

EXPIREMENT - 01

Step 1: Open browser search for jira login

Step 2 : Continue with your g-mail account or login jira

Step3: Click on jira software & select project from top menu bar then select create project from drop down menu-

Step 4: Select scrum click on template & check on create

Step 5: Give a name to your project & give a description of your project.

Step 6: Click on create.

Step 7: Select Issues from top menu bar and select issue type (this will be the default settings)

Step 8:- Give a summary to your project.

Step9:- Now write a user story in description box

Step 10: Now your study will go into the backlog to be assign and action by the propet manager, product owner or other stake holders & then click on start print.

Step 1: Click on board & select insights.

Ship 12: Click on insights & click sprint burn dawn

Step: And click on dearn more.

EXPIREMEN – 02

Create and manage product backlog using appropriate tool like jira for a shopping cart application.

→ Summary: Customer Registration functionality.

Description

As a customer

I want to have registration functionality.
So that I can successfully register

Scope

1. Build a registration page.
2. Customer validation
3. Customer should be able to change the phone number
4. It should work in all browser
5. It should also work as mobile devices

Pre-condition

Customer should have an email & phone number

Acceptance Criteria.

Scenario 1: Customer can successfully register

1. Given I am on registration page.
2. And I provide a valid customer name & phone number & I click on "sign in"
3. Then I will successfully register.

Scenario 2: Customer cannot successfully register

1. Given I am on the registration page
2. And I provide an invalid

customer name & phone number.

3.Then I will get an error message as registration failed incorrect customer name.

2.Summary: Password reset functionality.

Description

As a customer

I want to have the ability to reset my password.

So that I can regain access to my account

Scope:

Build a password reset page

Email validation for reset Ability to set a new password.

Precondition:-

Customer should have a registered email

Acceptance Criteria:

Scenario 1 : Successful password reset

Given I am on the password reset page.

And I click on "Send reset link"

Then I will receive a resent link in my email.

When I click the reset link & set a new password

Then my password is successfully updated

Scenario 2 - Unsuccessful password reset due to unregistered email.

Given I am on the password reset page.

And I provide an unregistered email

Then I will get an error message as "email not found".

3: shopping cart functionality:

Deception:

As a customer

I want to add items to my shopping cart

So that I can purchase them later.

Scope:

Build a shopping cart page. Ability to add /remove items. Cart should show total price.

Preconditions:

Customer Should be logged in

Acceptance Criteria:

Scenario1: Add item to cart.

given I am logged in

and i am on a product page when I click on "Add to cart" Then the item is added to my cart

And the cart shows the updated totel price

Scenario 2: Remove item from cart:

Given I am the shopping cart page.

And I have item's in my cart. When I click on "Remove" next to an item

Then the item is removed from my cast

And The cart shows the updated total price.

4) Summary: Order Tracking functionality.

Description:

As a customer

I want to track my order status

So that I can know my order will arrive.

Scope:

Build an order tracking page Integrate with shopping provider API

Display order status updates

Precondition:-

Customer should have an order ID

Acceptance criteria

Scenarios1 : Successful Order tracking

Given I am on the order tracking page

And I provide a valid order ID

Then I see the current status of my order

Scenario 2:- Unsuccessful order tracking device to invalid order ID.

Given I am on the order tracking page

And I provide invalid order ID Then I will get an error message as "order not found".

EXPIREMENT - 03

1. Sign Up and Log In:

Go to the Visily website and create an account.
Log in to access the design tools.

2. Start a New Project:

Click on "New Project" from the dashboard.
Choose a template or start with a blank canvas.
Name your project and select the design type (Web or Mobile).

3.. Create a Basic Layout:

Drag and drop UI components (buttons, text boxes, images) onto the canvas.
Arrange these components to build the basic structure of your design.

4. . Customize Your Design:

Adjust colors, fonts, and sizes to fit your design preferences.
Use the alignment tools to ensure everything is neatly organized.

