DAR001 DMeta Mobile App using Reverse React Notation – Tunnel Redirection

Figure 1

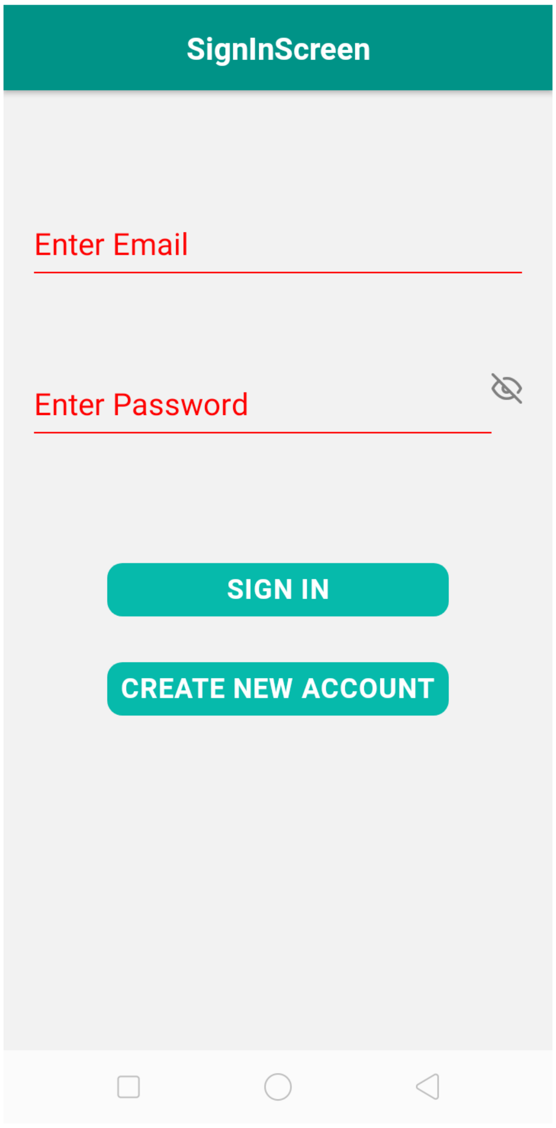
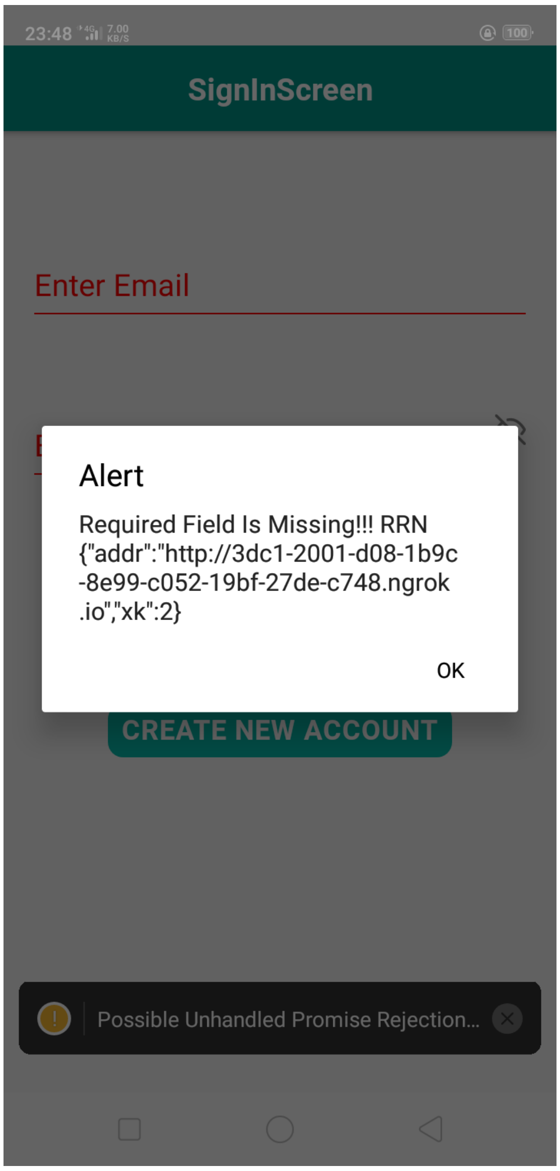


Figure 2



Alert

Required Field Is Missing!!! RRN {"addr":"http://3dc1-2001-d08-1b9c-8e99-c052-19bf-27de-c748.ngrok.io","xk":2}

OK

Possible Unhandled Promise Rejection...

