Test Plan and Test Cases

User Action No.	Test Suite	Test Case	Steps	Expected Outcome	Type
1.1	**Sign Up**	**Successful Sign Up**	1. Navigate to Sign-Up page 2. Enter valid email, password 3. Click "Sign Up"	User is signed up successfully, and a confirmation email is sent	Functional
1.2		**Sign Up with Existing Email** (Negative)	1. Enter already registered email 2. Click "Sign Up"	Error message: "Email already exists."	Functional
1.3		**Sign Up with Invalid Email** (Negative)	1. Enter invalid email format (e.g., user@com) 2. Click "Sign Up"	Error message: "Invalid email format."	Functional
1.4		**Sign Up with Weak Password** (Negative)	1. Enter weak password (e.g., 123) 2. Click "Sign Up"	Error message: "Password too weak."	Functional
2.1	**Sign In**	**Successful Sign In**	1. Navigate to Sign-In page 2. Enter valid email, password 3. Click "Sign In"	User is signed in and redirected to the dashboard	Functional
2.2		**Sign In with Incorrect Password** (Negative)	1. Enter valid email but incorrect password 2. Click "Sign In"	Error message: "Incorrect password."	Functional
2.3		**Sign In with Unregistered Email** (Negative)	1. Enter unregistered email 2. Click "Sign In"	Error message: "User does not exist."	Functional

3.1	**Onboarding OCN Nodes**	**Add Valid Nodes to Blockchain**	1. Log in 2. Enter valid Node ID (e.g., NodeID-1234), public IP(X.X.X.X where 0<=X<=255), select node type. 3. Click "ADD NODE" 4. Repeat for multiple nodes 5. Next	Nodes are added successfully and added nodes are reflected in the Preview Table.	Functional
3.2		**Invalid Node ID** (Negative)	1. Enter **invalid Node ID** format (**e.g.,NodeID1234**) and enter valid IP. Different Node ID examples for Negative Scenario: 1. NodeID1244 2.**NodeID-abcd(Not working as expected)** **3**.NodeId-1234	Verify Add Node button is disabled.	Functional
3.3		**Invalid Public IP (**Negative**)**	1. Enter valid Node ID format (e.g.,NodeID-1234) and enter **invalid IP**(X.X.X.X where 1.255<X<1000(**Not working as expected).** 2.X is any alphabets.	Verify Add Node button is disabled.	Functional
3.4	**Provide Wallet Details**	**Provide Valid Wallet address**	1. Enter valid Wallet Address(valid checksum hexadecimal) and select wallet permission. 2. Add Wallet.	Verify Add Wallet Button is enabled.	Functional
3.5		**Provide Invalid Wallet Address. (**Negative)	1. Enter **Invalid Wallet Address(invalid checksum hexadecimal).**	Verify Add Wallet button is disabled.	Functional
3.6		**Review and Submit Blockchain details**	1.Enter valid Node ID, valid IP address, select valid Node type(repeat this for process for as much as Node required).click on Next. 2.Enter valid Wallet address and select wallet permission and add required number of wallets. 3. Review and Submit details.	Verify the entered details is reflected in Review and Submit page.	Functional

4.1	**Create New Private Blockchain**	**Provide Network name and valid Wallet Address**	1.Enter Network Name and Valid Wallet Address. 2.Click on Next	Verify after entering valid details Next button is enabled and User is able to click it.	Functional
4.2		**Provide Invalid Wallet Address. (**Negative)	1.Enter Network Name and **Invalid** Wallet Address.	Verify Next button is disabled.	Functional
4.3		**Add Valid Nodes to Private Blockchain**	1. Enter valid Node ID (e.g., NodeID-1234), public IP(X.X.X.X where 0<=X<=255), select node type. 3. Click "ADD NODE" 4. Repeat for multiple nodes 5. Next	Nodes are added successfully and added nodes are reflected in the Preview Table.	Functional
4.4		**Invalid Node ID** (Negative)	1. Enter **invalid Node ID** format (**e.g.,NodeID1234**) and enter valid IP. Different Node ID examples for Negative Scenario: 1. NodeID1244 2.**NodeID-abcd(Not working as expected)** **3**.NodeId-1234	Verify Add Node button is disabled.	Functional
4.5		**Invalid Public IP (**Negative**)**	1. Enter valid Node ID format (e.g.,NodeID-1234) and enter **invalid IP**(X.X.X.X where 1.255<X<1000(**Not working as expected).** 2.X is any alphabets.	Verify Add Node button is disabled.	Functional
4.6		**Review and Submit Private Blockchain details**	1. Enter Network name and valid wallet address and click on Next. 2.Enter valid Node ID, valid IP address, select valid Node type(repeat this for process for as much as Node required).click on Next. 3. Review and Submit details.	Verify the entered details is reflected in Review and Submit page.	Functional
5	**Sign Out**	**Successful Sign Out**	1. After signing in, click "Sign Out"	User is successfully signed out and	Functional

				sign in button is displayed	
6.1	**UI/UX Testing**	**Check Layout on Desktop & Mobile**	1. Open the app on different devices and resolutions	UI elements are correctly aligned, responsive design is functional	UI/UX
6.2		**Check Error Messages**	1. Perform actions leading to errors (e.g., Sign Up with invalid data)	Error messages should be clear and appropriately positioned	UI/UX
6.3		**Password Encryption**	1. Inspect browser storage 2. Check that passwords are encrypted securely	Password should be stored securely and never in plain text	Security
7.1	**Get the API after submittin g the Node Details**	**Automate the API using Postman with valid Details(Node ID and Wallet Address)**	1.Get the API which gets called after submitting Node Details and Automate it using Postman with valid details	API is successfully submitted and sends 200 as response code.	API Testing
7.2		**Automate the API using with Invalid Details(Node ID and Wallet Address) (**Negative)	1.Get the API which gets called after submitting Node Details and Automate it using Postman with Invalid details	API response should throw corresponding error.	API Testing
8.1	**Cross-Bro wser Testing**	**Run on Chrome, Firefox, Safari, Edge**	1. Open the app on each browser 2. Perform basic functions (Sign Up, Sign In, Add Nodes)	The application should work consistently across all browsers	Cross-Bro wser
9.1	**Usability Testing**	**Check Workflow for Onboarding Nodes**	1. Simulate a user adding multiple nodes and wallets 2. Check if the workflow is intuitive and easy to navigate	User should find the process intuitive and receive proper feedback	Usability

9.2		**Review Error Handling**	1. Intentionally make errors (e.g., submit without entering node ID)	Error handling should be clear, with appropriate messages and recovery options	Usability