



LORDS INSTITUTE OF ENGINEERING AND TECHNOLOGY

(UGC AUTONOMOUS INSTITUTION)

Approved by AICTE / Affiliated to OU / Estd. 2003 / Accredited with 'A' grade by NAAC
All Eligible UG Programs are Accredited by NBA

Laboratory : Full stack Web Development lab		Department : CSE	
Roll No.: 160923733152	Class : 04 - A2(C)	Semester : 07	Academic Year : 26-27

INDEX

S.No.	Name of the Experiment	Date	Page No.	Marks	Faculty Sign. with Date
01	Personal Portfolio Page using HTML		1-4		
02	styling the portfolio with CSS		5-7		
03	Javascript Basics - operators, control flow, arrays, functions, strings.		8-11		
04	DOM manipulation and events		12-14		
05	contact form validation using regex and exception handling		15-17		
06	Fetch API and async and await		18-20		
07	Rebuilding the portfolio with react.js		21-25		

EXPERIMENT-01

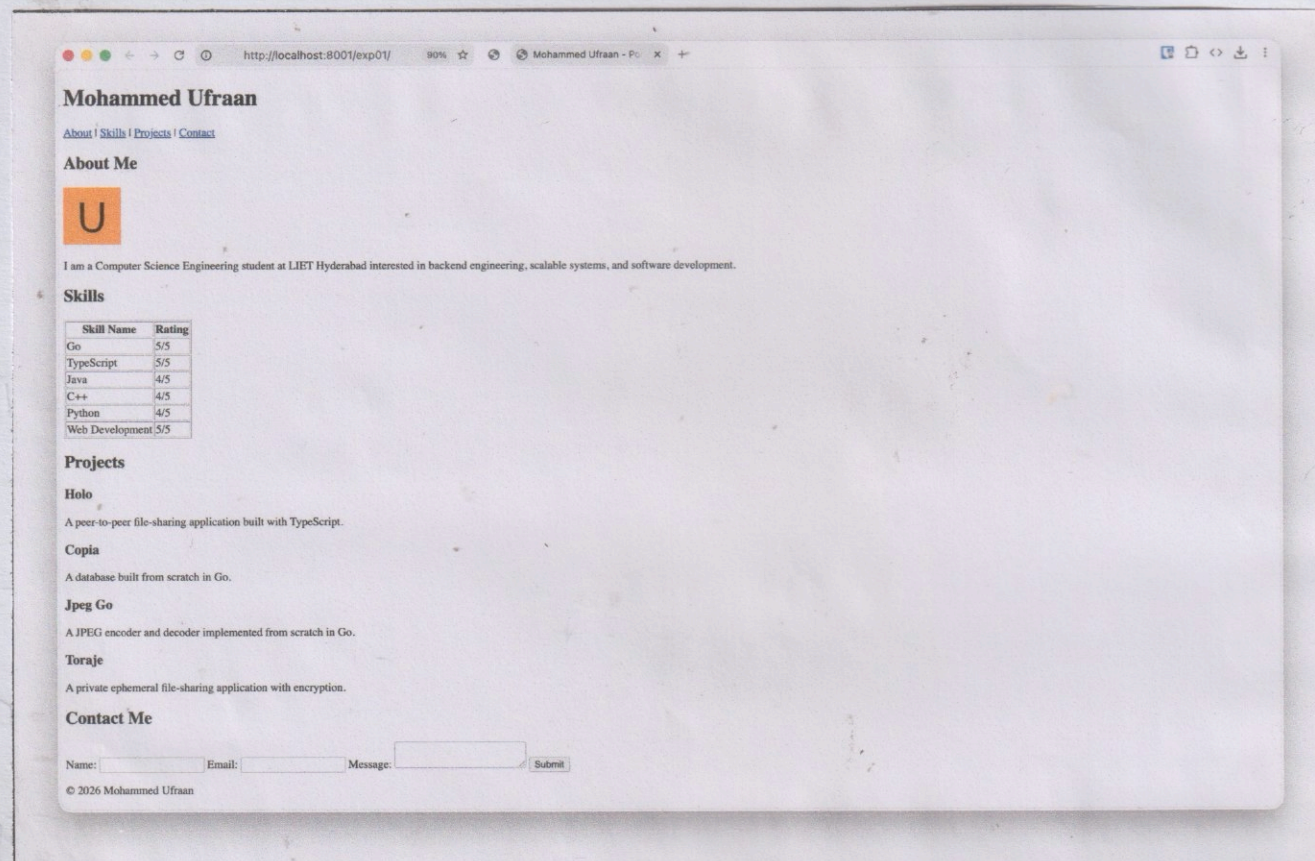
PERSONAL PORTFOLIO PAGE USING HTML

Aim:

To design and develop a personal portfolio web page using HTML, incorporating semantic tags, an image, a form, and a table to display skills and ratings.

Algorithm:

1. Create a new HTML5 boilerplate file (index.html) with the DOCTYPE declaration, html, head, and body elements.
2. Add metadata in the head section: title, character set meta tag, and viewport meta tag.
3. Build a header containing the student's name and a nav element with anchor links to page sections.
4. Create an about section with a profile image tag and a short bio paragraph.
5. Create a skills section containing a table with columns for skill name and rating.
6. Create a projects section using article elements as placeholders for each project.
7. Create a contacts section with a form containing a text input for name, email, textarea, submit button.
8. Add a footer with copyright text.
9. Validate the HTML using the W3C markup validator.



```

<section id="skills">
  <h2>Skills </h2>
  <table border="1">
    <tr><th>Skill Name</th><th>Rating</th></tr>
    <tr><td>Go</td><td>4/5</td></tr>
    <tr><td>Typescript</td><td>4/5</td></tr>
    <tr><td>C++</td><td>3/5</td></tr>
  </table>
</section>

<section id="projects">
  <h2>Projects </h2>
  <article>
    <h3>Holo </h3>
    <p>P2P file sharing application </p>
  </article>
  <article>
    <h3>Jpeg Go </h3>
    <p>Jpeg encoder and decoder in go </p>
  </article>
</section>

<section id="contact">
  <h2>Contact Me </h2>
  <form>
    <label for="name">Name: </label>
    <input type="text" id="name" required>
    <label for="email">Email: </label>
    <input type="email" id="email" required>
    <label for="message">Message </label>
    <input type="button" value="Submit" />
    <textarea id="message" </textarea>
  </form>

