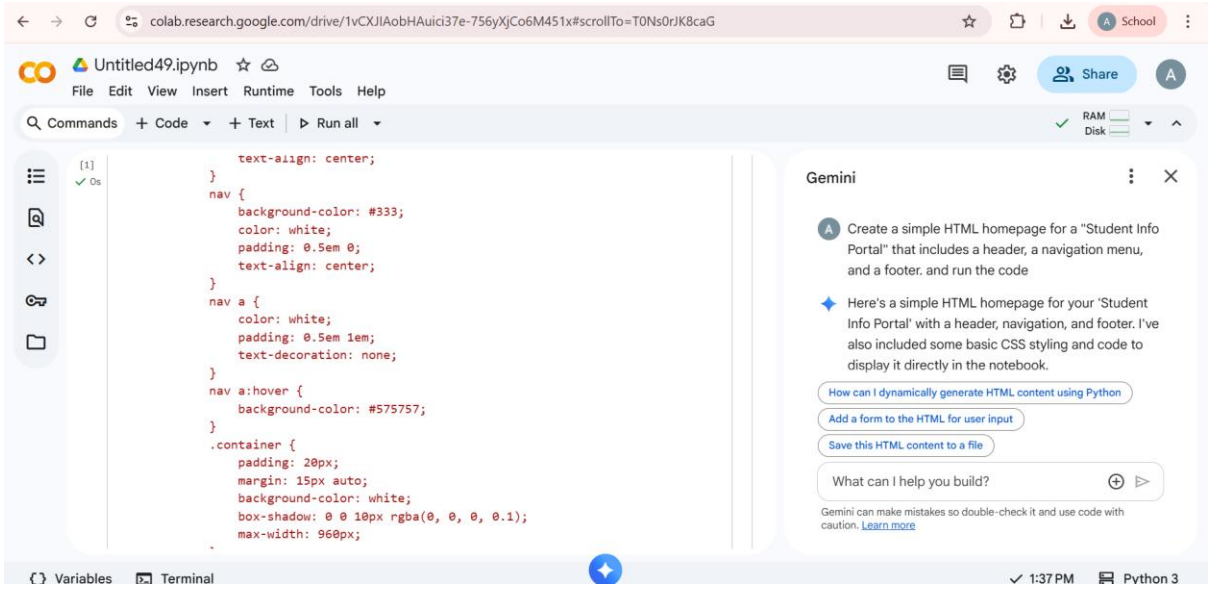
AI-generated HTML Page
Task: Ask AI to generate a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer.

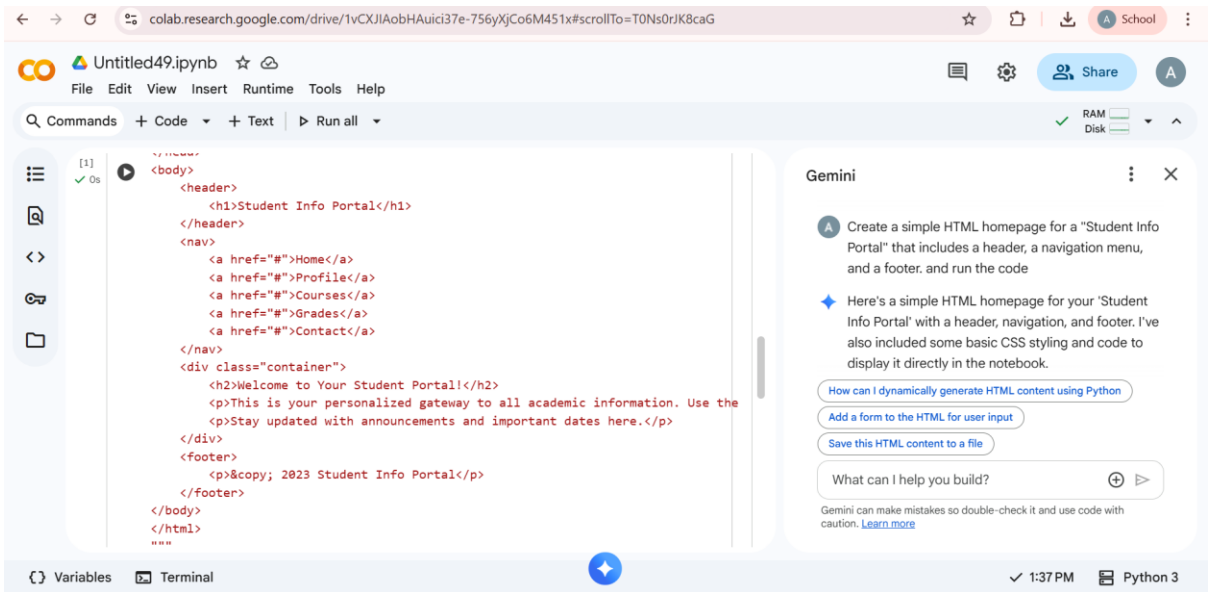
➢ PROMPT:

Create a simple HTML homepage for a "Student Info Portal" that includes a header, a navigation menu, and a footer.