5. Add Interactions:

Link different screens or elements to create a clickable prototype.
Test these interactions by previewing your design.

6. Review and Export:

Check your design in Preview Mode to see how it looks and works.
Make any necessary changes.
Export your design as an image, PDF, or share it as a prototype link.

EXIREMENT – 04

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Registration and Feedback Form</title>
<link rel="stylesheet" href="styles.css" />
<style>
body {
font-family: Arial, sans-serif;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
background-color: #f0f0f0;
}
#form-container {
background: lightblue;
padding: 20px;
border-radius: 8px;
box-shadow: 0 0 10px red;
width: 300px;
}
h2 {
margin-top: 0;
}
label {
display: block;
margin: 10px 0 5px;
}
input {
width: 100%;
padding: 8px;
margin-bottom: 10px;
border: 1px solid black;
border-radius: 4px;
}
```

```
}

button {
padding: 10px 15px;
border: none;
border-radius: 4px;
background-color: #28a745;
color: white;
cursor: pointer;
}
button:hover {
background-color: #218838;
}

#feedback-form {
text-align: center;
}

.hidden {
display: none;
}

</style>
</head>
<body>
<div id="form-container">
<!-- Registration Form -->
<div id="registration-form">
<h2>Registration Form</h2>
<form id="regForm">
<label for="name">Name:</label>
<input type="text" id="name" name="name" required />
<label for="email">Email:</label>
<input type="email" id="email" name="email" required />
<button type="submit">Submit</button>
</form>
</div>
<!-- Feedback Form -->
<div id="feedback-form" class="hidden">
<h2>Feedback</h2>
<p id="feedback"></p>
</div>
</div>
<script>
```

```
document.getElementById("regForm").onsubmit = function (event) {  
  event.preventDefault();  
  document.getElementById("registration-form").classList.add("hidden");  
  const { value: name } = document.getElementById("name");  
  const { value: email } = document.getElementById("email");  
  document.getElementById(  
    "feedback"  
  ).innerText = `Thank you for registering, ${name}! We have sent a confirmation  
  email to ${email}.`;  
  document.getElementById("feedback-form").classList.remove("hidden");  
};  
</script>  
</body>  
</html>
```

EXPIREMNT – 05

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Basketball Scoreboard</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Basketball Scoreboard</h1>
  <div class="scoreboard">
    <div>
      <h2>Team A</h2>
      <p id="teamA-score">0</p>
      <button onclick="increaseScore('teamA')">+1 Point</button>
    </div>
    <div>
      <h2>Team B</h2>
      <p id="teamB-score">0</p>
      <button onclick="increaseScore('teamB')">+1 Point</button>
    </div>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

Css --

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
}
```

```
.scoreboard {  
    display: flex;  
    justify-content: space-around;  
    margin-top: 20px;  
}  
  
.scoreboard div {  
    padding: 10px;  
    border: 1px solid #000;  
    width: 100px;  
}  
  
h2 {  
    font-size: 1.2em;  
}  
  
button {  
    margin-top: 10px;  
    padding: 5px;  
}
```

JS

```
let teamAScore = 0;  
let teamBScore = 0;  
  
function increaseScore(team) {  
    if (team === 'teamA') {  
        teamAScore++;  
        document.getElementById('teamA-score').textContent = teamAScore;  
    } else if (team === 'teamB') {  
        teamBScore++;  
        document.getElementById('teamB-score').textContent = teamBScore;  
    }  
}
```

EXPIREMEN – 06

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JS Objects Example</title>
</head>
<body>
  <h1>Player Info</h1>
  <div id="player-output"></div> <!-- This will display the output -->

<script>
  // Constructor function for a Player object
  function Player(name, team, points) {
    this.name = name;
    this.team = team;
    this.points = points;

    // Method to display player info
    this.displayInfo = function() {
      return ${this.name} from team ${this.team} has scored ${this.points}
    points.;
    };

