

launch_code



- Codergirl – Frontend
- Unit 2 - Class 16
- Feb 1, 2021

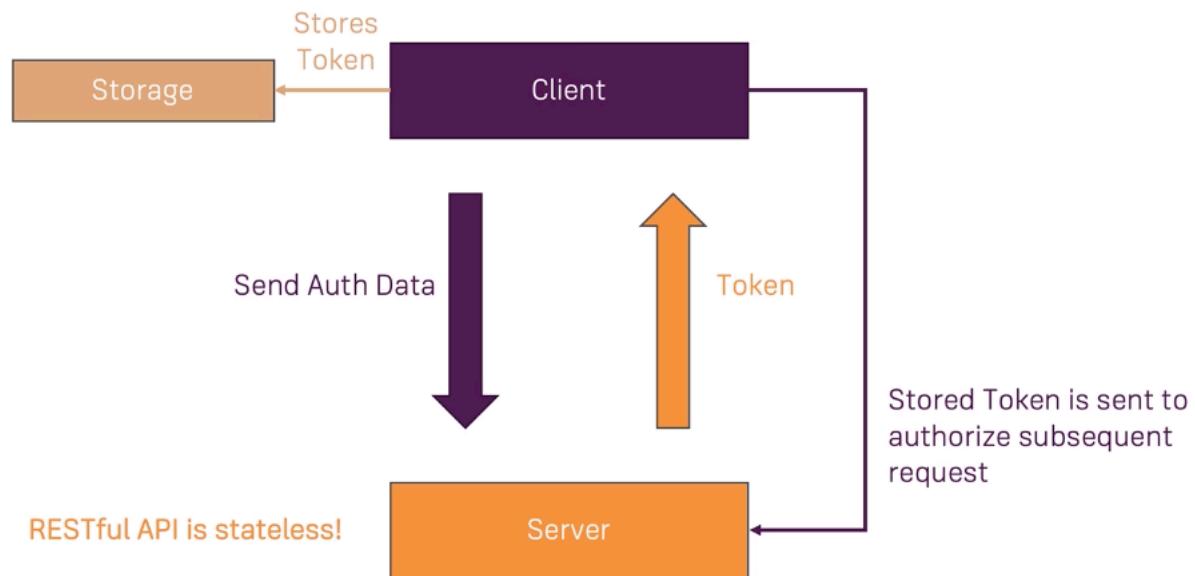


AGENDA

- **Section 20: Authentication & Route Protection in Angular**
- **Section 21: Dynamic Components**



How Authentication Works



RECIPE LIST

- New area in the application called Auth - AuthComponent.

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <form>
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input type="email" id="email" class="form-control" />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" class="form-control" />
      </div>
      <div>
        <button class="btn btn-primary">Sign Up</button> |
        <button class="btn btn-primary">Switch to Login</button>
      </div>
    </form>
  </div>
</div>
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {}
```



APP MODULE

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { RecipesComponent } from './recipes/recipes.component';
import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
import { ShoppingListComponent } from './shopping-list/shopping-list.component';
import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
import { DropdownDirective } from './shared/dropdown.directive';
import { ShoppingListService } from './shopping-list/shopping-list.service';
import { AppRoutingModule } from './app-routing.module';
import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
import { RecipeService } from './recipes/recipe.service';
import { AuthComponent } from './auth/auth.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    RecipesComponent,
    RecipeListComponent,
    RecipeDetailComponent,
    RecipeItemComponent,
    ShoppingListComponent,
    ShoppingEditComponent,
    DropdownDirective,
    RecipeStartComponent,
    RecipeEditComponent,
    AuthComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    AppRoutingModule
  ],
  providers: [ShoppingListService, RecipeService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```



HEADER COMPONENT

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a routerLink="/" class="navbar-brand">Recipe Book</a>
    </div>

    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li routerLinkActive="active"><a routerLink="/recipes">Recipes</a></li>
        <li routerLinkActive="active">
          <a routerLink="/auth">Authenticate</a>
        </li>
        <li routerLinkActive="active">
          <a routerLink="/shopping-list">Shopping List</a>
        </li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        <li class="dropdown" appDropdown>
          <a style="cursor: pointer;" class="dropdown-toggle" role="button"
             >Manage <span class="caret"></span>
          </a>
          <ul class="dropdown-menu">
            <li>
              <a style="cursor: pointer;" (click)="onSaveData()">Save Data</a>
            </li>
            <li>
              <a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



AUTH COMPONENT

Recipe Book Recipes Authenticate Shopping List

E-Mail

abc@abc.com

Password

.....

[Sign Up](#) | [Switch to Login](#)



SWITCHING BETWEEN AUTH MODES

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {

  isLoginPage = true;

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

}
```

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <form>
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input type="email" id="email" class="form-control" />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" class="form-control" />
      </div>
      <div>
        <button class="btn btn-primary" type="submit">{{ isLoginPage ? 'Login' : 'Sign up' }}</button> |
        <button class="btn btn-primary" type="button" (click)="onSwitchMode()">Switch to {{ isLoginPage ? 'Sign up' : 'Login' }}</button>
      </div>
    </form>
  </div>
</div>
```



SWITCHING BETWEEN AUTH MODES

← → ⌂ ⓘ http://localhost:4200/auth

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Recipe Book Recipes Authenticate Shopping List

E-Mail

Password

[Sign up](#) | [Switch to Login](#)

← → ⌂ ⓘ http://localhost:4200/auth

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Recipe Book Recipes Authenticate Shopping List

E-Mail

Password

[Login](#) | [Switch to Sign up](#)



HANDLING FORM INPUT

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <form #authForm="ngForm">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input type="email" id="email" class="form-control" name="email" email required/>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" class="form-control" name="password" required minlength="6"/>
      </div>
      <div>
        <button class="btn btn-primary" type="submit" [disabled]="!authForm.valid">{{ isLoginMode ? 'Login' :
'Sign up'}}</button> |
        <button class="btn btn-primary" type="button" (click)="onSwitchMode()">Switch to {{ isLoginMode ?
'Sign up' : 'Login'}}</button>
      </div>
    </form>
  </div>
</div>
```



HANDLING FORM INPUT

```
import { NgForm } from '@angular/forms';
import { Component } from '@angular/core';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {

  isLoginPage = true;

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    console.log(form.value);
    form.reset();
  }
}
```



HANDLING FORM INPUT

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input type="email" id="email" class="form-control" name="email" email required/>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" class="form-control" name="password" required minlength="6"/>
      </div>
      <div>
        <button class="btn btn-primary" type="submit" [disabled]="!authForm.valid">{{ isLoginMode ? 'Login' : 'Sign up' }}</button> |
        <button class="btn btn-primary" type="button" (click)="onSwitchMode()">Switch to {{ isLoginMode ? 'Sign up' : 'Login' }}</button>
      </div>
    </form>
  </div>
</div>
```



PREPARING THE BACKEND

```
{  
  "rules":{  
    ".read": "now < 1614578400000", // 2021-3-  
    1  
    ".write": "now < 1614578400000", // 2021-3-  
    1  
  }  
}
```



PREPARING THE BACKEND

```
{  
  "rules":{  
    ".read": auth != null,// 2021-3-1  
    ".write": auth != null, // 2021-3-1  
  }  
}
```



PREPARING THE BACKEND

The screenshot shows a web browser window with the URL `http://localhost:4200/recipes`. The page content includes a green button labeled "New Recipe" and a message "Please select a Recipe!". In the top right corner, there are buttons for "Save Data" and "Fetch Data". At the bottom, the browser's developer tools are open, specifically the Console tab, displaying an error message:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
core.js:2563
GET https://ng-course-recipe-book-e7924-default-firebase.com/recipes.json 401 (Unauthorized)
zone.js:324
core.js:400
ERROR ▾ HttpErrorResponse {headers: HttpHeaders, status: 401, statusText: "Unauthorized", url: "https://ng-course-recipe-book-e7924-default-firebase.com/recipes.json", ok: false, ...}
core.js:402
defaultErrorHandler
@ core.js:4002
push../node_modules/@angular/core/fesm5/core.js.ErrorHandler.handleError
@ core.js:4050
next
@ core.js:26759
schedulerFn
@ core.js:23735
push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber._tryOrUnsub
@ Subscriber.js:192
push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber.next
@ Subscriber.js:130
push../node_modules/rxjs/_esm5/internal/Subscriber.js.Subscriber._next
@ Subscriber.js:76
push../node_modules/rxjs/_esm5/internal/Subscriber.js.Subscriber.next
@ Subscriber.js:53
push../node_modules/rxjs/_esm5/internal/Subject.js.Subject.next
@ Subject.js:47
push../node_modules/@angular/core/fesm5/core.js.EventEmitter.emit
@ core.js:23719
(anonymous)
@ core.js:26278
push../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke
@ zone.js:391
push../node_modules/zone.js/dist/zone.js.Zone.run
@ zone.js:158
push../node_modules/@angular/core/fesm5/core.js.NgZone.runOutsideAngular
@ core.js:26215
onHandleError
@ core.js:26278
push../node_modules/zone.js/dist/zone.js.ZoneDelegate.handleError
@ zone.js:395
push../node_modules/zone.js/dist/zone.js.Zone.runTask
@ zone.js:198
push../node_modules/zone.js/dist/zone.js.ZoneTask.invokeTask
@ zone.js:498
```



PREPARING THE BACKEND

https://console.firebaseio.google.com/project/ng-course-recipe-book-e7924/authentication/providers

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Firebase

ng-course-recipe-book ▾

Authentication

Users Sign-in method Templates Usage

Sign-in providers

Provider	Status	Edit
Email/Password	Disabled	
Phone	Disabled	
Google	Disabled	
Play Games	Disabled	
Game Center	Disabled	
Facebook	Disabled	
Twitter	Disabled	
Github	Disabled	
Yahoo	Disabled	
Microsoft	Disabled	
Apple	Disabled	
Anonymous	Disabled	

Authorized domains

Add domain

Project Overview

Build

Authentication

Cloud Firestore

Realtime Database

Storage

Hosting

Functions

Machine Learning

Release & Monitor

Crashlytics, Performance, Test Lab, ...

Analytics

Dashboard, Realtime, Events, Conve...

Engage

Predictions, A/B Testing, Cloud Mes...