```



```
<button type="submit">Submit </button>
</form>
</section>
</main>
<footer>
  <p> ©copy; 2026 Mohammed Uraan </p>
</footer>
</body>
</html>
```


EXPERIMENT-02

STYLING THE PORTFOLIO WITH CSS (BOX MODEL, FLEXBOX, GRID)

Aim:

To style the personal portfolio page using CSS, applying the Box Model, Flexbox for layout, and CSS grid for a responsive project gallery, using media queries for responsiveness.

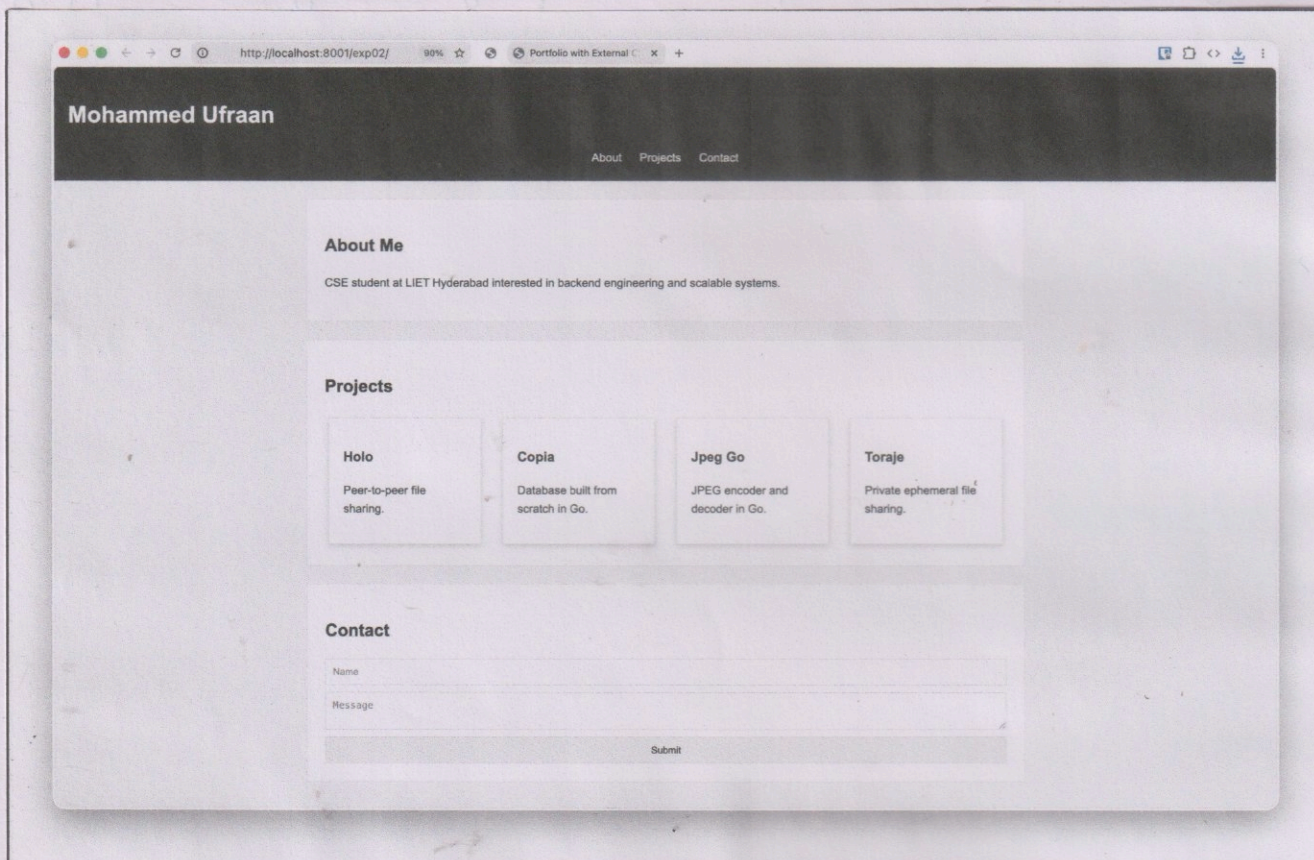
Algorithm:

1. Create a styles.css file and link it to index.html using a link tag.
2. Apply a universal box sizing: border-box rule to normalise the box model across all elements.
3. Style the body with a base font-family, colour scheme, and line-height.
4. Style the nav using display: flex along with justify-content and align-items to align nav links horizontally.
5. Style the projects section container using display: grid to create a responsive gallery.
6. Add padding, margin, and border styling to each card project, and use box-shadow for depth.
7. Style the contact form using Flexbox to stack labels.
8. Add a media query to collapse the navigation and switch the gallery to a single column on small screen.
9. Test the page at multiple viewport width using browser's responsive design mode.

