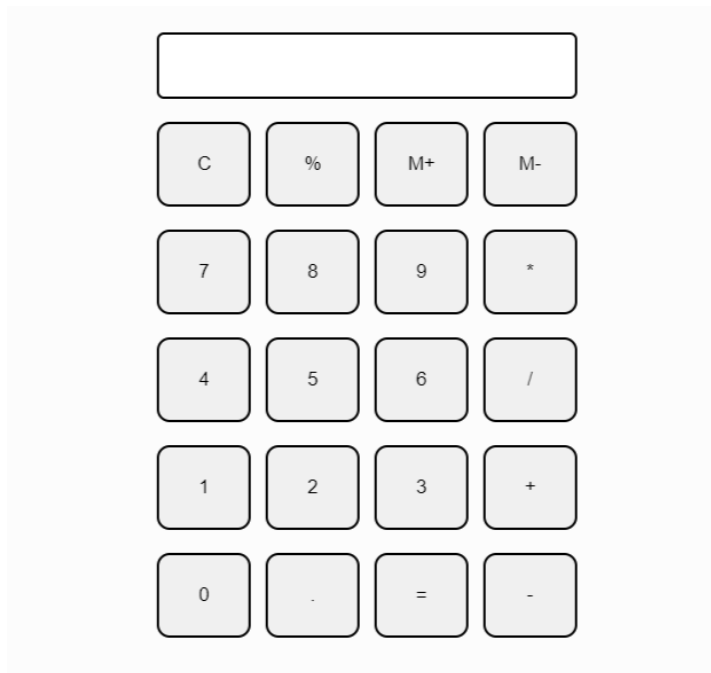


Hands – On Lab

Workshop 4

SIMPLE CALCULATOR

Create a UI of calculator using HTML and CSS and perform addition, subtraction, multiplication and division operations. Also handle the errors and exceptions. While clicking on C button, it should clear the textbox.



Git Hub Link:

[https://github.com/Navn-eet/Internet-Software-Architecture/tree/main/Navneet Workshop/Workshop4/Calculator](https://github.com/Navn-eet/Internet-Software-Architecture/tree/main/Navneet%20Workshop/Workshop4/Calculator)

DIGITAL CLOCK

Create a Digital Clock using setInterval and Date function in JavaScript.

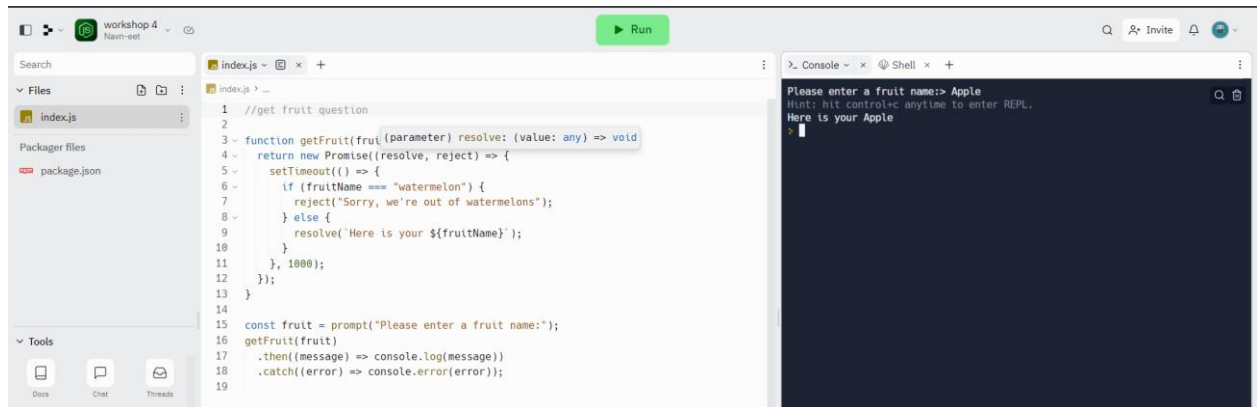
Note: Date object can be used to get the date, time, hours and seconds which can be updated using setInterval (). Try to keep the UI good looking and different from each other.

