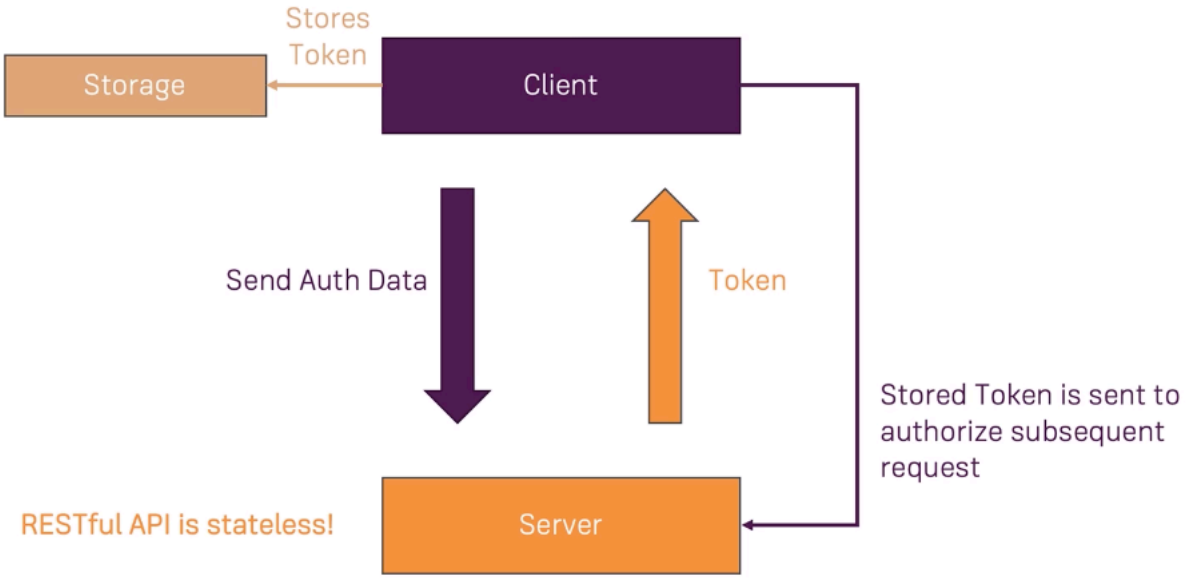
launch_code

- 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 {}
```

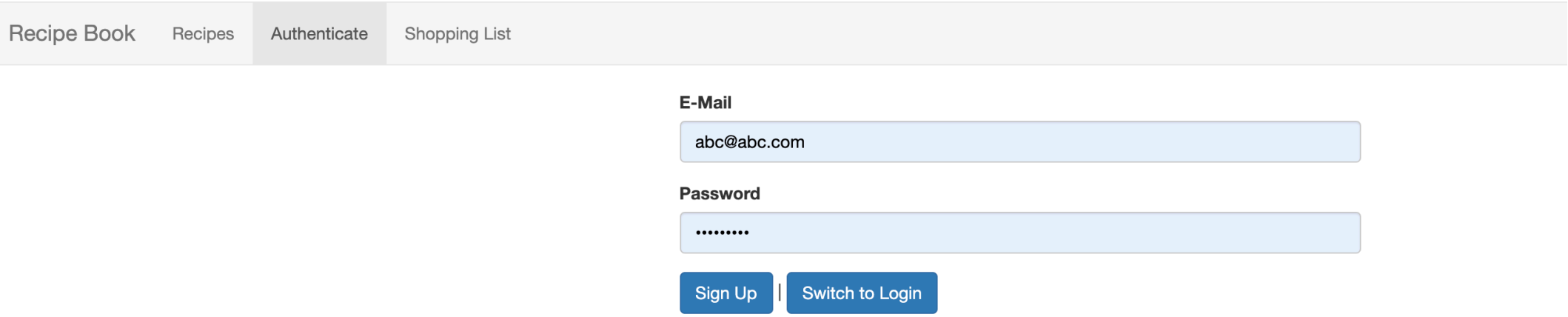
APP MODULE

