

Kebutuhan Sistem .....	2
Node.js .....	2
Stytch .....	3
Visual Studio Code .....	4
Instalasi Kebutuhan Sistem .....	5
Membuat Navigasi .....	7
Navigasi Vue Router .....	7
Membuat Halaman Views .....	9
Intercept Halaman Login .....	11
Membuat Style Umum .....	12
Membuat Desain Login .....	13
Membuat Desain Register .....	18
Membuat Desain Home .....	21
Membuat Server Backend .....	23
Membuat Konfigurasi Environment .....	25
Membuat Server Node dengan Express .....	26
Membuat Akses dari Vue ke Server .....	29
Halaman Register .....	29
Membuat Logout .....	33
Membuat Login .....	34
Mengembalikan Proses Intercept Login .....	36

## Kebutuhan Sistem

### Node.js

Node.js adalah platform buatan Ryan Dahl untuk menjalankan aplikasi web berbasis JavaScript yang dikenalkan pada tahun 2009. Dengan platform ini, Anda dapat mengeksekusi kode JavaScript dari sisi server. Untuk mendukung kemampuan tersebut, Node.js dibangun dengan engine Javascript V8 milik Google. Disamping itu, Node.js memiliki pustaka server sendiri sehingga Anda tidak perlu menggunakan program server web seperti Nginx dan Apache.

Dengan model event-driven dan non-blocking I/O-nya, Node.js adalah platform yang lebih mampu menangani banyak proses secara bersamaan daripada platform bersifat thread-based networking. Selain itu juga, Node.js memiliki NPM (Node Package Manager). Jika Node.js adalah runtime environment JavaScript, NPM merupakan package manager JavaScript dari Node.js. Singkatnya, NPM adalah aplikasi yang bisa digunakan user untuk menyimpan dan saling berbagi modul node. Tentunya, ini akan membuat proses pengembangan aplikasi Anda semakin mudah dan efisien.

Meskipun keduanya berkaitan, JavaScript dan Node.js adalah dua hal yang berbeda. Untuk mengetahui perbedaan Node.js dan JavaScript, mari kita mulai dari pemahaman tentang JavaScript terlebih dahulu.

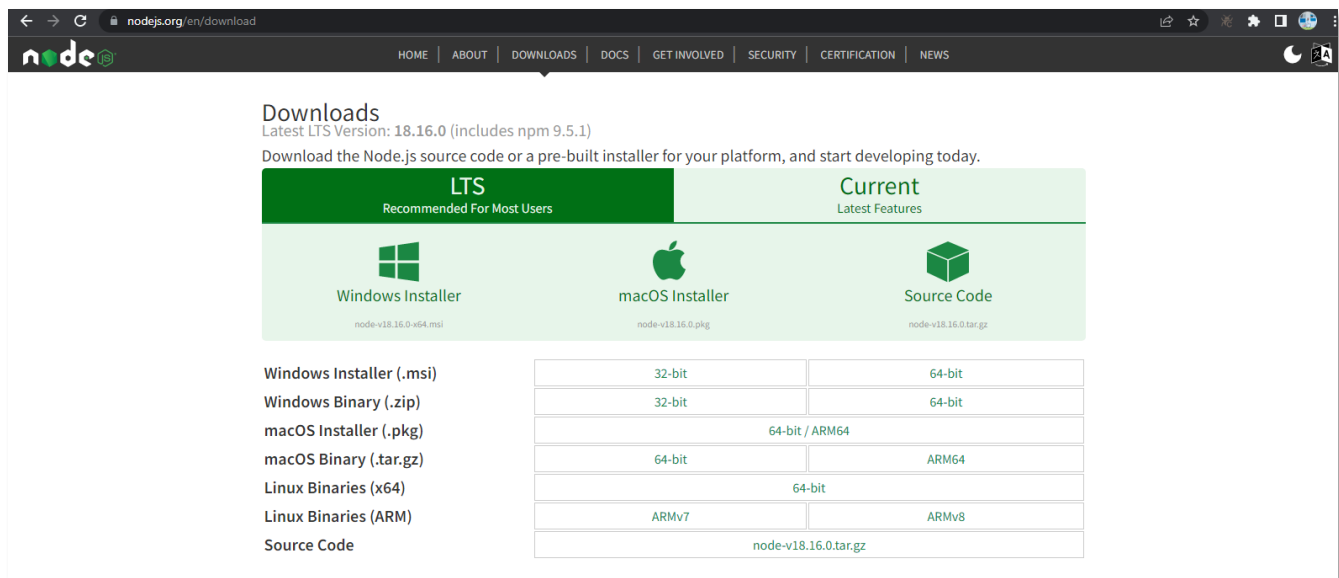
JavaScript adalah bahasa pemrograman yang digunakan bersamaan dengan HTML dan CSS untuk menciptakan halaman website yang bersifat interaktif. HTML menghasilkan struktur dan tampilan teks, sedangkan CSS bertanggung jawab atas tampilan grafis sebuah halaman. Nah, JavaScript berkontribusi atas animasi dan konten-konten interaktif yang ada di dalamnya.

Eksekusi kode JavaScript bergantung pada engine yang ada pada browser. Oleh karena itu, ia disematkan pada kode HTML. Inilah alasan mengapa JavaScript disebut bahasa pemrograman yang bekerja pada sisi client.

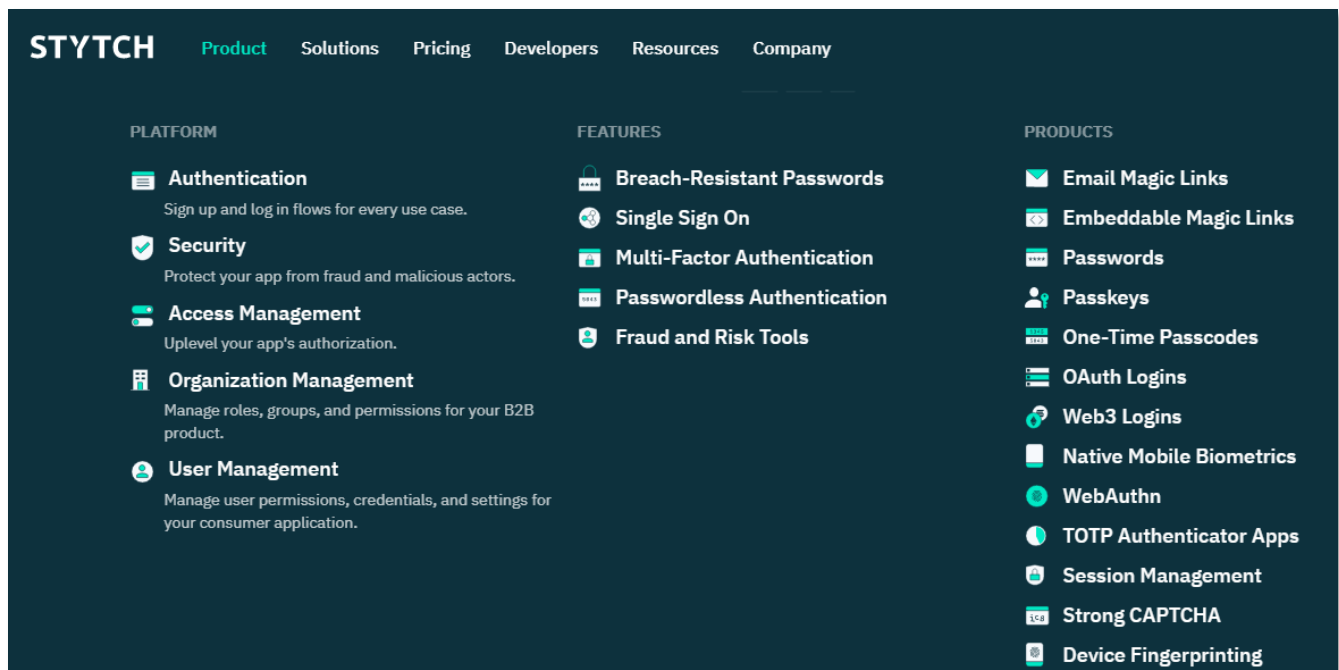
Disisi lain, Node.js adalah platform untuk menjalankan kode JavaScript pada sisi server yang bersifat open source. Ia bertugas untuk mengeksekusi kode JavaScript sebelum halaman website

ditampilkan di browser. Dengan demikian, Node.js dapat menjalankan situs, aplikasi web, dan game berbasis browser dengan performa tinggi.

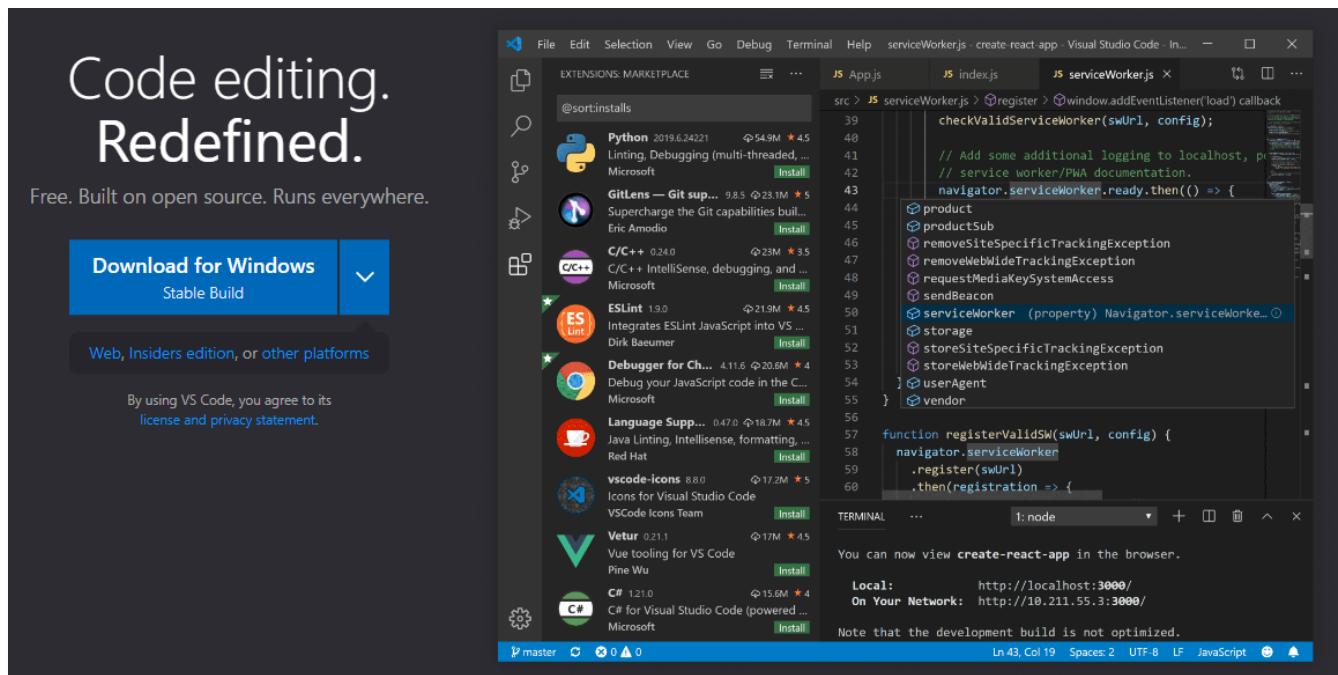
Buka situs resminya pada <https://nodejs.org/en/download>, gunakan versi yang disukai atau yang paling baru agar mendapatkan dukungan secara penuh.



