

DAR001 DMeta Mobile App using Reverse React Notation

– Tunnel Redirection

Figure 1

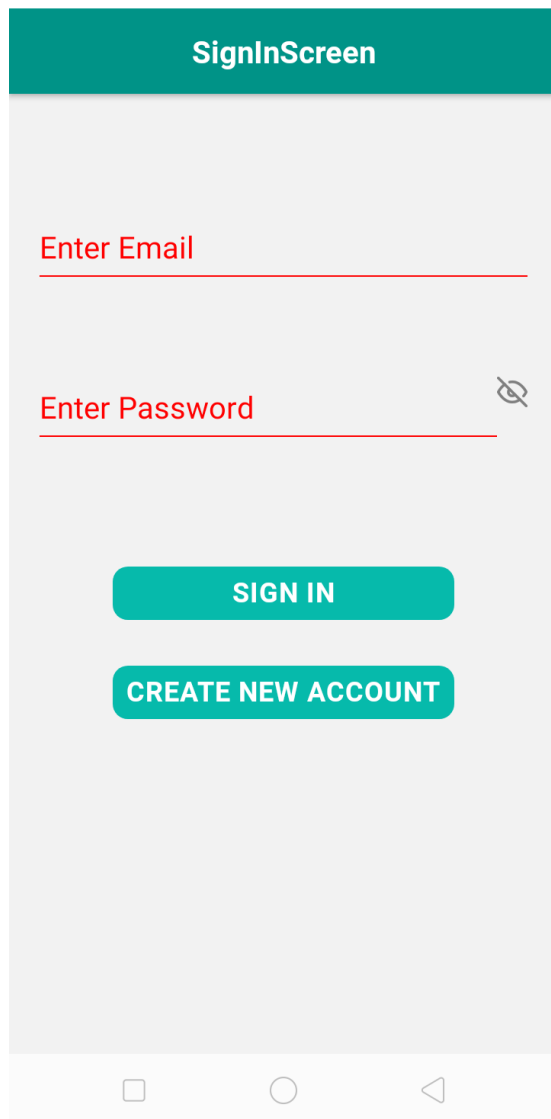
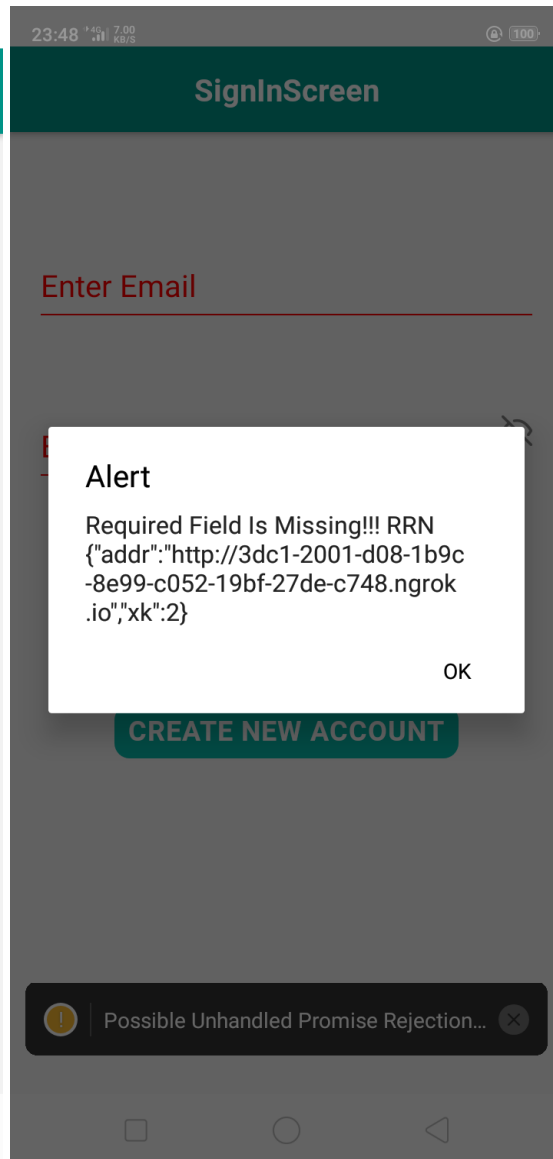