```
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

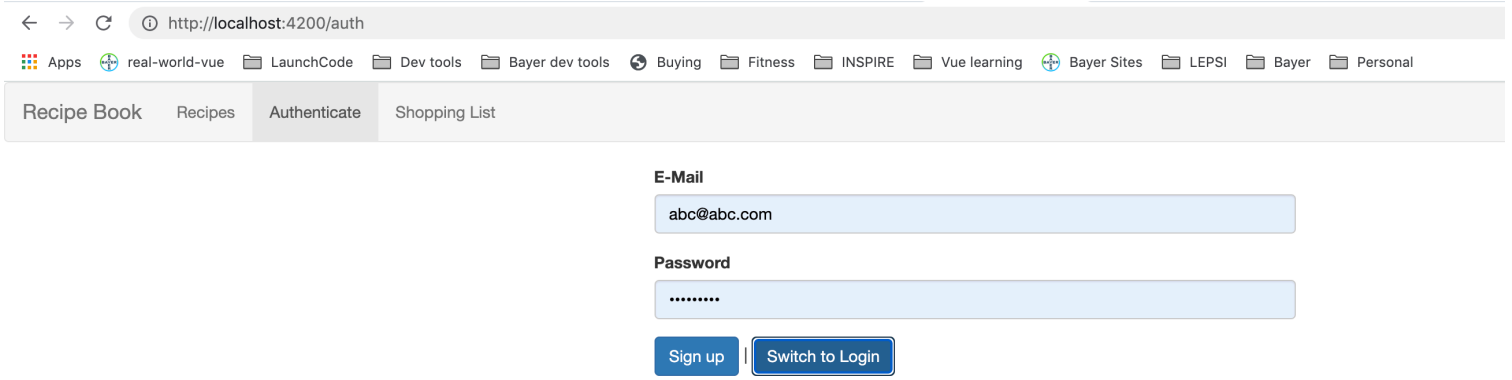


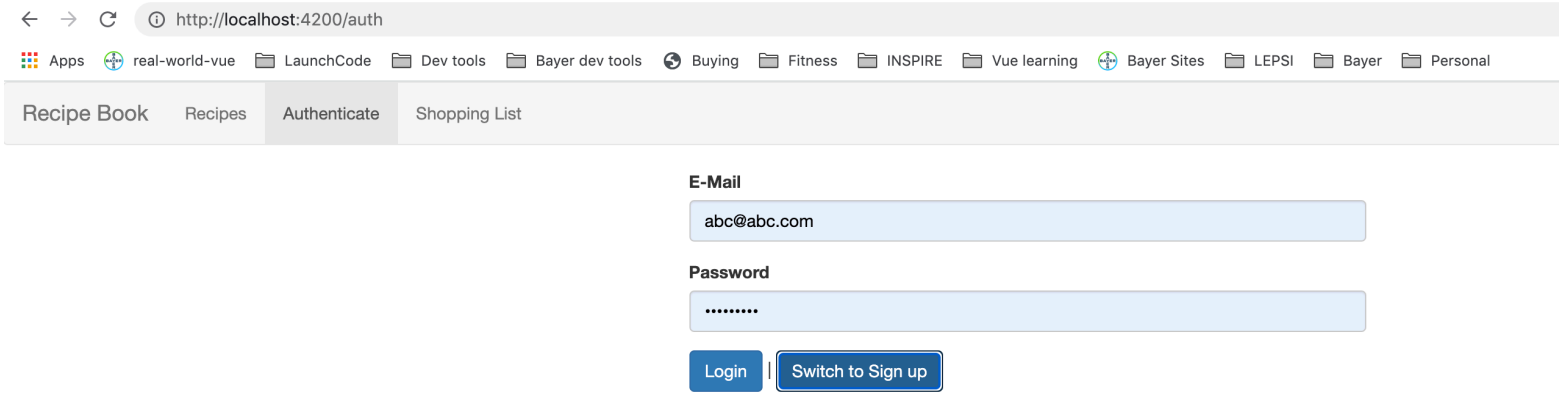
SWITCHING BETWEEN AUTH MODES

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

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

  isLoginMode = true;

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

}
```

```
<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">{{ 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>
```

SWITCHING BETWEEN AUTH MODES





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 {

  isLoginMode = true;

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

  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>
```