Conventional web services are usually hosted using Domain Name System (DNS), where domain names need to be registered with domain registrar at a cost.

Mobile devices or personal desktop computers can however host web or database services and become accessible to public Internet users using "tunnel services" such as *ngrok*.

However, tunnel addresses are randomly assigned and difficult to remember, e.g.
- https://3dc1-2001-d08-1b9c-8e99-c052-19bf-27de-c748.ngrok.io/dmeta/dmeta.php?tnn=adam&id=001&nn=anon

As such, they can be redirected from fixed and free personal websites such as github pages, using easy to remember usernames and directory names, e.g.

- https://godmeta.github.io/dmeta

In this article, we demonstrate how tunnel redirection can be done using React Native on Android devices, simplified using Reverse React Notation, a metaprogramming script based on reverse polish notation, derived from FORTH and its variant Phoscript.

DAR001 is the first in a series of DMeta App using Reverse React Notation (hence DAR), a long planned tutorial series on full stack app development, aimed at educating and helping beginners as well as senior programmers, on the benefits of metaprogramming, that quite literally enables programmers to write code collaboratively "one word at a time" ("word" means function name in FORTH), across diverse environments, with different programming languages.

Figure 1 shows the initial view of DMeta App, adapted from:

- https://medium.com/@ashfaaqahamed17/creating-signup-and-login-system-using-react-native-php-and-mysql-f176a1de2c73
- https://github.com/AshfaaqAhamed17/SignUp-Login

Upon pressing 'SIGN IN' button, InserRecord in line 130 in figure 4 will be called, to display the alert message as shown in figure 2, which contains the target tunnel address, redirected from:

```
- http://godmeta.github.io/dmeta.json
```

The original JavaScript code for fetch() is shown in figure 3, which wrapped and simplified with Reverse React Notation (RRN) as shown in figure 4:

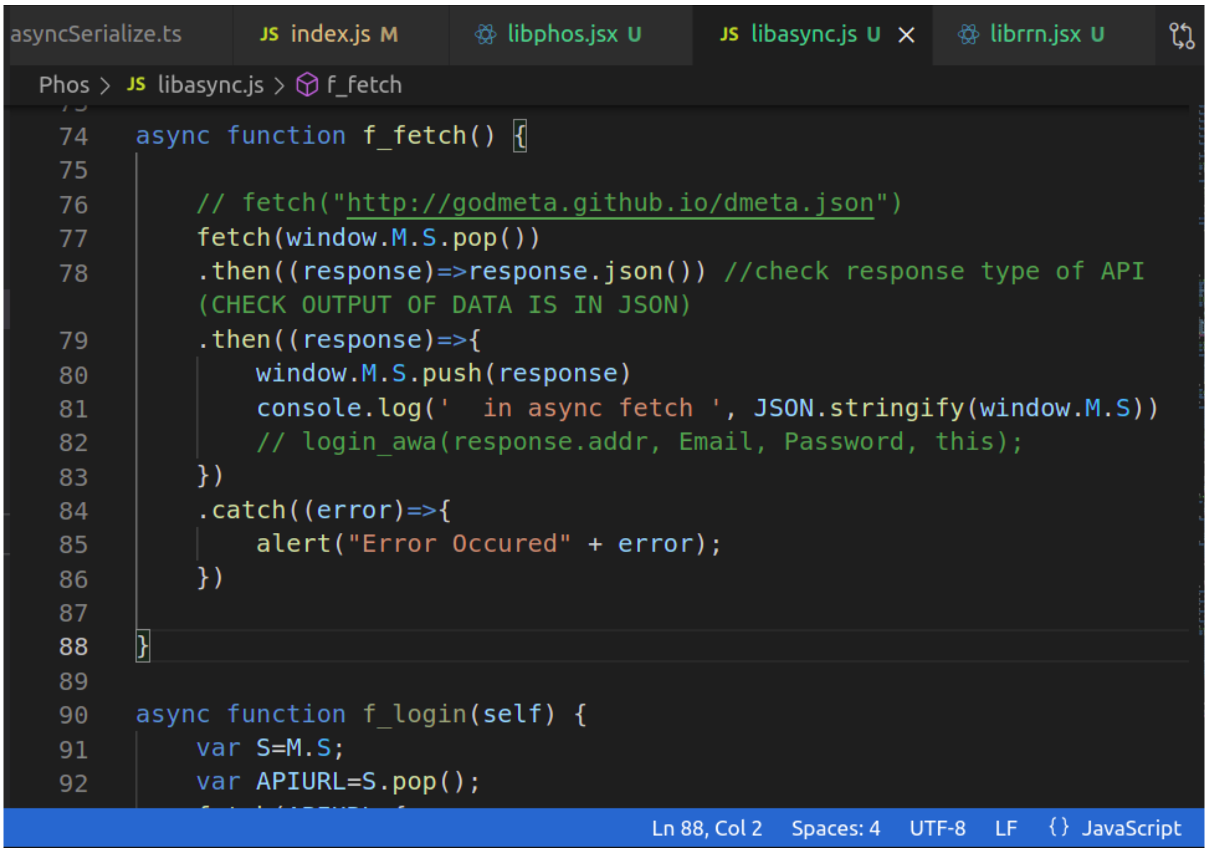
```
    f('http://godmeta.github.io/dmeta.json')
    alert("Required Field Is Missing!!! RRN "+r('awa: fetch: je:'));
```