EXPERIMENT-02

STYLING THE PORTFOLIO WITH

CSS (BOX MODEL, FLEXBOX, GRID)



Date :

Page No. : 06

Source Code:

```

* {
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  color: #222;
  background: #f5f5f5;
  line-height: 1.6;
  margin: 0;
}

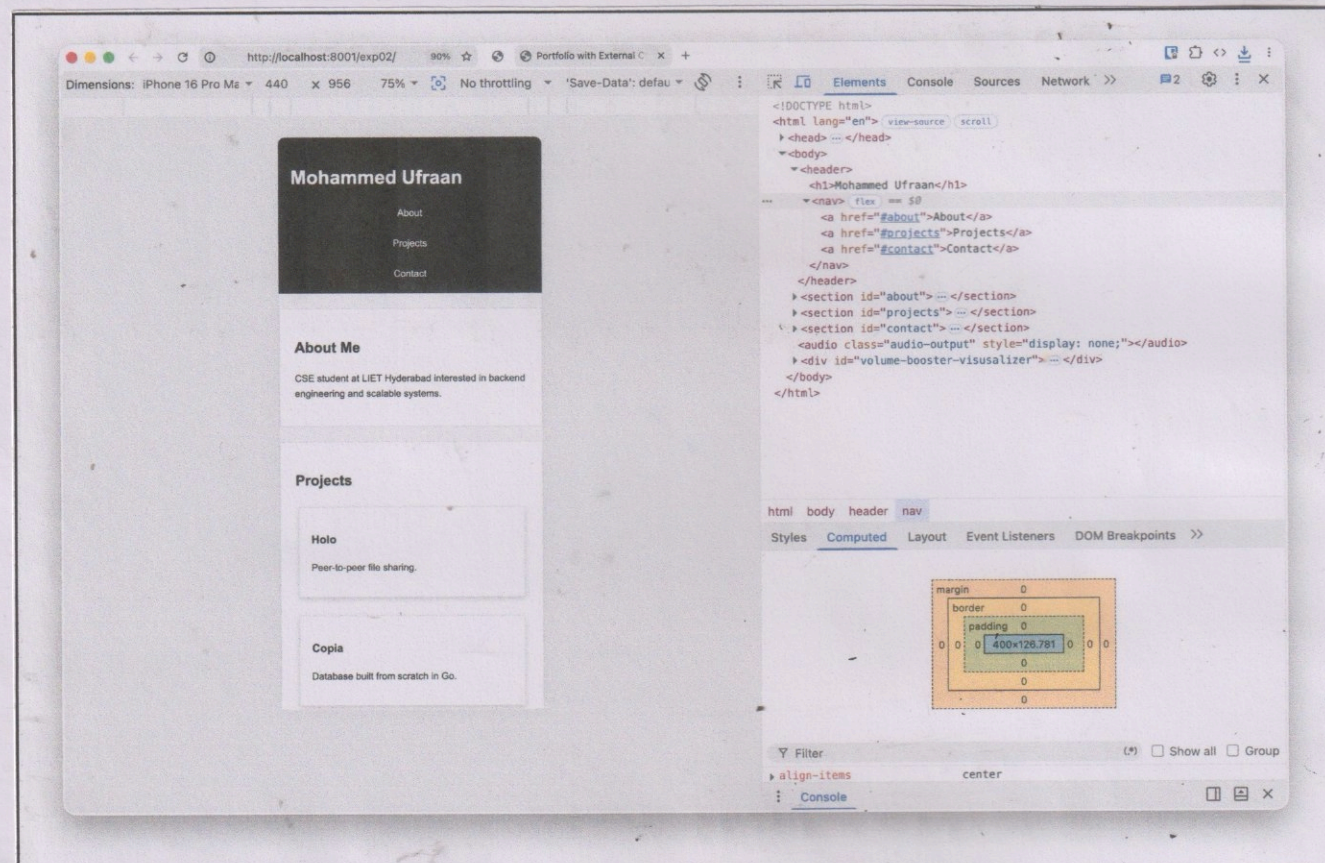
header {
  padding: 20px;
  background: #222;
  color: white;
}

nav {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 25px;
}

nav a {
  color: white;
  text-decoration: none;
}

section {
  max-width: 1000px;
  margin: 25px auto;

```

Date :

Page No. : 07

```
padding: 25px;
background: white;
border: 1px solid #ddd;
```

```
}
```

```
• project-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit,
    minmax(220px, 1fr));
  gap: 20px;
```

```
}
```

```
• project-grid article {
  padding: 20px;
  margin: 5px;
  border: 1px solid #ddd;
  box-shadow: 0 3px 8px #ccc;
```

```
}
```

```
form {
  display: flex;
  flex-direction: column;
  gap: 10px;
```

```
}
```

```
input, textarea, button {
  padding: 10px;
  border: 1px solid #ccc;
```

```
}
```

```
@media (max-width: 768px) {
  nav { flex-direction: column }
  • project-grid { grid-template-columns: 1fr; }
```

```
}
```


EXPERIMENT-03

JAVASCRIPT BASICS - OPERATORS, CONTROL FLOW, ARRAYS, FUNCTIONS, STRINGS

Aim:

To explore core javascript fundamentals: operators, control flow, arrays, functions, and string manipulation through standalone exercises.

Algorithm:

1. Create a script.js and link it to a test HTML page using a script tag with a defer attribute.
2. Declare variables and demonstrate arithmetic, comparison, and logical operators.
3. Write an if-else statement and a switch statement to demonstrate branching logic, such as a score.
4. Write a for-loop and a while loop for iteration.
5. Demonstrate push, pop, map, filter, and forEach array methods.
6. Write a function declaration and an arrow function that perform the same task.
7. Demonstrate string manipulation: concatenation, template literals, split, trim, toUppercase and slice.
8. Combine the above into a short script that processes an array of skill objects and logs a formatted summary string for each.
9. Test each snippet in the browser console.


```

Terminal - 82x38 - 11
ufraan@CS298 fswd-lab-exps % bun exp03/script.js
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
Comparison: true false
Logical: true
Grade: B
Very Good
For: 1
For: 2
For: 3
For: 4
For: 5
While: 1
While: 2
While: 3
While: 4
While: 5
After push: [ "Holo", "Copla", "Jpeg Go", "Toraje" ]
After pop: [ "Holo", "Copla", "Jpeg Go" ]
Map: [ "HOLO", "COPIA", "JPEG GO" ]
Filter: [ "Jpeg Go" ]
Project: Holo
Project: Copla
Project: Jpeg Go
Average: 4.333333333333333
Arrow Average: 4.333333333333333
Concatenation: Hello, Mohammed Ufraan
Template Literal: Welcome, Mohammed Ufraan
Trim: Mohammed Ufraan
Uppercase: MOHAMMED UFRAAN
Split: [ "Mohammed", "Ufraan" ]
Slice: Mohammed Ufraan
Go: 5/5
TypeScript: 5/5
Java: 4/5
C++: 4/5