CODE With OUTPUT:

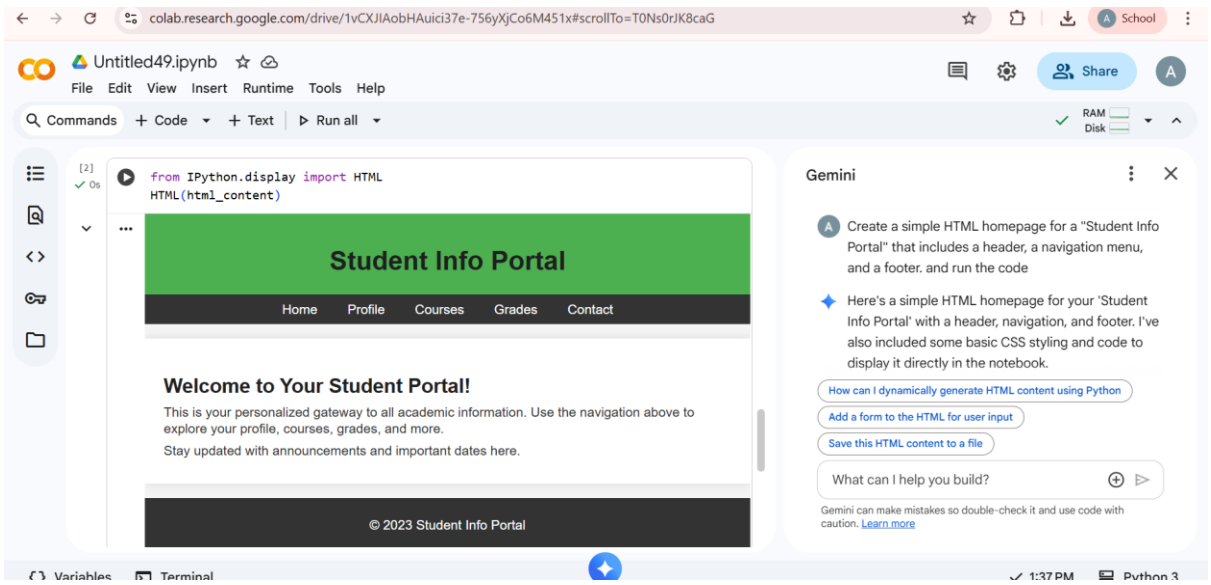



colab.research.google.com/drive/1vCXJIAobHAuici37e-756yXjCo6M451x#scrollTo=T0Ns0rJK8caG
Untitled49.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[1] 0s
text-align: center;
}
nav {
background-color: #333;
color: white;
padding: 0.5em 0;
text-align: center;
}
nav a {
color: white;
padding: 0.5em 1em;
text-decoration: none;
}
nav a:hover {
background-color: #575757;
}
.container {
padding: 20px;
margin: 15px auto;
background-color: white;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
max-width: 960px;
Gemini
Create a simple HTML homepage for a "Student Info Portal" that includes a header, a navigation menu, and a footer. and run the code
Here's a simple HTML homepage for your 'Student Info Portal' with a header, navigation, and footer. I've also included some basic CSS styling and code to display it directly in the notebook.
How can I dynamically generate HTML content using Python
Add a form to the HTML for user input
Save this HTML content to a file
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables Terminal
1:37 PM Python 3


colab.research.google.com/drive/1vCXJIAobHAuici37e-756yXjCo6M451x#scrollTo=T0Ns0rJK8caG
Untitled49.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[1] 0s
<body>
<header>
<h1>Student Info Portal</h1>
</header>
<nav>
Home
Profile
Courses
Grades
Contact
</nav>

<h2>Welcome to Your Student Portal!</h2>
This is your personalized gateway to all academic information. Use the
Stay updated with announcements and important dates here.

<footer>
© 2023 Student Info Portal
</footer>
</body>
</html>
"""
Gemini
Create a simple HTML homepage for a "Student Info Portal" that includes a header, a navigation menu, and a footer. and run the code
Here's a simple HTML homepage for your 'Student Info Portal' with a header, navigation, and footer. I've also included some basic CSS styling and code to display it directly in the notebook.
How can I dynamically generate HTML content using Python
Add a form to the HTML for user input
Save this HTML content to a file
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables Terminal
1:37 PM Python 3


colab.research.google.com/drive/1vCXJIAobHAuici37e-756yXjCo6M451x#scrollTo=T0Ns0rJK8caG
Untitled49.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[2] 0s
from IPython.display import HTML
HTML(html_content)
Student Info Portal
Home Profile Courses Grades Contact
Welcome to Your Student Portal!
This is your personalized gateway to all academic information. Use the navigation above to explore your profile, courses, grades, and more.
Stay updated with announcements and important dates here.
© 2023 Student Info Portal
Gemini
Create a simple HTML homepage for a "Student Info Portal" that includes a header, a navigation menu, and a footer. and run the code
Here's a simple HTML homepage for your 'Student Info Portal' with a header, navigation, and footer. I've also included some basic CSS styling and code to display it directly in the notebook.
How can I dynamically generate HTML content using Python
Add a form to the HTML for user input
Save this HTML content to a file
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables Terminal
1:37 PM Python 3

EXPLANATION:

- **%%html**: This is a Colab magic command that tells the notebook to render the cell's content as HTML.
- **<!DOCTYPE html>**: Declares the document type to be HTML5.
- **<html>**: The root element of the HTML page.
- **<head>**: Contains meta-information about the HTML document, such as the title that appears in the browser tab (<title>). It also includes a <style> block for CSS rules that define the appearance of the page.
- **<body>**: Contains the visible content of the HTML page.
- **<header>**: Represents the introductory content or a set of navigational links. In this case, it contains the main heading "Student Info Portal" (<h1>).
- **<nav>**: Represents a section of navigation links. Here, it contains links for "Home", "Profile", "Courses", and "Grades" (<a> tags).
- ** **: A generic container element used to group content.
  The class="container" is used by the CSS to apply specific styling (padding in this case).
- **<h2>**: A second-level heading.
- **<p>**: A paragraph of text.
- **<footer>**: Represents the footer for its nearest sectioning content or the root element. It typically contains information like copyright notices.

The <style> section contains CSS rules that define how the HTML elements are displayed:

- **body**: Sets the default font, removes default margins and padding, and sets a background color.
- **header**: Styles the header with a background color, text color, padding, and center-aligned text.
- **nav**: Styles the navigation bar similarly to the header, with a different background color.
- **nav a**: Styles the navigation links (the <a> tags within the <nav>) with text color, margins, and removes the default underline.
- **nav a:hover**: Adds an underline when the mouse hovers over a navigation link.
- **.container**: Adds padding to the container div.
- **footer**: Styles the footer with a background color, text color, center-aligned text, padding, and positions it at the bottom of the page.

➢ TASK DESCRIPTION-2:

Use AI to add CSS styling to

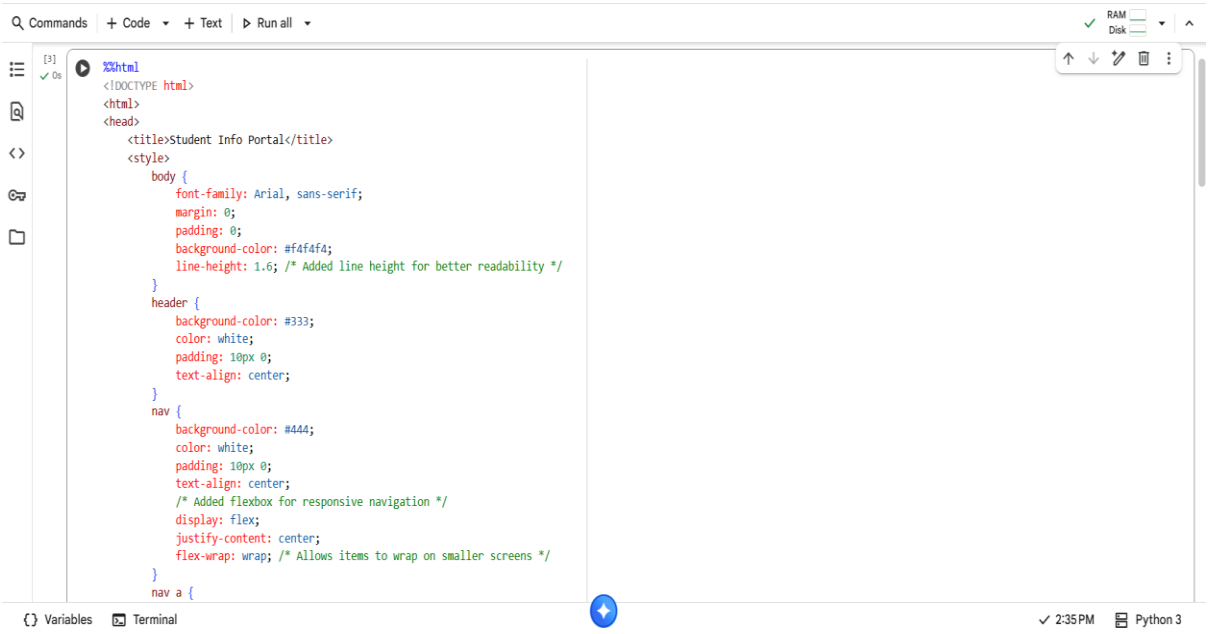
Task #1 homepage for:

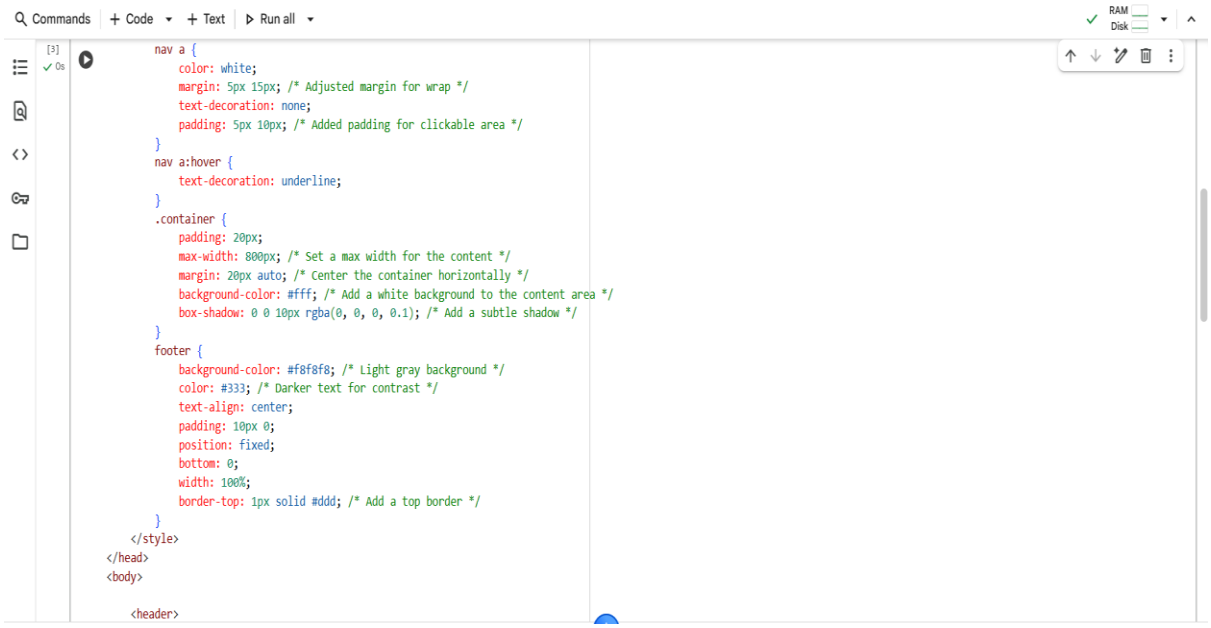
- Responsive navigation bar.
- Centered content section.
- Footer with light gray background.

➢ PROMPT:

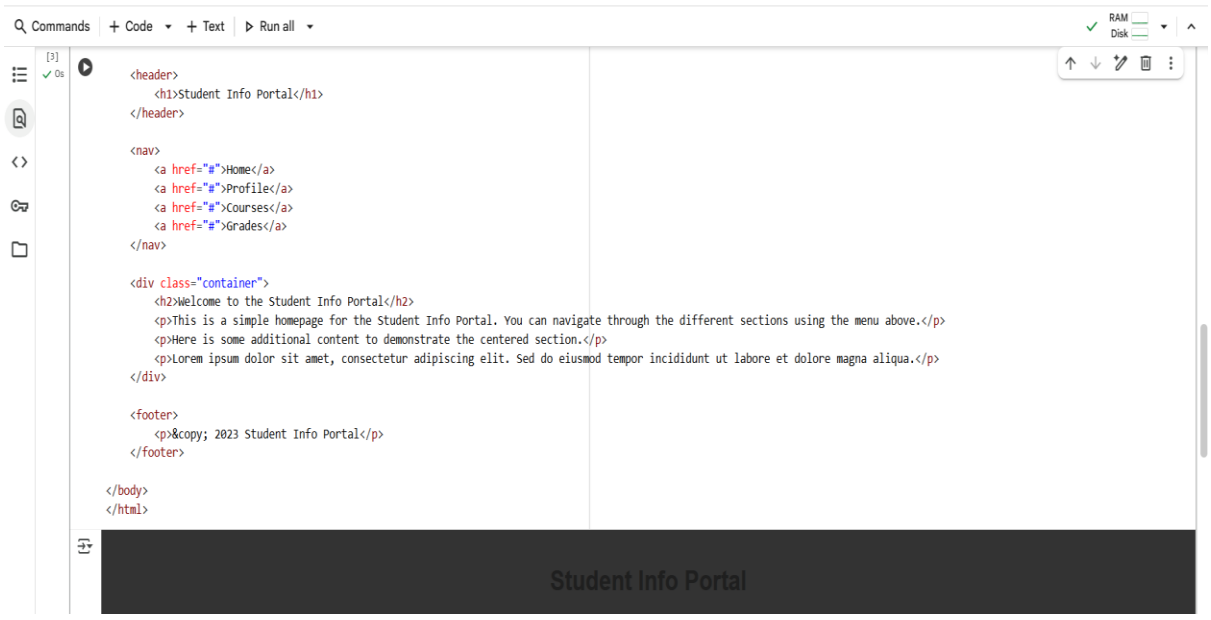
Add CSS styling to the "Student Info Portal" homepage to include:A responsive navigation bar.A centered content section.A footer with a light gray background.

CODE With OUTPUT:



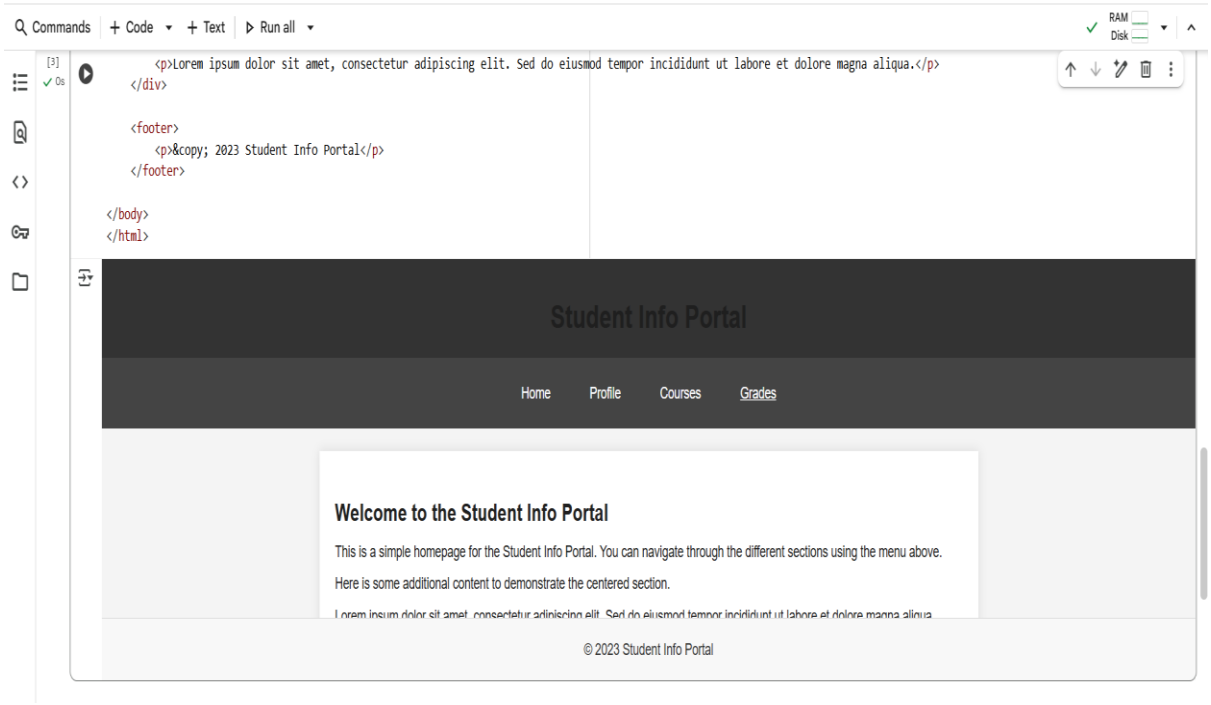


```
        nav a {
            color: white;
            margin: 5px 15px; /* Adjusted margin for wrap */
            text-decoration: none;
            padding: 5px 10px; /* Added padding for clickable area */
        }
        nav a:hover {
            text-decoration: underline;
        }
        .container {
            padding: 20px;
            max-width: 800px; /* Set a max width for the content */
            margin: 20px auto; /* Center the container horizontally */
            background-color: #fff; /* Add a white background to the content area */
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
        }
        footer {
            background-color: #f8f8f8; /* Light gray background */
            color: #333; /* Darker text for contrast */
            text-align: center;
            padding: 10px 0;
            position: fixed;
            bottom: 0;
            width: 100%;
            border-top: 1px solid #ddd; /* Add a top border */
        }
    </style>
</head>
<body>

    <header>
```



```
    <header>
        <h1>Student Info Portal</h1>
    </header>

    <nav>
        <a href="#">Home</a>
        <a href="#">Profile</a>
        <a href="#">Courses</a>
        <a href="#">Grades</a>
    </nav>

    <div class="container">
        <h2>Welcome to the Student Info Portal</h2>
        <p>This is a simple homepage for the Student Info Portal. You can navigate through the different sections using the menu above.</p>
        <p>Here is some additional content to demonstrate the centered section.</p>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    </div>

    <footer>
        <p>&copy; 2023 Student Info Portal</p>
    </footer>

</body>
</html>
```

Student Info Portal



EXPLANATION:

- **body**: line-height: 1.6; was added for better readability of text.
- **nav**:
    - display: flex;: This makes the navigation links arrange themselves in a row.
    - justify-content: center;: This centers the navigation links horizontally within the nav bar.
    - flex-wrap: wrap;: This allows the navigation links to wrap onto the next line if the screen is too narrow to fit them all on one line, making the navigation bar responsive.
- **nav a**:
    - margin: 5px 15px;: The margin was adjusted to provide some space around the links when they wrap.
    - padding: 5px 10px;: Padding was added to increase the clickable area of the links.
- **.container**:
    - max-width: 800px;: Sets a maximum width for the content area, preventing it from becoming too wide on large screens.

- margin: 20px auto;: This is a common technique to center a block-level element horizontally. auto for the left and right margins tells the browser to distribute the available space equally on both sides.
- background-color: #fff;: Sets the background color of the content container to white.
- box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);: Adds a subtle shadow around the container to make it stand out.

- **footer**:
  - background-color: #f8f8f8;: Sets the background color of the footer to a light gray.
  - color: #333;: Sets the text color in the footer to a dark gray for better contrast.
  - border-top: 1px solid #ddd;: Adds a light gray border at the top of the footer.

Additional HTML Content:

- More <p> tags with placeholder text (Lorem ipsum...) were added within the .container div to better demonstrate the centered content section with more content.