References:

Reverse React Notation (RRN): Simplifying React syntax with FORTH-like Reverse Polish Notation and Stack Machine Architecture

- https://github.com/udexon/RRN/blob/main/Reverse%20React%20Notation%20RRN%20Part%20I.pdf

Figure 3



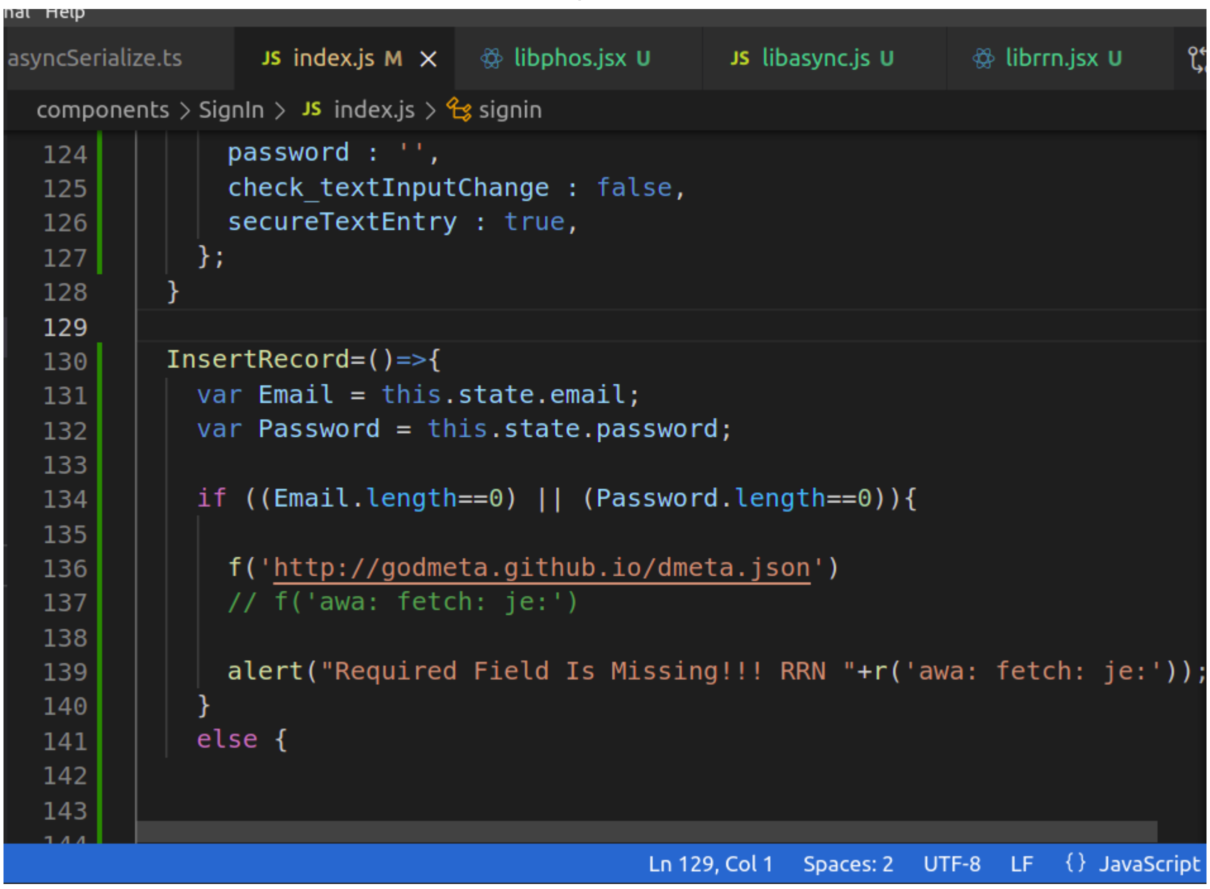
asyncSerialize.ts | JS index.js M | libphos.jsx U | JS libasync.js U | librrn.jsx U