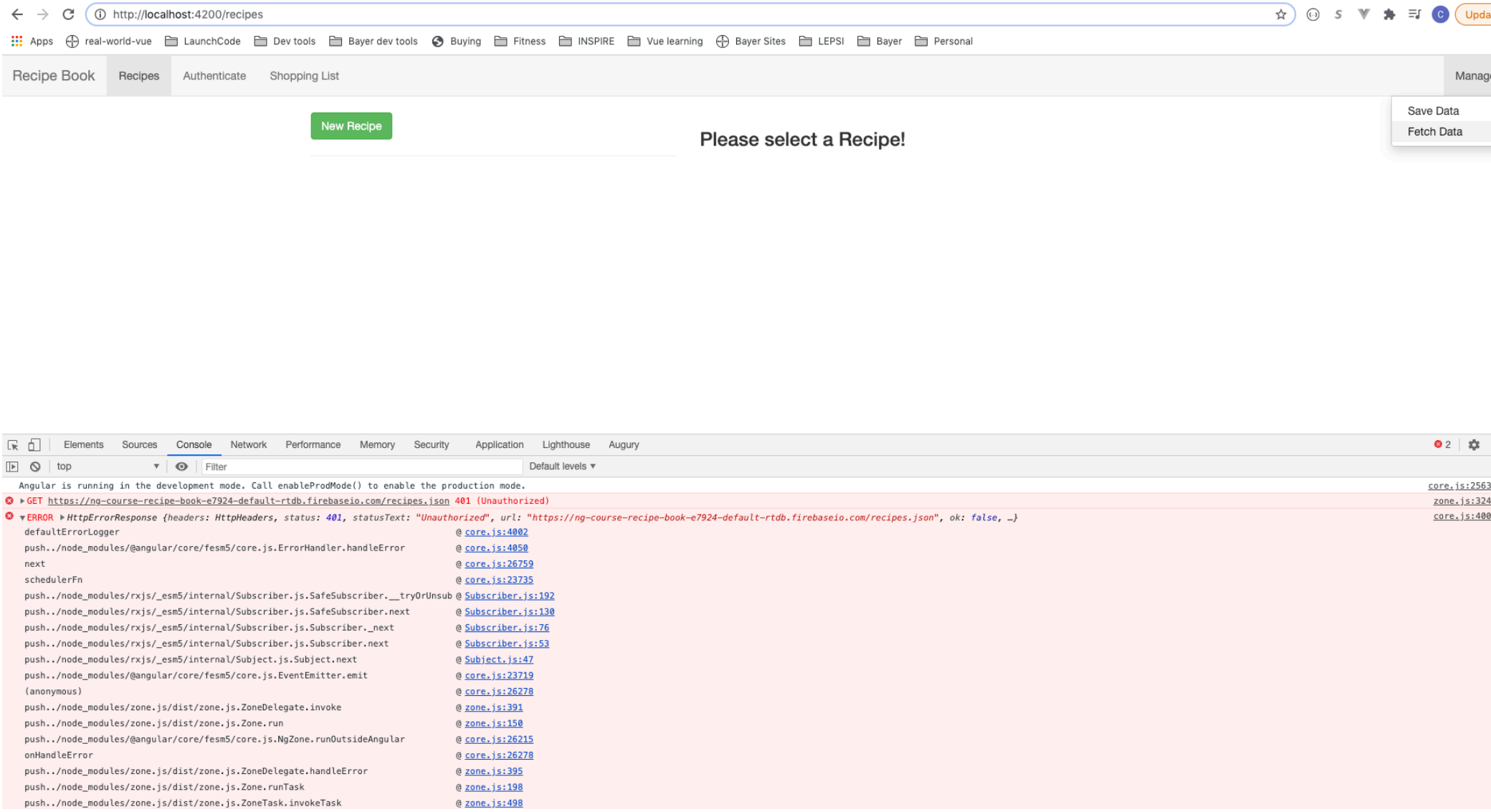
PREPARING THE BACKEND

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

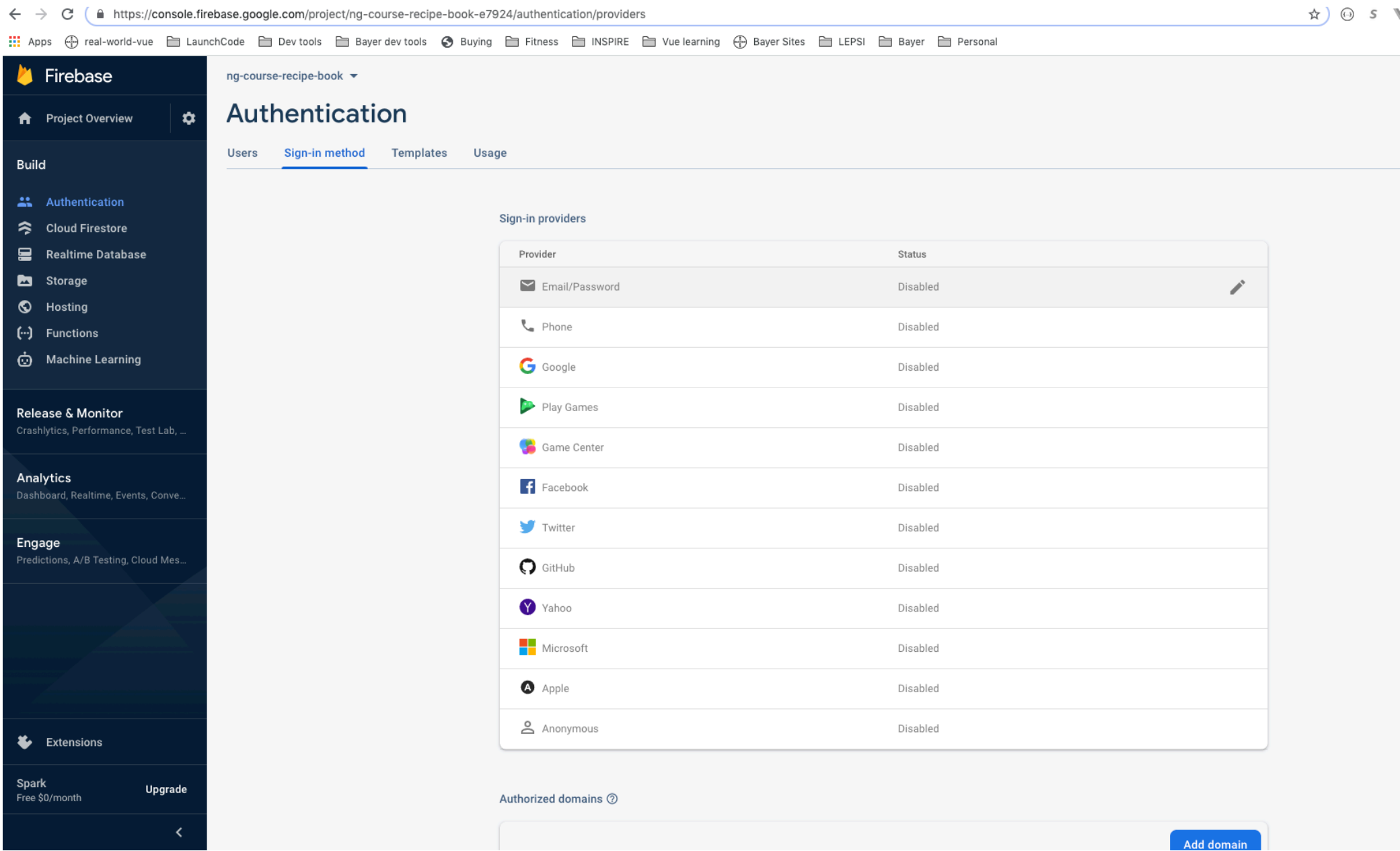
PREPARING THE BACKEND

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

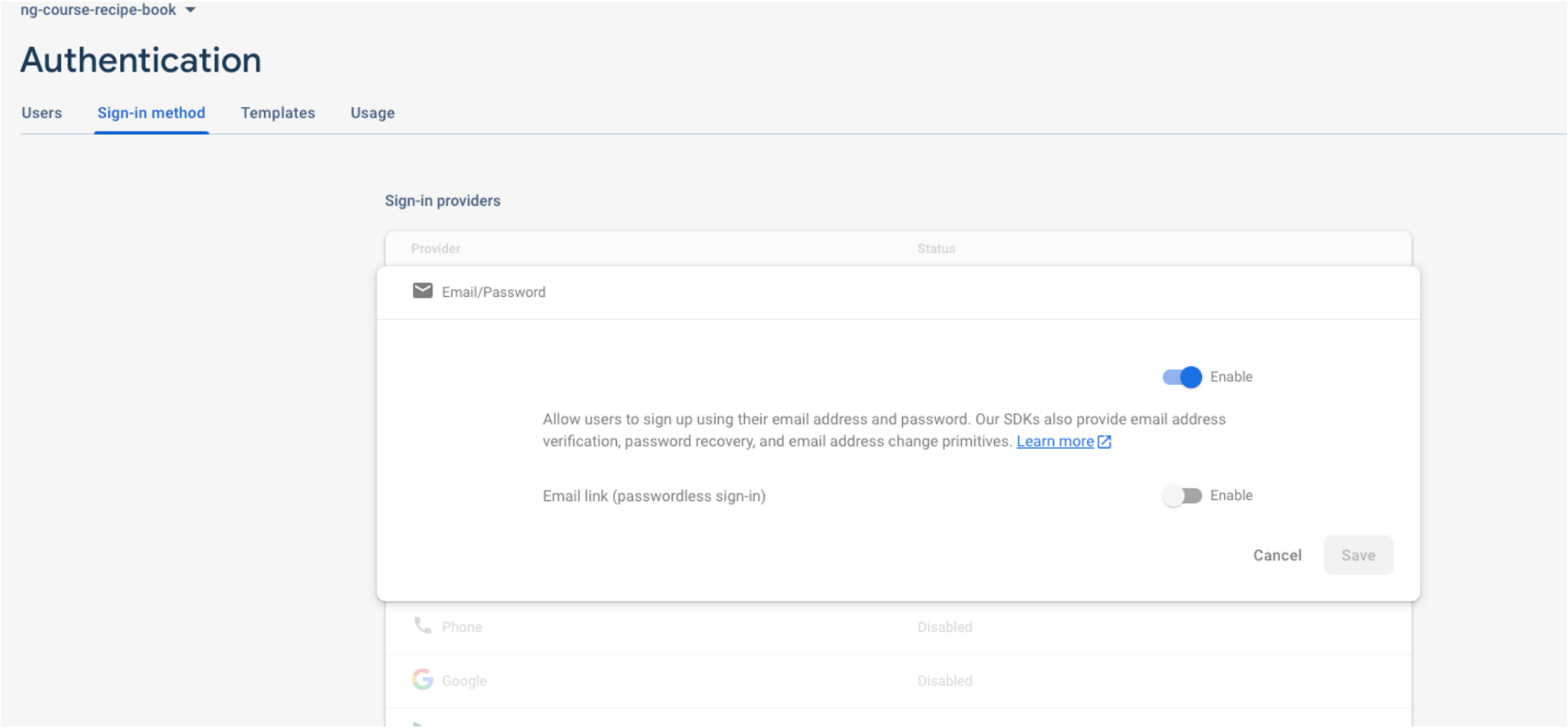