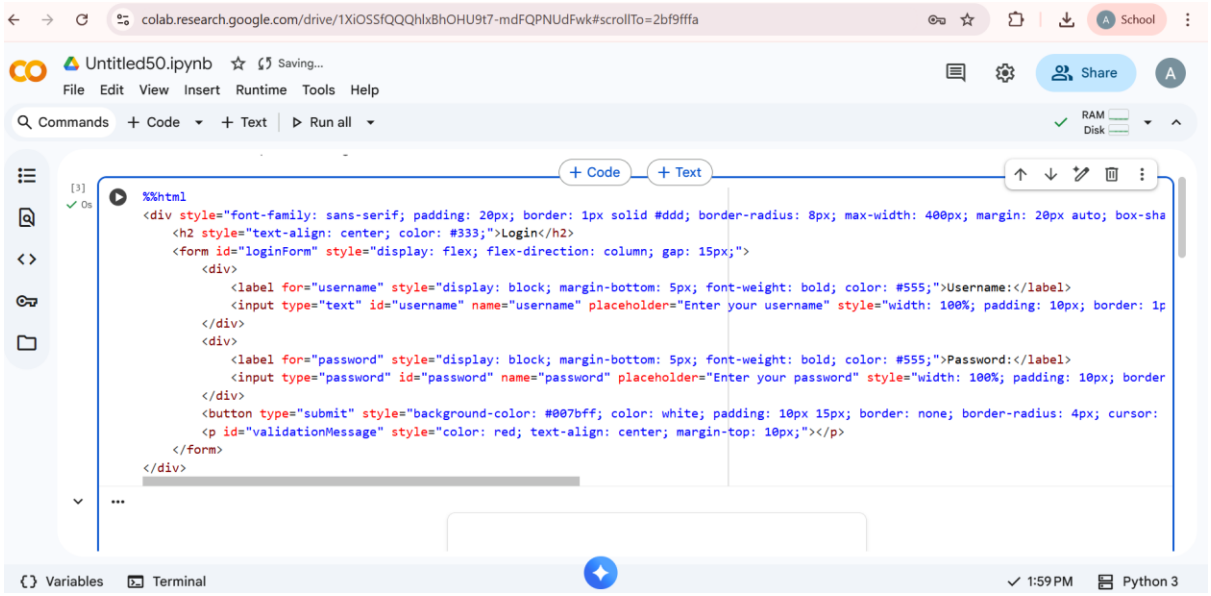
➢ TASK DESCRIPTION-3:

JavaScript Interactivity
Task: Prompt AI to generate a JS script that validates a simple login form (non-empty username/password).

➢ PROMPT:

Write a JavaScript script that validates a simple login form to ensure the username and password fields are not empty before submission.

CODE With OUTPUT:


colab.research.google.com/drive/1XiOSSfQQQhlxBhOHU9t7-mdFQPNUdFwk#scrollTo=2bf9fffa
Untitled50.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
+ Code + Text
[3]
%%html
<div style="font-family: sans-serif; padding: 20px; border: 1px solid #ddd; border-radius: 8px; max-width: 400px; margin: 20px auto; box-sha
<h2 style="text-align: center; color: #333;">Login</h2>
<form id="loginForm" style="display: flex; flex-direction: column; gap: 15px;">

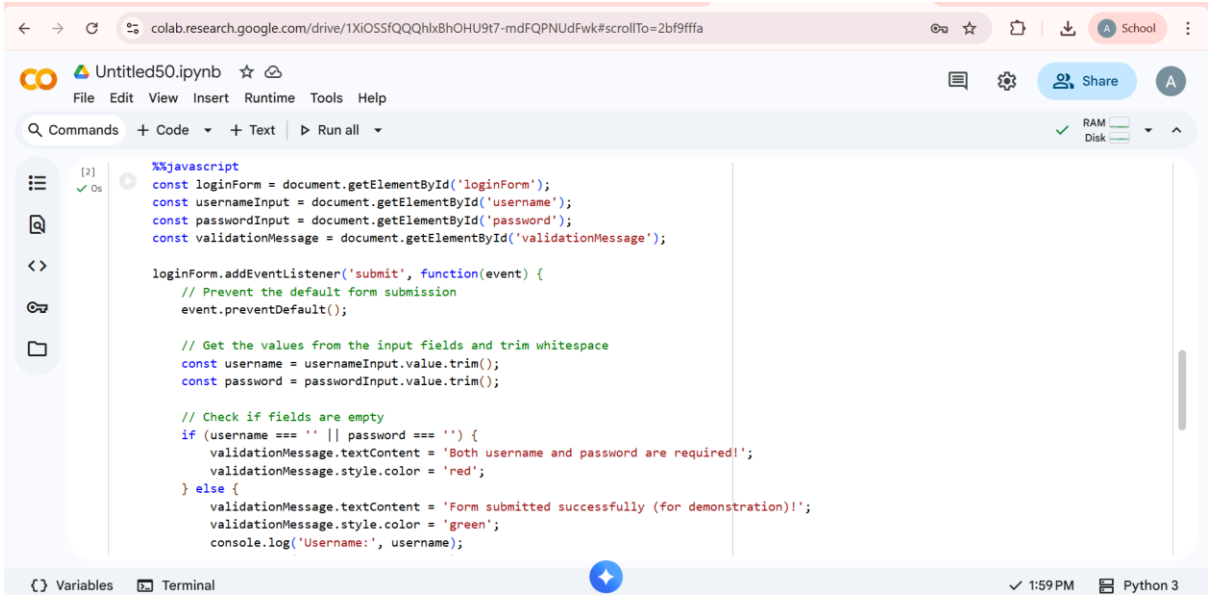
<label for="username" style="display: block; margin-bottom: 5px; font-weight: bold; color: #555;">Username:</label>
<input type="text" id="username" name="username" placeholder="Enter your username" style="width: 100%; padding: 10px; border: 1p

