Portfolio Dashboard Template using Angular

Aayush Goel

**Introduction**

This project is a simple and clean dashboard template with robust log in and registration authentication.

The session is maintained using JSON Web Token(JWT)

A dashboard for any purpose requiring data and statistics can be built upon this template.

The app features a side-nav where multiple page links can be added and dynamically replaced on the same page without reloading.

The front-end for this project is built using Angular and the backend is handled using NodeJS with many of its modules.

MySQL has been selected as the database option.

**Front-End Development**

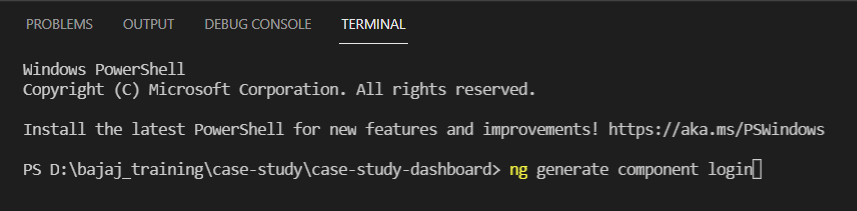
* **Modules Used**
* [Angular Material](https://material.angular.io/)
* [Highcharts](https://www.highcharts.com/)
* [Flex-layout](https://github.com/angular/flex-layout)
* [Routing Module](https://angular.io/guide/router)
* [HTTP Client Module](https://blog.angular-university.io/angular-http/)
* [Angular Forms Module](https://angular.io/guide/forms)
* [Angular Reactive Forms](https://angular.io/guide/reactive-forms)

**Commands Used**

The following commands were used to create components, modules, services, and auth-guards for the app app:

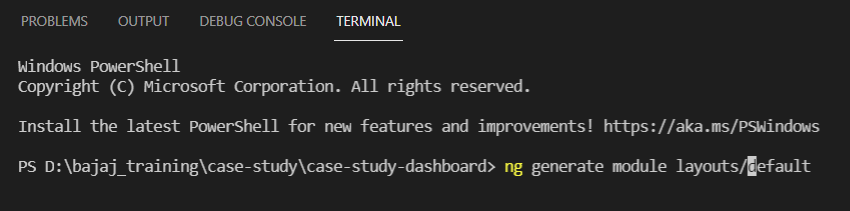
**Create Component**

* ng generate component <component name>



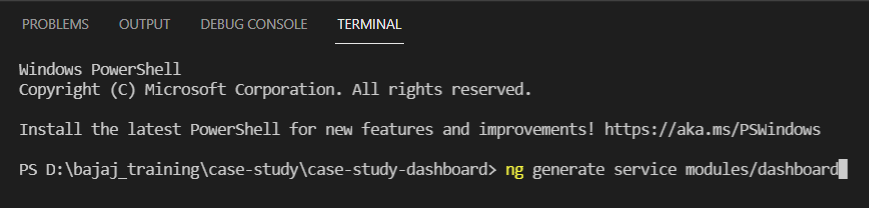
**Create Module**

* ng generate module <module name>



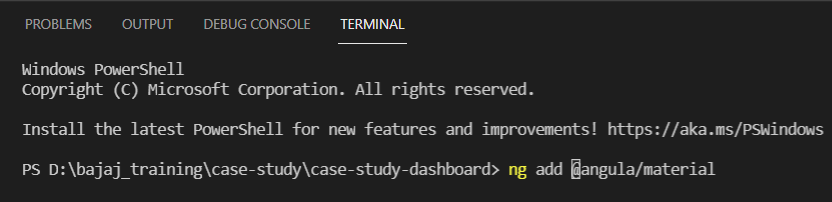
**Create Service**

* ng generate service <service name>



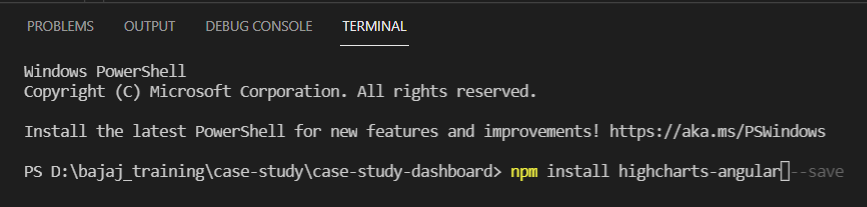
**Adding Angular Material**

* ng add @angular/material

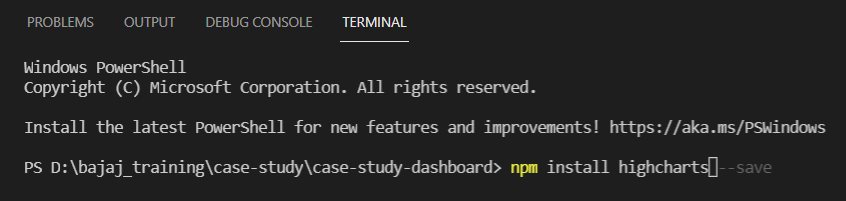


**Adding Highcharts**

* npm install highcharts-angular --save

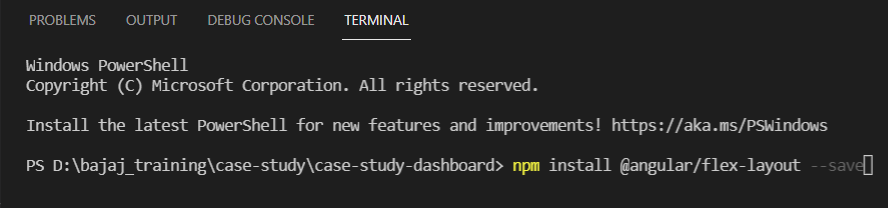


* npm install highcharts --save



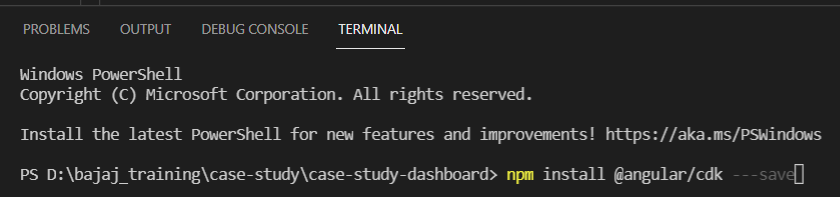
**Adding flex-layout**

* npm install @angular/flex-layout --save



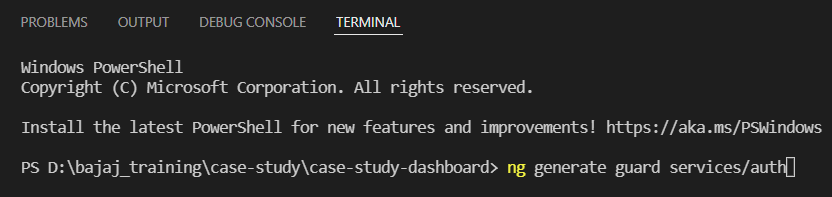
**Adding angular CDK(Component Dev Kit)**