Extensions

Spark Free \$0/month Upgrade

PREPARING THE BACKEND

ng-course-recipe-book ▾

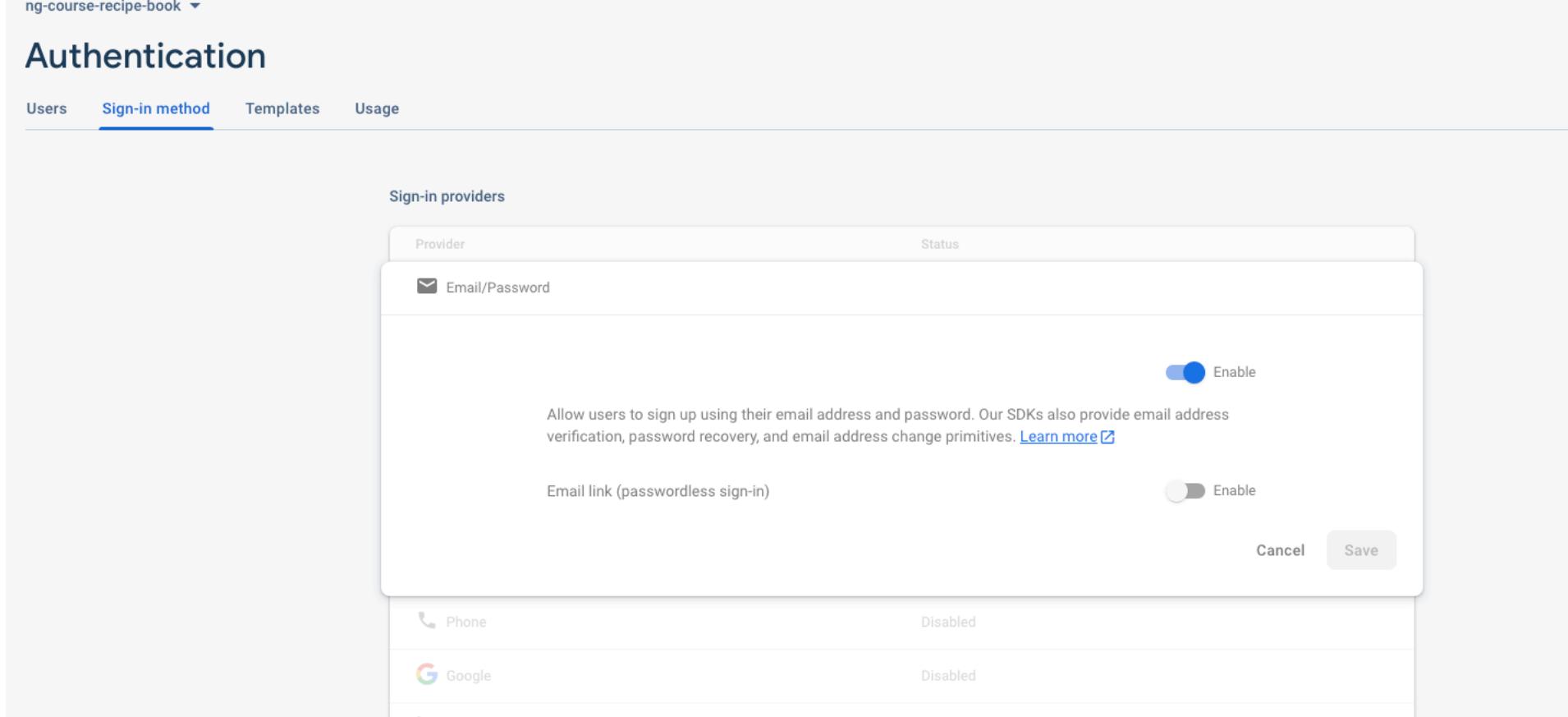
Authentication

Users **Sign-in method** Templates Usage

Sign-in providers

Provider	Status
✉ Email/Password	<input checked="" type="checkbox"/> Enable Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more
Email link (passwordless sign-in)	<input type="checkbox"/> Enable
📞 Phone	Disabled
Google	Disabled

Cancel **Save**



PREPARING THE BACKEND

ng-course-recipe-book ▾

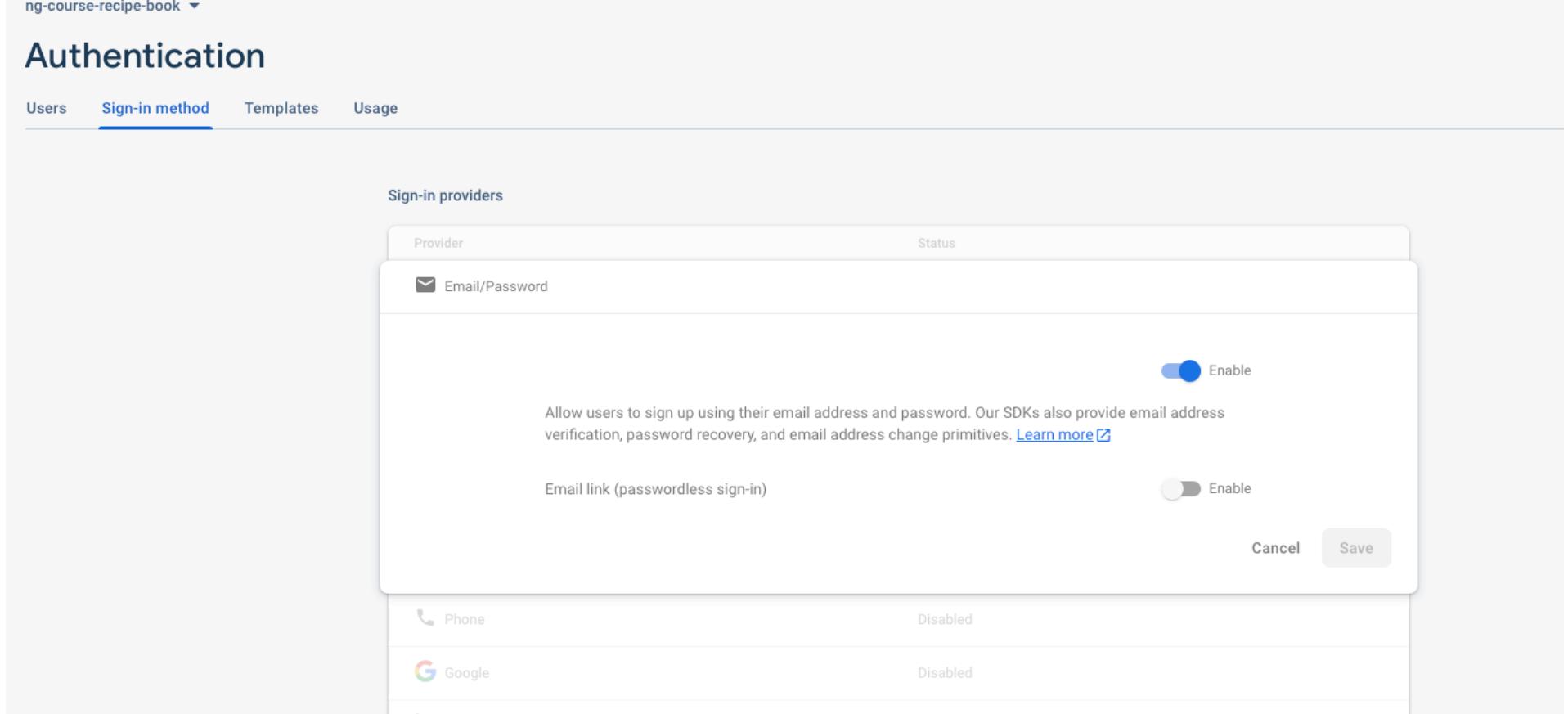
Authentication

Users **Sign-in method** Templates Usage

Sign-in providers

Provider	Status
✉ Email/Password	<input checked="" type="checkbox"/> Enable Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more
Email link (passwordless sign-in)	<input type="checkbox"/> Enable
📞 Phone	Disabled
Google	Disabled

Cancel **Save**



PREPARING THE BACKEND

ng-course-recipe-book ▾

Authentication

Users Sign-in method Templates Usage

Sign-in providers

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center	Disabled
Facebook	Disabled
Twitter	Disabled
Github	Disabled
Yahoo	Disabled
Microsoft	Disabled
Apple	Disabled



PREPARING THE SINGUP REQUEST



PREPARING THE SINGUP REQUEST

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";

interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string
}

@Injectable({providedIn: 'root'})
export class AuthService {

  constructor( private http: HttpClient ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    })
  }
}
```



PREPARING THE SINGUP REQUEST

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";

interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string
}

@Injectable({providedIn: 'root'})
export class AuthService {

  constructor( private http: HttpClient ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    })
  }
}
```



```
import { AuthService } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;

  constructor(private authService: AuthService) {

  }

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }
    if(this.isLoginPage) {

    } else {
      this.authService.signup(form.value.email, form.value.password).subscribe(resData => {
        console.log(resData);
      },
      error => {
        console.log(error);
      })
    }

    // console.log(form.value);
    form.reset();
  }
}
```



PREPARING THE SINGUP REQUEST

Screenshot of a web browser showing a sign-up form and the Network tab of the developer tools.

The browser address bar shows `http://localhost:4200/auth`. The page title is "E-Mail".

The sign-up form fields are:

- E-Mail (empty input field)
- Password (empty input field)

Buttons at the bottom of the form are "Sign Up" (blue) and "Switch to Login" (blue).

The Network tab in the developer tools shows a list of requests. One request is highlighted in blue:

- Name: accounts:singUp?key=AlzaSyB4GCPgTMKORoIcC070Z5pfhDVKOP3LJ2A
- Request URL: `https://identitytoolkit.googleapis.com/v1/accounts:singUp?key=AlzaSyB4GCPgTMKORoIcC070Z5pfhDVKOP3LJ2A`
- Request Method: POST
- Status Code: 200 OK
- Remote Address: 208.87.237.211:80
- Referrer Policy: strict-origin-when-cross-origin
- Response Headers:
 - Access-Control-Allow-Origin: `http://localhost:4200`
 - Access-Control-Expose-Headers: date, vary, vary, vary, content-encoding, server, content-length
 - Alt-Svc: `h3-29=":443"; ma=2592000, h3-T051=":443"; ma=2592000, h3-Q050=":443"; ma=2592000, h3-Q046=":443"; ma=2592000, h3-Q043=":443"; ma=2592000, quic=":443"; ma=2592000; v="46,43"`
 - Cache-Control: no-cache, no-store, max-age=0, must-revalidate
 - Connection: keep-alive
 - Content-Encoding: gzip
 - Content-Length: 1035
 - Content-Type: application/json; charset=UTF-8
 - Date: Sat, 30 Jan 2021 23:42:34 GMT
 - Expires: Mon, 01 Jan 1990 00:00:00 GMT
 - Pragma: no-cache
 - Server: ESF
 - Vary: Origin



PREPARING THE SINGUP REQUEST

→ C ⓘ http://localhost:4200/auth

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Recipe Book Recipes Authenticate Shopping List Manage ▾

E-Mail
Password

Sign Up | Switch to Login

Elements Sources Console Network Performance Memory Security Application Lighthouse Augury