PREPARING THE BACKEND



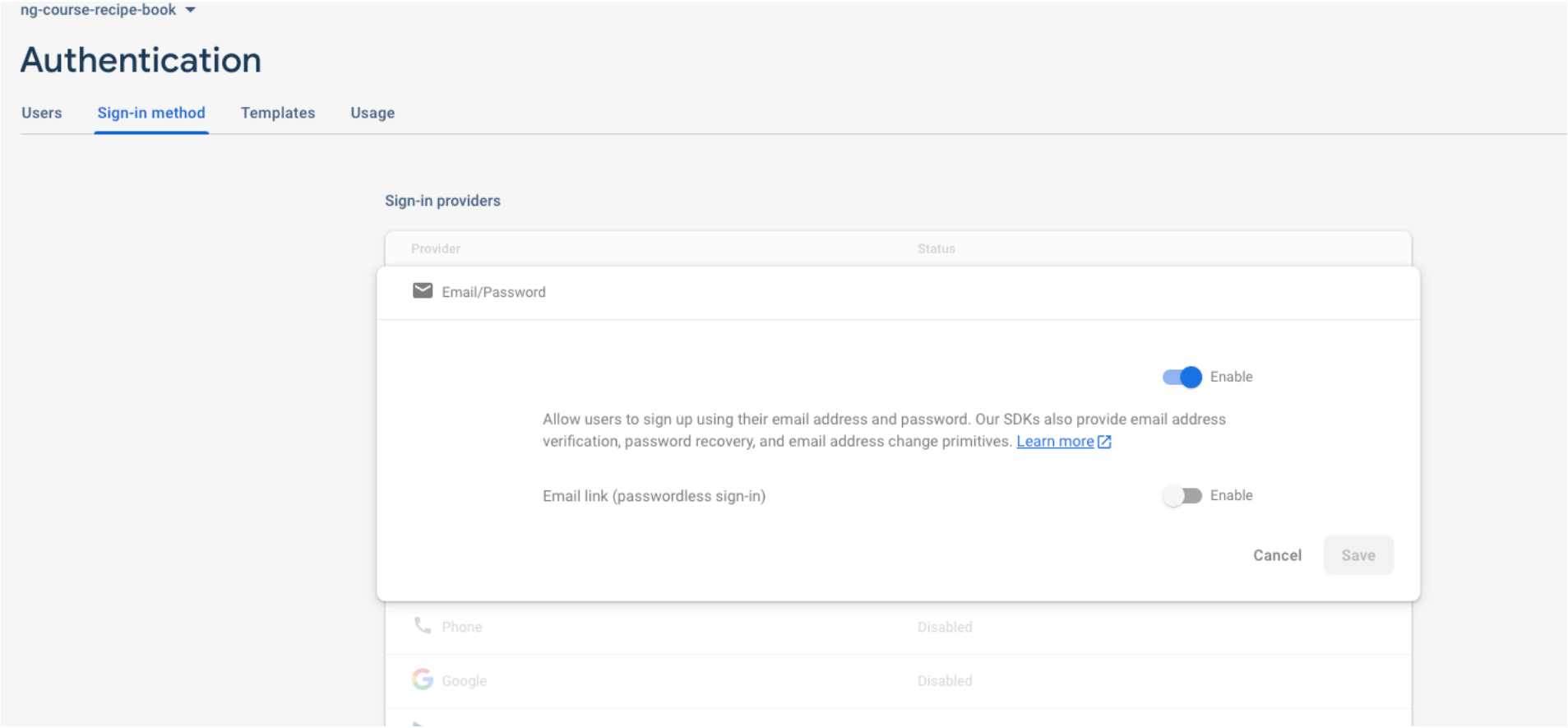
PREPARING THE BACKEND



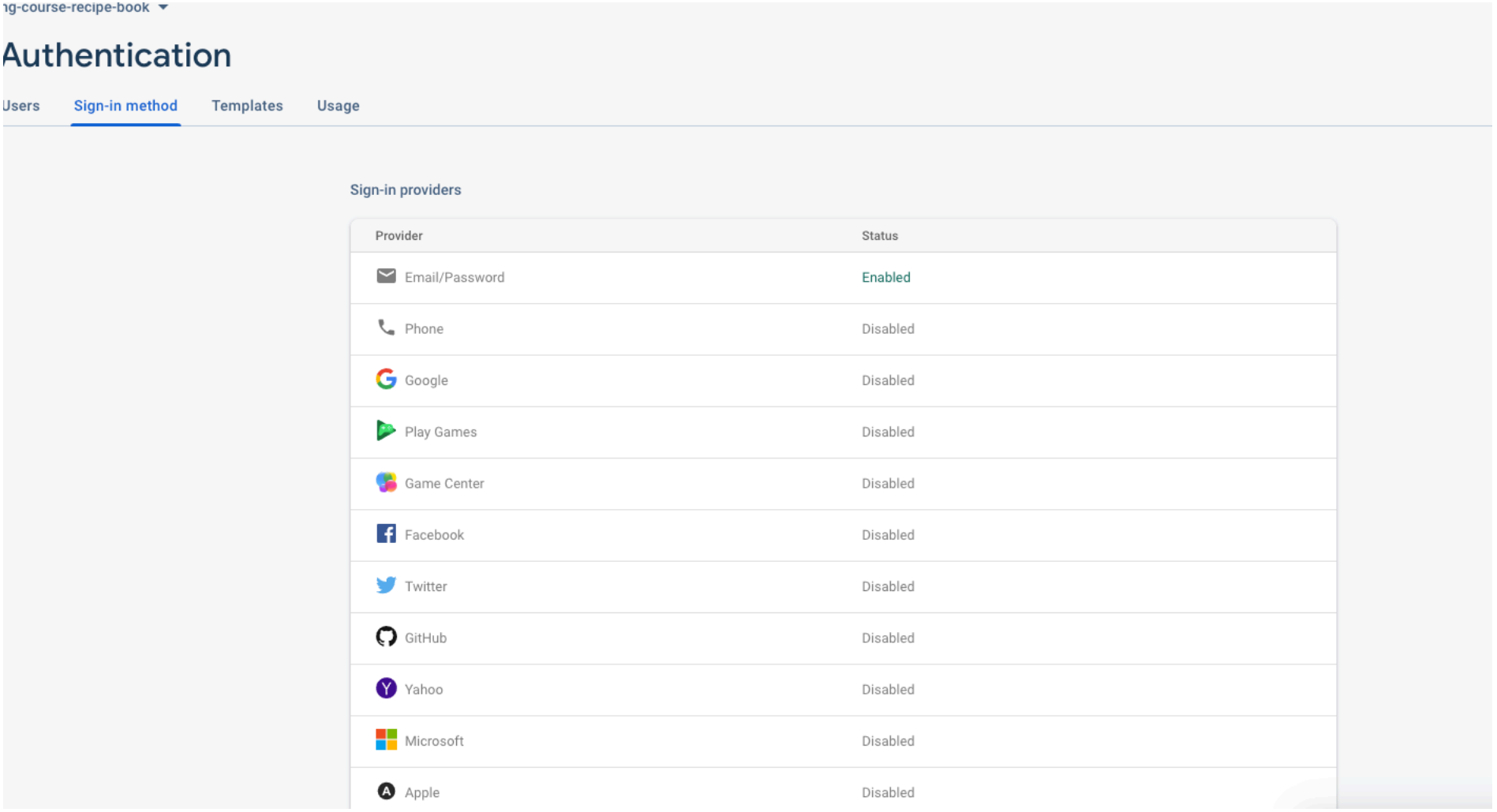
PREPARING THE BACKEND



PREPARING THE BACKEND



PREPARING THE BACKEND



PREPARING THE SINGUP REQUEST

PREPARING THE SINGUP REQUEST