Git Hub Link:

[https://github.com/Navn-eet/Internet-Software-Architecture/tree/main/Navneet Workshop/Workshop4/Clock](https://github.com/Navn-eet/Internet-Software-Architecture/tree/main/Navneet%20Workshop/Workshop4/Clock)

ASYNCHRONOUS JAVASCRIPT

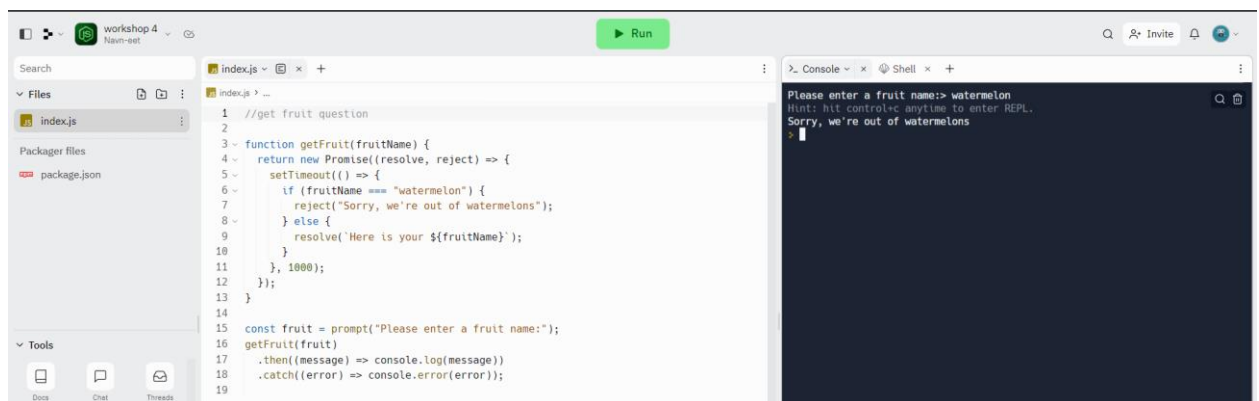
- a. Create a function called **getFruit** that takes in a fruit name as a parameter and returns a Promise that resolves after 1 second with a message saying "Here is your [fruit]". If the fruit name is "watermelon", the Promise should reject after 2 seconds with an error message saying "Sorry, we're out of watermelons"



```
1 //get fruit question
2
3 function getFruit(fruitName) {
4   return new Promise((resolve, reject) => {
5     setTimeout(() => {
6       if (fruitName === "watermelon") {
7         reject("Sorry, we're out of watermelons");
8       } else {
9         resolve("Here is your " + fruitName);
10      }
11    }, 1000);
12  });
13 }
14
15 const fruit = prompt("Please enter a fruit name:");
16 getFruit(fruit)
17   .then((message) => console.log(message))
18   .catch((error) => console.error(error));
19
```

Console output:

```
Please enter a fruit name: Apple
Here is your Apple
```

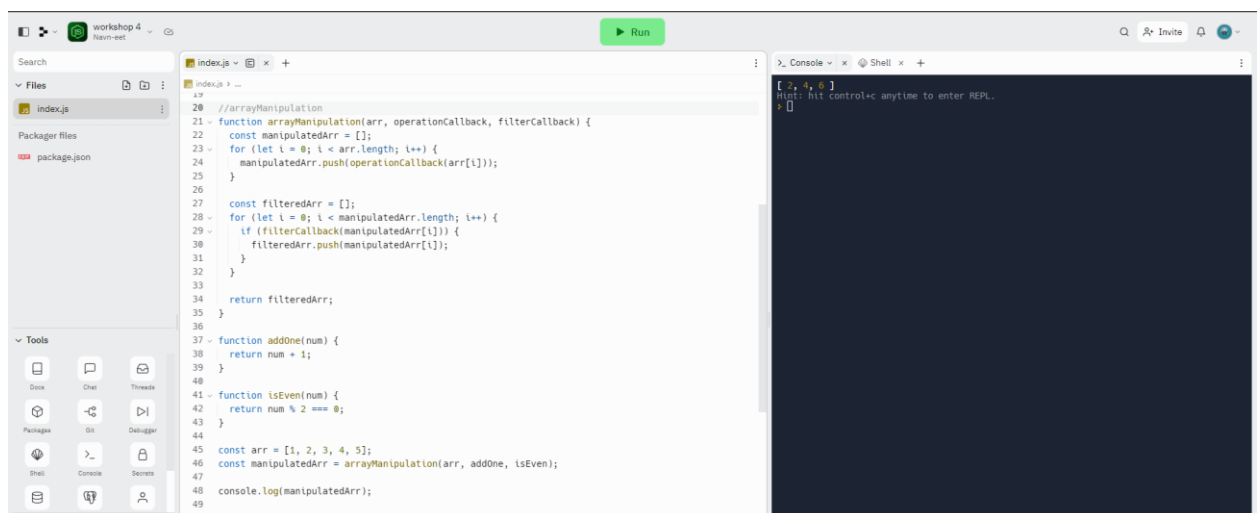


```
1 //get fruit question
2
3 function getFruit(fruitName) {
4   return new Promise((resolve, reject) => {
5     setTimeout(() => {
6       if (fruitName === "watermelon") {
7         reject("Sorry, we're out of watermelons");
8       } else {
9         resolve("Here is your " + fruitName);
10      }
11    }, 1000);
12  });
13 }
14
15 const fruit = prompt("Please enter a fruit name:");
16 getFruit(fruit)
17   .then((message) => console.log(message))
18   .catch((error) => console.error(error));
19
```