## Stytch



## Visual Studio Code



## Instalasi Kebutuhan Sistem

Langkah – langkah untuk instalasi kebutuhan sistem sebagai berikut:

- 1) Cek versi Node.js dan NPM yang digunakan pada sistem dengan perintah CMD berikut:



```
Administrator: Command Prompt

D:\>node --version
v18.14.0

D:\>npm --version
9.6.6

D:\>_
```

- 2) Buat sebuah direktori baru menggunakan CMD sebagai tempat untuk menyimpan project Vue, misalkan pada **D:/workshop-vue** sebagai berikut:



```
Administrator: Command Prompt

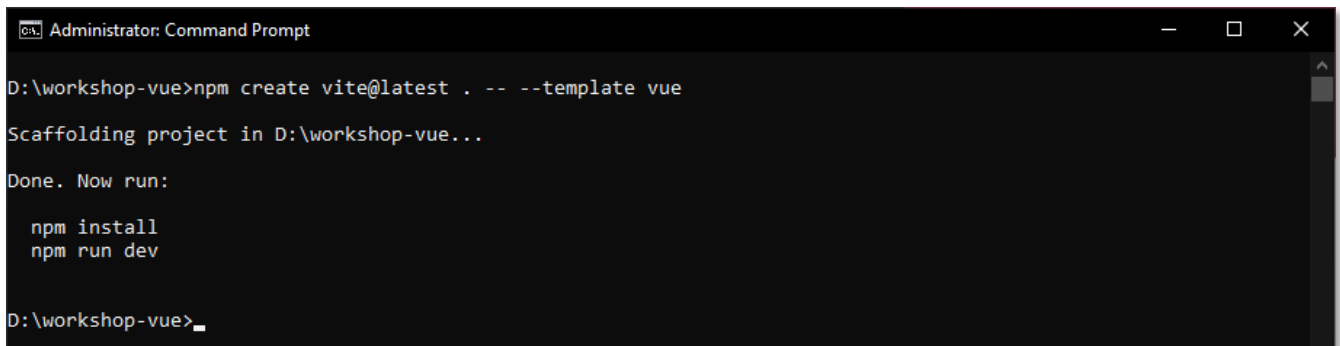
D:\>mkdir workshop-vue

D:\>cd workshop-vue

D:\workshop-vue>_
```

- 3) Tuliskan perintah dibawah untuk membuat project Vue pertama kali:

\$ **create vite@latest . -- --template vue**



```
Administrator: Command Prompt

D:\workshop-vue>npm create vite@latest . -- --template vue

Scaffolding project in D:\workshop-vue...

Done. Now run:

  npm install
  npm run dev

D:\workshop-vue>_
```

- 4) Instalasi library vue-router yang digunakan untuk berpindah dari satu halaman ke halaman lain:

\$ **npm install vue-router**

```
Administrator: Command Prompt
D:\workshop-vue>npm install vue-router
added 27 packages, and audited 28 packages in 12s
3 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
D:\workshop-vue>
```

Buka project Vue pertama tersebut pada teks editor VSCode dengan menulis \$ **code** .

Kemudian buka file **package.json** untuk melihat berbagai dependencies yang digunakan pada project Vue yang sudah dibuat:

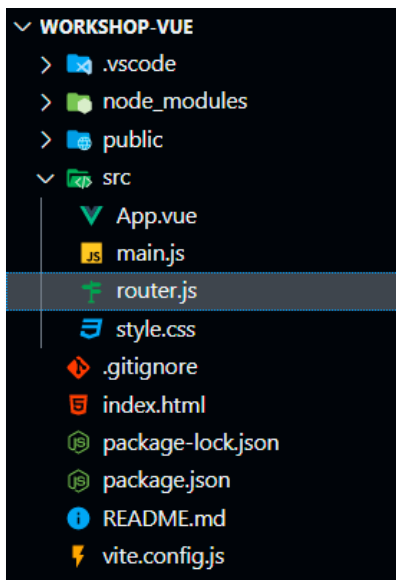
```
package.json > ...
1  {
2    "name": "workshop-vue",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "vite build",
9      "preview": "vite preview"
10   },
11   "dependencies": {
12     "vue": "^3.2.47",
13     "vue-router": "^4.2.1"
14   },
15   "devDependencies": {
16     "@vitejs/plugin-vue": "^4.1.0",
17     "vite": "^4.3.2"
18   }
19 }
```

## Membuat Navigasi

### Navigasi Vue Router

Navigasi dilakukan untuk dapat berpindah dari satu halaman menuju halaman yang lain, oleh Vue telah disediakan library **vue-router**, maka langkah membuat navigasi adalah:

- 1) Hapus direktori yang tidak diperlukan agar kode lebih ringkas, seperti direktori **assets** dan **components** yang saat ini belum digunakan, dan buat sebuah file baru pada direktori **src/router.js** sebagai berikut:



- 2) Tuliskan kode dibawah ini pada file **src/router.js**:

```
import { createRouter, createWebHistory } from "vue-router";

const routes = [
  {
    path: "/",
    name: "Home",
    component: () => import("./views/Home.vue"),
    meta: {
      requiresAuth: true,
    },
  },
  {
    path: "/login",
    name: "Login",
    component: () => import("./views/Login.vue"),
  },
]
```

```
{
  path: "/register",
  name: "Register",
  component: () => import("../views/Register.vue"),
},
];

const router = createRouter({
  history: createWebHistory(),
  routes,
});

export default router;
```

3) Buka file **src/App.vue** dan ubah kode didalamnya sebagai berikut:

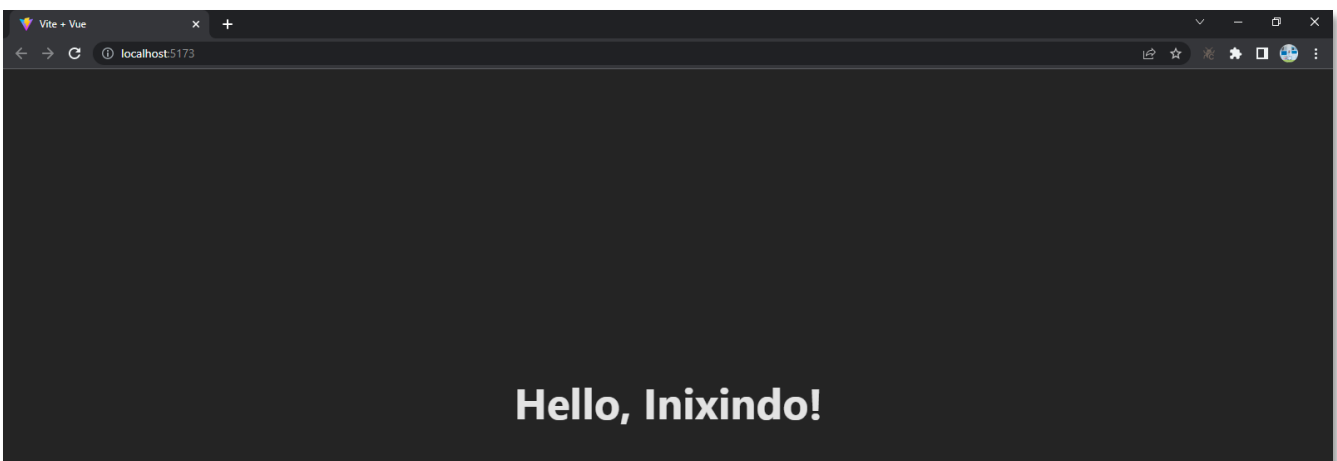
```
<template>
  <div>
    <h1>Hello, Inixindo!</h1>
    <router-view></router-view>
  </div>
</template>
```

Buka terminal pada VSCode dan jalankan aplikasi Vue dengan perintah \$ **npm run dev**

```
VITE v4.3.8 ready in 587 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h to show help

█
```





## Membuat Halaman Views

Pada bagian **src/router.js** sebelumnya telah didefinisikan 3 (tiga) routes, meliputi **path: '/'**, **path:'/login'** dan **path:'/register'** yang masing – masing path mengarah ke component view-nya, maka kita perlu membuat halaman view tersebut.