Figure 2



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?nn=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

The screenshot shows a VS Code editor with the file `libasync.js` open. The editor is displaying the `f_fetch` function, which is an asynchronous function that fetches data from `http://godmeta.github.io/dmeta.json`. The function uses `fetch` to get the data, then checks if the response is JSON. If it is, it pushes the response to `window.M.S`, logs it, and calls `login_awa`. If there is an error, it alerts the user. The status bar at the bottom indicates the cursor is at line 88, column 2, with 4 spaces, UTF-8 encoding, and LF line endings.

```

74  async function f_fetch() {
75
76      // fetch("http://godmeta.github.io/dmeta.json")
77      fetch(window.M.S.pop())
78      .then((response)=>response.json()) //check response type of API
79      (CHECK OUTPUT OF DATA IS IN JSON)
80      .then((response)=>{
81          window.M.S.push(response)
82          console.log(' in async fetch ', JSON.stringify(window.M.S))
83          // login_awa(response.addr, Email, Password, this);
84      })
85      .catch((error)=>{
86          alert("Error Occured" + error);
87      })
88  }
89
90  async function f_login(self) {
91      var S=M.S;
92      var APIURL=S.pop();
  
```

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

Figure 4

The screenshot shows a VS Code editor with the file `index.js` open. The editor is displaying the `InsertRecord` function, which is a synchronous function that checks if the email and password fields are filled. If they are, it calls `f_awa` to fetch data. If they are not, it alerts the user. The status bar at the bottom indicates the cursor is at line 129, column 1, with 2 spaces, UTF-8 encoding, and LF line endings.

```

124      password : '',
125      check_textInputChange : false,
126      secureTextEntry : true,
127  };
128  }
129
130  InsertRecord=()=>{
131      var Email = this.state.email;
132      var Password = this.state.password;
133
134      if ((Email.length==0) || (Password.length==0)){
135
136          f('http://godmeta.github.io/dmeta.json')
137          // f('awa: fetch: je:')
138
139          alert("Required Field Is Missing!!! RRN "+r('awa: fetch: je:'));
140      }
141      else {
142
143
144
  
```

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

Figure 5: InsertRecord()

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure for 'SIGNUP-PHOS'. The main editor window shows the 'index.js' file with the 'InsertRecord()' function. The function is an arrow function that takes no arguments and returns a Promise. It declares 'Email' and 'Password' variables from 'this.state'. It checks if either field is empty and shows an alert if so. Otherwise, it makes a fetch request to 'http://godmeta.github.io/dmeta.json' and shows an alert with the response. The status bar at the bottom indicates 'Ln 129, Col 1'.

```
124 password : '',
125 check_textInputChange : false,
126 secureTextEntry : true,
127 };
128 }
129
130 InsertRecord=()=>{
131   var Email = this.state.email;
132   var Password = this.state.password;
133
134   if ((Email.length==0) || (Password.length==0)){
135
136     f('http://godmeta.github.io/dmeta.json')
137     // f('awa: fetch: je:')
138
139     alert("Required Field Is Missing!!! RRN "+r('awa: fetch: je:'));
140   }
141   else {
142
143   }
```

Figure 6: async function f_fetch()

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure for 'SIGNUP-PHOS'. The main editor window shows the 'libasync.js' file with the 'f_fetch()' function. The function is an async function that makes a fetch request to 'http://godmeta.github.io/dmeta.json'. It checks the response type and pushes the response to 'window.M.S'. It also has a catch block for errors. The status bar at the bottom indicates 'Ln 88, Col 2'.

```
74 async function f_fetch() {
75
76   // fetch("http://godmeta.github.io/dmeta.json")
77   fetch(window.M.S.pop())
78   .then((response)=>response.json()) //check response type of API
79   (CHECK OUTPUT OF DATA IS IN JSON)
80   .then((response)=>{
81     window.M.S.push(response)
82     console.log(' in async fetch ', JSON.stringify(window.M.S))
83     // login_awa(response.addr, Email, Password, this);
84   })
85   .catch((error)=>{
86     alert("Error Occured" + error);
87   })
88 }
89
90 async function f_login(self) {
91   var S=M.S;
92   var APIURL=S.pop();
```