SUPERVISOR: AHMED HANIF

NARWAL AUTH API DELIVERABLE 12

MUHAMMAD NASEEM
SAMI NAEEM
KYNAT MANSHA

API HOSTING

Resolved all previous issues with api integration.

- issues with Global Access to the API (Zero Knowledge) services

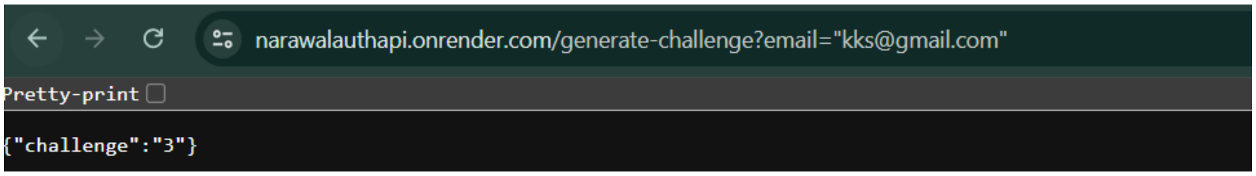
GitHub Repository:

- A private Github repository created :
    - Containing the Node.js Project
    - API file named narawalauthapi.js
    - Python code

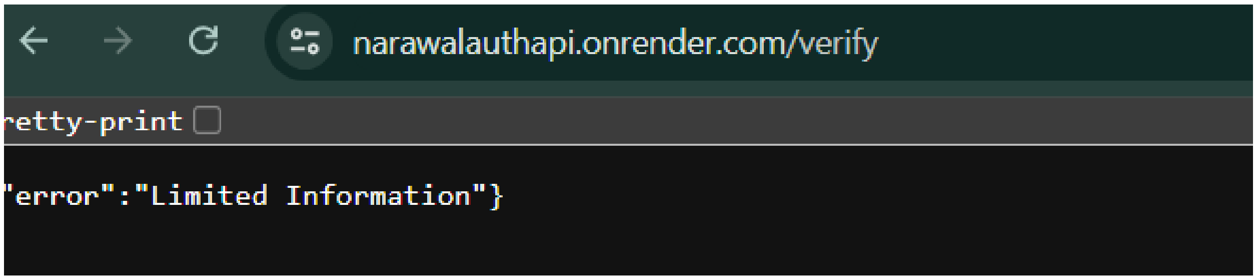
Render Configuration:

- Hosted our API repository on render and ran it successfully.
- App domain:
  https://narawalauthapi.onrender.com

Demo:



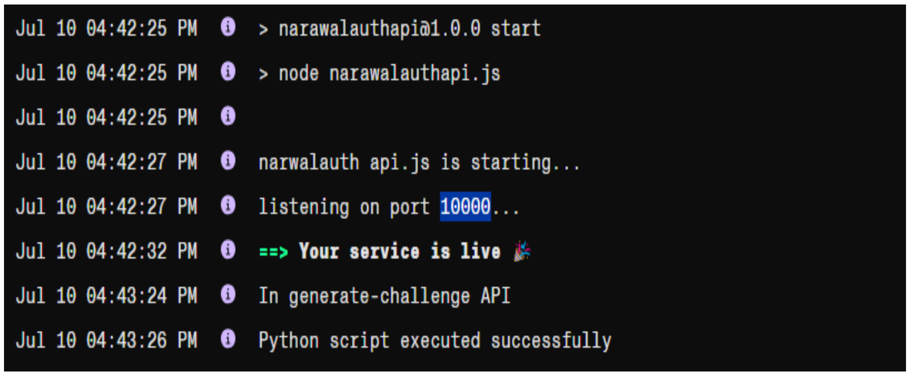
```
narawalauthapi.onrender.com/generate-challenge?email="kks@gmail.com"
Pretty-print
{"challenge":"3"}
```



```
narawalauthapi.onrender.com/verify
retty-print
"error":"Limited Information"}
```

Server Initialization:

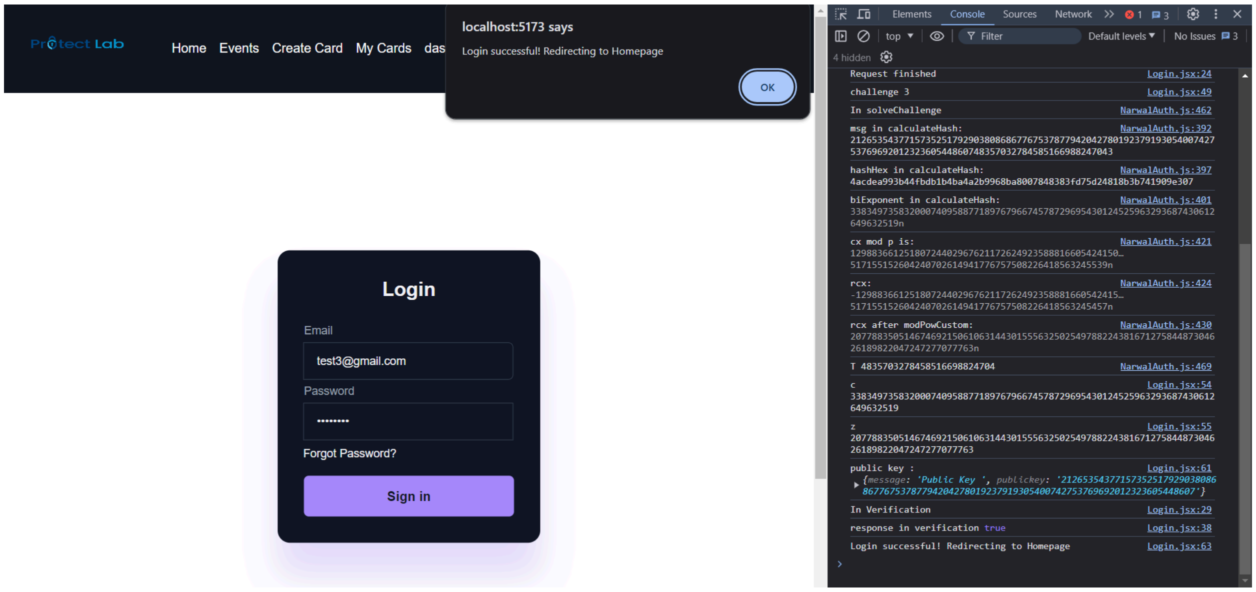
- Server Initialization: Starts the Express server on the rendered port (10000).
- Console log: Logs server start-up message for monitoring.



```
Jul 10 04:42:25 PM    > narawalauthapi@1.0.0 start
Jul 10 04:42:25 PM    > node narawalauthapi.js
Jul 10 04:42:25 PM
Jul 10 04:42:27 PM    narwalauth api.js is starting...
Jul 10 04:42:27 PM    listening on port 10000...
Jul 10 04:42:32 PM    ==> Your service is live 🎉
Jul 10 04:43:24 PM    In generate-challenge API
Jul 10 04:43:26 PM    Python script executed successfully
```

Code update:

- API domain changed.