    // Method to add points to the player's score
    this.addPoints = function(newPoints) {
      this.points += newPoints;
      return ${this.name} now has ${this.points} points.;
    };
  }

  // Creating new player objects
  const player1 = new Player("John", "Team A", 10);
  const player2 = new Player("Alex", "Team B", 12);

  // Get the output div
  const outputDiv = document.getElementById('player-output');
```

```
// Display player info in the div
outputDiv.innerHTML += "<p>" + player1.displayInfo() + "</p>";
outputDiv.innerHTML += "<p>" + player1.addPoints(5) + "</p>";
outputDiv.innerHTML += "<p>" + player2.displayInfo() + "</p>";
outputDiv.innerHTML += "<p>" + player2.addPoints(3) + "</p>";
</script>
</body>
</html>
```

EXPIREMNT – 07

Step-1 Install Node.js. It is used to set up Typescript on our local computer. To install Node.js on Windows, Open google chrome and search: node.js download(click on first link).

Step-2 Install Typescript. To install Typescript, enter the following command in Command prompt or node js command prompt

- npm install -g typescript

Step-3 To verify the installation was successful, enter the command \$ tsc -v in the Terminal Window.

Install Live server in VS code (Extension)

Create and run first program in Typescript

- Open visual studio code
- Create a New Folder : typescript
- Create a New File inside the folder typescript : app.ts
- Write the below code and save it (inside the file app.ts)

```
// TypeScript function to generate Fibonacci series
function fibonacci(n: number): number[] {
    let series: number[] =[0, 1];
    // Initial two numbers in the series

    for (let i = 2; i < n; i++) {
        series[i] = series[i - 1] + series[i - 2];
    // Calculate the next number
    }

    return series;
}

// Example usage
let count: number = 10;
// Define how many numbers in the Fibonacci series you want
let result: number[] = fibonacci(count);
console.log(Fibonacci series of ${count} numbers:, result);
```

Go to Terminal and type the below commands
cd typescript Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass tsc app.ts node app.js
• Observe the output

NOTE : When we use the command “tsc app.ts” it converts the program into JavaScript by creating a new file app.js.

EXPIREMENT - 08

```
App.js----- // src/App.js
import React from 'react';
import RegistrationForm from './RegistrationForm';
import './App.css'; // Ensure you have your CSS for styling

const App = () => {
  return (
    <div className="app-container">
      <h1>Registration Form</h1>
      <RegistrationForm />
    </div>
  );
};

export default App;
```

```
App.css ----- /* src/App.css */
body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: linear-gradient(135deg, #80eab0, #90bac0); /* Gradient
background */
}

.app-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
}

h1 {
  color: #042e7a; /* Dark teal color */
  margin-bottom: 20px;
}
```

```
.form-container {  
    background-color: white; /* White background for the form */  
    border-radius: 12px; /* Rounded corners */  
    padding: 30px; /* More padding */  
    box-shadow: 0 4px 20px grey; /* Enhanced shadow for a 3D effect */  
    transition: transform 0.2s; /* Smooth transform effect */  
}  
  
.form-container:hover {  
    transform: translateY(-5px); /* Lift effect on hover */  
}  
  
input {  
    width: 100%;  
    padding: 12px;  
    margin: 10px 0;  
    border: 1px solid #ccc;  
    border-radius: 6px; /* Rounded corners */  
    transition: border-color 0.3s; /* Smooth border color change */  
}  
  
input:focus {  
    border-color: #00796b; /* Dark teal border on focus */  
    outline: none; /* Remove outline */  
}  
  
button {  
    background-color: #00796b; /* Dark teal color */  
    color: white;  
    padding: 12px;  
    border: none;  
    border-radius: 6px; /* Rounded corners */  
    cursor: pointer;  
    transition: background-color 0.3s; /* Smooth background color change */  
}  
  
button:hover {  
    background-color: #004d40; /* Darker teal on hover */  
}
```