- 1) Buat sebuah file baru pada direktori **src/views/Home.vue** dan tuliskan kode didalamnya:

```
<template>
  <main>
    <h1>This is Home Page</h1>
  </main>
</template>
```

- 2) Buat sebuah file baru pada direktori **src/views/Login.vue** dan tuliskan kode didalamnya:

```
<template>
  <main>
    <h1>This is Login Page</h1>
  </main>
</template>
```

- 3) Buat sebuah file baru pada direktori **src/views/Register.vue** dan tuliskan kode didalamnya:

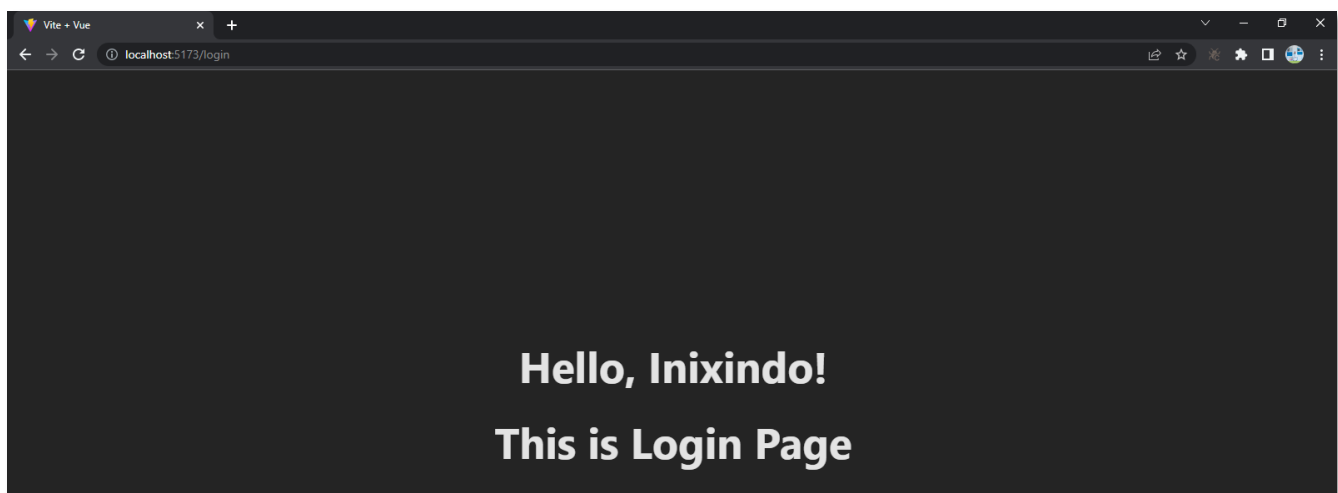
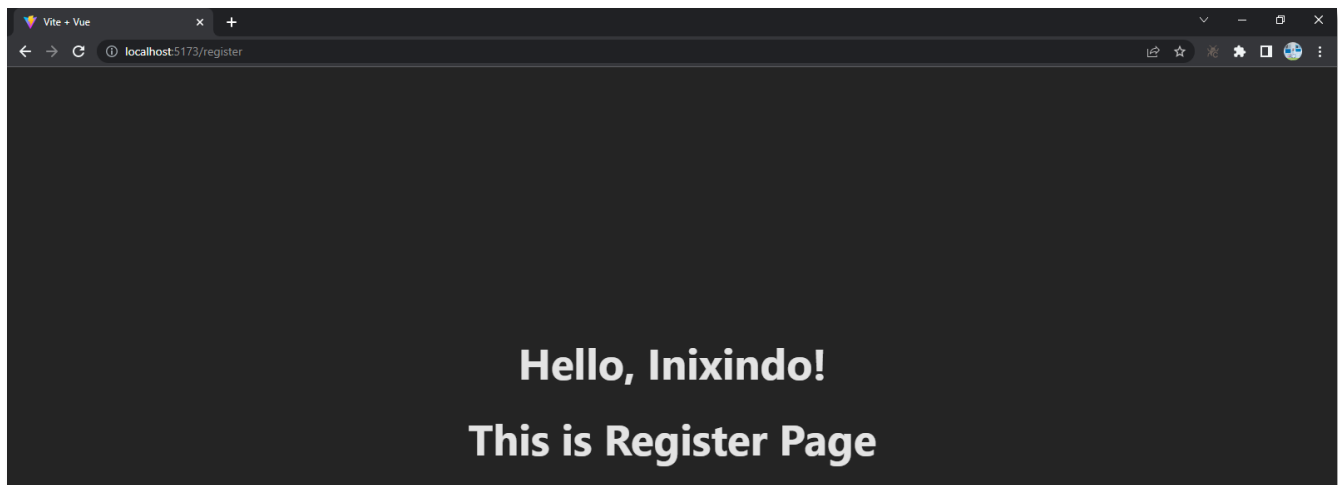
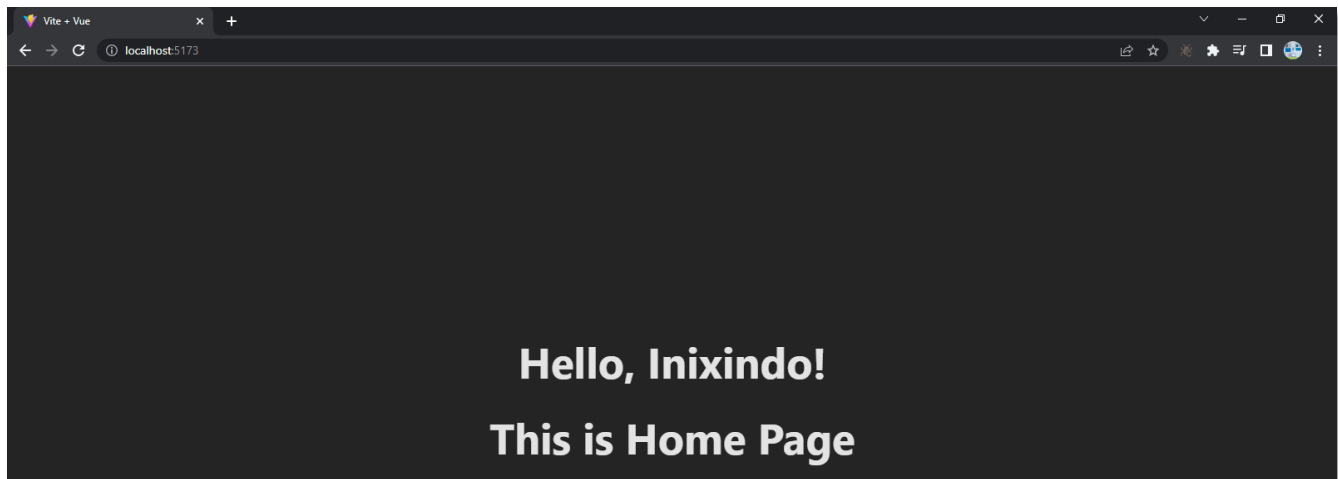
```
<template>
  <main>
    <h1>This is Register Page</h1>
  </main>
</template>
```

- 4) Buka file **src/main.js** dan tambahkan kode berikut untuk menggunakan router:

```
import { createApp } from "vue";
import "./style.css";
import App from "./App.vue";
import router from "./router";

createApp(App).use(router).mount("#app");
```

- 5) Lihat hasilnya pada web browser dengan menambahkan pada url: **http://localhost:5173/**, **http://localhost:5173/register**, dan **http://localhost:5173/login**



## Intercept Halaman Login

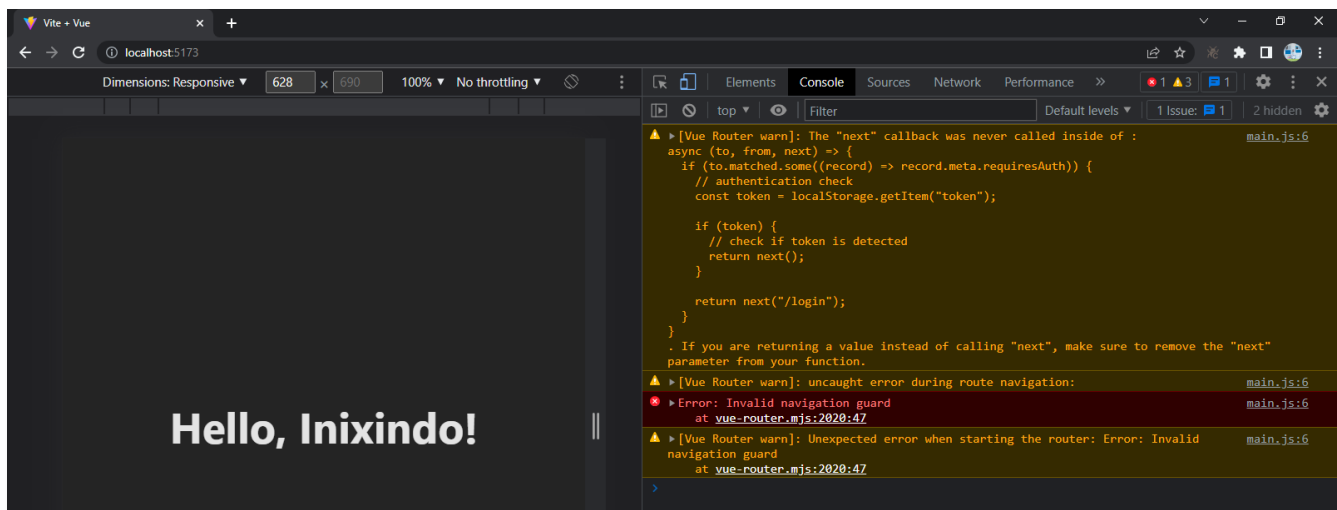
Sebelum bisa membuka halaman Home, pengguna terlebih dahulu diwajibkan untuk melakukan Login atau Register, disinilah letak autentikasi pengguna. Intercept ini dilakukan jika pengguna mengakses halaman depan **path: '/'** maka secara otomatis diarahkan ke **path: '/login'**. Untuk itu buka kembali file **src/router.js** dan tambahkan kode berikut:

```
router.beforeEach(async (to, from, next) => {
  if (to.matched.some((record) => record.meta.requiresAuth)) {
    // authentication check
    const token = localStorage.getItem("token");

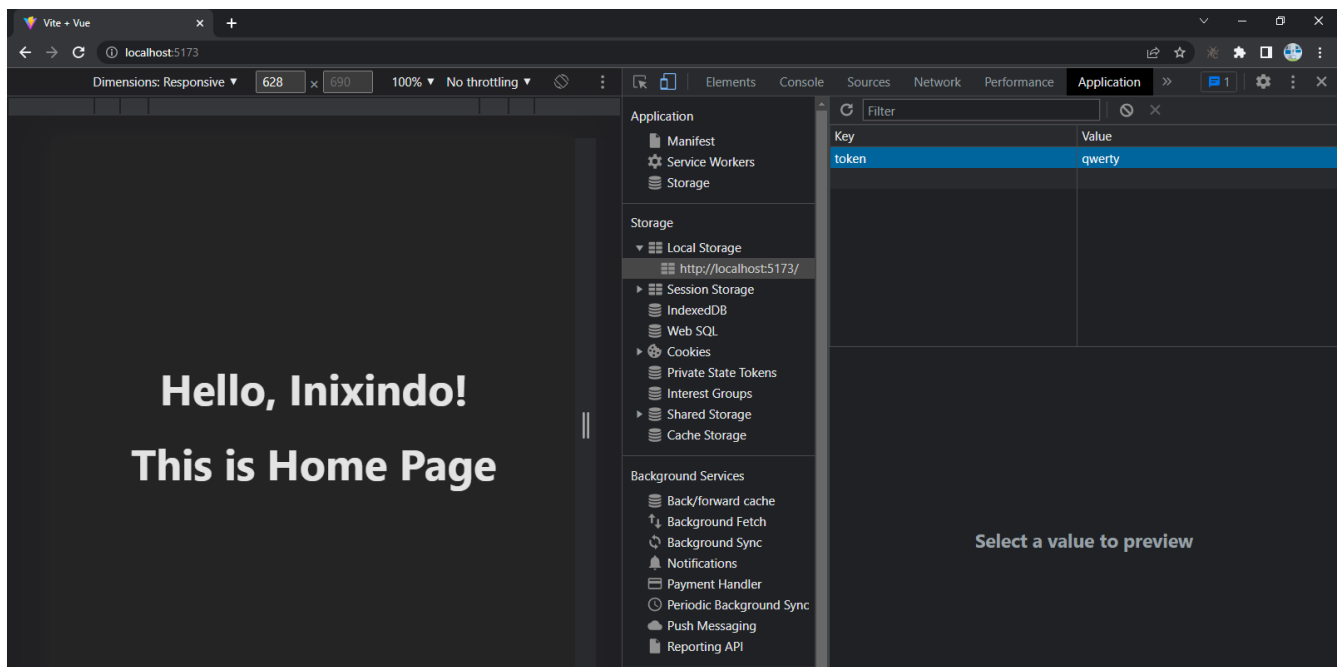
    if (token) {
      // check if token is detected
      return next();
    }

    return next("/login");
  }
});

export default router;
```