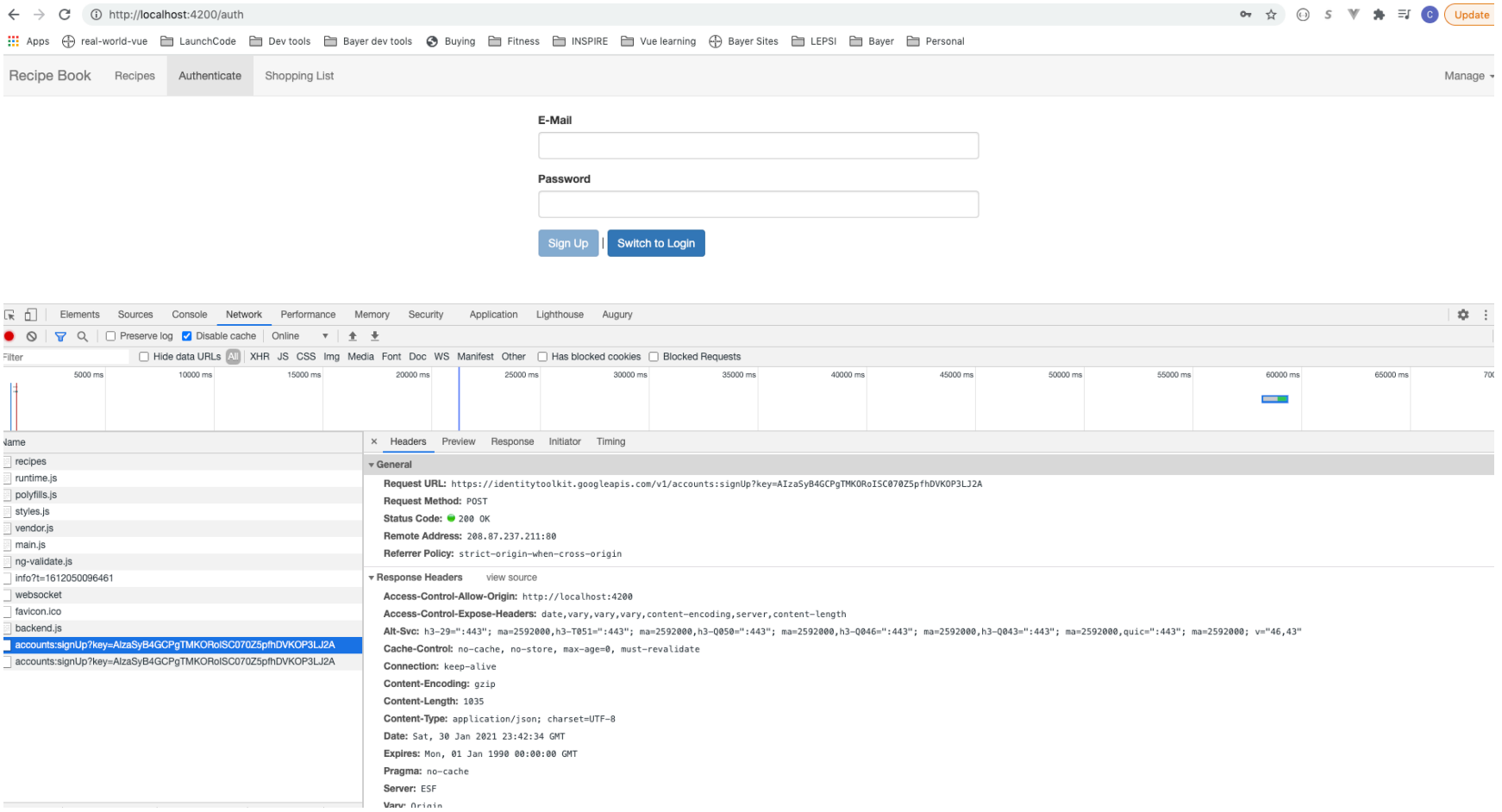
```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";

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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        })
    }

}
```

PREPARING THE SINGUP REQUEST

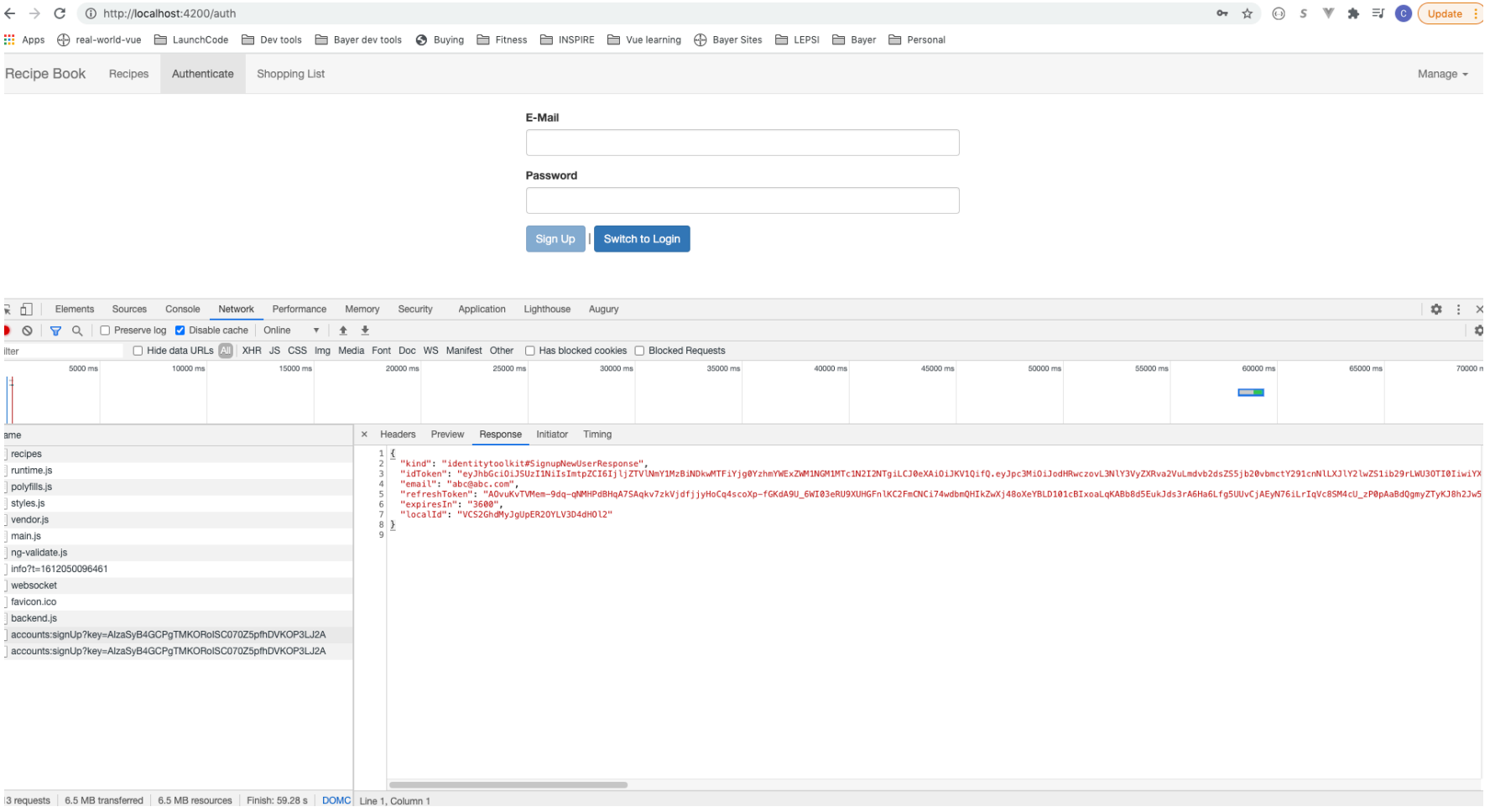
```
import { HttpClient } from '@angular/common/http';
import { Injectable } from "@angular/core";

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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            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 {
  isLoginMode = true;

  constructor(private authService: AuthService) {

  }

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

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

    } 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();
  }
}
```