Preserve log Disable cache Online

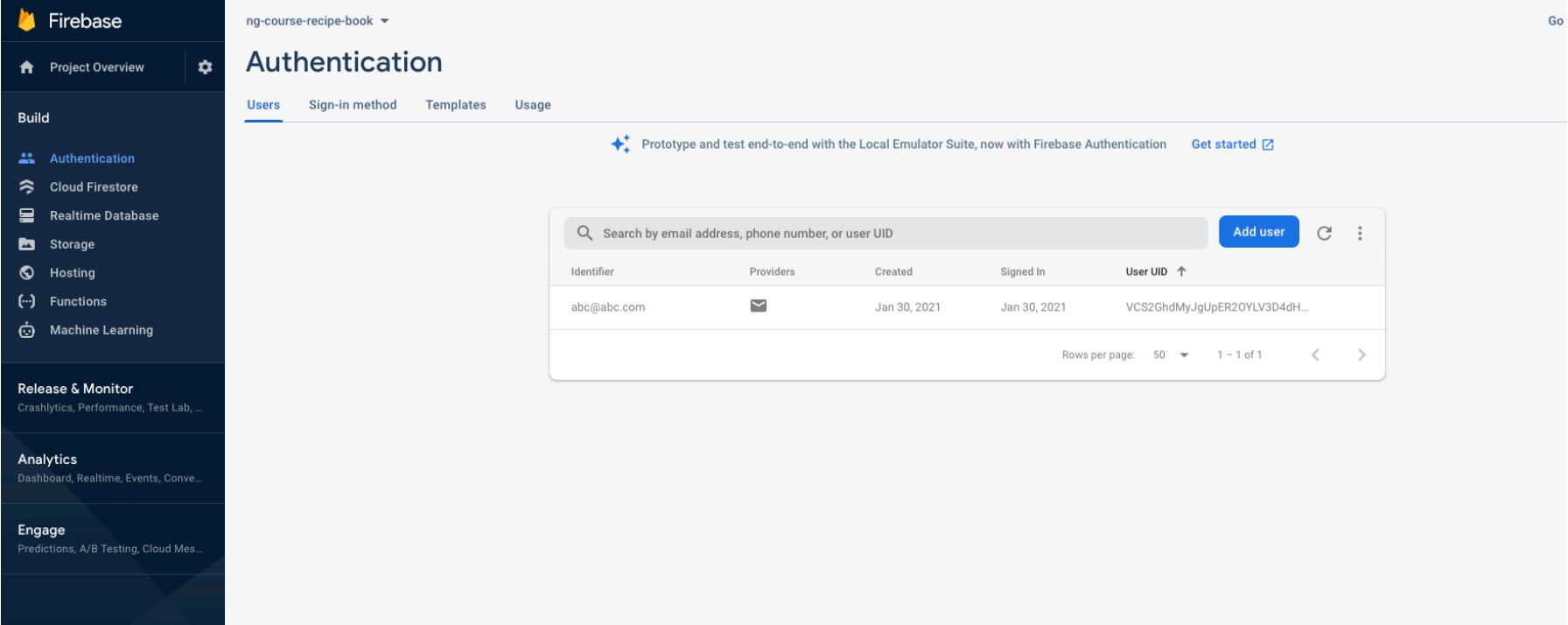
Iter 5000 ms 10000 ms 15000 ms 20000 ms 25000 ms 30000 ms 35000 ms 40000 ms 45000 ms 50000 ms 55000 ms 60000 ms 65000 ms 70000 ms

ame

	Headers	Preview	Response	Initiator	Timing
recipes	1 { 2 "kind": "identitytoolkit#SignupNewUserResponse", 3 "idToken": "eyJhbGciOiJSUzIiNlIsImtpZC161jUjZTV1NmY1MzBjNDkwMTF1Yjg0YzhmYWExZWM1NGM1MTc1N2I2Ng1LcJ0eXAi0jKV10lfQ.eyJpc3Mi0jodHRwczovL3Ny3VzXrva2VuLmdv2dsZ55jz20vbmcTy291cnNLX31y2lwZ51b29rLWU30Tl0i1wiYX", 4 "refreshToken": "AOuvkUTVMem-9dq-qNMHPdBHqA75Aqkv7zkVjdfjyHoCq4scxP-fKdA9U_6WI03eRU9XUHGFn1KC2FmNCi74wdbmQHkZwXj48oXeYBLD101cBIXoaLqKAbB8d5EukJds3rA6Ha6Lfg5UUVcjAEyN76iLrIqVc8SM4cU_zP0pAa8d0gmyZTyKJ8h2Jw", 5 "expiresIn": "3600", 6 "(localId": "VCS2GhdMyJgUpER20YLV3D4dH012" 7 } 8 }	9			
runtime.js					
polyfills.js					
styles.js					
vendor.js					
main.js					
ng-validate.js					
info?=>1612050096461					
websocket					
favicon.ico					
backend.js					
accounts:signUp?key=AlzaSyB4GCPgTMKORoISCo70Z5phDVKOP3LJ2A					
accounts:signUp?key=AlzaSyB4GCPgTMKORoISCo70Z5phDVKOP3LJ2A					

3 requests 6.5 MB transferred 6.5 MB resources Finish: 59.28 s DOMC Line 1, Column 1

PREPARING THE SINGUP REQUEST



The screenshot shows the Firebase console interface for the project "ng-course-recipe-book". The left sidebar contains navigation links for Project Overview, Authentication (selected), Cloud Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning, Release & Monitor, Analytics, and Engage. The main content area is titled "Authentication" and shows the "Users" tab selected. A banner at the top right says "Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication" with a "Get started" button. Below is a table with one row of data:

Identifier	Providers	Created	Signed In	User UID
abc@abc.com	✉	Jan 30, 2021	Jan 30, 2021	VCS2GhdMyJgUpER2OYLV3D4dH...

At the bottom of the table, there are pagination controls: "Rows per page: 50", "1 - 1 of 1", and navigation arrows.



PREPARING THE SINGUP REQUEST

Screenshot of a web browser showing a sign-up form and the Network tab of the developer tools.

The browser address bar shows `http://localhost:4200/auth`. The page title is "Authenticate". The navigation bar includes "Recipe Book", "Recipes", "Authenticate" (which is active), and "Shopping List".

The "Authenticate" form has fields for "E-Mail" and "Password", and buttons for "Sign Up" and "Switch to Login".

The Network tab of the developer tools shows a list of requests. One request, "accounts:signUp?key=AlzaSyB4GCPgTMKORoIsc070Z5pfhDVkop3Lj2A", is selected and expanded. Its Response tab displays the following JSON error message:

```
1 {  
2   "error": {  
3     "code": 400,  
4     "message": "EMAIL_EXISTS",  
5     "errors": [  
6       {  
7         "message": "EMAIL_EXISTS",  
8         "domain": "global",  
9         "reason": "invalid"  
10      }  
11    ]  
12  }  
13}  
14
```

ADDING A LOADING SPINNER AND ERROR HANDLING

```
import { AuthService } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;
  isLoading = false;
  constructor(private authService: AuthService) {

  }

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }
    this.isLoading = true;
    if(this.isLoginPage) {

    } else {
      this.authService.signup(form.value.email, form.value.password).subscribe(
        resData => {
          console.log(resData);
          this.isLoading = false;
        },
        error => {
          console.log(error);
          this.isLoading = false;
        })
    }
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```



ADDING A LOADING SPINNER AND ERROR HANDLING

```
import { Component } from "@angular/core";

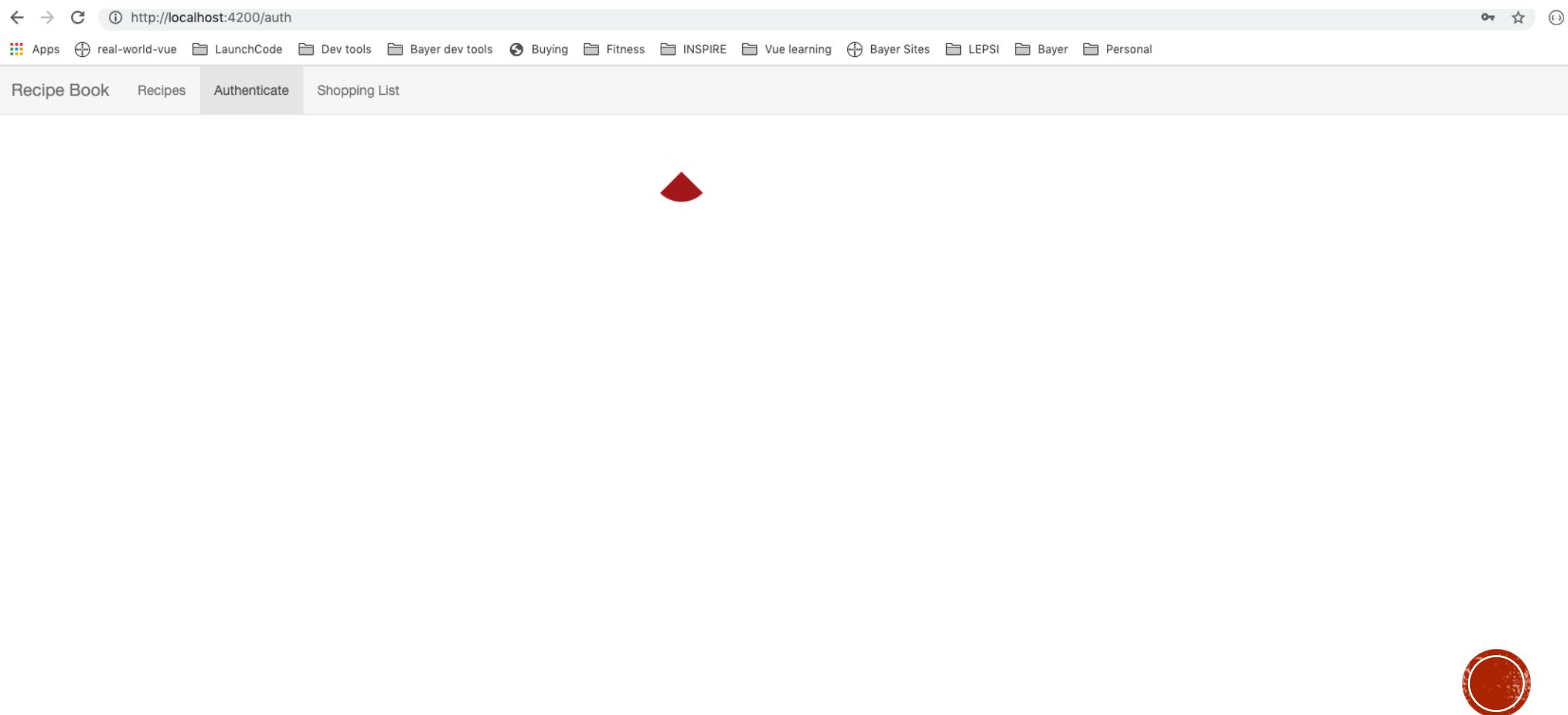
@Component({
  selector: 'app-loading-spinner',
  template: '<div class="lds-hourglass"></div>',
  styleUrls: ['./loading-spinner.component.css']
})
export class LoadingSpinnerComponent {

}

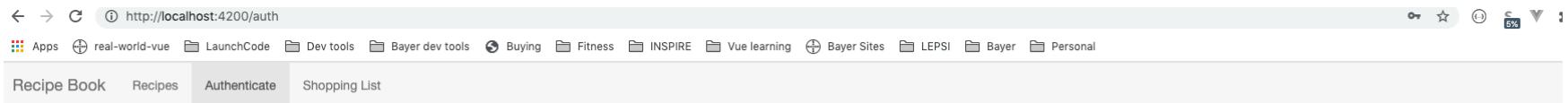
.lds-hourglass {
  display: inline-block;
  position: relative;
  width: 80px;
  height: 80px;
}
.lds-hourglass:after {
  content: " ";
  display: block;
  border-radius: 50%;
  width: 0;
  height: 0;
  margin: 8px;
  box-sizing: border-box;
  border: 32px solid rgb(79, 26, 163);
  border-color: rgb(79, 26, 163) transparent rgb(79, 26, 163) transparent;
  animation: lds-hourglass 1.2s infinite;
}
@keyframes lds-hourglass {
  0% {
    transform: rotate(0);
    animation-timing-function: cubic-bezier(0.55, 0.055, 0.675, 0.19);
  }
  50% {
    transform: rotate(900deg);
    animation-timing-function: cubic-bezier(0.215, 0.61, 0.355, 1);
  }
  100% {
    transform: rotate(1800deg);
  }
}
```



ADDING A LOADING SPINNER AND ERROR HANDLING



ADDING A LOADING SPINNER AND ERROR HANDLING



A screenshot of the Network tab in the browser's developer tools. The tab is titled "Network" and shows a list of requests. One request, "accounts:signUp?key=AlzaSyB4GCPgTMKORoISCo70Z5pfhDVKOP3LJ2A", has failed with a status of 400. The "Response" tab is selected, showing the JSON error response:{"error": {"code": 400, "message": "EMAIL_EXISTS", "errors": [{"message": "EMAIL_EXISTS", "domain": "global", "reason": "invalid"}]}}



ADDING A LOADING SPINNER AND ERROR HANDLING

```
import { AuthService } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService) {

  }

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }
    this.isLoading = true;
    if(this.isLoginPage) {

    } else {
      this.authService.signup(form.value.email, form.value.password).subscribe(
        resData => {
          console.log(resData);
          this.isLoading = false;
        },
        errorRes => {
          console.log(errorRes);
          switch( errorRes.error.error.message) {
            case 'EMAIL_EXISTS': this.error = 'This email already exists';
          }
          // this.error = 'an error occurred!';
          this.isLoading = false;
        })
    }
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```



ADDING A LOADING SPINNER AND ERROR HANDLING

← → C ⓘ http://localhost:4200/auth

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Recipe Book Recipes Authenticate Shopping List

This email already exists

E-Mail
abc@abc.com

Password
.....