```

Date :

Page No. : 09

Source Code :

```

let a = 10, b = 5;
const name = "Mohammed Ufraan";
console.log("Addition: ", a+b);
console.log("Subtraction: ", a-b);
console.log("Multiplication: ", a*b);
console.log("Division: ", a/b);
console.log("Comparison: "; a>b, a===b);
console.log("Logical: ", a>5 && b<10);

```

```

let score = 85;
if (score >= 90)
    console.log("Grade: A");
else if (score >= 75)
    console.log("Grade: B");
else if (score >= 60)
    console.log("Grade: C");
else
    console.log("Grade: F");

```

```

let grade = "B";
switch (grade) {
    case "A": console.log("Excellent"); break;
    case "B": console.log("Very Good"); break;
    case "C": console.log("Good"); break;
    default: console.log("Needs improvement");
}

```



```
for (let i=1; i<=5; i++) {  
  console.log("For: ", i);  
}  
let i=1;  
while (i<=5) {  
  console.log("While: " i);  
  i++;  
}  
let projects = ["Holo", "Copia", "Jpeg Cro"];  
projects.push("Torajie");  
console.log("After push: ", projects);  
projects.pop();  
console.log("After pop: ", projects);  
  
let upperProjects = projects.map(  
  project => project.toUpperCase()  
);  
console.log("Map: ", upperProjects);  
  
let filteredProjects = projects.filter(  
  project => project.length > 5  
);  
console.log("Filter: ", filteredProjects);  
projects.forEach(project => console.log(  
  "Projects: ", project));  
function average(a, b, c) {  
  return (a + b + c) / 3;  
}
```



```
const avgArrow = (a, b, c) => (a + b + c) / 3;  
console.log("Average: ", average(4, 5, 4));  
console.log("Arrow avg: ", avgArrow(4, 5, 4));
```

```
let text = "Mohammed Ufraan";  
console.log("Concatenation: ", "Hello, " + name);  
console.log("Template Literal: Welcome, ${name}");  
console.log("Trim: ", text.trim());  
console.log("Uppercase: ", text.toUpperCase());  
console.log("Split: ", text.trim().split(" "));  
console.log("Slice: ", text.trim().slice(0, 15));
```

```
const skills = [  
  { name: "Go", rating: 5 },  
  { name: "Typescript", rating: 4 },  
  { name: "C++", rating: 3 }  
];
```

```
skills.forEach(skill => {  
  console.log(`${skill.name}: ${skill.rating}/5`);  
});
```


EXPERIMENT-04

DOM MANIPULATION AND EVENTS

Aim:

To bring the static portfolio to life using DOM manipulation - dynamically adding and removing project cards, switching themes, and responding to the user events.

Algorithm:

1. Select key DOM elements such as project gallery container and theme toggle using query selector.
2. Create a JS array of project objects containing a title, description, and image URL.
3. Write a renderProjects function that loops through the array, dynamically creates elements for each project, and appends to gallery container.
4. Add a remove button to each generated card and attach a click event listener to remove card.
5. Add project control that pushes object to array.
6. Implement a theme toggle button to toggle a dark-theme class on the body element.
7. Store selected theme to persist across sessions.
8. Test all interactions in the browser.

Source Code:

HTML:

```
<body>
```

```
  <h1>Mohammed Ufraan - Projects</h1>
```


Mohammed Ufraan - Projects

Toggle Theme Add Project Search projects

Holo

Peer-to-peer file sharing.

Remove

Jpeg Go

JPEG encoder and decoder in Go.

Remove

Toraje

Private file-sharing application.

Remove

Mohammed Ufraan - Projects

Toggle Theme Add Project Search projects

Holo

Peer-to-peer file sharing.

Remove

Jpeg Go

JPEG encoder and decoder in Go.

Remove

Toraje

Private file-sharing application.

Remove

New Project

A newly added project.

Remove

Date :

Page No. : 13

```
<button id="themeToggle">Toggle theme </button>
<button id="addProject">Add project </button>
<input id="search" placeholder="search">
<div id="gallery"></div>
```

```
</body>
```

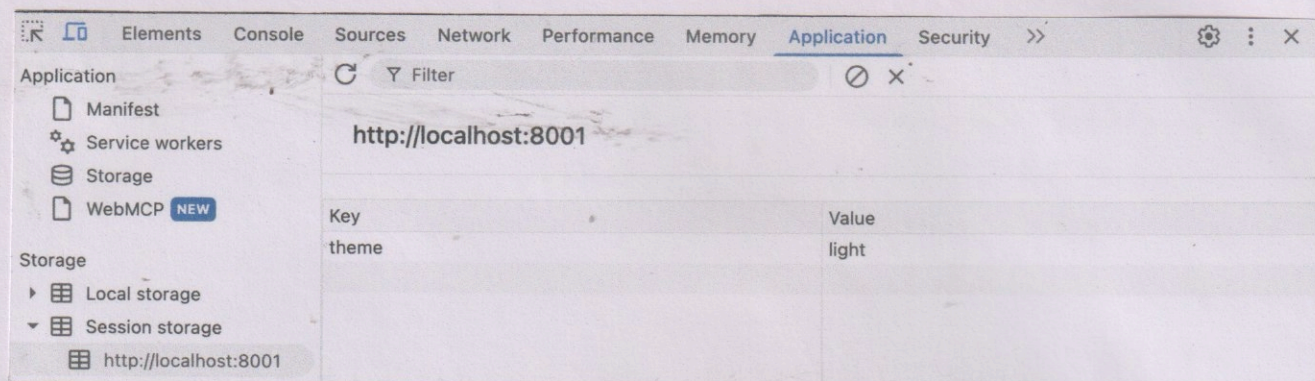
~~script~~

JS:

```
const gallery = document.querySelector("#gallery");
const themeToggle = document.querySelector("#themeToggle");
const addProject = document.querySelector("#addProject");
const search = document.querySelector("#search");
```

```
let projects = [
  {title: "Holo", description: "P2P file sharing"},
  {title: "Jpeg Go", description: "JPEG encoder/decoder"},
];
```

```
function renderProjects(list = projects) {
  gallery.innerHTML = "";
  list.forEach((project, index) => {
    const card = document.createElement("article");
    card.className = "card";
    card.innerHTML = `<h2>${project.title}</h2>
    <p>${project.description}</p>
    <button>Remove</button>`;
    card.querySelector("button").onclick = () => {
      projects.splice(index, 1);
      renderProjects();
    };
  });
}
```