* npm install @angular/cdk --save



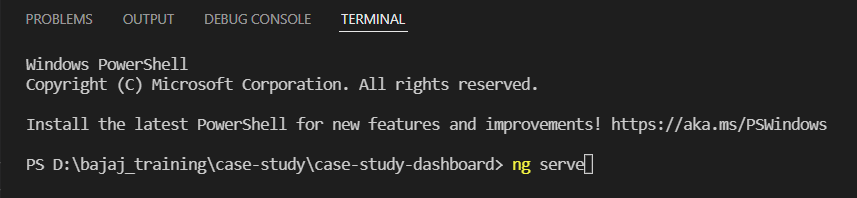
**Creating and Auth Guard**

* ng generate guard services/auth



**Running the project**

* ng server

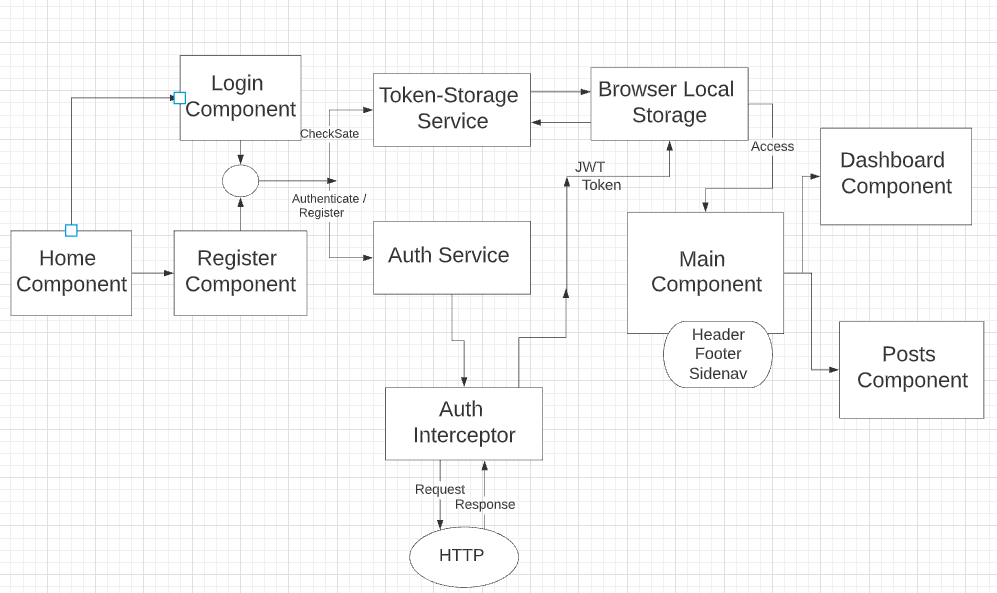


**Note :** Run the project on port 8081 using ng server –port 8081 as CORS has been configured to port 8081

**Angular Concepts Used**

* Components
* Directives
* Observables
* Templates
* Dependency injection
* Routing
* Modules
* Interpolation
* Event Binding
* @Injectable and Custom Service
* HTTP Client - GET POST PUT DELETE

**Application Flow**



**Back-end Development**

# Modules Used

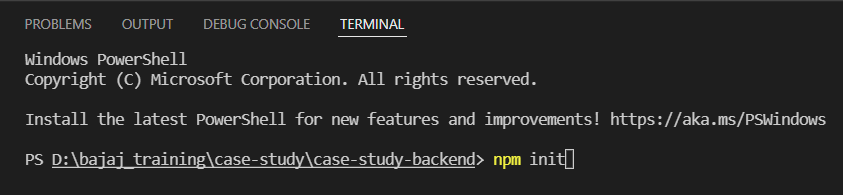
* [ExpressJS](http://expressjs.com/)
* [MySQL](https://www.npmjs.com/package/mysql2)
* [Sequelize](https://sequelize.org/)
* [Bcryptjs](https://www.npmjs.com/package/bcryptjs)
* [Jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken)
* [CORS](https://www.npmjs.com/package/cors)

# **Commands Used**

The following commands can be used to initialize the project, install the dependencies required and then run the program