Diketahui pada browser terjadi error yaitu: **Invalid navigation guard**, hal ini terjadi karena tidak ada token yang terdeteksi oleh browser, dan halaman menampilkan defaultnya, App.vue. namun, jika terdeteksi adanya suatu token maka dapat membuka halaman Home, seperti pada gambar dibawah ini:



Buat token secara manual pada browser dengan cara membuka **Inspect Element** dan pilih tab menu **Application**, pada **Local Storage** tambahkan sebuah atribut baru dengan **Key: token** dan **Value: sembarang**. Setelah token dideteksi oleh browser, kemudian refresh halaman browser dan secara otomatis dapat membuka halaman Home. Token yang terdeteksi merupakan token dengan nilai sembarang bersifat sementara, untuk kemudian dapat diganti dengan token valid yang berasal dari Styth.

## Membuat Style Umum

Style umum ini digunakan untuk disetiap halaman, berisikan style sederhana untuk menampilkan warna dan teks, buka file **src/style.css** dan tuliskan kode CSS berikut:

```
:root {
  --primary: #13099b;
  --primary-dark: #0f086f;
  --gray: #9ca3af;
  --light: #f3f4f6;
  --dark: #111827;
}
```

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Montserrat", sans-serif;
}

a {
  color: var(--primary);
}

input {
  appearance: none;
  outline: none;
  border: none;
  background: none;
}

.logo {
  display: inline-block;
  font-size: 1rem;
  font-weight: 900;
  letter-spacing: 5px;
  padding: 0.5rem;
  padding-right: calc(0.5rem - 5px);
  border: 2px solid #ffffff;
  color: #ffffff;
  margin-bottom: 1.5rem;
}
```

## Membuat Desain Login

Buka kembali file **src/views/Login.vue** dan tambahkan kode seperti dibawah ini:

```
<script setup>
import { ref } from 'vue'

const email = ref('')
const password = ref('')
</script>

<template>
  <main>
```

```

<header>
  <h1 class="logo">AUTH APP</h1>
  <h2>Login Page</h2>
  <p>
    Login or create an account to start using the
    <strong>Authentication App</strong>
  </p>
</header>

<form @submit.prevent="">
  <label for="email">
    <span>Enter your email</span>
    <input type="email" placeholder="test@test.com" v-model="email" />
  </label>
  <label for="password">
    <span>Enter your password</span>
    <input type="password" placeholder="*****" v-model="password" />
  </label>
  <input type="submit" value="Login" />
</form>

<footer>
  <p>
    Don't have an account?
    <router-link to="/register">
      Register here
    </router-link>
  </p>
</footer>
</main>
</template>

<style scoped>
main {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  justify-content: flex-start;
  height: 100vh;
  background-color: var(--primary);
  color: #ffffff;
}

```

```
header {
  padding: 1.5rem;
}

footer {
  background-color: #ffffff;
  width: 100%;
  color: var(--dark);
  text-align: center;
  padding: 1.5rem;
  padding-bottom: 3rem;
}

h2 {
  font-size: 2.125rem;
  margin-bottom: 1rem;
}

h2~p {
  font-weight: 500;
  font-size: 1rem;
}

form {
  flex: 1 1 0%;
  display: block;
  border-radius: 1.5rem 1.5rem 0 0;
  background-color: #ffffff;
  box-shadow: 0px -4px 12px 4px rgba(0, 0, 0, 0.16);
  color: var(--dark);
  padding: 4rem 1.5rem;
  width: 100%;
}

label {
  display: block;
  margin-bottom: 1.5rem;
}

label span {
  display: block;
  color: var(--gray);
  font-size: 1rem;
  font-weight: 500;
}
```

```
margin-bottom: 0.5rem;
}

input:not([type="submit"]) {
  display: block;
  width: 100%;
  /* border: 1px solid var(--gray); */
  border-radius: 1rem;
  padding: 1.5rem 1rem;
  font-size: 1.125rem;
  font-weight: 500;
  color: var(--dark);
  background-color: var(--light);
}

input:not([type="submit"]) {
  color: var(--gray);
  font-style: italic;
}

input[type="submit"] {
  display: block;
  width: fit-content;
  margin: 0 auto;
  font-size: 1.5rem;
  font-weight: 700;
  color: #ffffff;
  background-color: var(--primary);
  padding: 1rem;
  border-radius: 1rem;
  cursor: pointer;
  transition: 0.5s ease;
}

input[type="submit"]:hover {
  background-color: var(--primary-dark);
}
</style>
```

Untuk melihat hasil desain halaman Login ini, kita perlu menghilangkan dahulu interceptnya, maka buka kembali file `src/router.js` dan berikan komentar pada bagian `router.beforeEach()`:

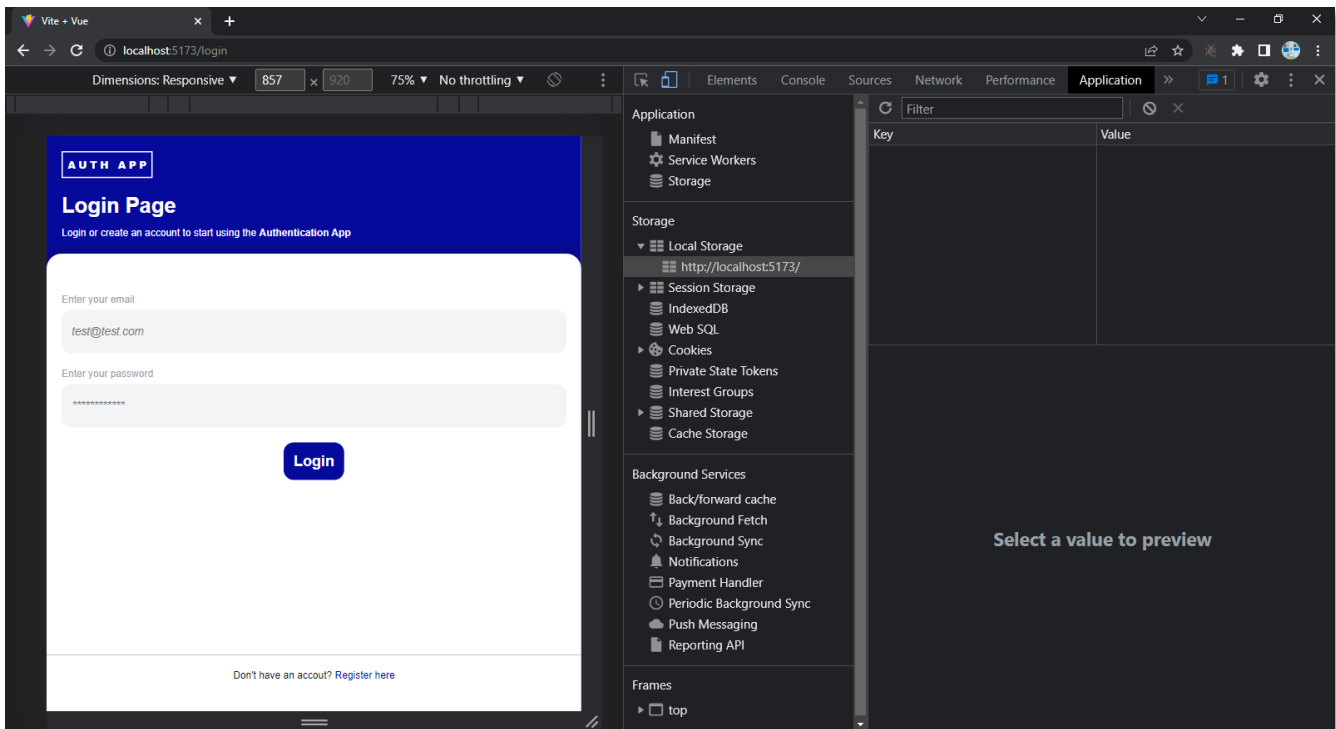