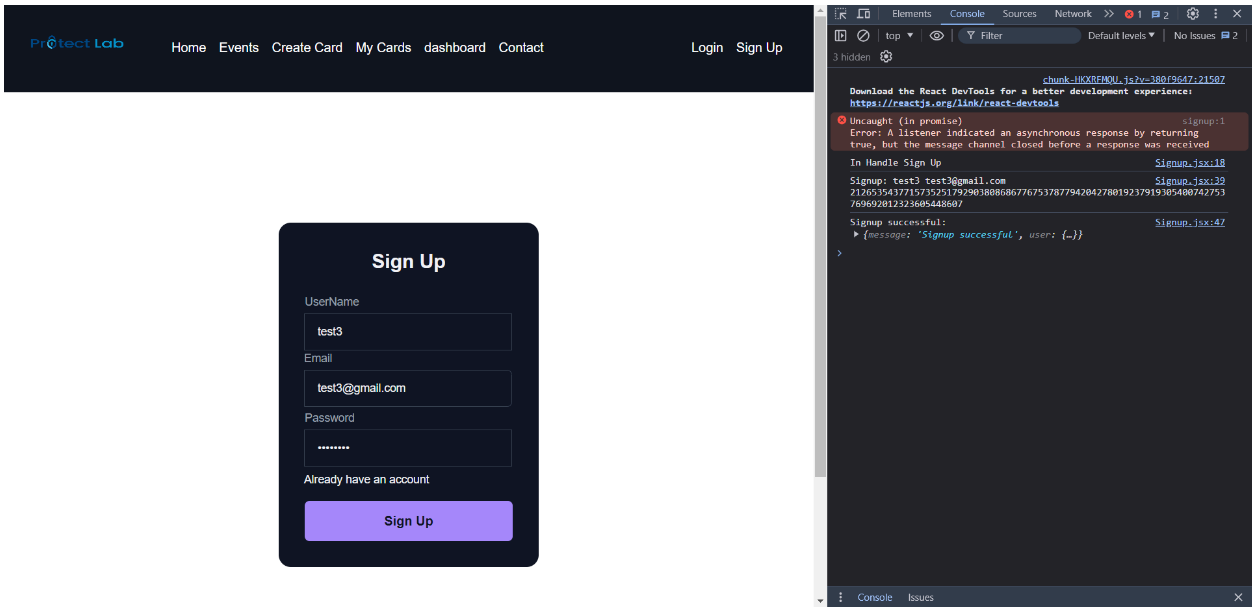


localhost:5173 says

Login successful! Redirecting to Homepage

Email: test3@gmail.com

```
Request finished                                           Login.jsx:24
challenge 3                                                Login.jsx:49
In solveChallenge                                   NarwalAuth.js:462
msg in calculateHash:                               NarwalAuth.js:392
212653543771573525179290380868677675378779420427801923791930540074275376969201232360544860748357032784585166988247043
hashHex in calculateHash:                           NarwalAuth.js:397
4acdea993b44fbdb1b4ba4a2b9968ba8007848383fd75d24818b3b741909e307
biExponent in calculateHash:                        NarwalAuth.js:401
338349735832000740958877189767966745787296954301245259632936874306126496325190n
cx mod p is:                                        NarwalAuth.js:421
12988366125180724402967621172624923588816605424150…
5171551526042407026149417767575082264185632455539n
rcx:                                                NarwalAuth.js:424
-1298836612518072440296762117262492358881660542415…
5171551526042407026149417767575082264185632455457n
rcx after modPowCustom:                             NarwalAuth.js:430
2077883505146746921506106314430155563250254978822438167127584487304626189822047247277077763n
T 4835703278458516698824704                         NarwalAuth.js:469
c                                                          Login.jsx:54
33834973583200074095887718976796674578729695430124525963293687430612649632519
z                                                          Login.jsx:55
2077883505146746921506106314430155563250254978822438167127584487304626189822047247277077763
public key :                                               Login.jsx:61
▸ {message: 'Public Key ', publickey: '2126535437715735251792903808686776753787794204278019237919305400742753769692012323605448607'}
In Verification                                            Login.jsx:29
response in verification true                              Login.jsx:38
Login successful! Redirecting to Homepage                  Login.jsx:63
```



UserName: test3

Email: test3@gmail.com

```
                                  chunk-HKXRFMQU.js?v=380f9647:21507
Download the React DevTools for a better development experience:
https://reactjs.org/link/react-devtools
Uncaught (in promise)                                        signup:1
Error: A listener indicated an asynchronous response by returning
true, but the message channel closed before a response was received
In Handle Sign Up                                       Signup.jsx:18
Signup: test3 test3@gmail.com                           Signup.jsx:39
2126535437715735251792903808686776753787794204278019237919305400742753
7696920123236054486O7
Signup successful:                                      Signup.jsx:47
▸ {message: 'Signup successful', user: {…}}
```

Initialization and Challenge Storage:

- challenges: **Map** to store generated challenges associated with email addresses.
- Console log: Indicates server start for monitoring purposes.

```
const challenges = new Map();
console.log("narwalauth api.js is starting...");
```

Endpoint: Generate Challenge:

- Endpoint **/generate-challenge**: Handles GET requests to generate and send a challenge
- Request Body: Expects **email** field.
- Validation: Checks if **email** is provided; returns error if not.
- Python Script Invocation: Calls runPythonScript to execute **zkp_operations.py** with **generate_challenge** argument.
- Response: Stores generated challenge in **challenges** map and sends it as JSON response

```
app.get('/generate-challenge', (req, res) => {
  console.log("In generate-challenge API");

  // Extract email from query parameters
  const email = req.query.email;
  if (!email) {
    return res.status(400).json({ error: 'Email is required' });
  }

  runPythonScript('zkp_operations.py', ['generate_challenge'], (err, results)
=> {
    if (err) {
      return res.status(500).json({ error: 'Internal Server Error' });
    }
    const challenge = results[0];
    challenges.set(email, challenge);
    res.json({ challenge: challenge.toString() });
  });
});
```

Endpoint: Verify Authentication:

- Endpoint **/verify**: Handles GET requests to verify authentication data (**publicKey**, **c**, **z**)
- Validation: Ensures required fields (**email**, **publicKey**, **c**, **z**) are present; returns error if any are missing.
- Python Script Invocation: Calls **runPythonScript** to execute **zkp_operations.py** with

verify argument and authentication data.

- Response Handling: Parses output from Python script to determine authentication success or failure based on the first element of returned data.

```
app.get('/verify', (req, res) => {
  const { email, publicKey, c, z } = req.query;

  const challenge = challenges.get(email);
  if (!challenge || !email || !publicKey || !c || !z) {
    return res.status(400).json({ error: 'Limited Information' });
  }

  console.log(req.query);
  console.log("challenge", challenge.toString());

  runPythonScript('zkp_operations.py',
    ['verify', publicKey, c, z, challenge.toString()],
    (err, results) => {
      if (err) {
        console.error('Error running Python script:', err);
        return res.status(500).json({ error: 'Internal Server Error' });
      }

      // Log the raw output from the Python script
      const rawOutput = results.split('python verification script')[1].trim();
      console.log('Processed output from Python script:', rawOutput);

      try {
        const resultsArray = rawOutput.slice(1, -1).split(',').map(item =>
item.trim());
        const firstElement = parseInt(resultsArray[0], 10);
        console.log('First element from Python script output:', firstElement);
        if (firstElement === 1) {
          console.log('isSuccess: is true here');
          res.json({ success: true });
        } else {
          console.log('isSuccess: is false here');
          res.json({ success: false });
        }
      } catch (parseError) {
        console.error('Error parsing JSON:', parseError);
        res.status(500).json({ error: 'Invalid response from verification
process' });
      }
    }
  );
});
```