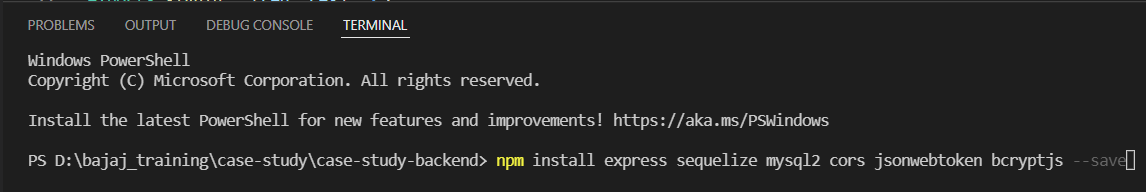
**Initializing the Project**

* npm init



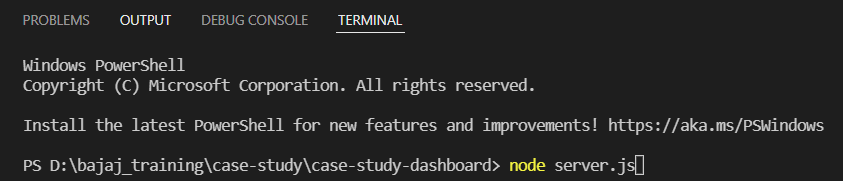
**Installing the Dependency packages**

* npm install express sequelize mysql2 cors jsonwebtoken bcryptjs –save

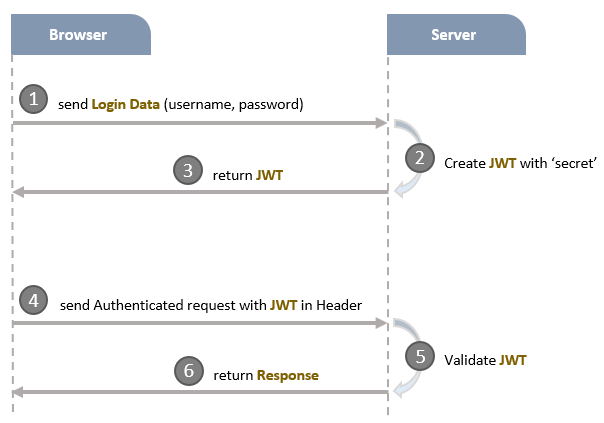


**Running the back-end NodeJS program**

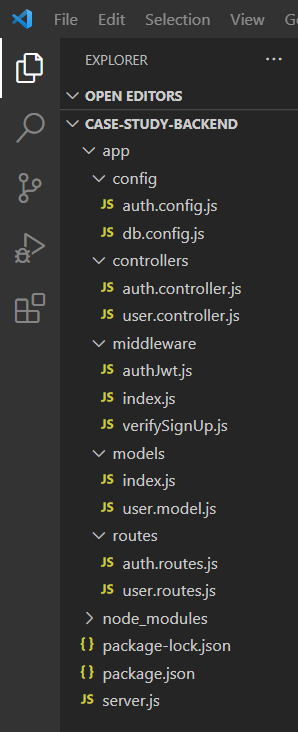
* node server.js



**Application Flow**



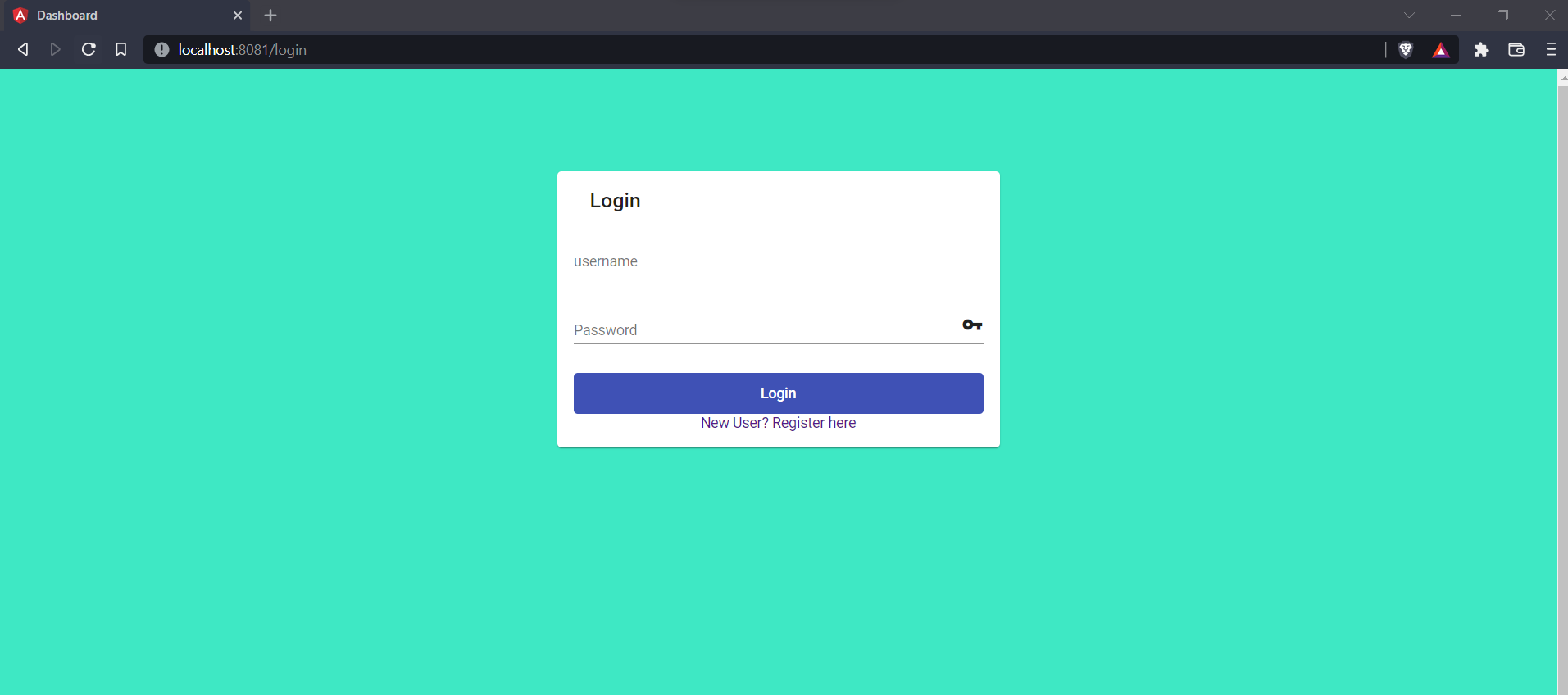
# **Project Structure**



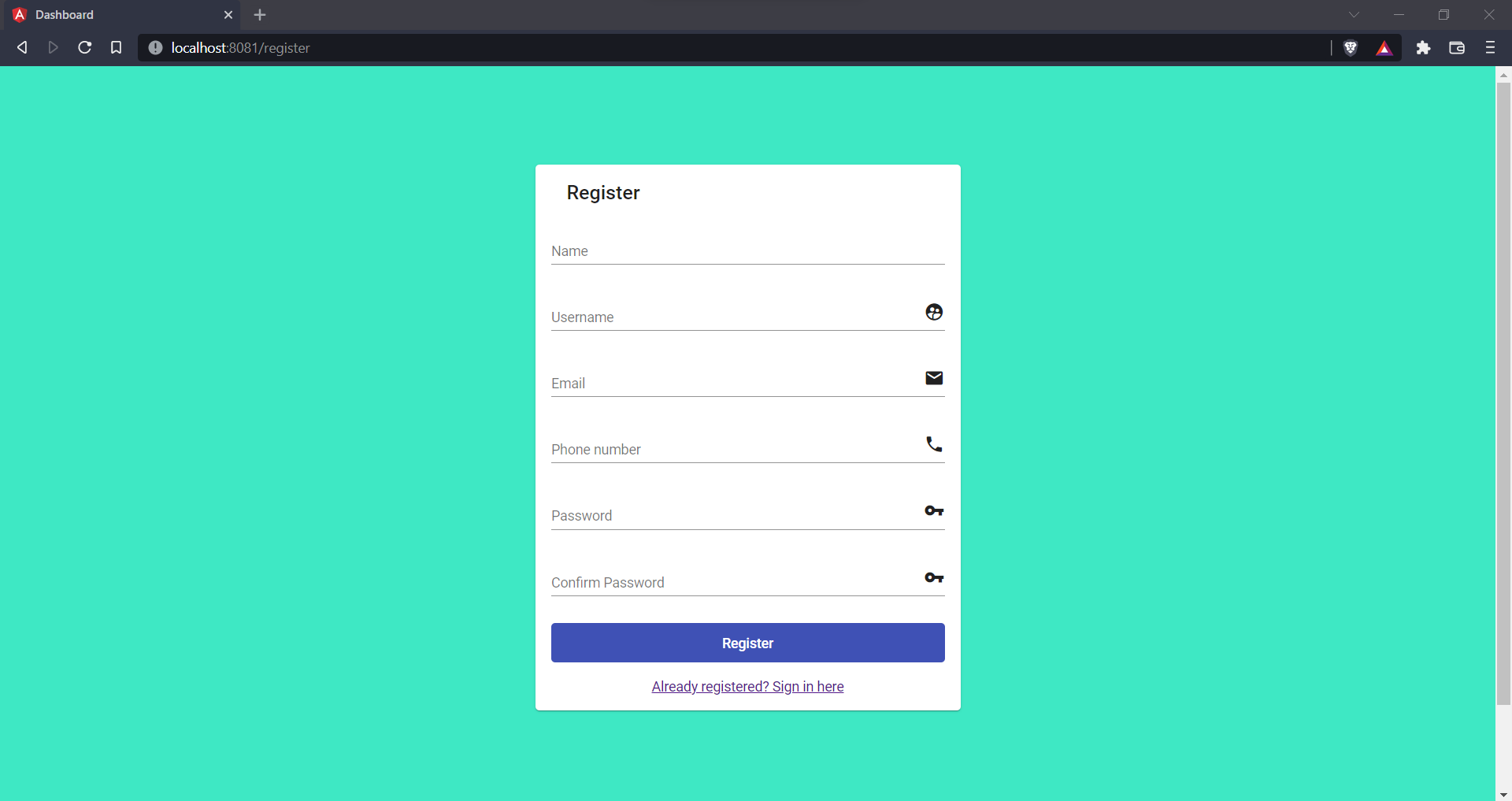
* **config**
* configure MySQL database & Sequelize
* configure Auth Key
* **routes**
* *auth.routes.js*: POST signup & signin
* *user.routes.js*: GET public & protected resources
* **middlewares**
* *verifySignUp.js*: check duplicate Username or Email
* *authJwt.js*: verify Token
* **controllers**
* *auth.controller.js*: handle signup & signin actions
* *user.controller.js*: return public & protected content
* **models** for Sequelize Models
* *user.model.js*
* *server.js*: import and initialize necessary modules and routes, listen for connections.