```
// router.beforeEach(async (to, from, next) => {
//   if (to.matched.some((record) => record.meta.requiresAuth)) {
//     // authentication check
//     const token = localStorage.getItem("token");

//     if (token) {
//       // check if token is detected
//       return next();
//     }

//     return next("/login");
//   }
// });
```

Lihat hasilnya pada web browser, seperti gambar dibawah ini:



## Membuat Desain Register

Buka kembali file `src/views/Register.vue` dan tambahkan kode seperti dibawah ini:

```
<script setup>
import { ref } from 'vue'

const email = ref('')
const password = ref('')
const conf_password = ref('')
</script>

<template>
  <main>
    <header>
      <h1 class="logo">AUTH APP</h1>
      <h2>Register Page</h2>
      <p>
        Login or create an account to start using the
        <strong>Authentication App</strong>
      </p>
    </header>

    <form @submit.prevent="">
      <label for="email">
        <span>Enter your email</span>
        <input type="email" placeholder="test@test.com" v-model="email" />
      </label>
      <label for="password">
        <span>Enter your password</span>
        <input type="password" placeholder="*****" v-model="password" />
      </label>
      <label for="conf_password">
        <span>Confirm your password</span>
        <input type="password" placeholder="*****" v-
model="conf_password" />
      </label>
      <input type="submit" value="Register" />
    </form>

    <footer>
      <p>Already have an account?
        <router-link to="/login">Login here</router-link>
      </p>
    </footer>
  </main>
</template>
```

```
</main>
</template>

<style scoped>
main {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  justify-content: flex-start;
  height: 100vh;
  background-color: var(--primary);
  color: #ffffff;
}

header {
  padding: 1.5rem;
}

footer {
  background-color: #ffffff;
  width: 100%;
  color: var(--dark);
  text-align: center;
  padding: 1.5rem;
  padding-bottom: 3rem;
}

h2 {
  font-size: 2.125rem;
  margin-bottom: 1rem;
}

h2~p {
  font-weight: 500;
  font-size: 1rem;
}

form {
  flex: 1 1 0%;
  display: block;
  border-radius: 1.5rem 1.5rem 0 0;
  background-color: #ffffff;
  box-shadow: 0px -4px 12px 4px rgba(0, 0, 0, 0.16);
  color: var(--dark);
}
```

```
padding: 4rem 1.5rem;
width: 100%;
}

label {
  display: block;
  margin-bottom: 1.5rem;
}

label span {
  display: block;
  color: var(--gray);
  font-size: 1rem;
  font-weight: 500;
  margin-bottom: 0.5rem;
}

input:not([type="submit"]) {
  display: block;
  width: 100%;
  /* border: 1px solid var(--gray); */
  border-radius: 1rem;
  padding: 1.5rem 1rem;
  font-size: 1.125rem;
  font-weight: 500;
  color: var(--dark);
  background-color: var(--light);
}

input:not([type="submit"]) {
  color: var(--gray);
  font-style: italic;
}

input[type="submit"] {
  display: block;
  width: fit-content;
  margin: 0 auto;
  font-size: 1.5rem;
  font-weight: 700;
  color: #ffffff;
  background-color: var(--primary);
  padding: 1rem;
  border-radius: 1rem;
}
```

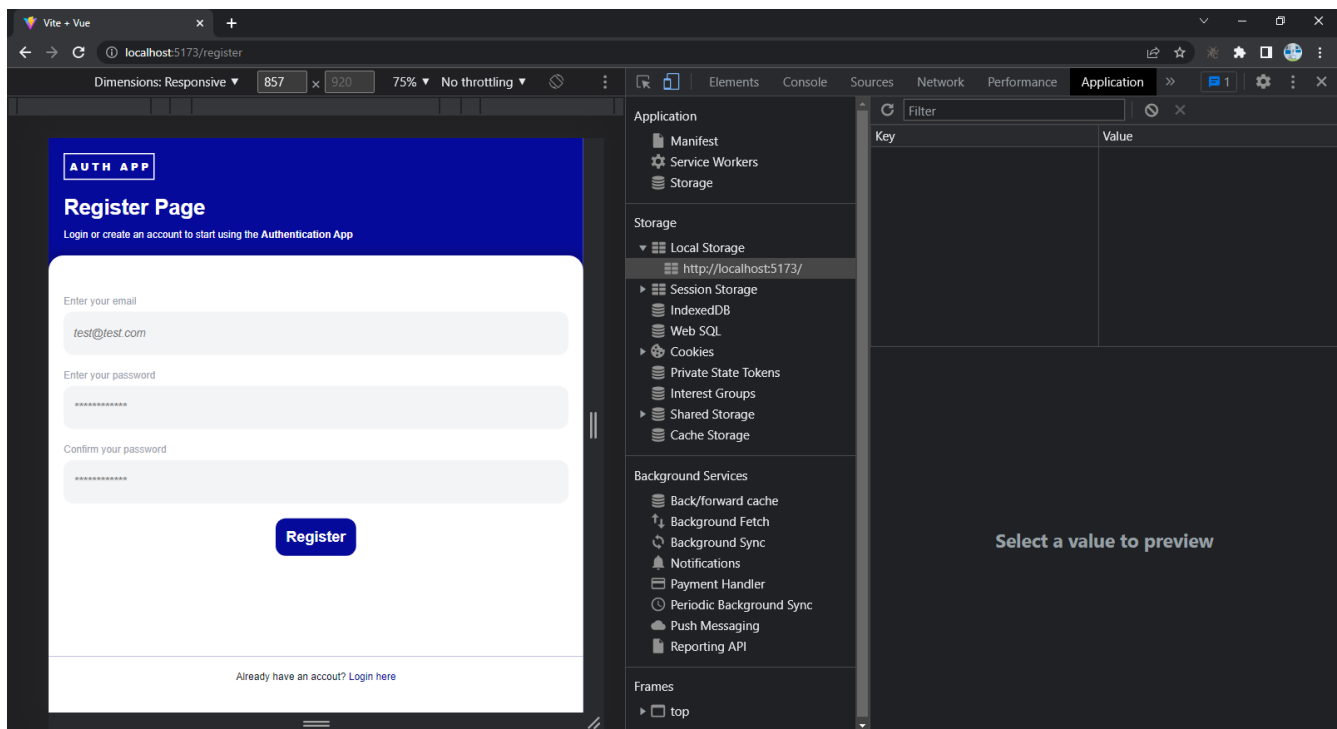
```

    cursor: pointer;
    transition: 0.5s ease;
  }

  input[type="submit"]:hover {
    background-color: var(--primary-dark);
  }
</style>

```

Lihat hasil desain halaman Register pada web browser:



## Membuat Desain Home

Buka kembali file **src/views/Home.vue** dan tambahkan kode seperti dibawah ini:

```

<script setup>
</script>

<template>
  <main>
    <h1>This is Home Page</h1>

    <button @click="">Logout</button>
  </main>

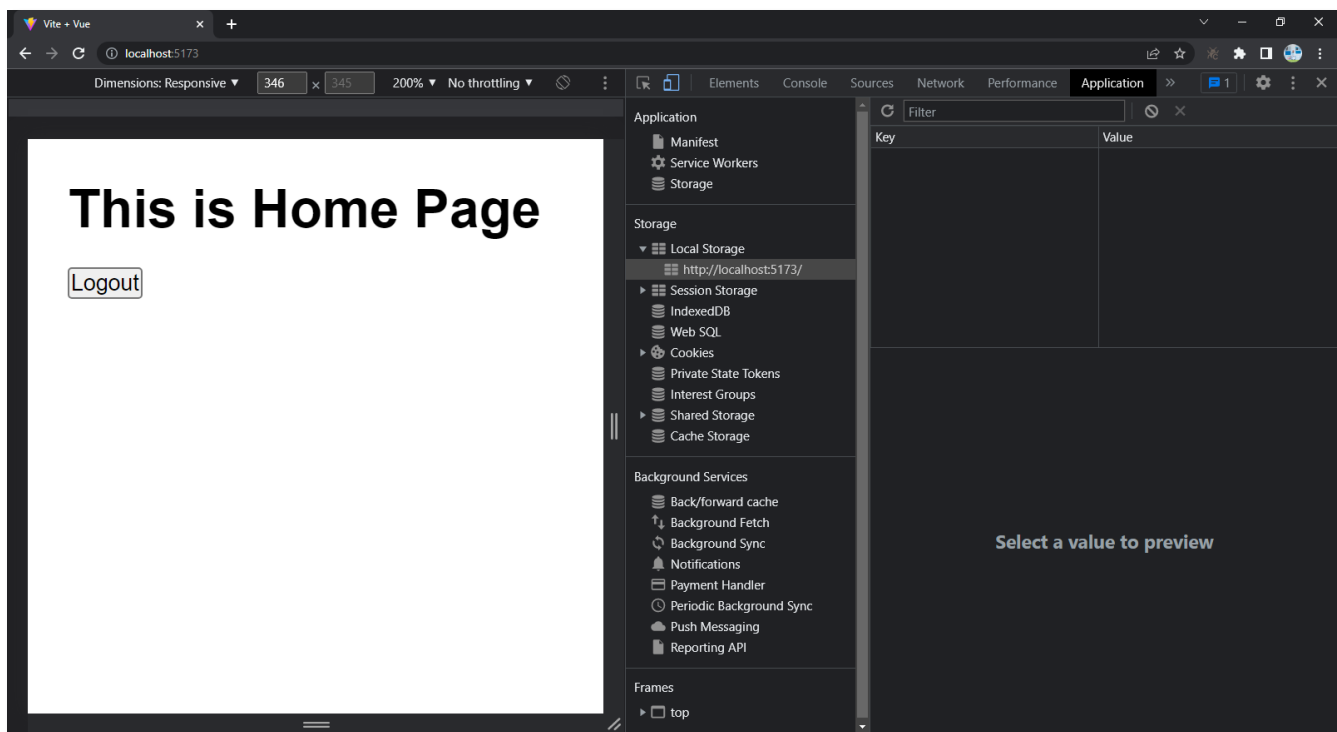
```

```
</template>

<style scoped>
main {
  padding: 1.5rem;
}

h1 {
  margin-bottom: 1rem;
}
</style>
```

Lihat hasilnya pada web browser sebagai berikut:

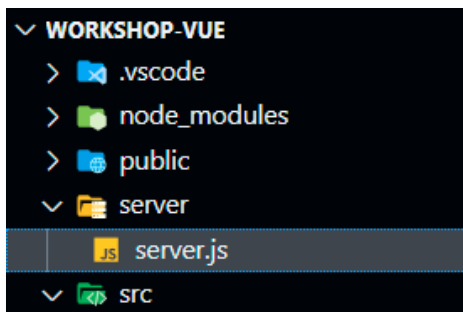


Kemudian buka kembali komentar pada bagian `router.beforeEach()` agar intercept di halaman Login aktif kembali.

## Membuat Server Backend

Server backend ini akan berisikan kode – kode pemrograman untuk mengakses **Stytch** sebagai penyedia autentikasi user.

- 1) Buat direktori baru pada folder root dengan nama **server** dan buat file baru didalamnya dengan nama **server.js**:



- 2) Buka terminal VSCode dan tuliskan kode berikut untuk membuat file **package.json** baru:  
**\$ npm init -y**