Console output:

```
Please enter a fruit name: watermelon
Sorry, we're out of watermelons
```

- b. Create a function called **arrayManipulation** that takes in an array of numbers and two callback functions as parameters. The first callback function should perform an operation on each element in the array, and the second callback function should filter the resulting array based on a condition. The function should return the filtered array.

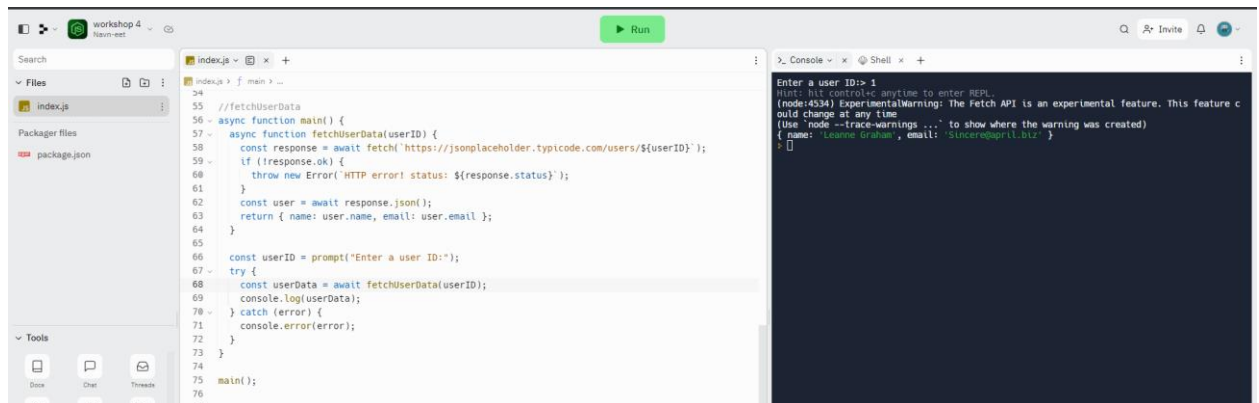


```
20 //arrayManipulation
21 function arrayManipulation(arr, operationCallback, filterCallback) {
22   const manipulatedArr = [];
23   for (let i = 0; i < arr.length; i++) {
24     manipulatedArr.push(operationCallback(arr[i]));
25   }
26
27   const filteredArr = [];
28   for (let i = 0; i < manipulatedArr.length; i++) {
29     if (filterCallback(manipulatedArr[i])) {
30       filteredArr.push(manipulatedArr[i]);
31     }
32   }
33
34   return filteredArr;
35 }
36
37 function addOne(num) {
38   return num + 1;
39 }
40
41 function isEven(num) {
42   return num % 2 === 0;
43 }
44
45 const arr = [1, 2, 3, 4, 5];
46 const manipulatedArr = arrayManipulation(arr, addOne, isEven);
47
48 console.log(manipulatedArr);
49
50
```

Console output:

```
[ 2, 4, 6 ]
```

- c. Create an asynchronous function called `fetchUserData` that uses `async/await` to fetch user data from a JSON API (<https://jsonplaceholder.typicode.com/users>). The function should take in a user ID as a parameter, and use that ID to fetch the user's data from the API. If the API returns an error, the function should throw an error. If the API returns the user data, the function should return an object containing the user's name and email