Screenshots

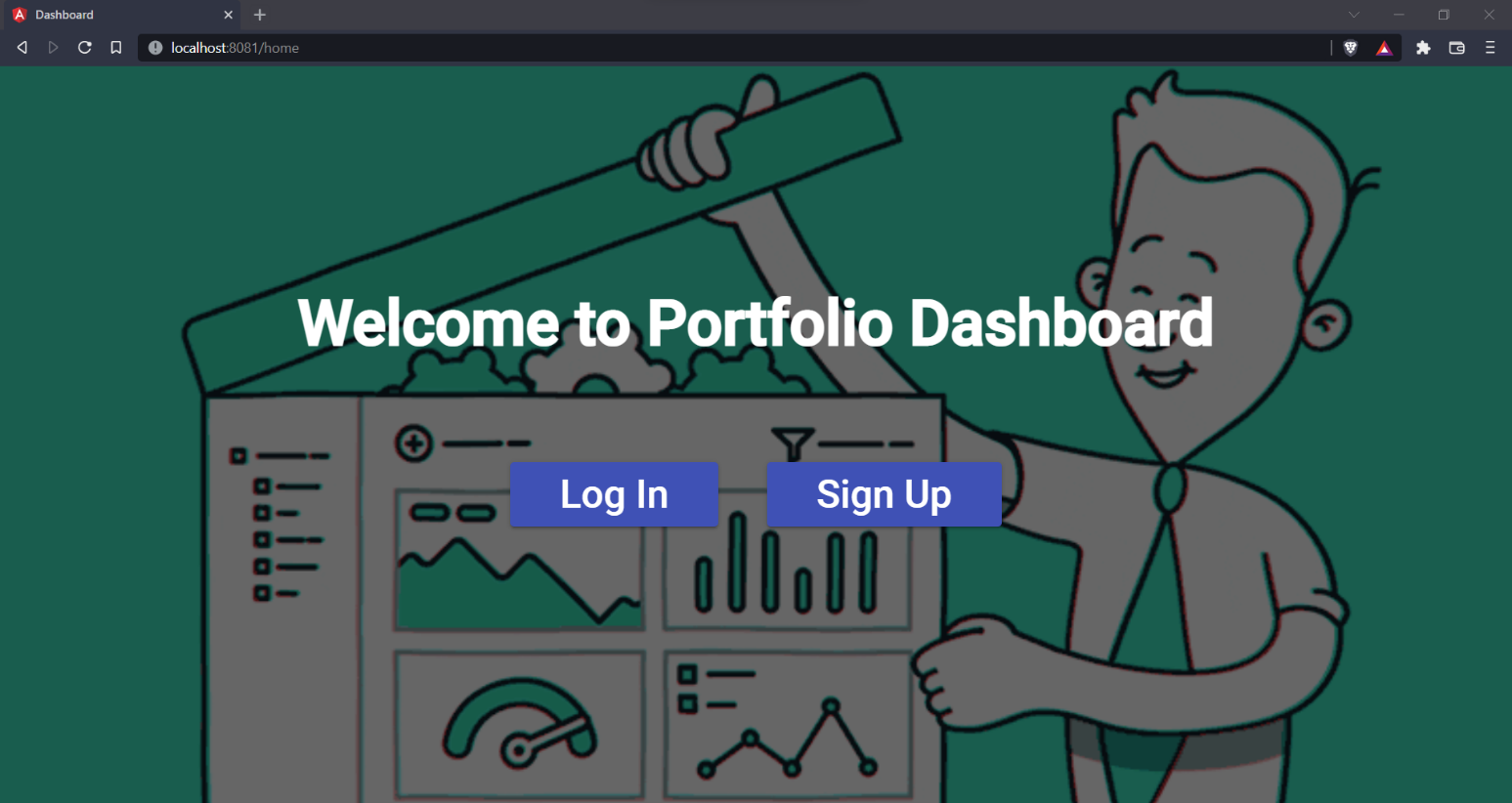
**Login Page**



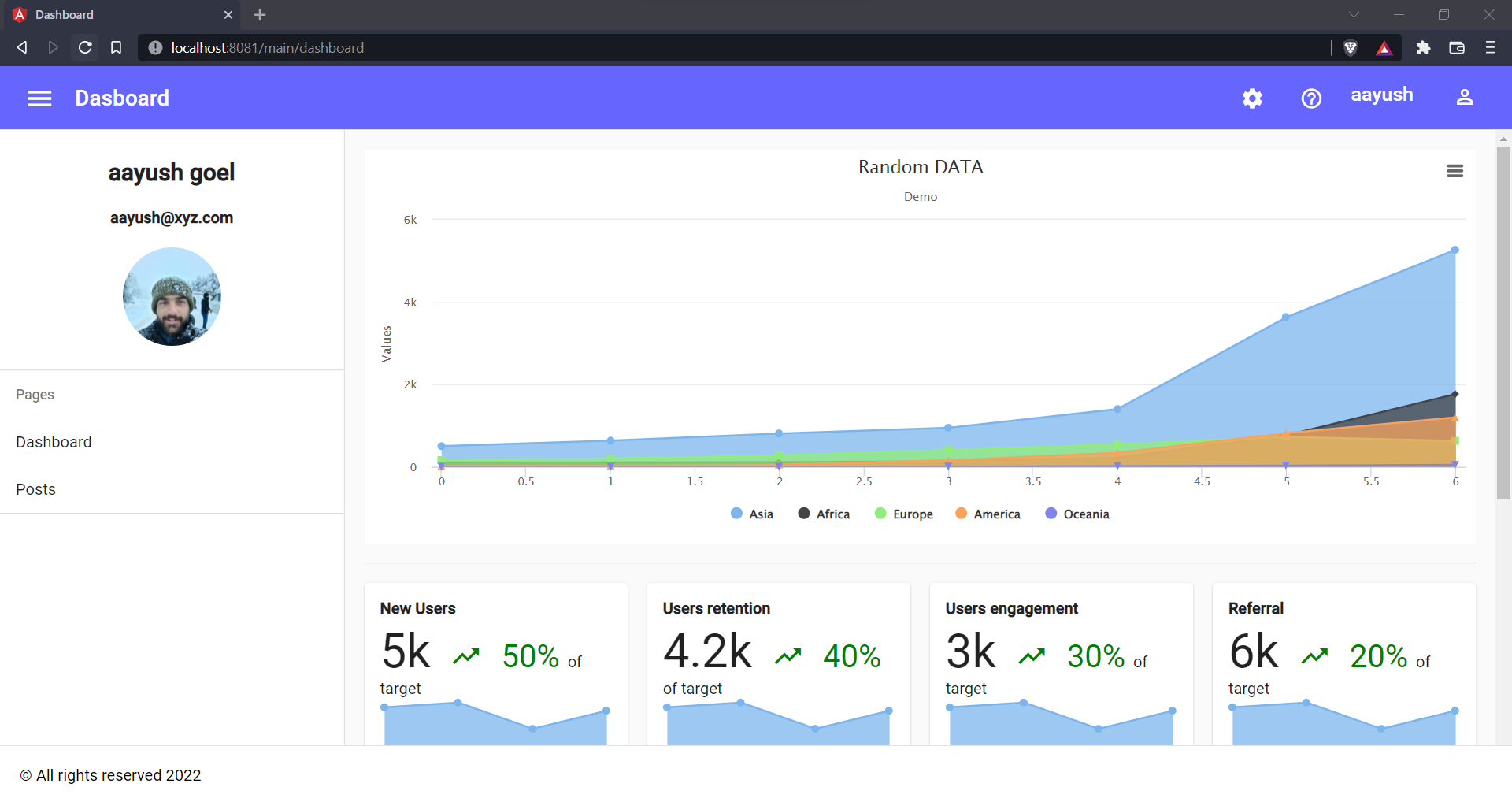
**Register Page**