Sign Up | Switch to Login

Elements Sources Console Network Performance Memory Security Application Lighthouse Augury

Angular is running in the development mode. Call enableProdMode() to enable the production mode.

① POST https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A 400 (Bad Request)

HTTPErrorResponse {headers: HttpHeaders, status: 400, statusText: "Bad Request", url: "https://identitytoolkit.googleapis.com/v1/accounts...signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A", ok: false, ...}

ADDING A LOADING SPINNER AND ERROR HANDLING

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError } from 'rxjs/operators';
import { throwError } from 'rxjs';

interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string
}

@Injectable({providedIn: 'root'})
export class AuthService {

  constructor( private http: HttpClient ) {}

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5fhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(errorRes => {
      let errorMessage = 'An unknown error occurred';
      if (!errorRes.error || !errorRes.error.error) {
        return throwError(errorMessage);
      }
      switch( errorRes.error.error.message ) {
        case 'EMAIL_EXISTS': errorMessage = 'This email already exists';
      }
      return throwError(errorMessage);
    }) )
  }
}
```



ADDING A LOADING SPINNER AND ERROR HANDLING

```
import { AuthService } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginMode = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService) {

  }

  onSwitchMode() {
    this.isLoginMode = !this.isLoginMode;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }
    this.isLoading = true;
    if(this.isLoginMode) {

    } else {
      this.authService.signup(form.value.email, form.value.password).subscribe(
        resData => {
          console.log(resData);
          this.isLoading = false;
        },
        errorMessage => {
          console.log(errorMessage);
          this.error = errorMessage;
          this.isLoading = false;
        }
      )
    }
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```



ADDING A LOADING SPINNER AND ERROR HANDLING

The screenshot shows a web browser window with the URL `http://localhost:4200/auth`. The page displays a sign-up form with the following elements:

- A red error message box at the top containing the text "This email already exists".
- An "E-Mail" input field containing the value "abc@abc.com".
- A "Password" input field containing the value "*****".
- Two blue buttons at the bottom: "Sign Up" and "Switch to Login".

Below the browser window, the developer tools are open, specifically the Network tab. The Network tab shows a single request to `https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMK0RoTSC070Z5pfhDVK0P3LJ2A`. The status of this request is "400 (Bad Request)". The response body is partially visible, showing the error message "This email already exists".

SENDING LOGIN REQUESTS

```
import { AuthService, AuthResponseData } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginMode = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService) {}

  onSwitchMode() {
    this.isLoginMode = !this.isLoginMode;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginMode) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        console.log(resData);
        this.isLoading = false;
      },
      errorMessage => {
        console.log(errorMessage);
        this.error = errorMessage;
        this.isLoading = false;
      }
    );
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```



SENDING LOGIN REQUESTS

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError } from 'rxjs/operators';
import { throwError } from 'rxjs';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object ]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  constructor( private http: HttpClient ) {}

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(errorRes => {
      let errorMessage = 'An unknown error occurred';
      if (!errorRes.error || !errorRes.error.error) {
        return throwError(errorMessage);
      }
      switch( errorRes.error.error.message) {
        case 'EMAIL_EXISTS': errorMessage = 'This email already exists';
      }
      return throwError(errorMessage);
    }) )
  }

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
  {
```



SENDING LOGIN REQUESTS

← → C ⓘ http://localhost:4200/auth

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites

Recipe Book Recipes Authenticate Shopping List Manage ▾

E-Mail
abc@abc.com

Password
.....

Login | Switch to Sign Up

Elements Sources Console Network Performance Memory Security Application Lighthouse Augury

top Filter Default levels ▾ 2 hidden

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode. core.js:25638
auth.component.ts:37
{kind: "identitytoolkit#VerifyPasswordResponse", localId: "VCS2GhdMyJgUpER20YLV3D4dH0l2", email: "abc@abc.com", displayName: "", idToken: "eyJhbGciOiJSUzI
1NiIsImtpZCI6InRCME0yQSJ9eyJpc3Mi0...Bum0Xv5XQ_-TKVaJ7-YokJHki0vnnSD1_wDI1qNzFXYtiuyZg", ...} 
displayName: ""
email: "abc@abc.com"
idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6InRCME0yQSJ9eyJpc3Mi0iJodHRwczovL2lkZW50aXR5dG9vbGtpdC5nb29nbGUuY29tLyIsImF1ZCI6Im5nLWVvdXJzZS1yZWVpcGUtYm9vay1...
kind: "identitytoolkit#VerifyPasswordResponse"
localId: "VCS2GhdMyJgUpER20YLV3D4dH0l2"
registered: true
__proto__: Object
```



SENDING LOGIN REQUESTS

```
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError } from 'rxjs/operators';
import { throwError } from 'rxjs';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object ]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  constructor( private http: HttpClient ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(this.handleError));
  }

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(this.handleError));
  }

  private handleError(errorRes: HttpErrorResponse) {
    let errorMessage = 'An unknown error occurred';
    if (!errorRes.error || !errorRes.error.error) {
      return throwError(errorMessage);
    }
  }
}
```



SENDING LOGIN REQUESTS

```
import { AuthService, AuthResponseData } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService) {

  }

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginPage) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        console.log(resData);
        this.isLoading = false;
      },
      errorMessage => {
        console.log(errorMessage);
        this.error = errorMessage;
        this.isLoading = false;
      }
    );
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```



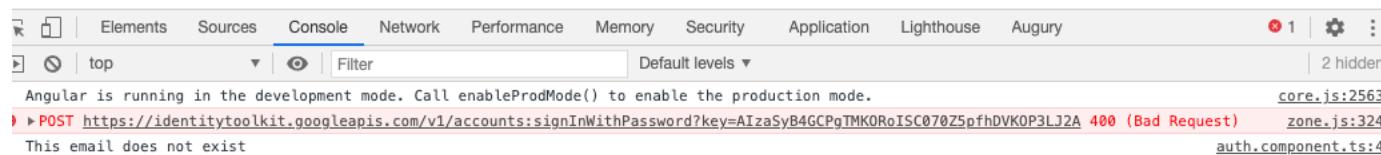
SENDING LOGIN REQUESTS

This email does not exist

E-Mail
abc@abc.com

Password
.....

Login | Switch to Sign Up



CREATING AND STORING THE USER DATA

```
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError, tap } from 'rxjs/operators';
import { throwError, Subject } from 'rxjs';
import { User } from './user.model';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object ]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  user = new Subject<User>();
  constructor( private http: HttpClient ) {

  }

  signup(email: string, password: string) {
    return
    this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(this.handleError), tap(resData => {
      this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
    }));
  }

  login(email: string, password: string) {
    return
    this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
    {
      email: email,
      password: password
    }).pipe(catchError(this.handleError), tap(resData => {
      this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
    }));
  }
}
```



REFLECTING THE AUTH STATE IN THE UI

```
import { Router } from '@angular/router';
import { AuthService, AuthResponseData } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService, private router: Router) {}

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

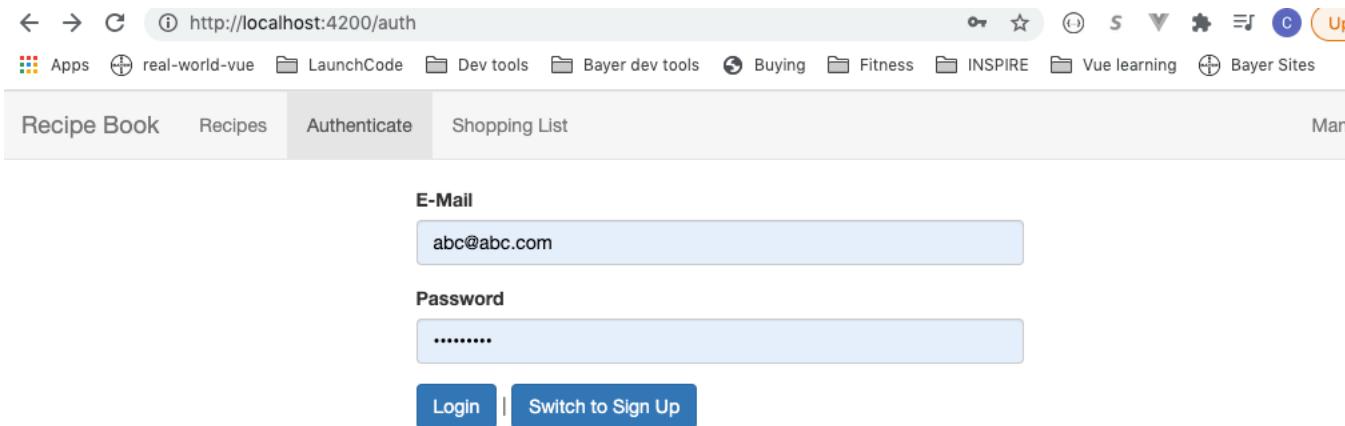
  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginPage) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        console.log(resData);
        this.isLoading = false;
        this.router.navigate(['/recipes']);
      },
      errorMessage => {
        console.log(errorMessage);
        this.error = errorMessage;
        this.isLoading = false;
      }
    );
    // console.log(form.value);
  }
}
```



REFLECTING THE AUTH STATE IN THE UI



REFLECTING THE AUTH STATE IN THE UI

```
import { Subscription } from 'rxjs';
import { AuthService } from './auth/auth.service';
import { Component, OnInit, OnDestroy } from '@angular/core';

import { DataStorageService } from '../shared/data-storage.service';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html'
})
export class HeaderComponent implements OnInit, OnDestroy{

  private userSub: Subscription;
  isAuthenticated = false;

  constructor(private dataStorageService: DataStorageService, private authService: AuthService) {}
  ngOnInit() {
    this.userSub = this.authService.user.subscribe(user => {
      this.isAuthenticated = !!user;
      console.log(!user);
      console.log (!!user);
    });
  }

  onSaveData() {
    this.dataStorageService.storeRecipes();
  }

  onFetchData() {
    this.dataStorageService.fetchRecipes().subscribe();
  }

  ngOnDestroy() {
    this.userSub.unsubscribe();
  }
}
```