```
PS D:\workshop-vue\server> npm init -y
Wrote to D:\workshop-vue\server\package.json:

{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

- 3) Install beberapa dependencies yang dibutuhkan untuk server, meliputi: **Express**, **dotenv**, **Cors**, dan **Stytch**:  
**\$ npm install express dotenv cors stytch**
- 4) Install dependencies tambahan yang berguna untuk otomatisasi restart server: **nodemon**  
**\$ npm install -D nodemon**
- 5) Lihat pada file **server/package.json** untuk memastikan seluruh dependencies telah terpasang:

```
server > package.json > ...
1  {
2    "name": "server",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1",
8      "start": "node server.js"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "cors": "^2.8.5",
15     "dotenv": "^16.0.3",
16     "express": "^4.18.2",
17     "stytch": "^7.0.1"
18   },
19   "devDependencies": {
20     "nodemon": "^2.0.22"
21   }
22 }
```

6) Ubah kode pada file package.json:

```
server > package.json > ...
1  {
2    "name": "server",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "type": "module",
7    "scripts": {
8      "dev": "nodemon server.js"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "cors": "^2.8.5",
15     "dotenv": "^16.0.3",
16     "express": "^4.18.2",
17     "stytch": "^7.0.1"
18   },
19   "devDependencies": {
20     "nodemon": "^2.0.22"
21   }
22 }
```



## Membuat Konfigurasi Environment

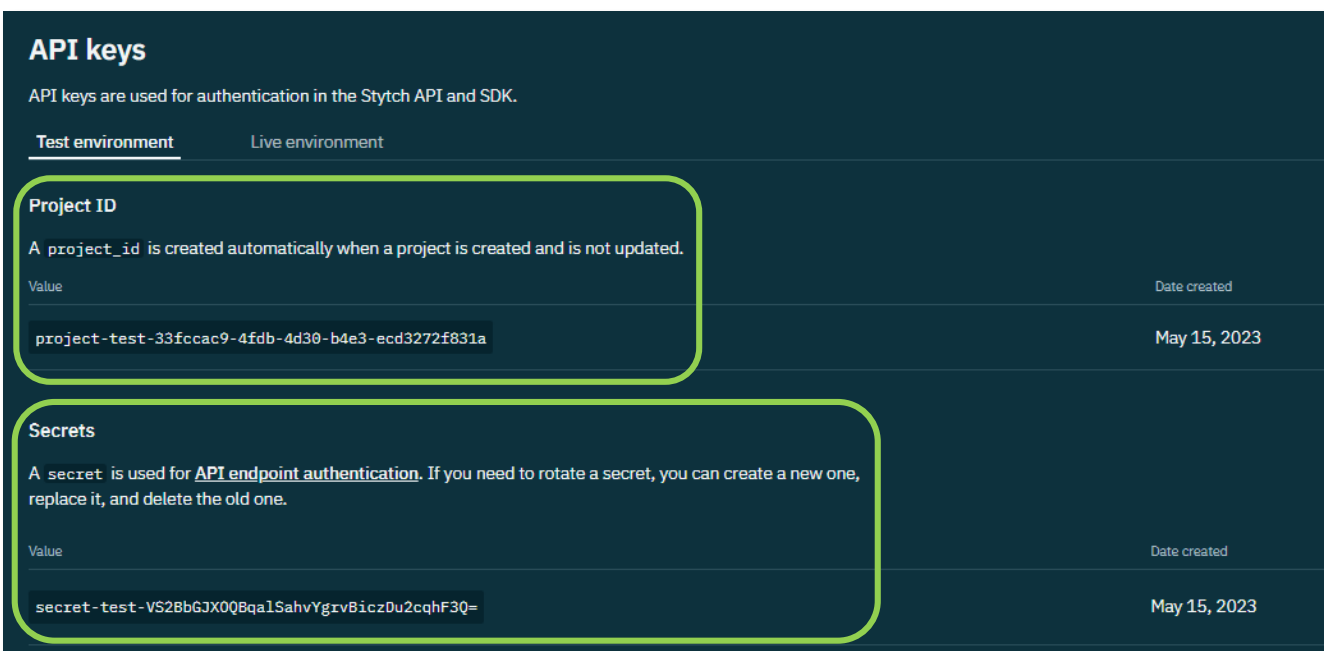
Konfigurasi environment disini berkaitan dengan Styth sebagai penyedia API untuk proses autentikasi user, oleh Styth telah disediakan Key sebagai konektor antara aplikasi Vue dengan server. Langkah – langkah konfigurasinya sebagai berikut:

- 1) Buat sebuah file baru pada **server/.env** dan isikan beberapa atribut berikut:

```
server > ❏ .env
1  PROJECT_ID="project-test-33fccac9-4fdb-4d30-b4e3-ecd3272f831a"
2  SECRET="secret-test-VS2BbGJX0QBqalSahvYgrvBiczDu2cqhF3Q="
```

**PROJECT\_ID** dan **SECRET** merupakan kode yang disediakan oleh Styth, untuk mendapatkannya terlebih dahulu buat akun pada <https://styth.com>

- 2) Daftar akun baru pada Styth untuk mendapatkan **API Keys**:



**API keys**

API keys are used for authentication in the Styth API and SDK.

Test environment   Live environment

**Project ID**

A `project_id` is created automatically when a project is created and is not updated.

Value	Date created
project-test-33fccac9-4fdb-4d30-b4e3-ecd3272f831a	May 15, 2023

**Secrets**

A `secret` is used for **API endpoint authentication**. If you need to rotate a secret, you can create a new one, replace it, and delete the old one.

Value	Date created
secret-test-VS2BbGJX0QBqalSahvYgrvBiczDu2cqhF3Q=	May 15, 2023

## Membuat Server Node dengan Express

Server Node membutuhkan sebuah library Express sebagai frameworknya, setelah Express terpasang pada server kemudian tuliskan kode berikut pada file **server/server.js**:

```
import express from "express";
import dotenv from "dotenv";
import cors from "cors";
import { envs, Client } from "stytch";

dotenv.config();

const app = express();

const client = new Client({
  project_id: process.env.PROJECT_ID,
  secret: process.env.SECRET,
  env: envs.test,
});

const port = process.env.PORT || 3333;

app.use(cors());
app.use(express.json());

app.post("/register", async (req, res) => {
  const { email, password } = req.body;

  try {
    const response = await client.passwords.create({
      email,
      password,
      session_duration_minutes: 60,
    });
    res.json({
      success: true,
      message: "User created successfully",
      token: response.session_token,
    });
  } catch (err) {
    console.log(err);
    res.json({ success: false, message: err.error_message, err: err });
  }
});
```

```
app.post("/login", async (req, res) => {
  const { email, password } = req.body;

  try {
    const response = await client.passwords.authenticate({
      email,
      password,
      session_duration_minutes: 60,
    });
    res.json({
      success: true,
      message: "User logged in successfully",
      token: response.session_token,
    });
  } catch (err) {
    console.log(err);
    res.json({ success: false, message: err.error_message, err: err });
  }
});

app.post("/authenticate", async (req, res) => {
  const { session_token } = req.body;

  try {
    await client.sessions.authenticate({
      session_token,
    });
    res.json({
      success: true,
      message: "Token is valid",
    });
  } catch (err) {
    console.log(err);
    res.json({ success: false, message: err.error_message, err: err });
  }
});

app.post("/logout", async (req, res) => {
  const { session_token } = req.body;

  try {
    await client.sessions.revoke({
      session_token,
    });
  }
});
```

```
res.json({
  success: true,
  message: "Successfully logged out",
});
} catch (err) {
  console.log(err);
  res.json({ success: false, message: err.error_message, err: err });
}
});

app.listen(port, () =>
  console.log(`Server started on http://127.0.0.1:${port}`)
);
```

Jalankan kode diatas dengan perintah \$ **npm run dev**

```
PS D:\workshop-vue\server> npm run dev

> server@1.0.0 dev
> nodemon server.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server started on http://127.0.0.1:3333
█
```