Phos > JS libasync.js > f_fetch

```
async function f_fetch() {

    // fetch("http://godmeta.github.io/dmeta.json")
    fetch(window.M.S.pop())
    .then((response)=>response.json()) //check response type of API
    (CHECK OUTPUT OF DATA IS IN JSON)
    .then((response)=>{
        window.M.S.push(response)
        console.log('  in async fetch ', JSON.stringify(window.M.S))
        // login_awa(response.addr, Email, Password, this);
    })
    .catch((error)=>{
        alert("Error Occured" + error);
    })

}

async function f_login(self) {
    var S=M.S;
    var APIURL=S.pop();
```

Ln 88, Col 2   Spaces: 4   UTF-8   LF   {} JavaScript

Figure 4



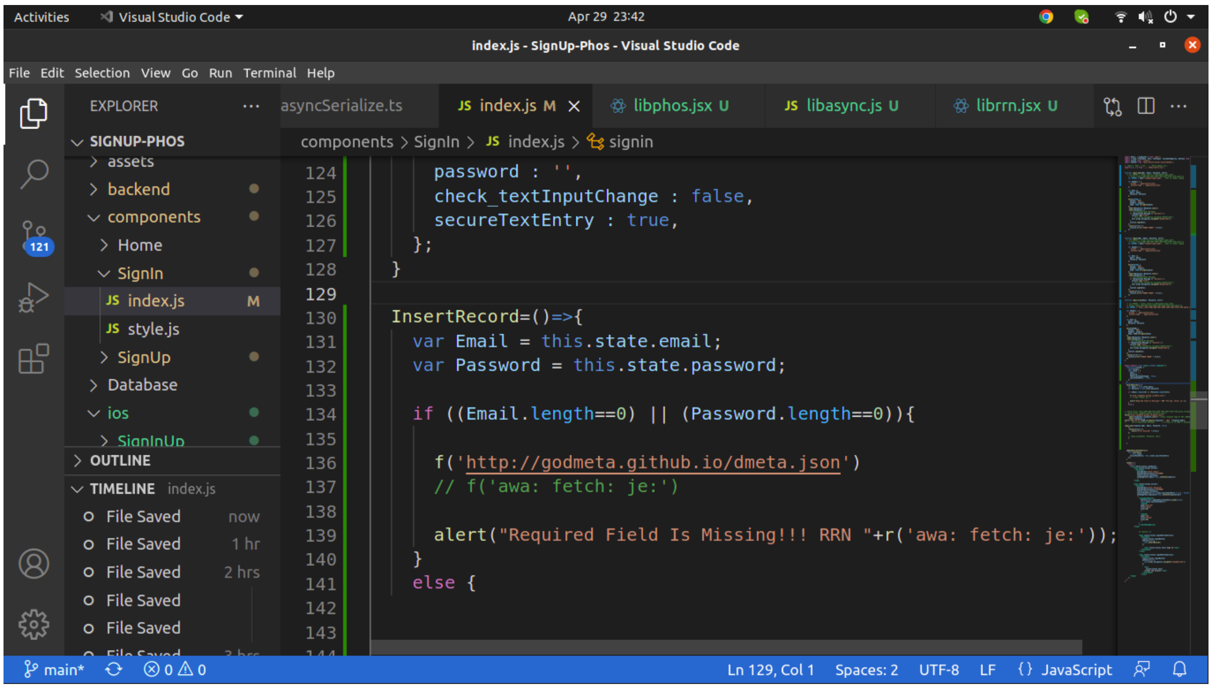
asyncSerialize.ts | JS index.js M | libphos.jsx U | JS libasync.js U | librrn.jsx U

components > SignIn > JS index.js > signin

```
            password : '',
            check_textInputChange : false,
            secureTextEntry : true,
        };
    }

    InsertRecord=()=>{
        var Email = this.state.email;
        var Password = this.state.password;

        if ((Email.length==0) || (Password.length==0)){

          f('http://godmeta.github.io/dmeta.json')
          // f('awa: fetch: je:')

          alert("Required Field Is Missing!!! RRN "+r('awa: fetch: je:'));
        }
        else {