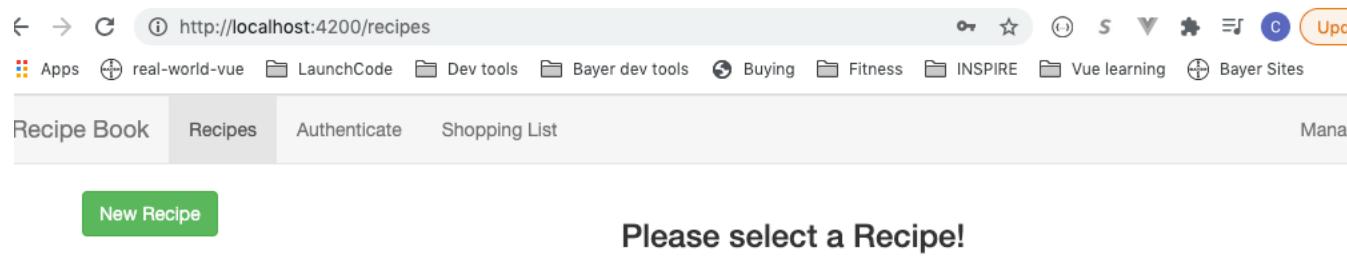
REFLECTING THE AUTH STATE IN THE UI

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a routerLink="/" class="navbar-brand">Recipe Book</a>
    </div>

    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li routerLinkActive="active"><a routerLink="/recipes" *ngIf="isAuthenticated">Recipes</a></li>
        <li routerLinkActive="active" *ngIf="!isAuthenticated">
          <a routerLink="/auth">Authenticate</a>
        </li>
        <li routerLinkActive="active">
          <a routerLink="/shopping-list">Shopping List</a>
        </li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        <li *ngIf="isAuthenticated">
          <a style="cursor: pointer;">Logout</a>
        </li>
        <li class="dropdown" appDropdown *ngIf="isAuthenticated">
          <a style="cursor: pointer;" class="dropdown-toggle" role="button"
            >Manage <span class="caret"></span>
          ></a>
          <ul class="dropdown-menu">
            <li>
              <a style="cursor: pointer;" (click)="onSaveData()">Save Data</a>
            </li>
            <li>
              <a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



REFLECTING THE AUTH STATE IN THE UI



ADDING THE TOKEN TO OUTGOING REQUESTS

```
import { AuthService } from './auth/auth.service';
import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { exhaustMap, map, take, tap } from 'rxjs/operators';

import { Recipe } from '../recipes/recipe.model';
import { RecipeService } from '../recipes/recipe.service';

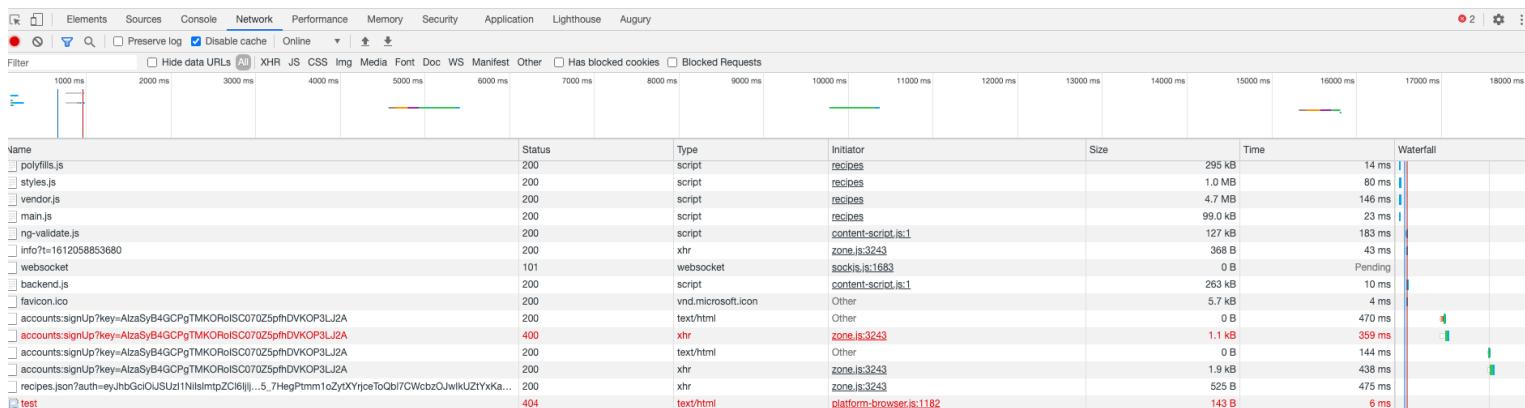
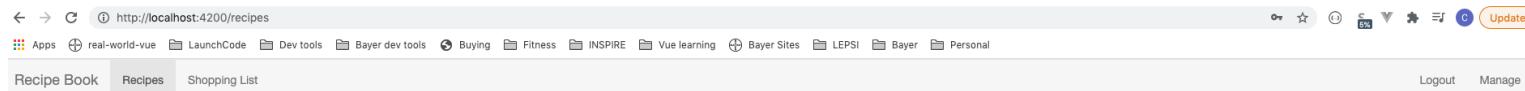
@Injectable({ providedIn: 'root' })
export class DataStorageService {
  constructor(private http: HttpClient, private recipeService: RecipeService, private authService: AuthService) {}

  storeRecipes() {
    const recipes = this.recipeService.getRecipes();
    this.http
      .put(
        'https://ng-course-recipe-book-e7924-default-firebase.com.firebaseio.com/recipes.json',
        recipes
      )
      .subscribe(response => {
        console.log(response);
      });
  }

  fetchRecipes() {
    return this.authService.user.pipe(take(1), exhaustMap(user => {
      return this.http
        .get<Recipe[]>(
          'https://ng-course-recipe-book-e7924-default-firebase.com.firebaseio.com/recipes.json',
          {
            params: new HttpParams().set('auth', user.token)
          }
        ),
        map(recipes => {
          return recipes.map(recipe => {
            return {
              ...recipe,
              ingredients: recipe.ingredients ? recipe.ingredients : []
            };
          });
        }),
        tap(recipes => {
          this.recipeService.setRecipes(recipes);
        })
      )
    });
  }
}
```



ADDING THE TOKEN TO OUTGOING REQUESTS



Name	Status	Type	Initiator	Size	Time	Waterfall
polyfills.js	200	script	recipes	295 kB	14 ms	
styles.js	200	script	recipes	1.0 MB	80 ms	
vendor.js	200	script	recipes	4.7 kB	146 ms	
main.js	200	script	recipes	99.0 kB	23 ms	
ng-validate.js	200	script	content-script.js:1	127 kB	183 ms	
info?r=1612058853680	200	xhr	zone.js:3243	368 B	43 ms	
websocket	101	websocket	sockjs:is:1683	0 B	Pending	
backend.js	200	script	content-script.js:1	263 kB	10 ms	
favicon.ico	200	image	vnd.microsoft.icon	5.7 kB	4 ms	
accounts:signUp?key=AlzaSyB4GCPgTMKORoIsc070Z5phDVkop3Lj2A	200	text/html	Other	0 B	470 ms	
accounts:signUp?key=AlzaSyB4GCPgTMKORoIsc070Z5phDVkop3Lj2A	400	xhr	zone.js:3243	1.1 kB	359 ms	
accounts:signUp?key=AlzaSyB4GCPgTMKORoIsc070Z5phDVkop3Lj2A	200	text/html	Other	0 B	144 ms	
accounts:signUp?key=AlzaSyB4GCPgTMKORoIsc070Z5phDVkop3Lj2A	200	xhr	zone.js:3243	1.9 kB	438 ms	
recipes.json?auth=eyJhbGciOiJSUzI1NiIsImtpZC16lj...5_7HgPtmm1oZyXYrjeToQbI7CwcbzOJwIkUZtYxKa...	200	xhr	zone.js:3243	525 B	475 ms	
test	404	text/html	platform-browser.js:1182	143 B	6 ms	



ADDING THE TOKEN TO OUTGOING REQUESTS

Screenshot of a web browser showing a recipe selection interface. The URL is <http://localhost:4200/recipes>. The page displays a "Please select a Recipe!" message above a list of available recipes. One recipe, "test", is highlighted.

Below the browser screenshot is the Network tab of the developer tools. It shows a timeline of requests. A specific request to "accounts/signUp" is selected, showing its Headers, Preview, Response, Initiator, and Timing details. The Headers include:

```
Origin: http://localhost:4200
Pragma: no-cache
Referer: http://localhost:4200/
sec-ch-ua: "Google Chrome";v="87", "Not;A Brand";v="99", "Chromium";v="87"
sec-ch-ua-mobile: ?0
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: cross-site
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36
```

The Preview section shows the JSON response body:

```
{"name": "test"}
```

The Response tab shows the raw JSON response:

```
{"name": "test"}
```

The Initiator tab shows the source file and line number for the request:

```
accounts/signUp?key=AlzaSyB4GCPgTMKORoISCo70Z5pfhDVKOP3LJ2A
```

The Timing tab shows the duration of the request:

```
1000 ms | 2000 ms | 3000 ms | 4000 ms | 5000 ms | 6000 ms | 7000 ms | 8000 ms | 9000 ms | 10000 ms | 11000 ms | 12000 ms | 13000 ms | 14000 ms | 15000 ms | 16000 ms | 17000 ms | 18000 ms
```



ADDING THE TOKEN WITH AN INTERCEPTOR

```
import { async } from '@angular/core/testing';
import { AuthService } from './auth.service';
import { HttpEvent, HttpHandler, HttpInterceptor, HttpParams, HttpRequest } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { take, exhaustMap } from 'rxjs/operators';

@Injectable()
export class AuthInterceptorService implements HttpInterceptor{

    constructor(private authService: AuthService) {

    }
    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        return this.authService.user.pipe(take(1), exhaustMap(user => {

            if(!user) {
                return next.handle(req);
            }
            const modifiedReq = req.clone({params: new HttpParams().set('auth', user.token)})
            return next.handle(modifiedReq);
        }));
        this.authService.user.subscribe();
    }
}
```



ADDING THE TOKEN WITH AN INTERCEPTOR

```
import { AuthService } from './auth/auth.service';
import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { exhaustMap, map, take, tap } from 'rxjs/operators';

import { Recipe } from '../recipes/recipe.model';
import { RecipeService } from '../recipes/recipe.service';

@Injectable({ providedIn: 'root' })
export class DataStorageService {
  constructor(private http: HttpClient, private recipeService: RecipeService, private authService: AuthService) {}

  storeRecipes() {
    const recipes = this.recipeService.getRecipes();
    this.http
      .put(
        'https://ng-course-recipe-book-e7924-default-firebase.com.firebaseio.com/recipes.json',
        recipes
      )
      .subscribe(response => {
        console.log(response);
      });
  }

  fetchRecipes() {
    return this.http
      .get<Recipe[]>(
        'https://ng-course-recipe-book-e7924-default-firebase.com.firebaseio.com/recipes.json'
      )
      .pipe(
        map(recipes => {
          return recipes.map(recipe => {
            return {
              ...recipe,
              ingredients: recipe.ingredients ? recipe.ingredients : []
            };
          });
        }),
        tap(recipes => {
          this.recipeService.setRecipes(recipes);
        })
      );
  }
}
```



ADDING THE TOKEN WITH AN INTERCEPTOR

```
import { AuthInterceptorService } from './auth/auth.interceptor.service';
import { LoadingSpinnerComponent } from './shared/loading-spinner/loading-spinnner.component';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { RecipesComponent } from './recipes/recipes.component';
import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
import { ShoppingListComponent } from './shopping-list/shopping-list.component';
import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
import { DropdownDirective } from './shared/dropdown.directive';
import { ShoppingListService } from './shopping-list/shopping-list.service';
import { AppRoutingModule } from './app-routing.module';
import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
import { RecipeService } from './recipes/recipe.service';
import { AuthComponent } from './auth/auth.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    RecipesComponent,
    RecipeListComponent,
    RecipeDetailComponent,
    RecipeItemComponent,
    ShoppingListComponent,
    ShoppingEditComponent,
    DropdownDirective,
    RecipeStartComponent,
    RecipeEditComponent,
    AuthComponent,
    LoadingSpinnerComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    AppRoutingModule
  ],
  providers: [ShoppingListService, RecipeService, {provide: HTTP_INTERCEPTORS, useClass: AuthInterceptorService, multi: true}],
})
```



ADDING THE TOKEN WITH AN INTERCEPTOR

← → ⌂ ⓘ http://localhost:4200/recipes

Apps real-world-vue LaunchCode Dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Logout Manage

Recipe Book Recipes Shopping List

New Recipe Please select a Recipe!