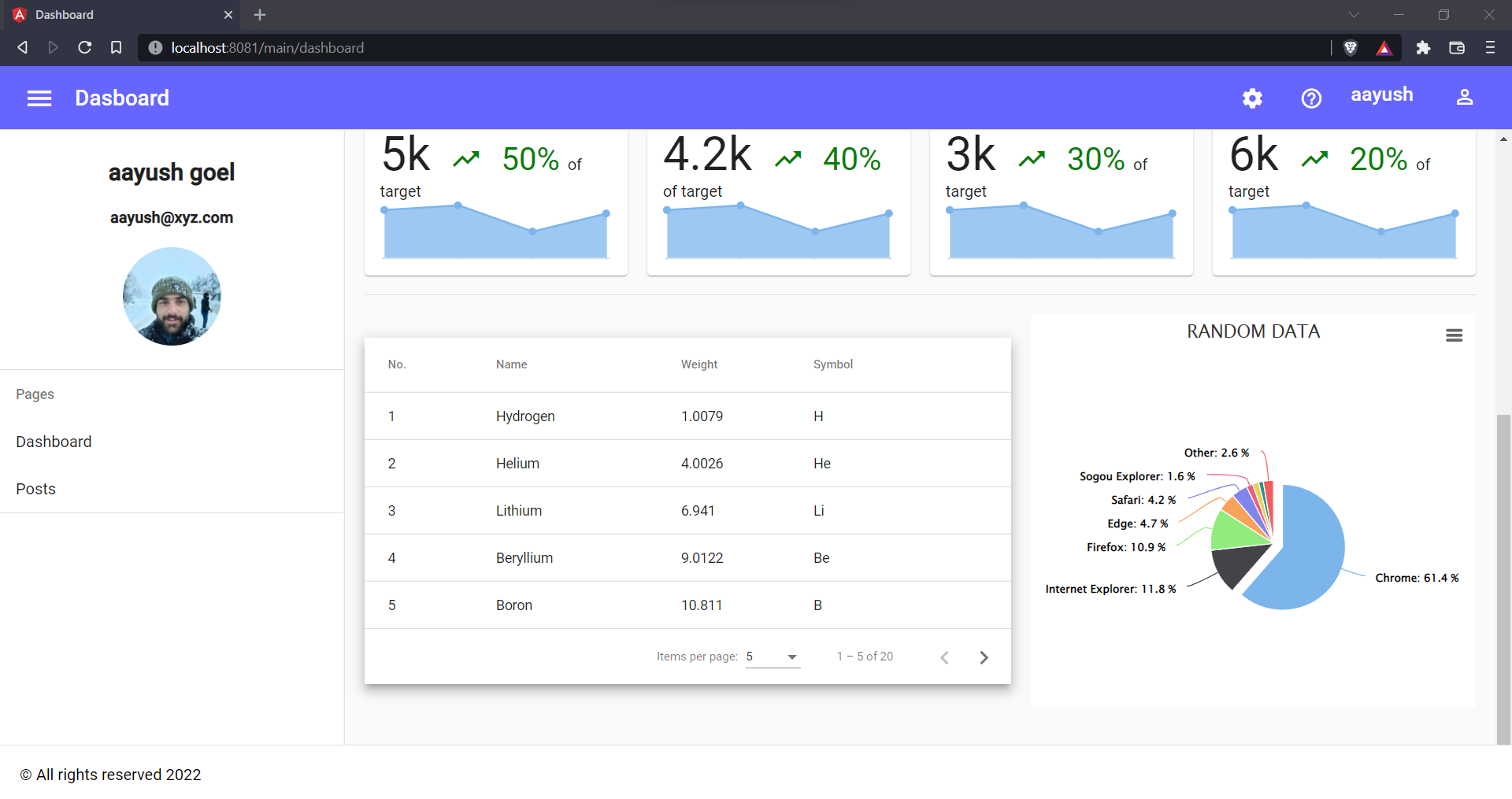


**Home Page**

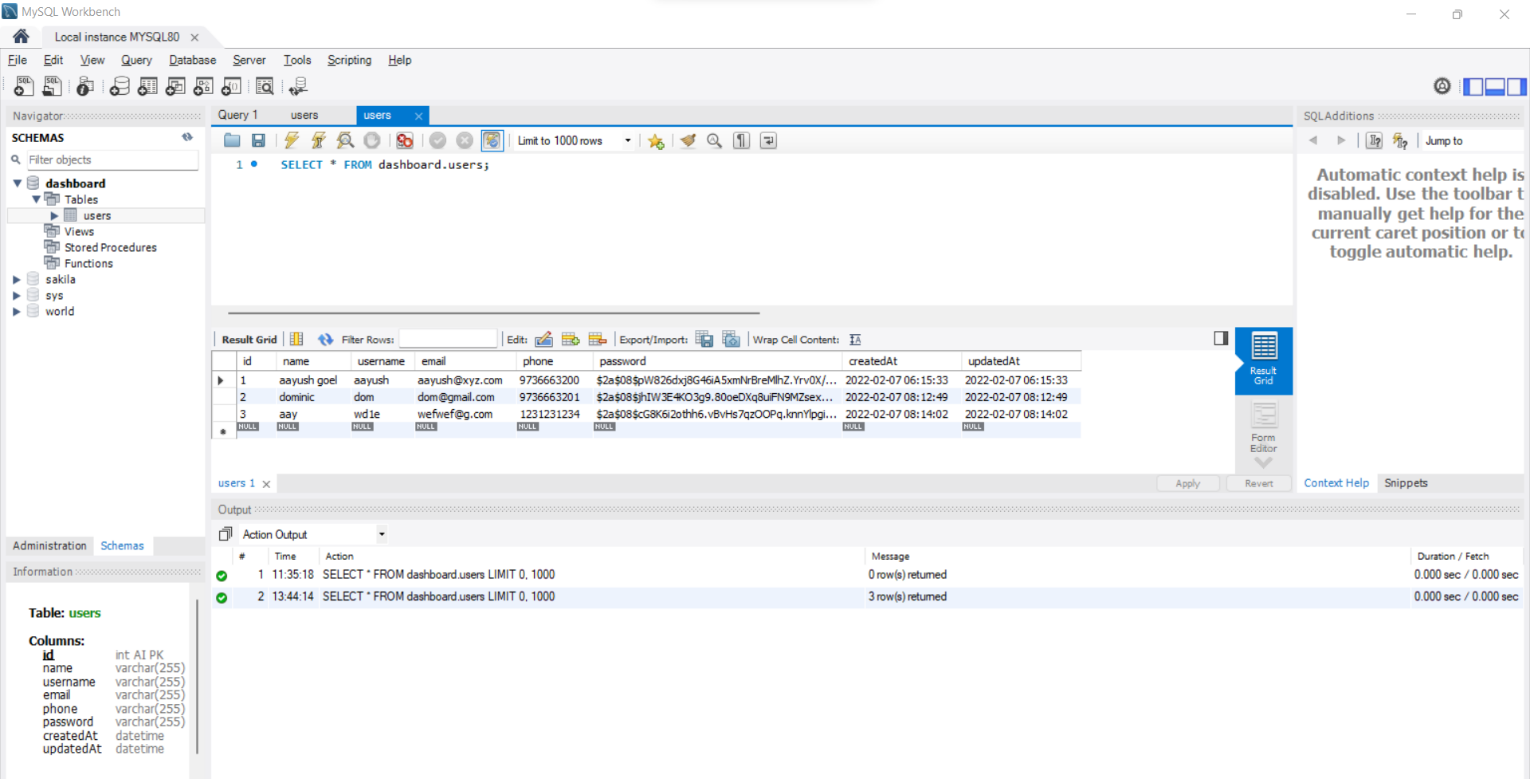


**Dashboard Page**