```
.success-message {  
  margin-top: 15px;  
  color: green;  
  font-weight: bold;  
}
```

```
RegistrationFrom.js ----- // src/RegistrationForm.js  
import React, { useState } from 'react';
```

```
const RegistrationForm = () => {  
  const [name, setName] = useState("");  
  const [email, setEmail] = useState("");  
  const [password, setPassword] = useState("");  
  const [message, setMessage] = useState("");  
  
  const handleSubmit = (e) => {  
    e.preventDefault();  
    // Here you can handle the registration logic (e.g., send data to an API)  
    setMessage('User registered successfully!');  
    // Clear the form fields  
    setName("");  
    setEmail("");  
    setPassword("");  
  };  
  
  return (  
    <div className="form-container">  
      <h2>Register</h2>  
      <form onSubmit={handleSubmit}>  
        <input  
          type="text"  
          placeholder="Name"  
          value={name}  
          onChange={(e) => setName(e.target.value)}  
          required  
        />  
        <input  
          type="email"  
          placeholder="Email"
```

```
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        required
      />
      <input
        type="password"
        placeholder="Password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        required
      />
      <button type="submit">Register</button>
    </form>
    {message && <p className="success-message">{message}</p>}
  </div>
);
};

export default RegistrationForm;
```

EXPIREMNT – 09

Step-1 : launch visual studio code

Step2 : open terminal

Step3: npx create-react-app myshop

Step4: create a file in src - Productlist.js

Step5 : create a file in src - AddProduct.js

Step6: link the ProductList.js and AddProduct.js in App.js

Step7: add css stylings in App.css file inside src

step8: in terminal

cd myshop

Step9: npm start

App.js ----- // src/App.js

```
import React, { useState } from 'react';
```

```
import ProductList from './ProductList';
```

```
import AddProduct from './AddProduct';
```

```
import './App.css'; // Import CSS for styling
```

```
const App = () => {
```

```
  const [products, setProducts] = useState([]);
```

```
  const addProduct = (product) => {
```

```
    setProducts([...products, product]);
```

```
  };
```

```
  const deleteProduct = (index) => {
```

```
    const newProducts = products.filter((_, i) => i !== index);
```

```
    setProducts(newProducts);
```

```
  };
```

```
  return (
```

```
    <div className="container">
```

```
      <h1>MyShop</h1>
```

```
      <AddProduct onAdd={addProduct} />
```

```
      <div className="product-box">
```

```
        <ProductList products={products} onDelete={deleteProduct} />
```

```
      </div>
```

```
    </div>
```

```
  );
```

```
};

export default App;

App.css ----- /* src/App.css */
body {
  margin: 0;
  font-family: Arial, sans-serif;
  background-color: #e0f7fa; /* Light blue background */
}

.container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
}

h1 {
  margin-bottom: 20px;
  color: #00796b; /* Dark teal color */
}

form {
  margin-bottom: 20px;
}

.product-box {
  background-color: white; /* White background for the product box */
  border: 1px solid #b2ebf2; /* Light border color */
  border-radius: 8px; /* Rounded corners */
  padding: 20px; /* Spacing inside the box */
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1); /* Shadow for a 3D effect */
}

h2 {
  margin: 0;
  color: #00796b; /* Dark teal color */
}
```

```
ProductList.js ----- // src/ProductList.js
import React from 'react';

const ProductList = ({ products, onDelete }) => {
  return (
    <div>
      <h2>Product List</h2>
      <ul>
        {products.map((product, index) => (
          <li key={index}>
            {product}
            <button onClick={() => onDelete(index)} style={{ marginLeft: '10px' }}>
              Delete
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
};