Network Performance Memory Security Application Lighthouse Augury

Preserve log Disable cache Online

Name	Status	Type	Initiator	Size	Time	Waterfall
auth	200	document	vendor.js:106654	792 B	22 ms	
runtime.js	200	script	auth	6.5 kB	23 ms	
polyfills.js	200	script	auth	295 kB	31 ms	
styles.js	200	script	auth	1.0 MB	53 ms	
vendor.js	200	script	auth	4.7 MB	184 ms	
main.js	200	script	auth	102 kB	12 ms	
ng-validate.js	200	script	content-script.js:1	127 kB	118 ms	
info?r=1612061495018	200	xhr	zone.js:3243	368 B	29 ms	
websocket	101	websocket	sockjs.js:1683	0 B	Pending	
backend.js	200	script	content-script.js:1	263 kB	10 ms	
favicon.ico	200	vnd.microsoft.icon	Other	5.7 kB	5 ms	
accounts:signInWithPassword?key=AlzaSyB4GCPgTMKORoISCo70Z5phDVKOP3LJ2A	200	xhr	zone.js:3243	1.6 kB	386 ms	
accounts:signInWithPassword?key=AlzaSyB4GCPgTMKORoISCo70Z5phDVKOP3LJ2A	200	text/html	Other	0 B	419 ms	



ADDING LOGOUT

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a routerLink="/" class="navbar-brand">Recipe Book</a>
    </div>

    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li routerLinkActive="active" *ngIf="isAuthenticated"><a routerLink="/recipes" *ngIf="isAuthenticated">Recipes</a></li>
        <li routerLinkActive="active" *ngIf="!isAuthenticated">
          <a routerLink="/auth">Authenticate</a>
        </li>
        <li routerLinkActive="active">
          <a routerLink="/shopping-list">Shopping List</a>
        </li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        <li *ngIf="isAuthenticated">
          <a style="cursor: pointer;" (click)="onLogout()">Logout</a>
        </li>
        <li class="dropdown" appDropdown *ngIf="isAuthenticated">
          <a style="cursor: pointer;" class="dropdown-toggle" role="button"
             >Manage <span class="caret"></span>
          </a>
          <ul class="dropdown-menu">
            <li>
              <a style="cursor: pointer;" (click)="onSaveData()">Save Data</a>
            </li>
            <li>
              <a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



ADDING LOGOUT

```
import { Router } from '@angular/router';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, tap } from 'rxjs/operators';
import { throwError, Subject, BehaviorSubject } from 'rxjs';
import { User } from './user.model';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object ]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  user = new BehaviorSubject<User>(null);
  constructor( private http: HttpClient, private router: Router ) {}

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
    this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
  }));
  }

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
    this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
  }));
  }
}
```



ADDING LOGOUT

A screenshot of a web browser window. The address bar shows the URL `http://localhost:4200/auth`. Below the address bar is a horizontal navigation bar with various links: Apps, real-world-vue, LaunchCode, Dev tools, Bayer dev tools, Buying, Fitness, INSPIRE, Vue learning, Bayer Sites, LEPSI, Bayer, and Personal. The main content area displays a login form for 'real-world-vue'. The form has two input fields: 'E-Mail' containing `abc@abc.com` and 'Password' containing several dots. Below the inputs are two buttons: 'Login' and 'Switch to Sign Up'.

E-Mail
abc@abc.com

Password
.....

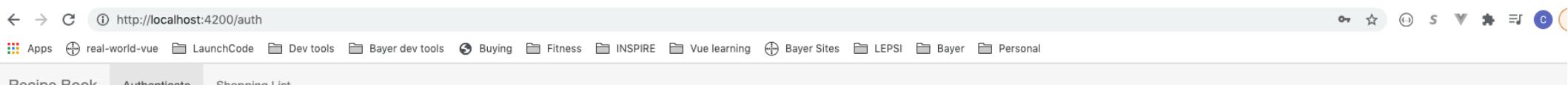
[Login](#) | [Switch to Sign Up](#)



ADDING LOGOUT

The screenshot shows a web browser window with the URL `http://localhost:4200/recipes` in the address bar. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons. The main content area of the browser displays a user interface for a "Recipe Book" application. At the top of this UI, there is a message: "Please select a Recipe!". Below this message is a green button labeled "New Recipe". The top navigation bar of the application has three items: "Recipe Book" (which is currently selected), "Recipes", and "Shopping List". On the far right of the application's header, there are links for "Logout" and "Manage". The overall layout is clean and modern, typical of a Vue.js application.

ADDING LOGOUT



E-Mail

Password

[Login](#) | [Switch to Sign Up](#)

ADDING AUTO LOGIN

```
import { Router } from '@angular/router';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError, tap } from 'rxjs/operators';
import { throwError, Subject, BehaviorSubject } from 'rxjs';
import { User } from './user.model';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  user = new BehaviorSubject<User>(null);
  constructor( private http: HttpClient, private router: Router ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVKOP3LJ2A',
{
  email: email,
  password: password
}).pipe(catchError(this.handleError), tap(resData => {
  this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
}));
  }

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVKOP3LJ2A',
{
  email: email,
  password: password
}).pipe(catchError(this.handleError), tap(resData => {
  this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
})
  }
}
```



ADDING AUTO LOGIN

New Recipe

Please select a Recipe!

Logout Manage

Application

token

userData

Key	Value
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJpbmxbmUgSldUIEJ1aWxkZXIiLCJpYXQiOjE1MzMjMyNzMSNjksImV4cCl6MTU2NDgxMDAwNSwiYX...
userData	{"email": "abc@abc.com", "id": "VCS2GhdMyJgUpER2OYLV3D4dH0l2", "_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6InRCME0yQSJ9.eyJpc3MiOiJodHRwczovL2lkZW50aXR5dG9vbGtpdC5nb29nbGUuY29tLyIsImF1ZCI6Im5nLWNvdXJzS1y2WNgcGUtYm9vay1UNzkyNCIsImlhCI6MTYxMjA2MjYzNiwiZXhwIjoxNjEzMjcyNjM2LCJ1c2VyX2lkIjo1VdNTMkd0ZE15SmdvEVSMk9ZTFYzRDRkSE9sM: null}



ADDING AUTO LOGIN

```
import { Router } from '@angular/router';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from "@angular/core";
import { catchError, tap, exhaustMap } from 'rxjs/operators';
import { throwError, Subject, BehaviorSubject } from 'rxjs';
import { User } from './user.model';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  user = new BehaviorSubject<User>(null);
  constructor( private http: HttpClient, private router: Router ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
    this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
  }));
}

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=AIzaSyB4GCPgTMK0RoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
    this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
  }));
}
```



ADDING AUTO LOGIN

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from './auth/auth.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit{
  constructor(private authService: AuthService) {

  }
  ngOnInit(): void {
    this.authService.autoLogin();
  }
  loadedFeature = 'recipe';

  onNavigate(feature: string) {
    this.loadedFeature = feature;
  }
}
```



ADDING AUTO LOGIN

A screenshot of a web browser window displaying a recipe management application. The URL in the address bar is `http://localhost:4200/recipes`. The page header includes navigation icons (back, forward, refresh) and a search bar. Below the header is a navigation bar with links: Apps, real-world-vue, LaunchCode, Dev tools, Bayer dev tools, Buying, Fitness, INSPIRE, Vue learning, Bayer Sites, LEPSI, Bayer, and Personal. The main content area has three buttons: "New Recipe" (green), "Authenticate" (blue), and "Shopping List" (grey). A large text message "Please select a Recipe!" is centered on the page. The bottom right corner features a red circular icon.

← → C ⓘ http://localhost:4200/recipes

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Recipe Book Authenticate Shopping List

New Recipe

Please select a Recipe!

ADDING AUTO LOGOUT

```
import { Router } from '@angular/router';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, tap, exhaustMap } from 'rxjs/operators';
import { throwError, Subject, BehaviorSubject } from 'rxjs';
import { User } from './user.model';

export interface AuthResponseData {
  kind: string,
  idToken: string,
  displayName: string,
  profilePicture?: string,
  email: string,
  refreshToken: string,
  expiresIn: string,
  localId: string,
  mfaPendingCredential?: string,
  mfaInfo?: [ object ]
}

@Injectable({providedIn: 'root'})
export class AuthService {

  user = new BehaviorSubject<User>(null);
  tokenExpirationTimer: any;
  constructor( private http: HttpClient, private router: Router ) {

  }

  signup(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
    this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
  }));
}

  login(email: string, password: string) {
    return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=AIzaSyB4GCPgTMKORoISC070Z5pfhDVK0P3LJ2A',
  {
    email: email,
    password: password
  }).pipe(catchError(this.handleError), tap(resData => {
```



ADDING AUTH GUARD

```
import { AuthService } from './auth.service';
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';

@Injectable({providedIn: 'root'})
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService) {}

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
    return this.authService.user.pipe(map(user => {
      //for truish values.
      return !!user;
    }));
  }
}
```



ADDING AUTH GUARD

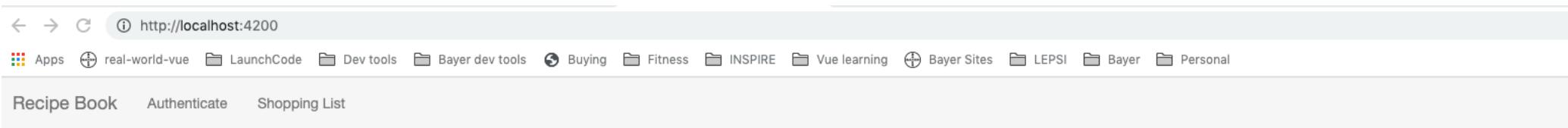
```
import { AuthGuard } from './auth/auth.guard';
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { RecipesComponent } from './recipes/recipes.component';
import { ShoppingListComponent } from './shopping-list/shopping-list.component';
import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
import { RecipesResolverService } from './recipes/recipes-resolver.service';
import { AuthComponent } from './auth/auth.component';

const appRoutes: Routes = [
  { path: '', redirectTo: '/recipes', pathMatch: 'full' },
  {
    path: 'recipes',
    component: RecipesComponent,
    canActivate: [AuthGuard],
    children: [
      { path: '', component: RecipeStartComponent },
      { path: 'new', component: RecipeEditComponent },
      {
        path: ':id',
        component: RecipeDetailComponent,
        resolve: [RecipesResolverService]
      },
      {
        path: ':id/edit',
        component: RecipeEditComponent,
        resolve: [RecipesResolverService]
      }
    ]
  },
  { path: 'shopping-list', component: ShoppingListComponent },
  { path: 'auth', component: AuthComponent }
];
@NgModule({
  imports: [RouterModule.forRoot(appRoutes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```



ADDING AUTH GUARD



ADDING AUTH GUARD

```
import { AuthService } from './auth.service';
import { Injectable } from "@angular/core";
import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree } from "@angular/router";
import { Observable } from "rxjs";
import { map, tap } from 'rxjs/operators';

@Injectable({providedIn: 'root'})
export class AuthGuard implements CanActivate {

    constructor(private authService: AuthService, private router: Router) {}

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.authService.user.pipe(map(user => {
            //for truish values.
            const isAuthenticated = !!user;
            if(isAuthenticated) return true;
            return this.router.createUrlTree(['/auth']);
        }))
        // tap(isAuthenticated => {
        //     if(!isAuthenticated) {
        //         this.router.navigate(['auth']);
        //     }
        // });
    }
}
```



DYNAMIC COMPONENTS

Components created runtime conditionally.