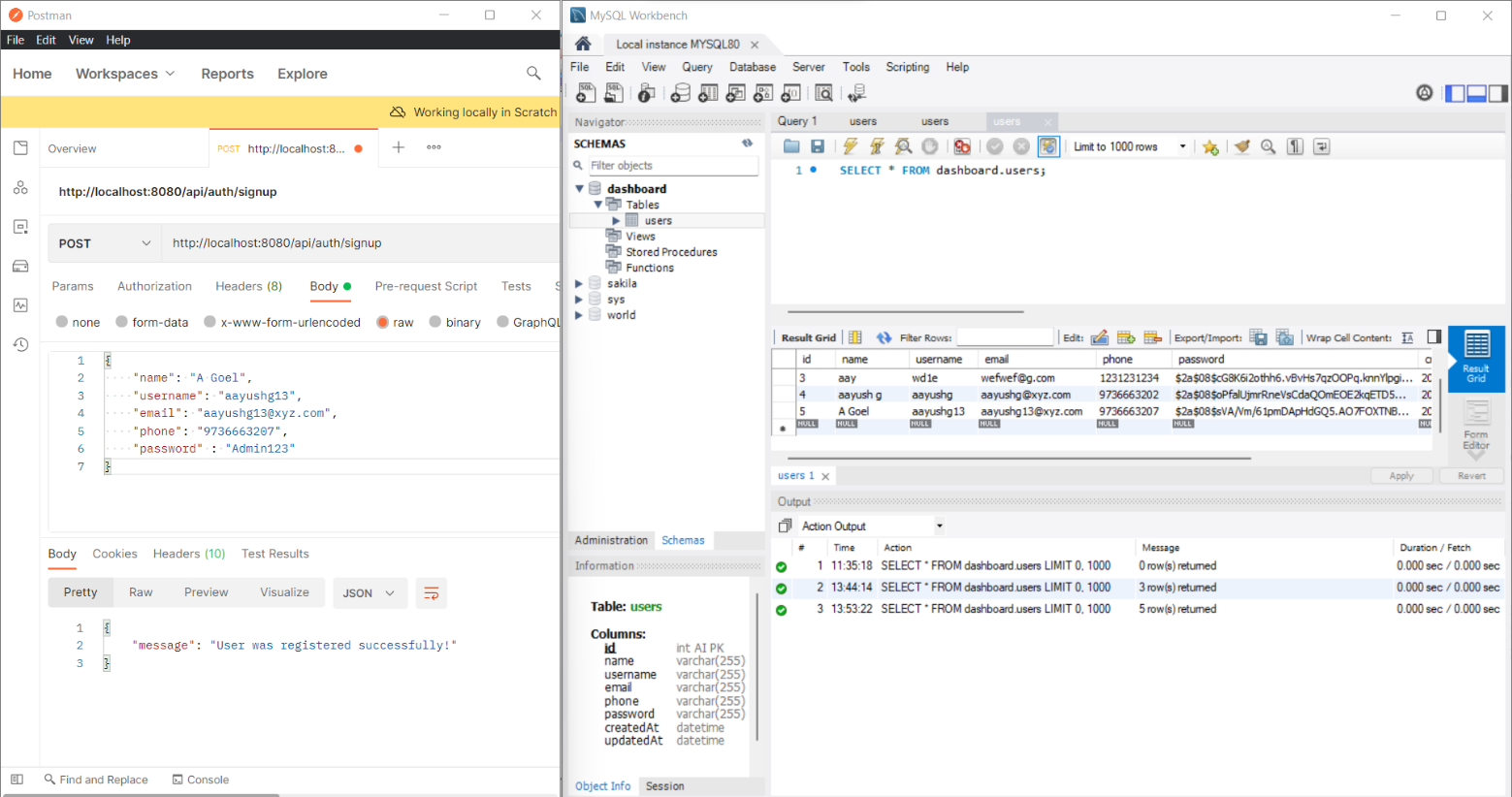


**Database(MySQL)**

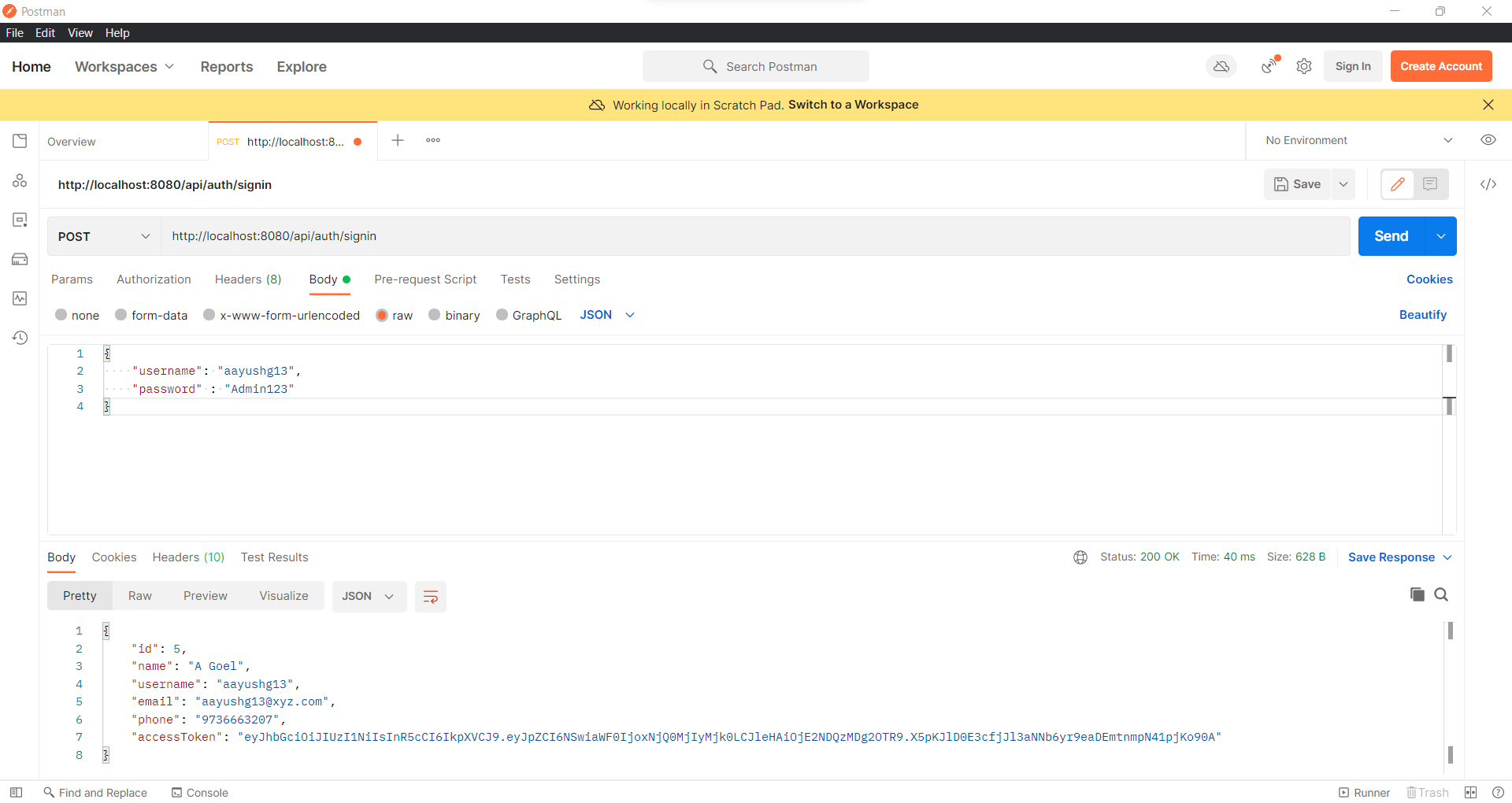


**Postman API Testing**

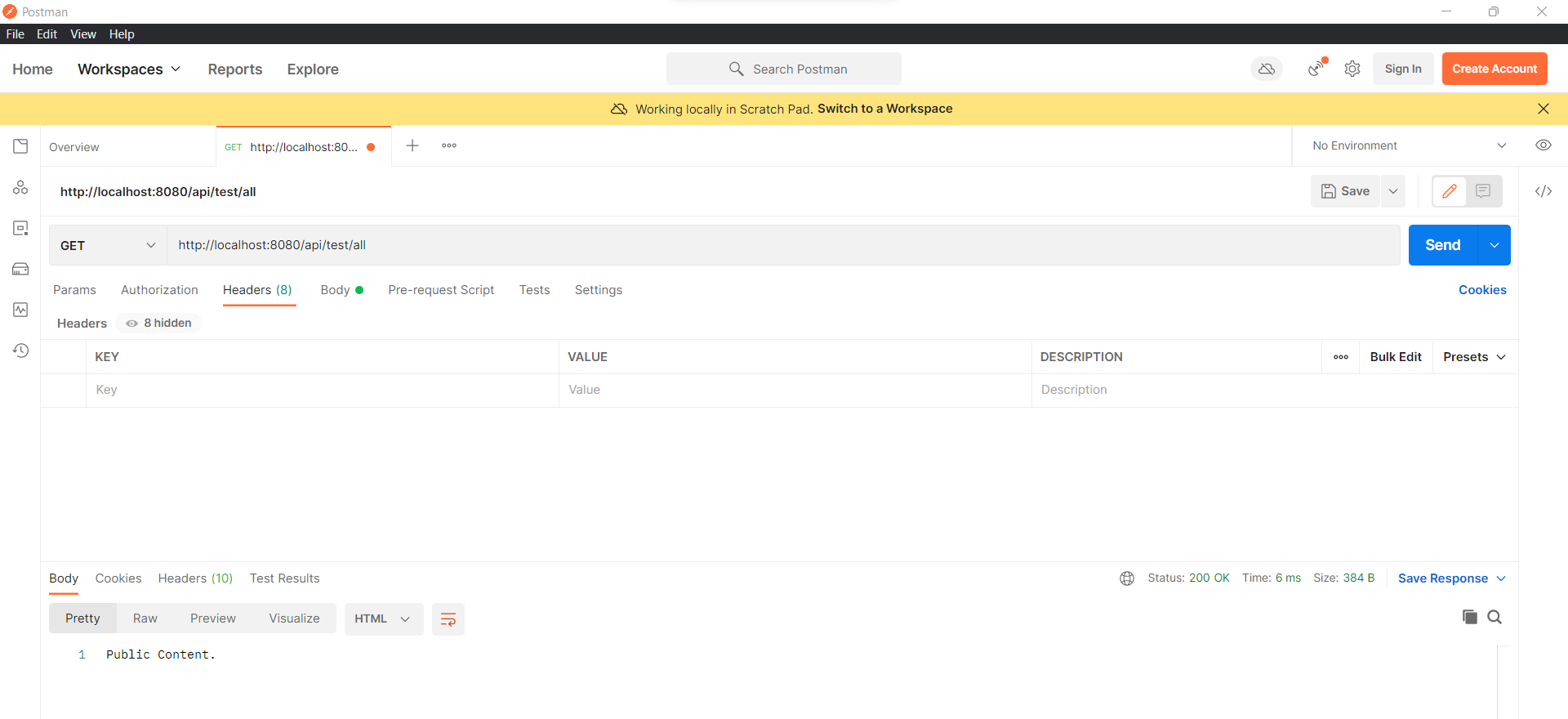