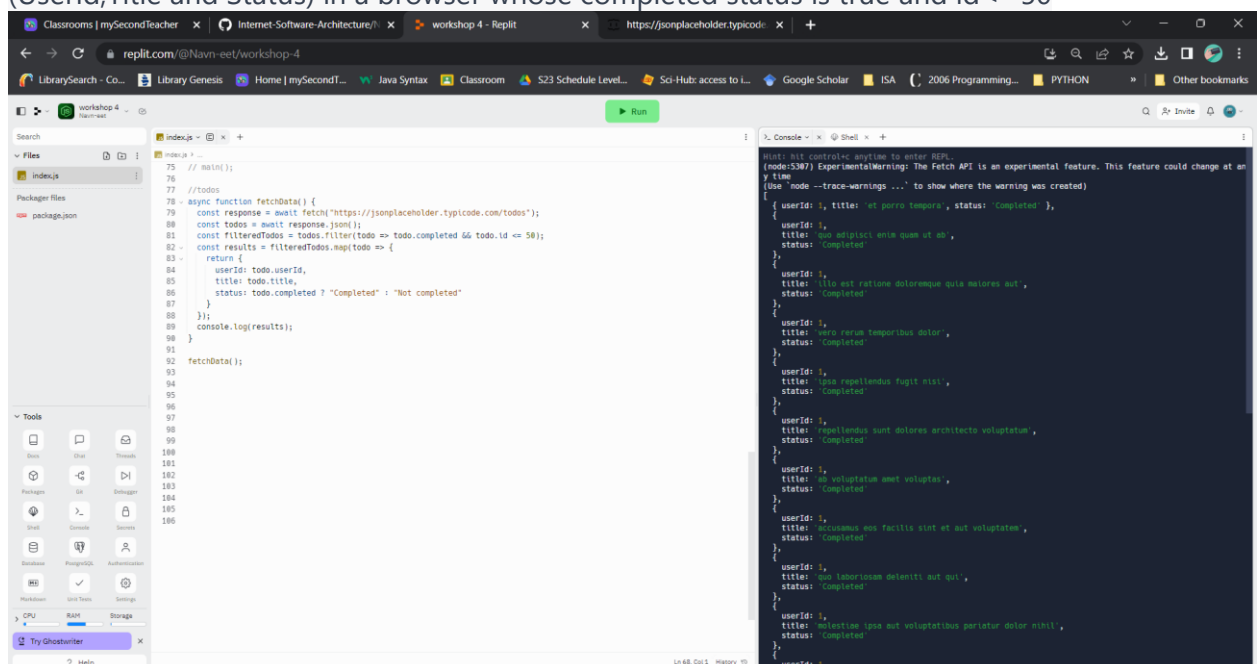


The screenshot shows a Replit workspace with a file explorer on the left, a code editor in the center, and a console on the right. The code editor contains the following JavaScript code:

```
34 //fetchUserData
35
36 - async function main() {
37   //fetchUserData
38   async function fetchUserData(userID) {
39     const response = await fetch('https://jsonplaceholder.typicode.com/users/${userID}');
40     if (!response.ok) {
41       throw new Error('HTTP error! status: ${response.status}');
42     }
43     const user = await response.json();
44     return { name: user.name, email: user.email };
45   }
46
47   const userID = prompt('Enter a user ID:');
48   try {
49     const userData = await fetchUserData(userID);
50     console.log(userData);
51   } catch (error) {
52     console.error(error);
53   }
54 }
55
56 main();
```

The console on the right shows the output of the program, including a prompt for a user ID and the resulting user data object.

- d. Fetch data from API (<https://jsonplaceholder.typicode.com/todos>) and display (UserId,Title and Status) in a browser whose completed status is true and id<=50



The screenshot shows a Replit workspace with a file explorer on the left, a code editor in the center, and a console on the right. The code editor contains the following JavaScript code:

```
75 //main
76
77 //fetchData
78 - async function main() {
79   //fetchData
80   async function fetchData() {
81     const response = await fetch('https://jsonplaceholder.typicode.com/todos');
82     const todos = await response.json();
83     const filteredTodos = todos.filter(todo => todo.completed && todo.id <= 50);
84     const results = filteredTodos.map(todo => {
85       userId: todo.userId,
86       title: todo.title,
87       status: todo.completed ? 'Completed' : 'Not completed'
88     });
89     console.log(results);
90   }
91
92   fetchData();
93 }
94
95
96
97
98
99
100
101
102
103
104
105
106
```

The console on the right shows the output of the program, displaying an array of objects containing user ID, title, and status for todos that are completed and have an ID less than or equal to 50.