For example, on Error.

Loaded via code.

Learn to create and load and communicate with it and
get rid of it.



ADDING AN ALERT MODAL COMPONENT

Instead of having error message on screen, develop a alert modal for error

The screenshot shows a web application interface. At the top, there is a navigation bar with three items: "Recipe Book", "Authenticate", and "Shopping List". Below the navigation bar, there is a sign-up form. The form has two input fields: "E-Mail" containing "abc@abc.com" and "Password" containing "*****". Above the password field, there is a red rectangular box with the text "This email already exists". At the bottom of the form, there are two buttons: "Sign Up" and "Switch to Login".



ADDING AN ALERT MODAL COMPONENT

```
import { Component, Input } from "@angular/core";

@Component({
  selector:'app-alert',
  templateUrl: './alert.component.html',
  styleUrls: ['./alert.component.css']
})
export class AlertComponent {
  @Input() message: string;
}
```

```
<div class="backdrop"></div>
<div class="alert-box">
  <p>{{ message }}</p>
  <div class="alert-box-actions">
    <button class="btn btn-primary">Close</button>
  </div>
</div>
```

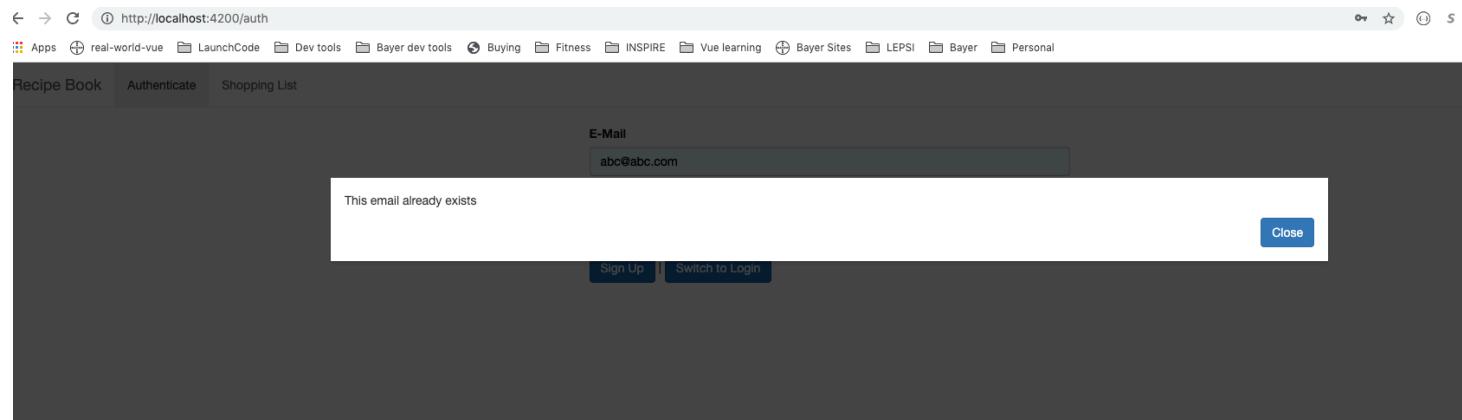


ADDING AN ALERT MODAL COMPONENT

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <!-- <div class="alert alert-danger" *ngIf="error">
      <p> {{ error }} </p>
    </div> -->
    <app-alert [message]="error" *ngIf="error"></app-alert>
    <div *ngIf="isLoading" style="text-align: center;">
      <app-loading-spinner></app-loading-spinner>
    </div>
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input
          type="email"
          id="email"
          class="form-control"
          ngModel
          name="email"
          required
          email
        />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          class="form-control"
          ngModel
          name="password"
          required
          minlength="6"
        />
      </div>
      <div>
        <button
          class="btn btn-primary"
          type="submit"
          [disabled]="!authForm.valid"
        >
          {{ isLoginMode ? 'Login' : 'Sign Up' }}
        </button>
        |
        <button class="btn btn-primary" (click)="onSwitchMode()" type="button">
          Switch to {{ isLoginMode ? 'Sign Up' : 'Login' }}
        </button>
      </div>
    </form>
```



ADDING AN ALERT MODAL COMPONENT



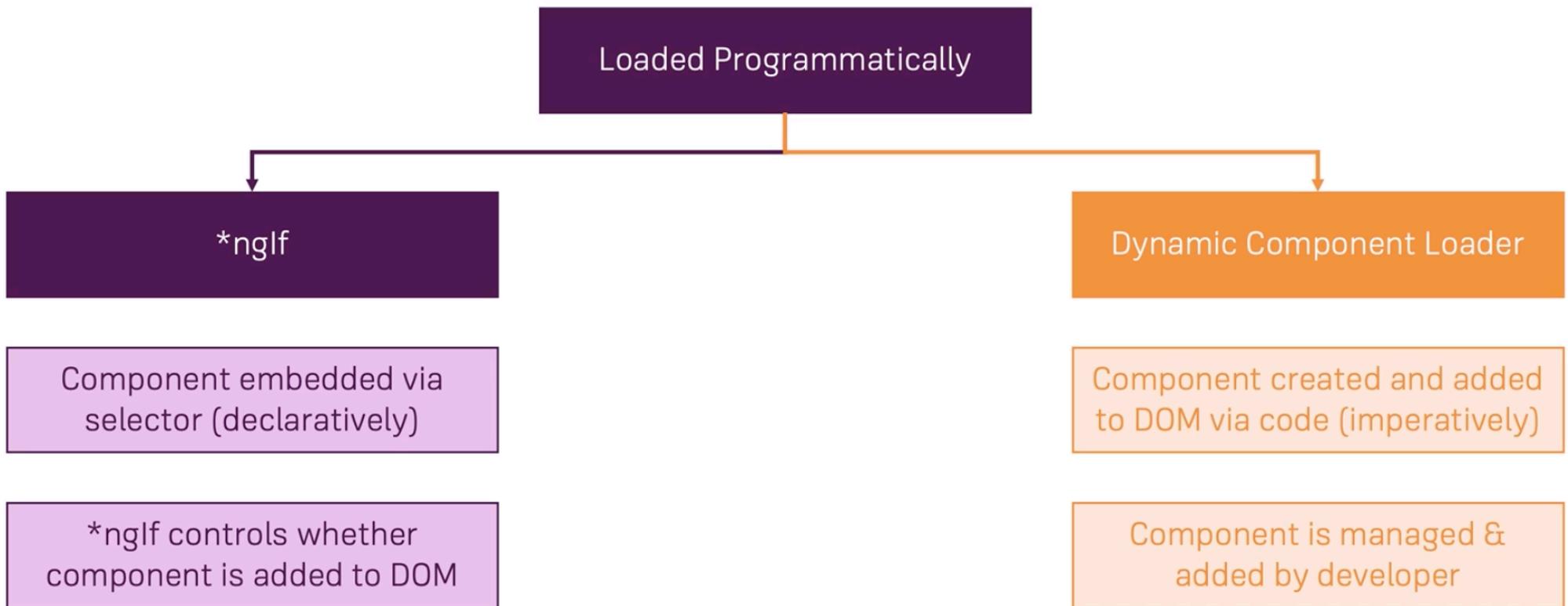
ADDING AN ALERT MODAL COMPONENT

```
.backdrop {  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 100vw;  
  height: 100vh;  
  background: rgba(0,0,0,0.75);  
  z-index: 50;  
}  
  
.alert-box {  
  position: fixed;  
  top: 30vh;  
  left: 20vw;  
  width: 60vw;  
  padding: 16px;  
  z-index: 100;  
  background: white;  
  box-shadow: 0 2px 8px rgba(0,0,0,0.26);  
}  
  
.alert-box-actions {  
  text-align: right;  
}
```



UNDERSTANDING THE DIFFERENT

What are “Dynamic Components”?



UNDERSTANDING THE DIFFERENT APPROACHES

```
import { Component, Input, Output, EventEmitter } from "@angular/core";

@Component({
  selector:'app-alert',
  templateUrl: './alert.component.html',
  styleUrls: ['./alert.component.css']
})
export class AlertComponent {
  @Input() message: string;
  @Output() close = new EventEmitter<void>();

  onClose() {
    this.close.emit();
  }
}
```



UNDERSTANDING THE DIFFERENT APPROACHES

```
import { Router } from '@angular/router';
import { AuthService, AuthResponseData } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginMode = true;
  isLoading = false;
  error: string = null;
  constructor(private authService: AuthService, private router: Router) {

  }

  onSwitchMode() {
    this.isLoginMode = !this.isLoginMode;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginMode) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        console.log(resData);
        this.isLoading = false;
        this.router.navigate(['/recipes']);
      },
      errorMessage => {
        console.log(errorMessage);
        this.error = errorMessage;
        this.isLoading = false;
      }
    );
    // console.log(form.value);
  }
}
```



UNDERSTANDING THE DIFFERENT APPROACHES

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <!-- <div class="alert alert-danger" *ngIf="error">
      <p> {{ error }} </p>
    </div> -->
    <app-alert [message]="error" *ngIf="error" (close)="onHandleError()"></app-alert>
    <div *ngIf="isLoading" style="text-align: center;">
      <app-loading-spinner></app-loading-spinner>
    </div>
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input
          type="email"
          id="email"
          class="form-control"
          ngModel
          name="email"
          required
          email
        />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          class="form-control"
          ngModel
          name="password"
          required
          minlength="6"
        />
      </div>
      <div>
        <button
          class="btn btn-primary"
          type="submit"
          [disabled]="!authForm.valid"
        >
          {{ isLoginMode ? 'Login' : 'Sign Up' }}
        </button>
        |
        <button class="btn btn-primary" (click)="onSwitchMode()" type="button">
          Switch to {{ isLoginMode ? 'Sign Up' : 'Login' }}
        </button>
      </div>
    </form>
  </div>
```



UNDERSTANDING THE DIFFERENT APPROACHES

The screenshot shows a web browser window with the URL `http://localhost:4200/auth`. The page displays a sign-up form with fields for 'E-Mail' (containing `abc@abc.com`) and 'Password'. Below the form are two buttons: 'Sign Up' and 'Switch to Login'. A modal dialog box is overlaid on the page, containing the text 'This email already exists' and a 'Close' button.

E-Mail
abc@abc.com

Password
.....

Sign Up | Switch to Login

This email already exists

Close



UNDERSTANDING THE DIFFERENT APPROACHES

A screenshot of a web browser window. The address bar shows the URL `http://localhost:4200/auth`. The page title is "real-world-vue". The navigation bar includes links for "Apps", "real-world-vue", "LaunchCode", "Dev tools", "Bayer dev tools", "Buying", "Fitness", "INSPIRE", "Vue learning", "Bayer Sites", "LEPSI", "Bayer", and "Personal". Below the navigation bar, there are three tabs: "Recipe Book" (selected), "Authenticate" (disabled), and "Shopping List". The main content area contains a login form with fields for "E-Mail" (containing "abc@abc.com") and "Password" (containing "....."). At the bottom of the form are two buttons: "Sign Up" and "Switch to Login".