## Membuat Akses dari Vue ke Server

### Halaman Register

Buka kembali file `src/views/Register.vue` dan tambahkan kode berikut:

```
<script setup>
import { ref } from 'vue'
import { useRouter } from 'vue-router'

const email = ref('')
const password = ref('')
const conf_password = ref('')

const router = useRouter()

const Register = async () => {
  if (!email.value || !password.value || !conf_password.value) {
    return alert('Please fill in all fields!')
  }
  if (password.value !== conf_password.value) {
    return alert('Passwords do not match!')
  }

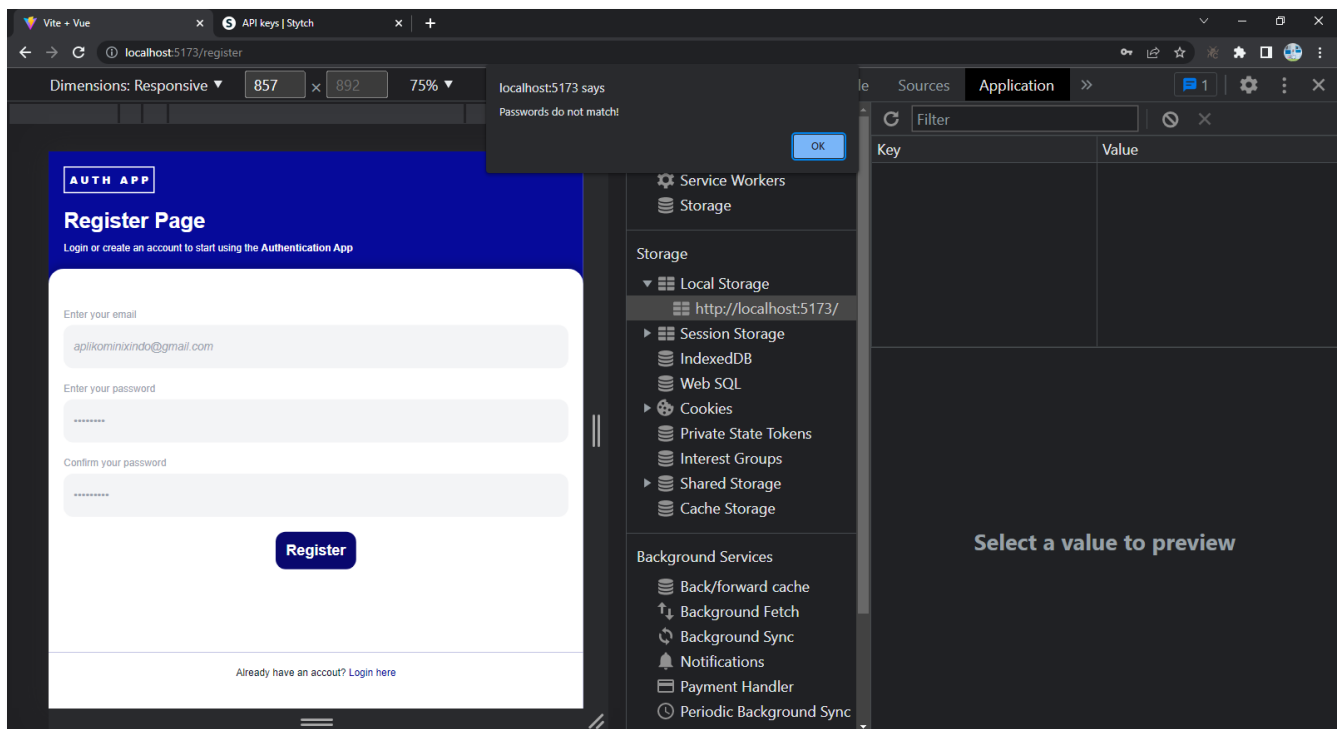
  const response = await fetch('http://127.0.0.1:3333/register', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      email: email.value, password: password.value,
    })
  }).then(response => response.json())

  if (response.success) {
    localStorage.setItem('token', response.token)
    router.push('/')
  } else {
    alert(response.message)
  }
}
</script>
```

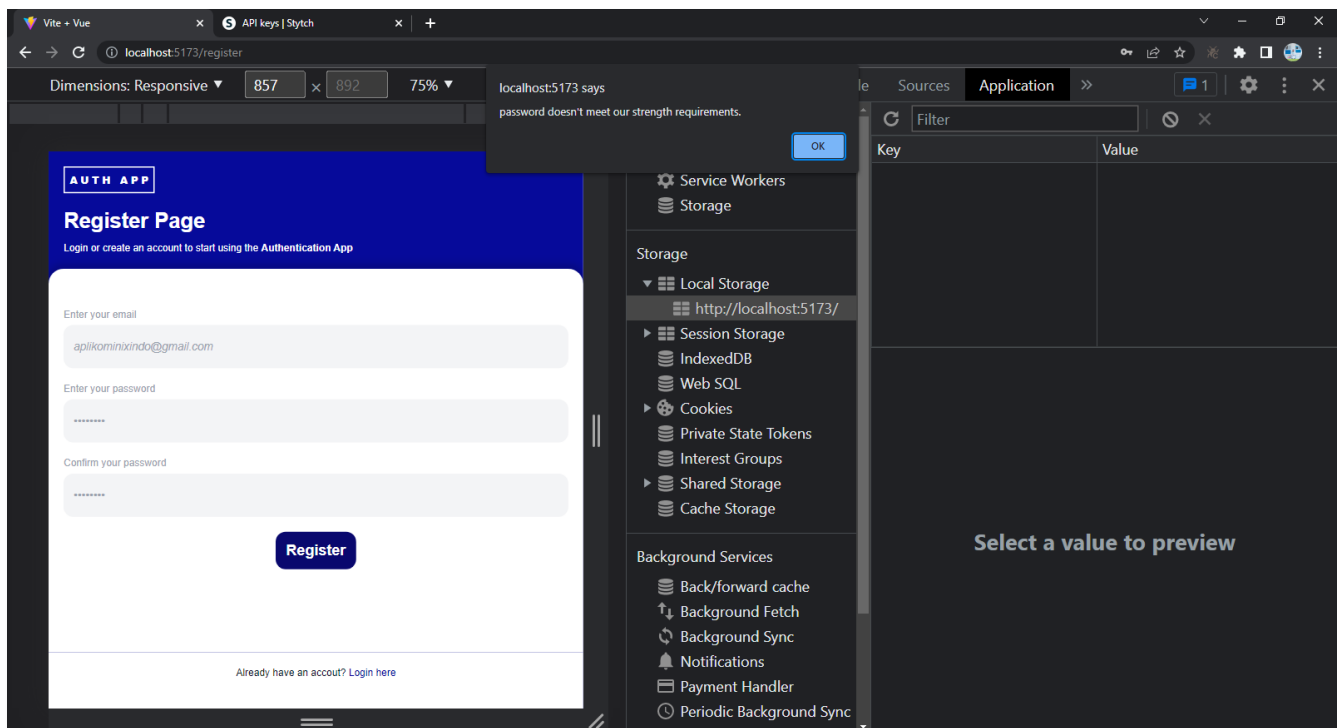
```
<template>
  <main>
    <header>
      <h1 class="logo">AUTH APP</h1>
      <h2>Register Page</h2>
      <p>
        Login or create an account to start using the
        <strong>Authentication App</strong>
      </p>
    </header>

    <form @submit.prevent="Register">
      .....
    </form>
  </main>
</template>
```

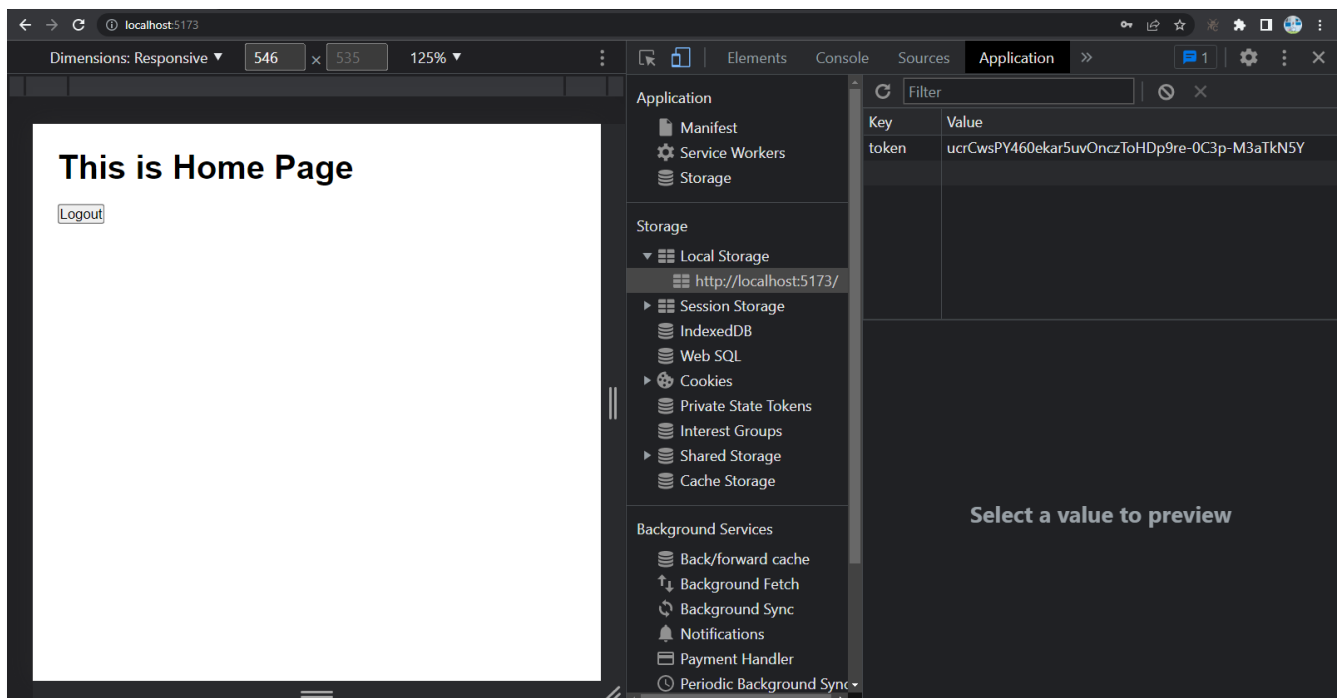
Untuk saat ini berikan kembali komentar pada `router.beforeEach()` untuk uji coba Register.



Gambar diatas menunjukkan bahwa Confirm Password tidak sesuai dengan Password.



Gambar diatas menunjukkan bahwa Password yang dibuat tidak memenuhi syarat kekuatan.



Gambar diatas menunjukkan bahwa proses Register telah berhasil dan masuk ke halaman Home.