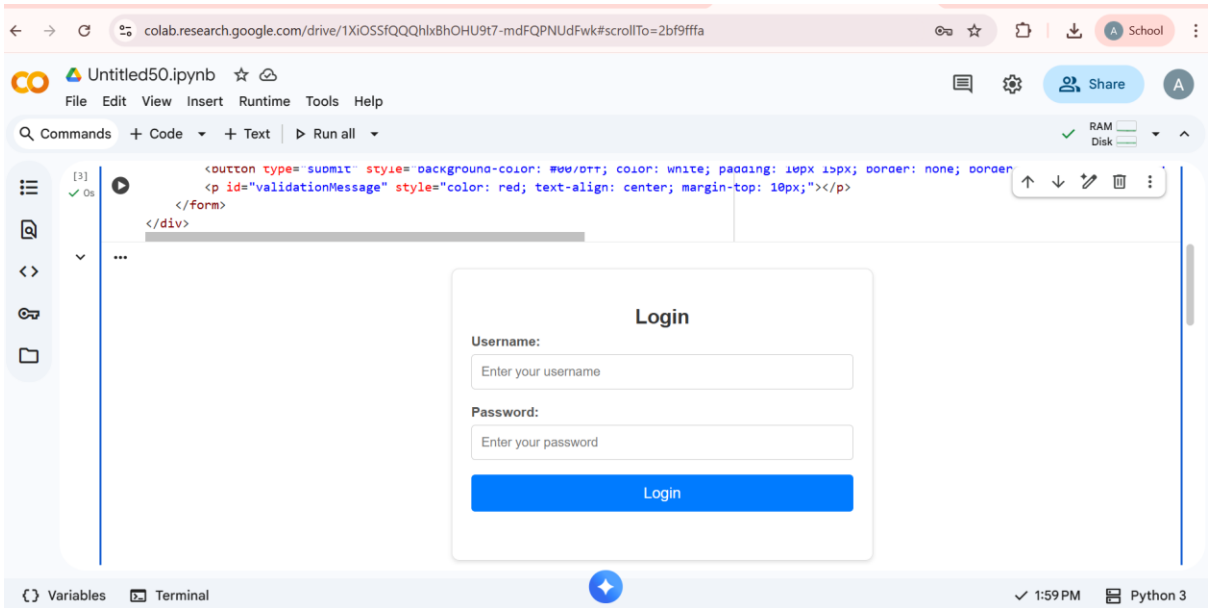
<label for="password" style="display: block; margin-bottom: 5px; font-weight: bold; color: #555;">Password:</label>
<input type="password" id="password" name="password" placeholder="Enter your password" style="width: 100%; padding: 10px; border

<button type="submit" style="background-color: #007bff; color: white; padding: 10px 15px; border: none; border-radius: 4px; cursor:

</form>

Variables Terminal
1:59 PM
Python 3


colab.research.google.com/drive/1XiOSSfQQQhlxBhOHU9t7-mdFQPNUdFwk#scrollTo=2bf9fffa
Untitled50.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[2]
%%javascript
const loginForm = document.getElementById('loginForm');
const usernameInput = document.getElementById('username');
const passwordInput = document.getElementById('password');
const validationMessage = document.getElementById('validationMessage');
loginForm.addEventListener('submit', function(event) {
// Prevent the default form submission
event.preventDefault();
// Get the values from the input fields and trim whitespace
const username = usernameInput.value.trim();
const password = passwordInput.value.trim();
// Check if fields are empty
if (username === '' || password === '') {
validationMessage.textContent = 'Both username and password are required!';
validationMessage.style.color = 'red';
} else {
validationMessage.textContent = 'Form submitted successfully (for demonstration)!';
validationMessage.style.color = 'green';
console.log('Username:', username);
Variables Terminal
1:59 PM
Python 3


colab.research.google.com/drive/1XiOSSfQQQhlxBhOHU9t7-mdFQPNUdFwk#scrollTo=2bf9fffa
Untitled50.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[3]

</form>

Login
Username:
Enter your username
Password:
Enter your password
Login
Variables Terminal
1:59 PM
Python 3

EXPLANANTION:

- **%%html**: This is a Colab magic command that tells the notebook to render the cell's content as HTML.
- **<!DOCTYPE html>, <html>, <head>, <body>**: These are standard HTML tags that define the basic structure of the document.
- **<title>Login Form</title>**: Sets the title of the HTML page, which appears in the browser tab.
- **<style> ... </style>**: This block contains the CSS rules that style the HTML elements.
    - .login-container: Styles the main container div that holds the login form. It sets the width, margins (to center it), padding, border, border radius, background color, and font family.
    - .login-container h2: Styles the heading inside the container, centering the text and adding a bottom margin.
    - .login-container label: Styles the labels for the input fields, making them block elements (so they appear on their own line), adding a bottom margin, and making the text bold.
    - .login-container input[type="text"], .login-container input[type="password"]: Styles the username and password input fields. It sets the width to 100%, adds padding, bottom margin, border, border radius, and uses box-sizing: border-box; to include padding and border in the element's total width.
    - .login-container button: Styles the submit button. It sets the width, padding, background color, text color, removes the border, adds a border radius, sets the cursor to a pointer on hover, and sets the font size.
    - .login-container button:hover: Changes the background color of the button when the mouse hovers over it.
    - .error-message: Styles the div where error messages will be displayed, setting the text color to red, adding a top margin, and centering the text.
- ** ... **: This div acts as a container for the login form and the heading. The CSS class login-container is applied to this div to style it.
- **<h2>Login</h2>**: The heading for the login form.
- **<form id="loginForm"> ... </form>**: This is the HTML form element. The id="loginForm" is used to reference this form in the JavaScript code.
- ** ... **: These divs are used to group each label and input field.
- **<label for="username">Username:</label>**, **<label for="password">Password:</label>**: These are labels for the input fields. The for attribute links the label to the input field with the corresponding id.

- **<input type="text" id="username" name="username">, <input type="password" id="password" name="password">**: These are the input fields where the user will type their username and password. The type attribute specifies the type of input (text or password), and the id and name attributes are used to identify the input fields.
- **<button type="submit">Login</button>**: This is the button that submits the form.
- ** **: This div is where the JavaScript will display any error messages. It has an id to be referenced by the JavaScript and a class for styling.

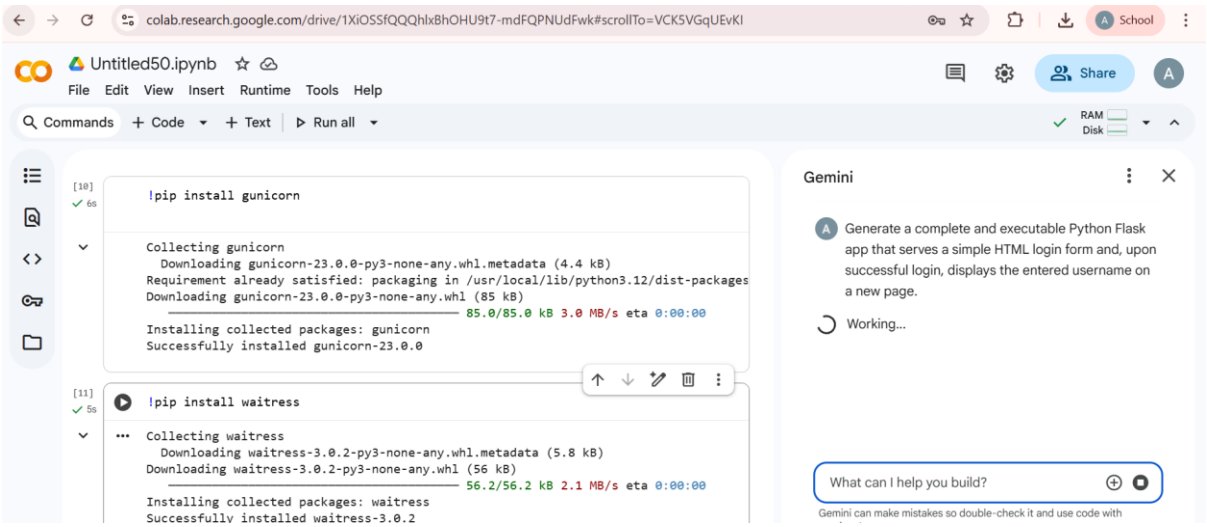
➢ TASK DISCRIPTION-4:

➢ Python Backend Integration-
Task: Ask AI to generate a Flask app that serves the HTML form (Task #3) and prints the username on successful login.

PROMPT:

Generate a complete and executable Python Flask app that serves a simple HTML login form and, upon successful login, displays the entered username on a new page.

CODE With OUTPUT:



ommands + Code + Text ▷ Run all

RAM Disk

[13] ✓ 0s

```python
if __name__ == '__main__':
    # This is for running the app directly with Flask's built-in server
    # For production, use a WSGI server like Gunicorn or Waitress
    # app.run(debug=True)
    pass # We will use waitress or gunicorn to serve the app later
```

```
Writing app.py
```

[8] ✓ 0s

```python
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
```

Commands + Code + Text ▷ Run all

RAM Disk

[8] ✓ 0s

```python
@app.route('/success/<username>')
def success(username):
    return render_template('success.html', username=username)

if __name__ == '__main__':
    # This is for running the app directly with Flask's built-in server
    # For production, use a WSGI server like Gunicorn or Waitress
    # app.run(debug=True)
    pass # We will use waitress or gunicorn to serve the app later
```

simple.html simple1.html test.h

C: > Users > deeks > success.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Success</title>
</head>
<body>
    <h2>Welcome, {{ username }}!</h2>
    <p>You have successfully logged in.</p>
</body>
</html>
```

Success

File C:/Users/deeks/success.html

Welcome, {{ username }}!

You have successfully logged in.

---Thank You---