PROGRAMMATIC CREATION

```
import { Directive, ViewContainerRef } from "@angular/core";
@Directive({
  selector: '[appPlaceholder'
})
export class PlaceholderDirective {
  constructor(public viewContainerRef: ViewContainerRef) {
  }
}
```



PROGRAMMATIC CREATION

```
<ng-template appPlaceholder></ng-template>
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <!-- <div class="alert alert-danger" *ngIf="error">
      <p> {{ error }} </p>
    </div> -->
    <!-- <app-alert [message]="error" *ngIf="error" (close)="onHandleError()"></app-alert> -->
    <div *ngIf="isLoading" style="text-align: center;">
      <app-loading-spinner></app-loading-spinner>
    </div>
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input
          type="email"
          id="email"
          class="form-control"
          ngModel
          name="email"
          required
          email
        />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          class="form-control"
          ngModel
          name="password"
          required
          minlength="6"
        />
      </div>
      <div>
        <button
          class="btn btn-primary"
          type="submit"
          [disabled]="!authForm.valid"
        >
          {{ isLoginMode ? 'Login' : 'Sign Up' }}
        </button>
        |
        <button class="btn btn-primary" (click)="onSwitchMode()" type="button">
          Switch to {{ isLoginMode ? 'Sign Up' : 'Login' }}
        </button>
      </div>
    </form>
  </div>
</div>
```



PROGRAMMATIC CREATION

```
import { PlaceholderDirective } from './placeholder/placeholder.directive';
import { AlertComponent } from './shared/alert/alert.component';

import { Router } from '@angular/router';
import { AuthService, AuthResponseData } from './auth.service';
import { Component, ComponentFactoryResolver, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginPage = true;
  isLoading = false;
  error: string = null;
  @ViewChild(PlaceholderDirective, null) alertHost: PlaceholderDirective;

  constructor(private authService: AuthService, private router: Router, private componentFactoryResolver: ComponentFactoryResolver) {}

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

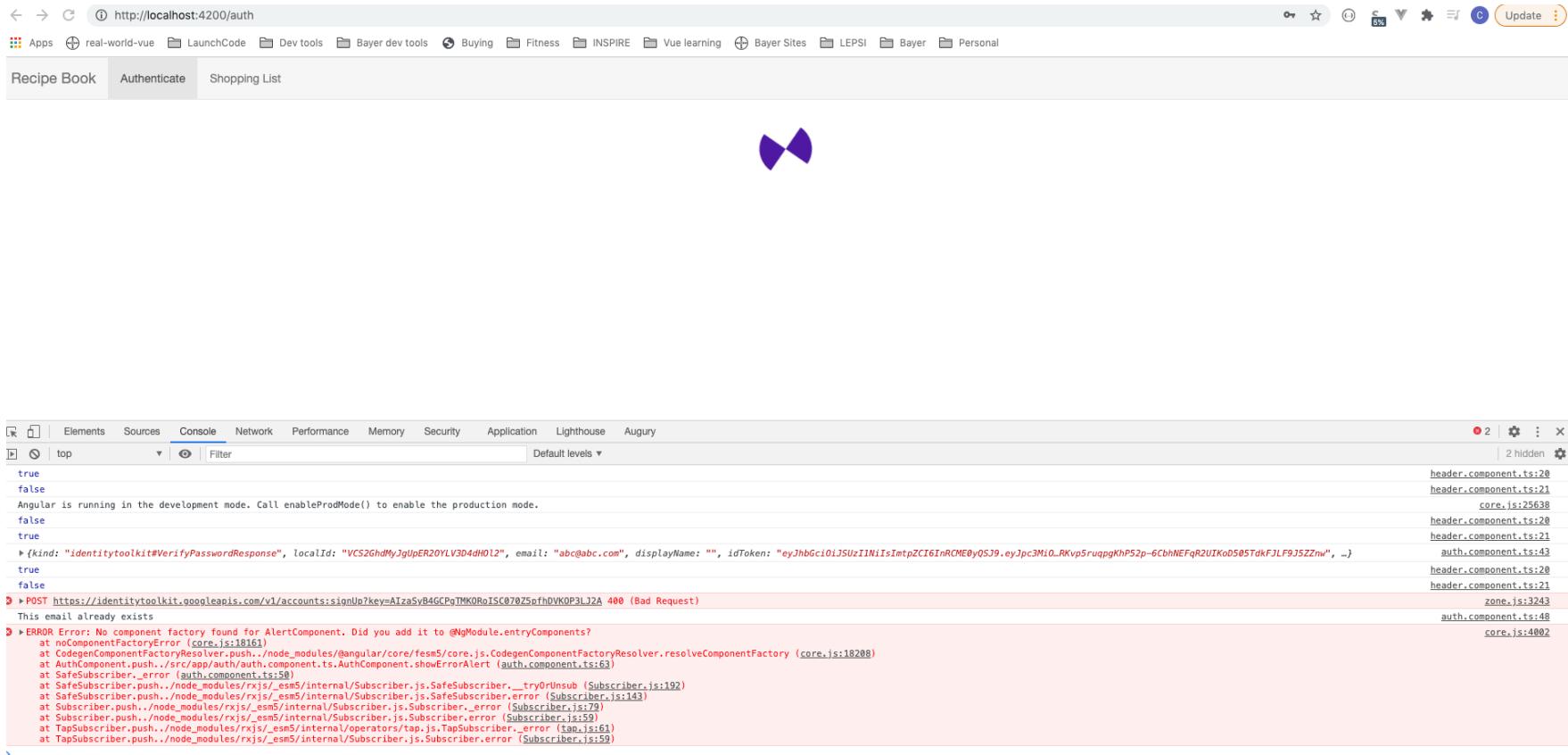
  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginPage) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        console.log(resData);
        this.isLoading = false;
        this.router.navigate(['/recipes']);
      },
      errorMessage => {
        this.error = errorMessage;
        this.alertHost.showAlert(errorMessage);
      }
    );
  }
}
```



PROGRAMMATIC CREATION



The screenshot shows a web browser window with the URL <http://localhost:4200/auth>. The page displays a simple form with fields for 'Email' and 'Password'. Below the form is a purple decorative icon. At the bottom of the page, there is a footer with links: 'About', 'Privacy Policy', 'Terms of Service', and 'Help'.

The browser's address bar shows the URL <https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=A1zaSyB4GCpTMK0RoISC070Z5pfhDVK0P3LJ2A> followed by a red error message: "400 (Bad Request)".

The developer tools console tab is open, showing the following log entries:

```
true
false
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
false
true
> {kind: "identitytoolkit#VerifyPasswordResponse", localId: "VCS2GhdMyJgUpER20YLV3D4dH012", email: "abc@abc.com", displayName: "", idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6InRCME0y0Sj9.eyJpc3Mi0.RKvp5ruqpgKhP52p-6CbhNEFqR2UIKoD505TdkFJLF9J5ZZnw", ...}
true
false
POST https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=A1zaSyB4GCpTMK0RoISC070Z5pfhDVK0P3LJ2A 400 (Bad Request)
This email already exists
ERROR Error: No component factory found for AlertComponent. Did you add it to @NgModule.entryComponents?
at noComponentFactoryError (core.js:1816)
at CodegenComponentFactoryResolver.push../node_modules/@angular/core/fesm5/core.js.CodegenComponentFactoryResolver.resolveComponentFactory (core.js:1828)
at SafeSubscriber.push../src/app/auth/auth.component.ts:AuthComponent.showErrorAlert (auth.component.ts:63)
at SafeSubscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber._tryOrInsub (Subscriber.js:192)
at SafeSubscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber.error (Subscriber.js:143)
at Subscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.Subscriber._error (Subscriber.js:79)
at Subscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.Subscriber.error (Subscriber.js:59)
at TapSubscriber.push../node_modules/rxjs/_esm5/internal/operators/tap.js.TapSubscriber._error (tap.js:61)
at TapSubscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.Subscriber.error (Subscriber.js:59)
```

PROGRAMMATIC CREATION

```
import { PlaceholderDirective } from './placeholder/placeholder.directive';
import { AlertComponent } from './shared/alert/alert.component';
import { AuthInterceptorService } from './auth/auth.interceptor.service';
import { LoadingSpinnerComponent } from './shared/loading-spinner/loading-spinner.component';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { RecipesComponent } from './recipes/recipes.component';
import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
import { ShoppingListComponent } from './shopping-list/shopping-list.component';
import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
import { DropdownDirective } from './shared/dropdown.directive';
import { ShoppingListService } from './shopping-list/shopping-list.service';
import { AppRoutingModule } from './app-routing.module';
import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
import { RecipeService } from './recipes/recipe.service';
import { AuthComponent } from './auth/auth.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    RecipesComponent,
    RecipeListComponent,
    RecipeDetailComponent,
    RecipeItemComponent,
    ShoppingListComponent,
    ShoppingEditComponent,
    DropdownDirective,
    RecipeStartComponent,
    RecipeEditComponent,
    AuthComponent,
    LoadingSpinnerComponent,
    AlertComponent,
    PlaceholderDirective
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [
    AuthInterceptorService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



DATA BINDING AND EVENT BINDING

```
import { PlaceholderDirective } from './placeholder/placeholder.directive';
import { AlertComponent } from '../shared/alert/alert.component';

import { Router } from '@angular/router';
import { AuthService, AuthResponseData } from './auth.service';
import { Component, ComponentFactoryResolver, ViewChild, OnDestroy } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable, Subscription } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent implements OnDestroy {
  isLoginPage = true;
  isLoading = false;
  error: string = null;
  @ViewChild(PlaceholderDirective, null) alertHost: PlaceholderDirective;
  private closeSub: Subscription;
  constructor(private authService: AuthService, private router: Router, private componentFactoryResolver: ComponentFactoryResolver) {}

  ngOnDestroy() {
    if(this.closeSub) {
      this.closeSub.unsubscribe();
    }
  }

  onSwitchMode() {
    this.isLoginPage = !this.isLoginPage;
  }

  onSubmit(form: NgForm) {
    if(!form.valid) {
      return;
    }

    let authObs: Observable<AuthResponseData>;
    this.isLoading = true;
    if(this.isLoginPage) {
      authObs = this.authService.login(form.value.email, form.value.password);
    } else {
      authObs = this.authService.signup(form.value.email, form.value.password);
    }

    authObs.subscribe(
      resData => {
        if(resData.error) {
          this.error = resData.error;
          this.alertHost.nativeElement.innerHTML = resData.message;
          this.closeSub = this.componentFactoryResolver.createFactory(AlertComponent).create(this.alertHost.nativeElement).subscribe(() => {
            this.error = null;
            this.alertHost.nativeElement.innerHTML = '';
          });
        } else {
          this.router.navigate(['/dashboard']);
        }
      },
      err => {
        this.error = 'Something went wrong!';
        this.alertHost.nativeElement.innerHTML = this.error;
        this.closeSub = this.componentFactoryResolver.createFactory(AlertComponent).create(this.alertHost.nativeElement).subscribe(() => {
          this.error = null;
          this.alertHost.nativeElement.innerHTML = '';
        });
      }
    );
  }
}
```



QUESTIONS?



STUDIO

Course Project – Unit Testing