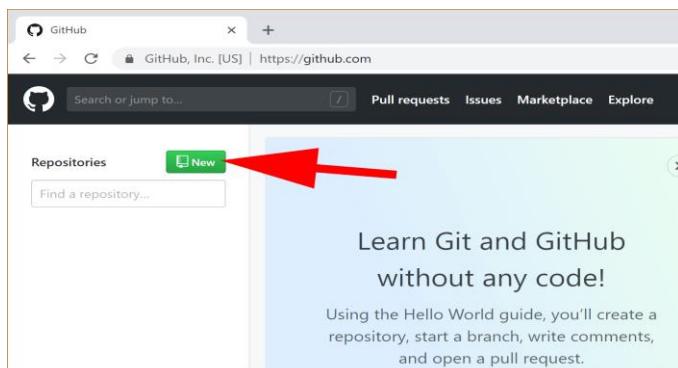
export default ProductList;
```

EXPIREMENT – 10

To create a new repository in GitHub:

1. Go to the GitHub website (<https://github.com/>) and sign in to your account.
2. Click on the "+" icon in the top right corner of the page, and select "New repository" from the dropdown menu.
3. On the "Create a new repository" page, enter a name "**GitDemo**", a short description (optional), and select whether the repository should be public or private.
4. If you want to create a repository from an existing local repository, you can select the "Import code" option and follow the prompts to import the code from your local repository.
5. If you want to create an empty repository, you can leave the "Initialize this repository with a README" option unchecked.
6. Click the "Create repository" button to create the repository.

After the repository is created, you will be taken to the repository page, where you can view the repository's code, make changes, and collaborate with others.



Cloning the Repository:

To clone a repository from GitHub:

1. Go to the repository page on the GitHub website and copy the repository's URL.
2. Open a terminal or command prompt on your local machine.
3. Change to the directory where you want to clone the repository.
4. Use the git clone command to clone the repository. For example:

```
ADMIN@DESKTOP-TQJUNQB MINGW64 ~
$ mkdir gitdemo

ADMIN@DESKTOP-TQJUNQB MINGW64 ~
$ cd gitdemo

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo
$ git init
Initialized empty Git repository in C:/Users/ADMIN/gitdemo/.git/

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git clone https://github.com/vaish12344/GitDemo.git
Cloning into 'GitDemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

5. Replace " https://github.com/vaish12344/GitDemo.git " with the URL of the repository that you copied in step 1.
6. Press Enter to start the cloning process.
7. When the cloning is complete, you will have a copy of the repository on your local machine, and you will be able to view the repository's code and make changes.

By following these steps, you should be able to clone a repository from GitHub to your local machine.

Pushing code to remote repository:

1. Enter into your local repository

```
ADMIN@DESKTOP-TQJUNQB MINGW64 ~
$ cd gitdemo
```

2. In your local repository, use the git remote add command to add the GitHub repository as a remote.

```
ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git remote add origin "https://github.com/vaish12344/GitDemo.git"
```

3. Create a new file that you want to push it into your remote repository and git status is used to check the status of the files:

```
ADMIN@DESKTOP-TQJUNQB MINGW64 ~
$ cd gitdemo
```

```

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ touch sourcecode1

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ ls
GitDemo/ sourcecode1

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    GitDemo/
      sourcecode1

nothing added to commit but untracked files present (use "git add" to track)

```

4. Add the file (file1.txt) to the repository
5. Commit the File and check the git status

```

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git add sourcecode1

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   sourcecode1
    ←

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    GitDemo/
      sourcecode1

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git commit -m "My First commit"
[master (root-commit) d848b65] My First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sourcecode1

ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    GitDemo/
      sourcecode1

nothing added to commit but untracked files present (use "git add" to track)

```

6. Now push your code using git push origin command followed by your branch name (this case : Master)

```
ADMIN@DESKTOP-TQJUNQB MINGW64 ~/gitdemo (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/vaish12344/GitDemo/pull/new/master
remote:
To https://github.com/vaish12344/GitDemo.git
 * [new branch]      master -> master
```

7. Establish connection between Remote and Local repository
8. Reload your github page (Remote Repository):