```

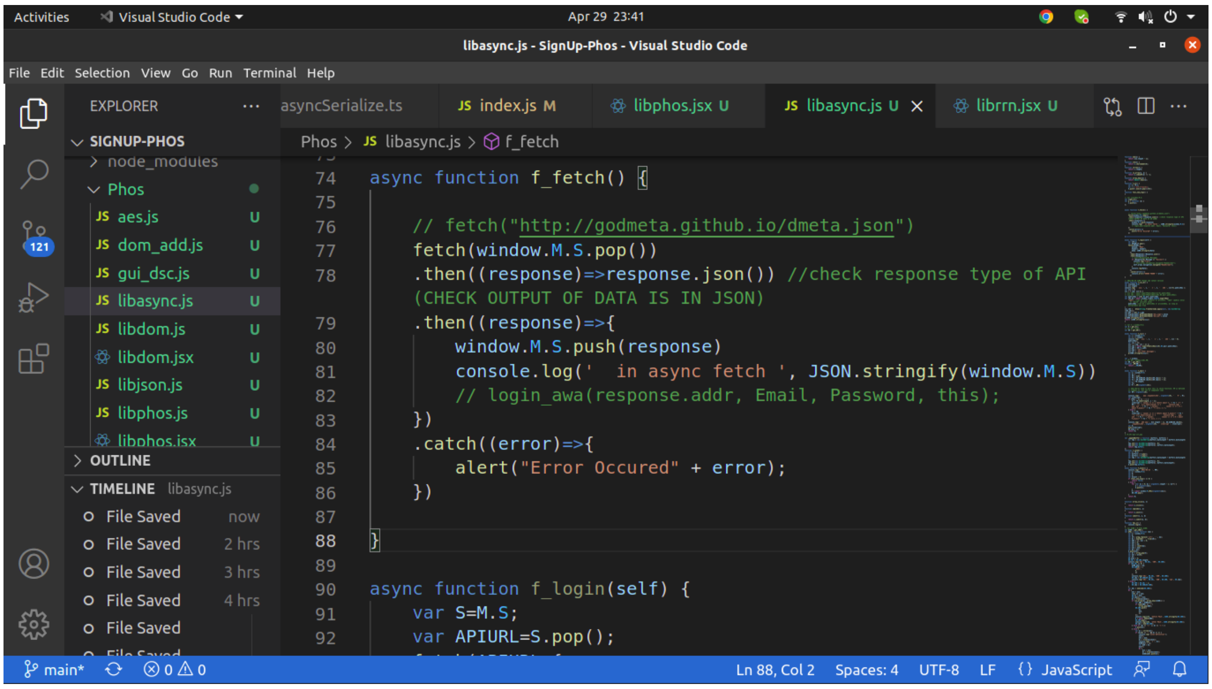
Ln 129, Col 1   Spaces: 2   UTF-8   LF   {} JavaScript

Figure 5: InsertRecord()



```
      password : '',
      check_textInputChange : false,
      secureTextEntry : true,
    };
  }

  InsertRecord=()=>{
    var Email = this.state.email;
    var Password = this.state.password;

    if ((Email.length==0) || (Password.length==0)){

      f('http://godmeta.github.io/dmeta.json')
      // f('awa: fetch: je:')

      alert("Required Field Is Missing!!! RRN "+r('awa: fetch: je:'));
    }
    else {
```

Figure 6: async function f_fetch()



```
async function f_fetch() {

    // fetch("http://godmeta.github.io/dmeta.json")
    fetch(window.M.S.pop())
    .then((response)=>response.json()) //check response type of API
    (CHECK OUTPUT OF DATA IS IN JSON)
    .then((response)=>{
        window.M.S.push(response)
        console.log('  in async fetch ', JSON.stringify(window.M.S))
        // login_awa(response.addr, Email, Password, this);
    })
    .catch((error)=>{
        alert("Error Occured" + error);
    })

}

async function f_login(self) {
    var S=M.S;
    var APIURL=S.pop();
```