Date :

Page No. : 14

```

    gallery.appendChild(card);
  });
}
addProject.onClick = () => {
  projects.push({
    title: "New project",
    description: "newly added project"
  });
  renderProjects();
};
themeToggle.onClick = () => {
  document.body.classList.toggle("dark-theme");
  sessionStorage.setItem(
    "theme",
    document.body.classList.contains("dark-theme") ?
    "dark" : "light"
  );
};
if (sessionStorage.getItem("theme") !== "dark")
  document.body.classList.add("dark-theme");
search.oninput = () => {
  const query = search.value.toLowerCase();
  renderProjects(projects.filter(p =>
    p.title.toLowerCase().includes(query)
  ));
};
renderProjects();

```


EXPERIMENT-05

CONTACT FORM VALIDATION USING REGEX AND EXCEPTION HANDLING

Aim:

To validate the contact form built in exp 01 using javascript regular expressions and implement try-catch based exception handling.

Algorithm:

1. Select the contact form and its input fields using DOM methods.
2. Attach a submit event listener to the form and prevent the default submission behaviour.
3. Define regex patterns: a name pattern allowing letters and spaces, email pattern.
4. Write a validation function that returns true or false based on testing input against pattern.
5. For each field, display inline error image if validation fails.
6. On successful validation, show a confirmation message.
7. Test with valid inputs, invalid inputs, and edge cases such as empty fields or malformed email to confirm correct handling.

Contact Mohammed Ufraan

Name:
Ufraan

Email:
naarfu@gmail.com

Message:
Good evening!

Submit

Form submitted successfully!

Date :

Page No. : 16

Source code:

HTML:

```
<form id="contactForm">
  <label> Name: </label>
  <input type="text" id="name">
  <span id="nameError" class="error"></span>
  <label> Email: </label>
  <input type="text" id="email">
  <span id="emailError" class="error"></span>
  <label> Message </label>
  <textarea id="message"></textarea>
  <span id="messageError" class="error"></span>
  <button type="submit"> Submit </button>
  <p id="formError" class="error"></p>
  <p id="success"></p>
</form>
```

JS:

```
const form = document.querySelector("#contactForm");
const name = document.querySelector("#name");
const email = document.querySelector("#email");
const message = document.querySelector("#message");
const nameRegex = /^[A-Za-z]+$ /;
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
```

```
function validate(input, regex, error, text) {
  if (!input.value.trim())
    throw new Error("Required can't be empty");
}
```


21 - 01 - 2023

123

abc

Message must contain at least 10 characters.

Submit

Date :

Page No. : 17

```

if(regex && !regex.test(input.value.trim())) {
    document.querySelector(error).textContent = text;
    return false;
}
document.querySelector(error).textContent = "";
return true;
}

form.addEventListener("submit", e => {
    e.preventDefault();
    try {
        document.querySelector("#formError").textContent = "";
        document.querySelector("#success").textContent = "";
        const validName = validate(
            name, nameRegex, "#nameError", "Enter valid name"
        );
        const validEmail = validate(
            email, emailRegex, "#emailError", "Enter valid mail"
        );
    } catch (error) {
        console.error(error);
        document.querySelector("#formError").textContent =
            "Please fill all fields correctly";
    } finally {
        console.log("Validation completed");
    }
});

```


EXPERIMENT-06

FETCH API AND ASYNC AWAIT

Aim:

To fetch live data from a public API, such as GitHub user statistics, and render it on the portfolio using the Fetch API and async/await.

Algorithm:

1. Identify a public API endpoint, such as the GitHub Users API, and inspect structure of JSON response.
2. Create an async function that accepts a username as a parameter.
3. Inside the function, use fetch with await to send the GET request and store the response.
4. Check whether the response is successful.
5. Parse response body as JSON using await, again handled as an asynchronous operation.
6. Use parsed data to update DOM elements.
7. Wrap fetch and parsing logic in a try-catch block to handle errors.
8. Call the async function when the page loads or when a button is clicked.
9. Test with valid and invalid usernames, and simulate a network failure to verify error handling works correctly.

Date :

Page No. : 19

Source Code:

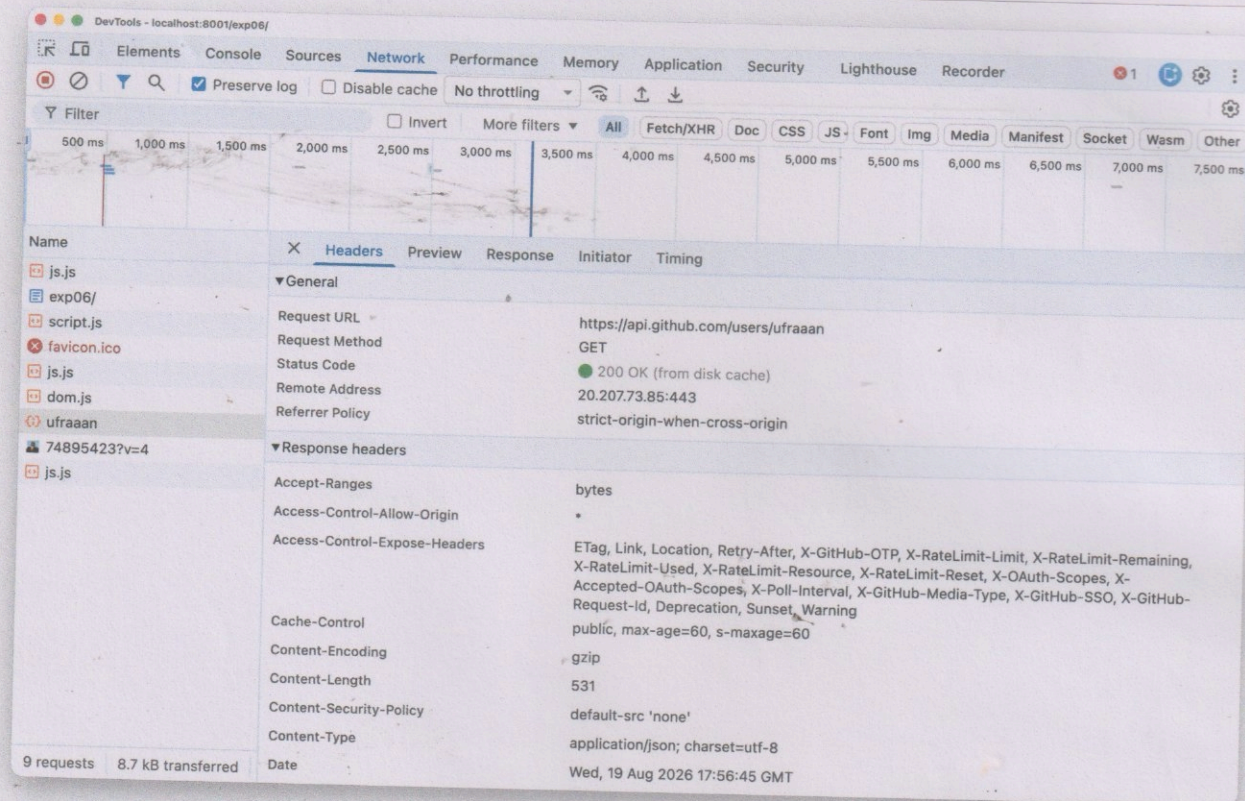
HTML:

```
<body>
  <h1>Ufraan - GitHub </h1>
  <img id="githubAvatar" width="100">
  <p id="githubStatus">Loading... </p>
  <p id="followers"></p>
  <p id="repos"></p>
</body>
```

JS:

```
const username = "ufraan";
async function fetchGithubUser(username) {
  const status = document.querySelector("#githubStatus");
  const avatar = document.querySelector("#githubAvatar");
  const followers = document.querySelector("#followers");
  const repos = document.querySelector("#repos");

  status.textContent = "Loading";
  try {
    const response = await fetch(
      'https://api.github.com/users/${username}');
    if (!response.ok)
      throw new Error("User not found");
    const data = await response.json();
```

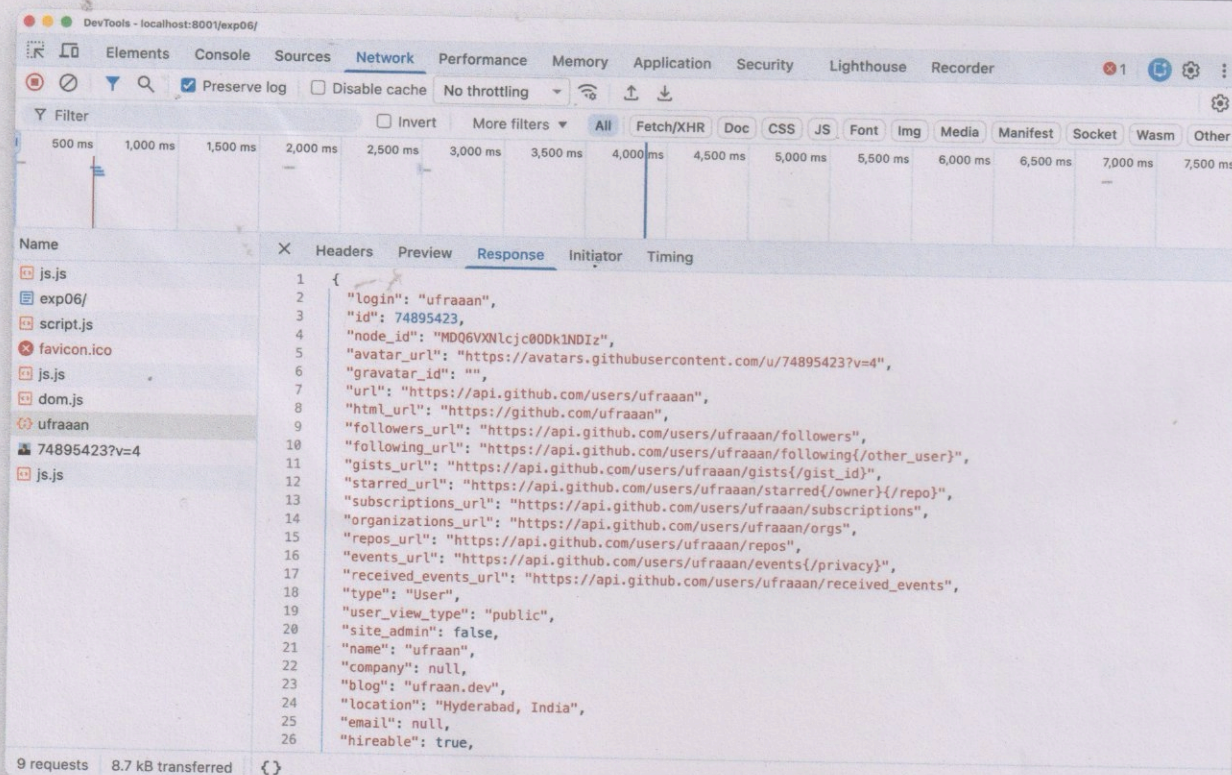
Date :

Page No. : 20

```

avatar.src = data.avatar_url;
status.textContent = `GitHub: ${data.login}`;
followers.textContent = `Followers: ${data.followers}`;
repos.textContent = `Public repos: ${data.public_repos}`;
} catch (error) {
  console.error(error);
  status.textContent = "Failed to load data";
}
}

```



EXPERIMENT - 07

REBUILDING THE PORTFOLIO WITH REACT.JS

Aim:

To rebuild the personal portfolio as a React application using functional components, props, state (useState), and effects (useEffect).