Kita juga dapat melihat User yang telah berhasil Register pada Dashboard Stytlch:

### User management

View your Stytlch users. Questions? Contact us

**Test environment** Live environment

All users ▼ Show: User ID, Email ID, E... ▼ 🔍 Search by Name / User ID / Email / Phone number

User ID	Email	Email ID	Phone number	Name	Created at
user-test...-e30dfda36bde	aplikominixindo@gmail.com	email-test...-424c83dc561a	—	—	May 24, 2023
user-test...-000449e66eee	yusufrizalh@gmail.com	email-test...-969079d714d0	—	—	May 15, 2023

[Back to all users](#)

## User aplikominixindo@gmail.com

View user information about aplikominixindo@gmail.com. Questions? Contact us Delete user

Name

—

User ID

user-test-31271b4a-5417-4fa5-99de-e30dfda36bde

[View event logs for user](#)

Status

Active

Email address

aplikominixindo@gmail.com

Email ID

email-test-11b01e19-269d-40bd-b787-424c83dc561a

Phone number

—

Phone number ID

—

Created at

May 24, 2023 (2023-05-24T13:57:25.000Z)

```
{
  "user_id": "user-test-31271b4a-5417-4fa5-99de-e30dfda36bde",
  "name": {
    "first_name": "",
    "middle_name": "",
    "last_name": ""
  },
  "emails": [
    {
      "email_id": "email-test-11b01e19-269d-40bd-b787-424c83dc561a",
      "email": "aplikominixindo@gmail.com",
      "verified": false
    }
  ],
  "status": "active",
  "phone_numbers": [],
  "webauthn_registrations": [],
  "created_at": "2023-05-24T13:57:25.000Z",
  "providers": [],
  "totps": [],
  "crypto_wallets": [],
  "password": {
    "password_id": "password-test-e0c9782a-261d-4782-8f74-68e62132252b",
    "requires_reset": false
  },
  "biometric_registrations": [],
  "trusted_metadata": {},
  "untrusted_metadata": {}
}
```



## Membuat Logout

Buka kembali file `src/views/Home.vue` dan tambahkan kode berikut:

```
<script setup>
import { useRouter } from 'vue-router'

const router = useRouter()

const Logout = async () => {
  const response = await fetch('http://127.0.0.1:3333/logout', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      session_token: localStorage.getItem('token')
    })
  }).then(response => response.json())

  if (response.success) {
    localStorage.removeItem('token')
    router.push('/login')
  } else {
    alert(response.message)
  }
}
</script>

<template>
  <main>
    <h1>This is Home Page</h1>
    <button @click="Logout">Logout</button>
  </main>
</template>
```

Ketika tombol Logout ditekan maka Stytc memberikan perintah untuk menghapus token dari web browser, sehingga autentikasi dihilangkan dan kembali ke halaman Login.

## Membuat Login

Buka kembali file **src/views/Login.vue** dan tambahkan kode berikut:

```
<script setup>
import { ref } from 'vue'
import { useRouter } from 'vue-router'

const email = ref('')
const password = ref('')

const router = useRouter()

const Login = async () => {
  if (!email.value || !password.value) {
    return alert('Please fill in all fields!')
  }

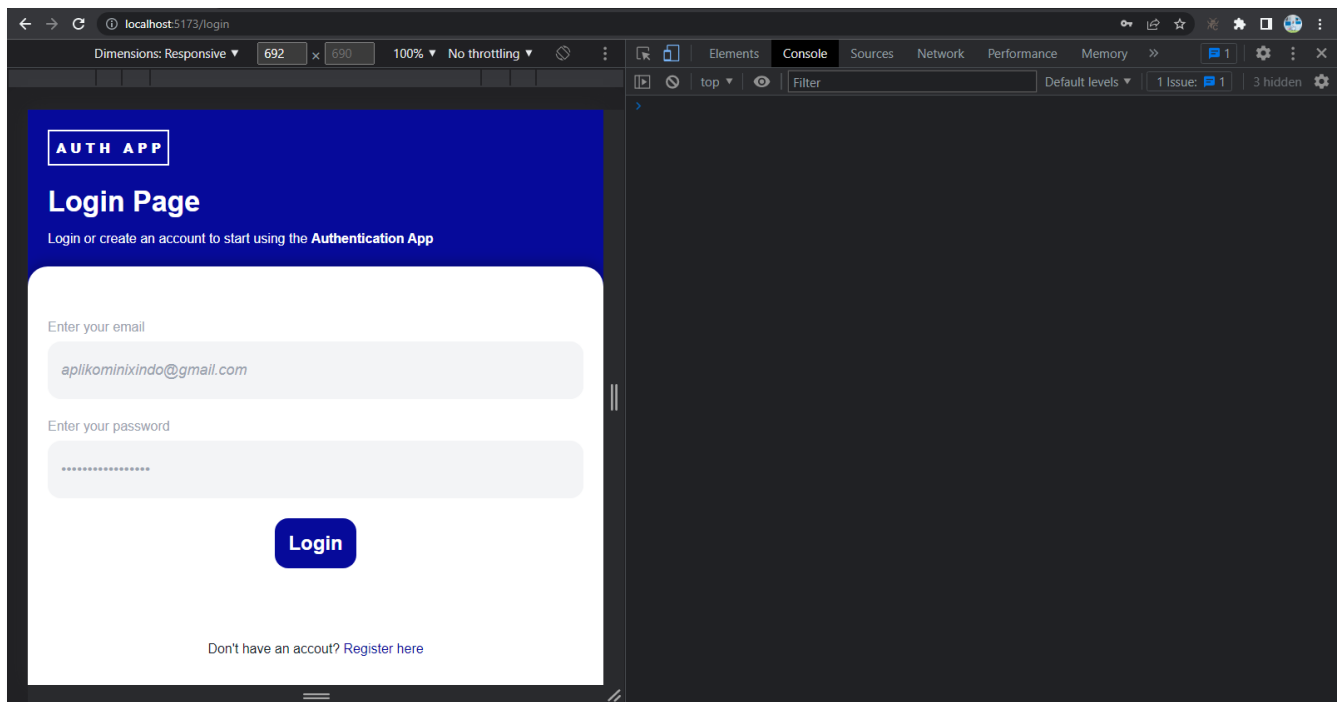
  const response = await fetch('http://127.0.0.1:3333/login', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      email: email.value, password: password.value,
    })
  }).then(response => response.json())

  if (response.success) {
    localStorage.setItem('token', response.token)
    router.push('/')
  } else {
    alert(response.message)
  }
}
</script>
```

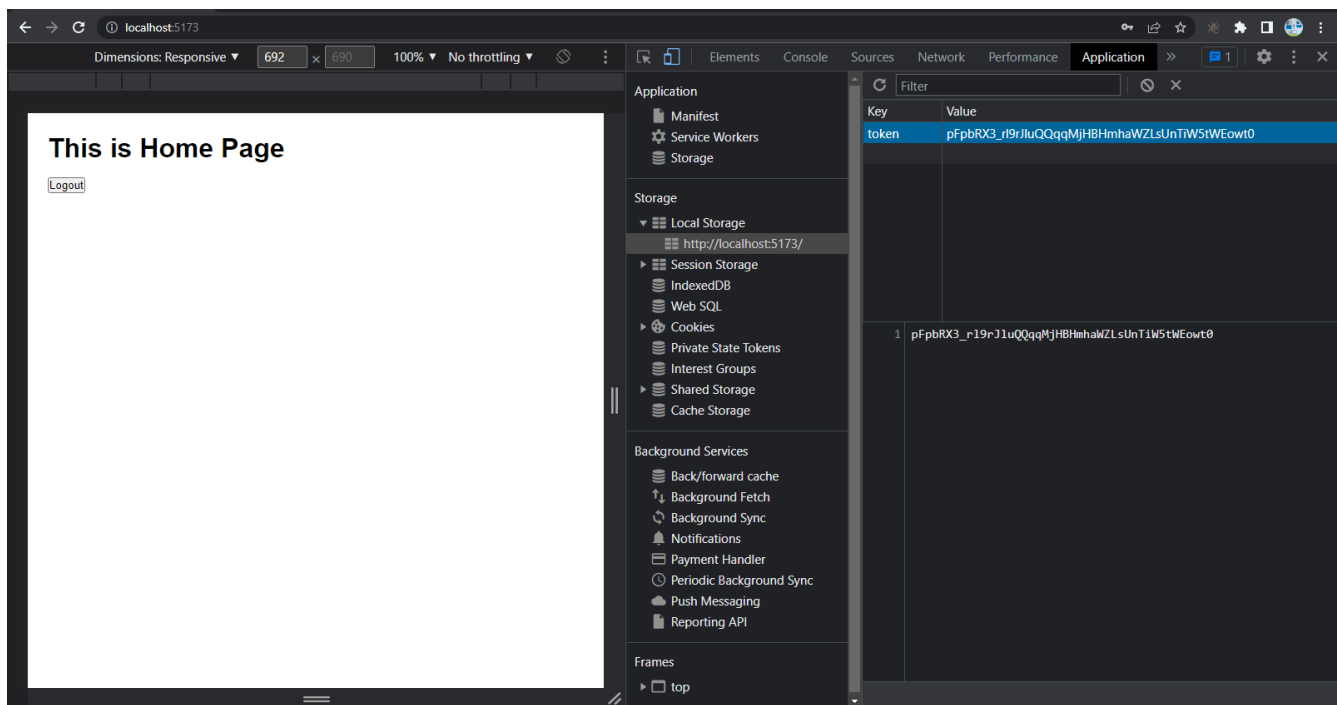
```
<template>
  <main>
    <header>
      <h1 class="logo">AUTH APP</h1>
      <h2>Login Page</h2>
      <p>
        Login or create an account to start using the
        <strong>Authentication App</strong>
      </p>
    </header>

    <form @submit.prevent="Login">
      .....
    </form>
  </main>
</template>
```

Buka browser dan arahkan pada halaman Login:



Gambar diatas menunjukkan Login dengan email dan password yang sudah terdaftar sebelumnya.



Gambar diatas menunjukkan proses Login berhasil dan muncul token autentikasi.

### Mengembalikan Proses Intercept Login

Buka kembali file **src/router.js** dan ubah kode pada **router.beforeEach()** sebagai berikut:

```
router.beforeEach(async (to, from, next) => {
  if (to.matched.some((record) => record.meta.requiresAuth)) {
    // authentication check
    const token = localStorage.getItem("token");
    if (token) {
      // check if token is valid
      return next();
    }
    return next("/login");
  }
  next();
});
```