**Signup**



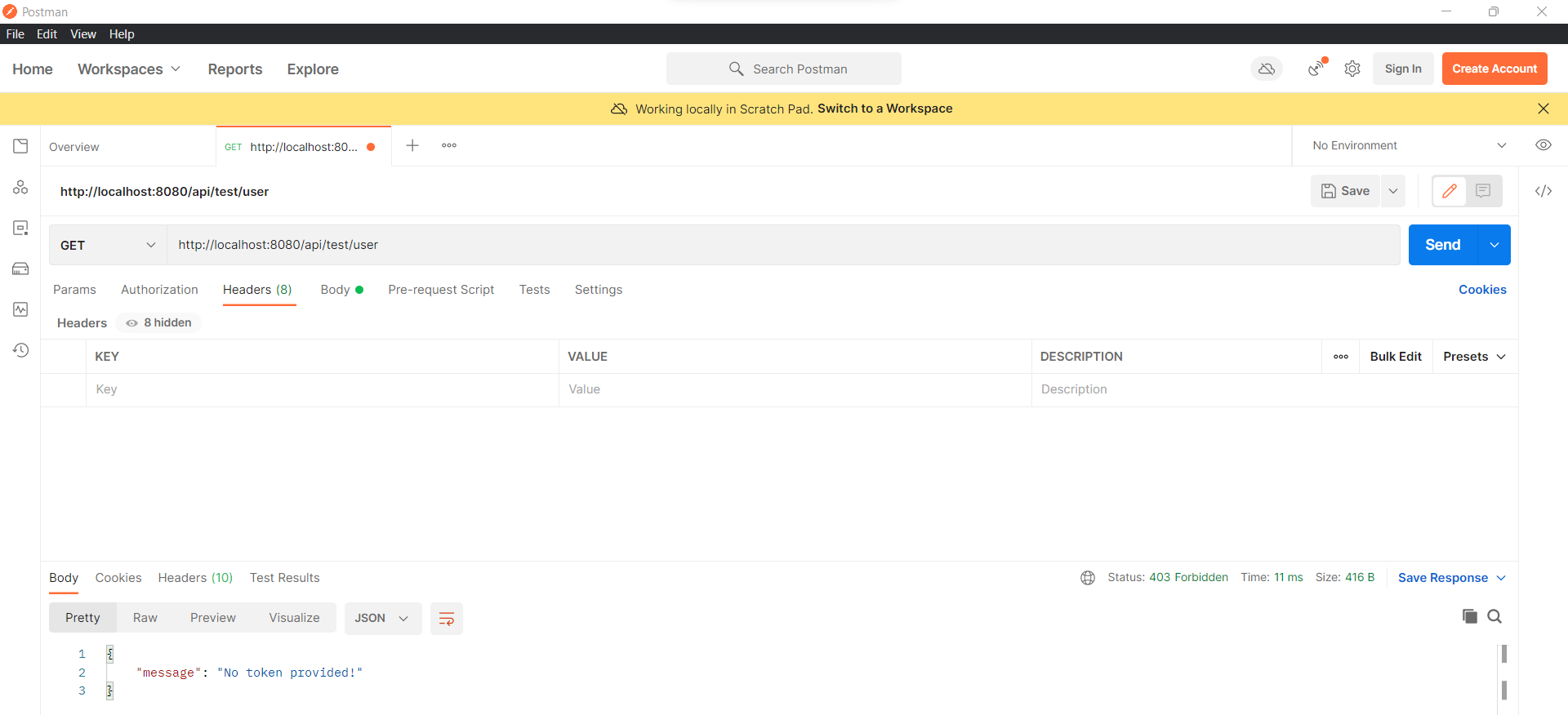
**Signin**



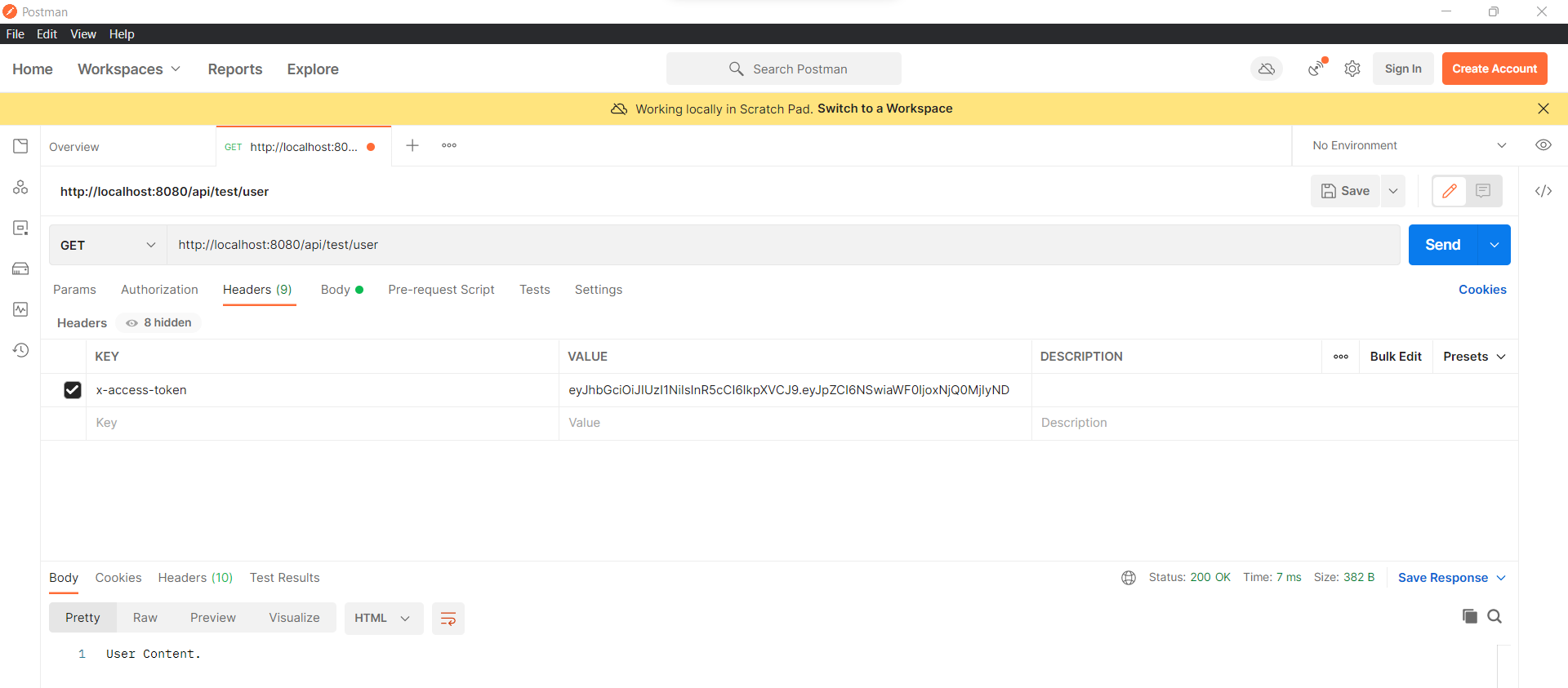
**Accessing Public Content**



**Accessing Private Content(failed without token)**

****

**Accessing Private Content(successful with token)**

****