Algorithm:

1. Scaffold a new React project.
2. Plan and create a component hierarchy: an App root component with child components such as Header, About, SkillsTable, ProjectGallery, ProjectCard, and ContactForm.
3. Convert the static HTML sections into JSX within their respective functional components.
4. Pass relevant arrays as props to components, rendering with unique key for each item.
5. Use useState hook to manage the theme.
6. Use useEffect hook to call GitHub fetch logic once component mounts.
7. Style components using CSS.
Run the development server and verify each component renders and updates correctly as props and state change.


```

Terminal - 84x13 - 1
ufraan@CS298 exp07 % npm run dev

> exp07@0.0.0 dev
> vite

VITE v5.4.21 ready in 79 ms

➔ Local:   http://localhost:5173/
➔ Network: use --host to expose
➔ press h + enter to show help

```

Date :

Page No. : 22

Source Code:

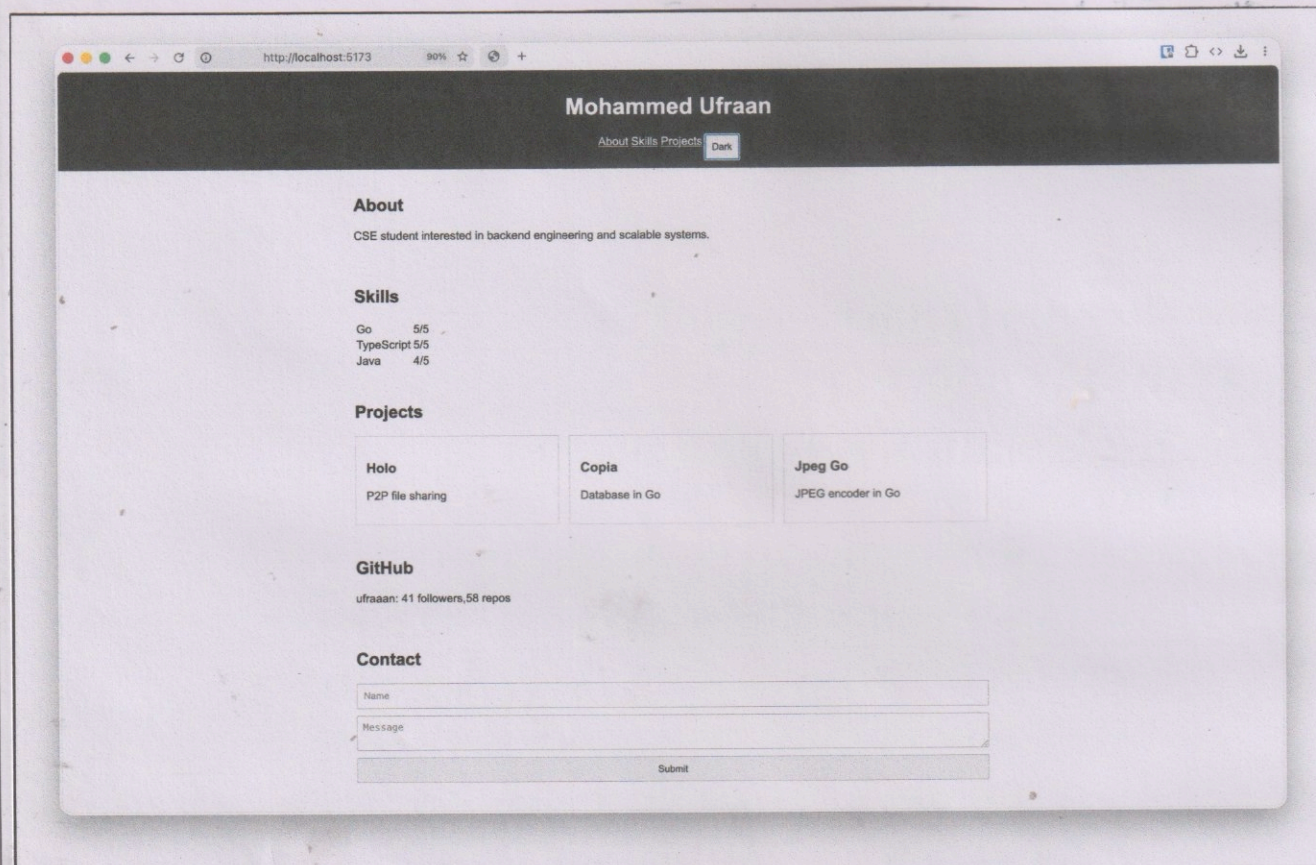
```

src/App.jsx
const skills = [
  ["Go", 5], ["Typescript", 4], ["Java", 3]
];
const projects = [
  ["Holo", "P2P file sharing"],
  ["Copia", "Database in Go"],
  ["Jpeg go", "JPEG encoder and decoder"]
];

function Header({dark, toggle}) {
  return <header>
    <h1>Mohammed Ufraan </h1>
    <nav>
      <a href="#about">About </a>
      <a href="#skills">Skills </a>
      <a href="#projects">Projects </a>
      <button onClick={toggle}>
        {dark ? "Light" : "Dark"}
      </button>
    </nav>
  </header>;
}

function Skills({data}) {
  return <section id="skills">
    <h2>Skills </h2>
    <table><tbody>
      {data.map(([,r]) =>
        <tr key={r}><td>{r}</td><td>{r}/5</td></tr>
      )}
    </tbody></table>
  </section>;
}

```

Date :

Page No. : 23

```

</tbody></table>
</section>;
}

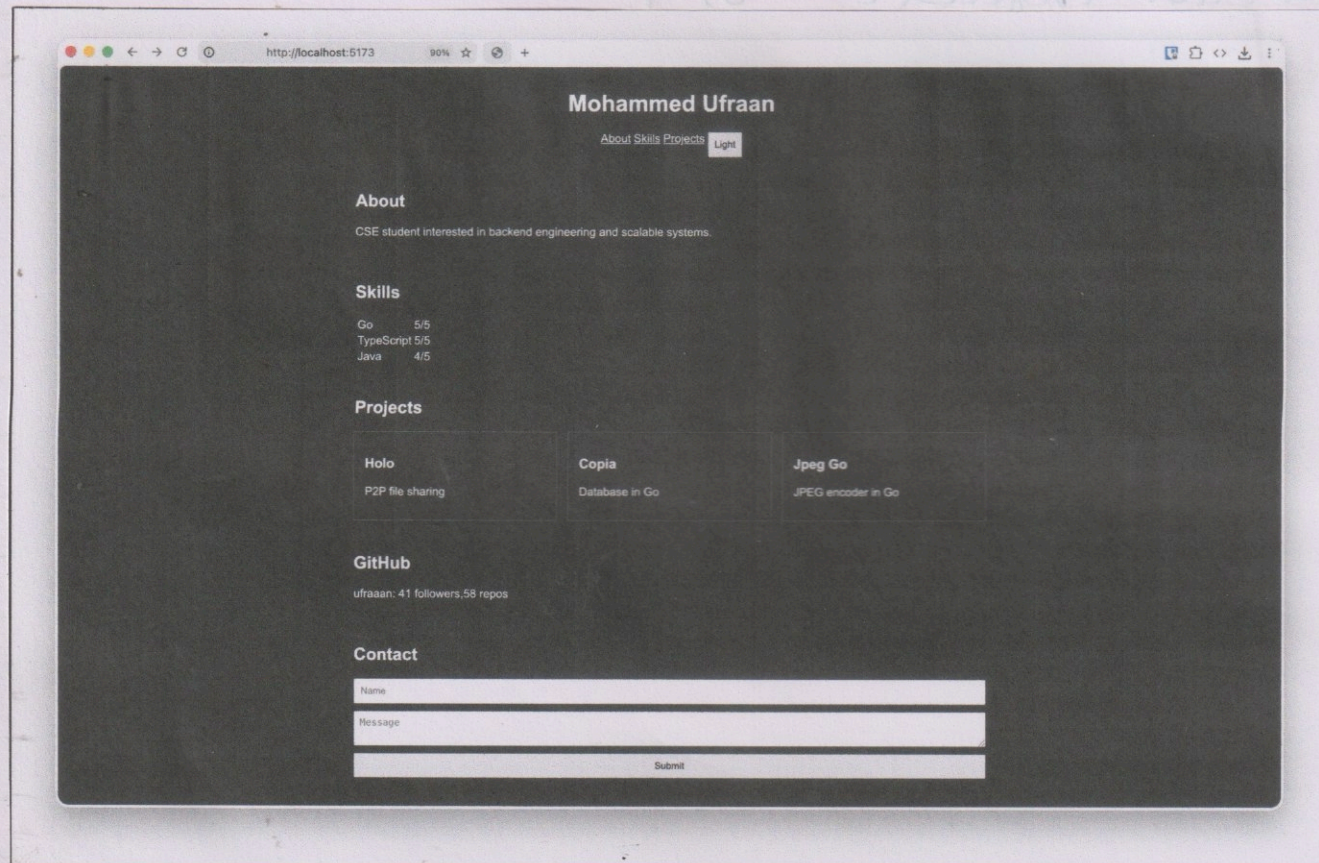
function Projects({data}) {
  return <section id="projects">
    <h2>Projects </h2>
    <div className="projects">
      {data.map(([,d]) =>
        <article key={n}>
          <h3>{n}</h3>
          <p>{d}</p>
        </article>
      )}
    </div>
  </section>;
}

```

```

function Contact() {
  const [name, Setname] = useState("");
  const [msg, Setmsg] = useState("");
  function submit(e) {
    e.preventDefault();
    alert(name ?? msg ? "submitted" : "fill all");
  }
  return <section id="contact">
    <h2>Contact </h2>
    <form onSubmit={submit}>

```

Date :

Page No. : 24

```
<form onSubmit={handleSubmit}>
  <input value={name} placeholder="Name"
    onChange={e => setName(e.target.value)} />
  <textarea value={msg} placeholder="msg"
    onChange={e => setMsg(e.target.value)} />
  <button>Submit </button>
</form>
```

```
</section>;
```

```
}
```

```
function GitHub() {
  const [user, setUser] = useState(null);
  useEffect(() => {
    fetch("https://api.github.com/users/ufraan")
      .then(r => r.json())
      .then(setUser);
  }, []);
```

```
return <section>
  <h2>GitHub </h2>
  {user ? <p>{user.login} : {user.followers}
    followers, {user.public_repos} repos </p> : }
</section>;
```

```
}
```

```
export default function App() {
  const [dark, setDark] = useState(false);
  return <div className={dark ? "dark" : ""}>
    <Header dark={dark} toggle={e => setDark(!dark)} />
```



```
<section id="about">
  <h2>About </h2>
  <p>CS undergrad interested in backend
    engineering and systems. </p>
  </section>
  <Skills data={skills}/>
  <Projects data={projects}/>
  <GitHub />
  <Contact />
</div> ;
}
```

```
src/index.css :
nav a {color: white;}
.projects {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 15px;
}
article {
  padding: 15px;
  border: 1px solid #aaa;
}
.dark {
  background: #222;
  color: white;
  min-height: 100vh;
}
```