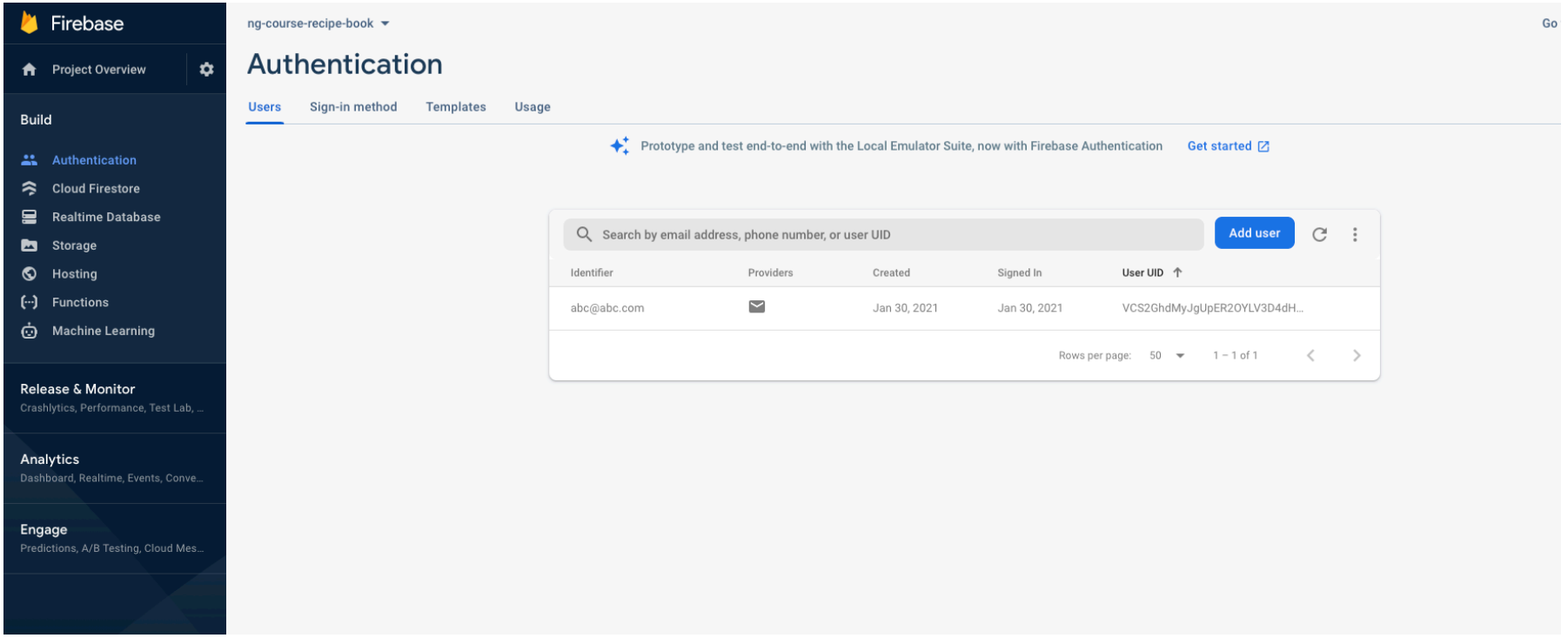
PREPARING THE SINGUP REQUEST



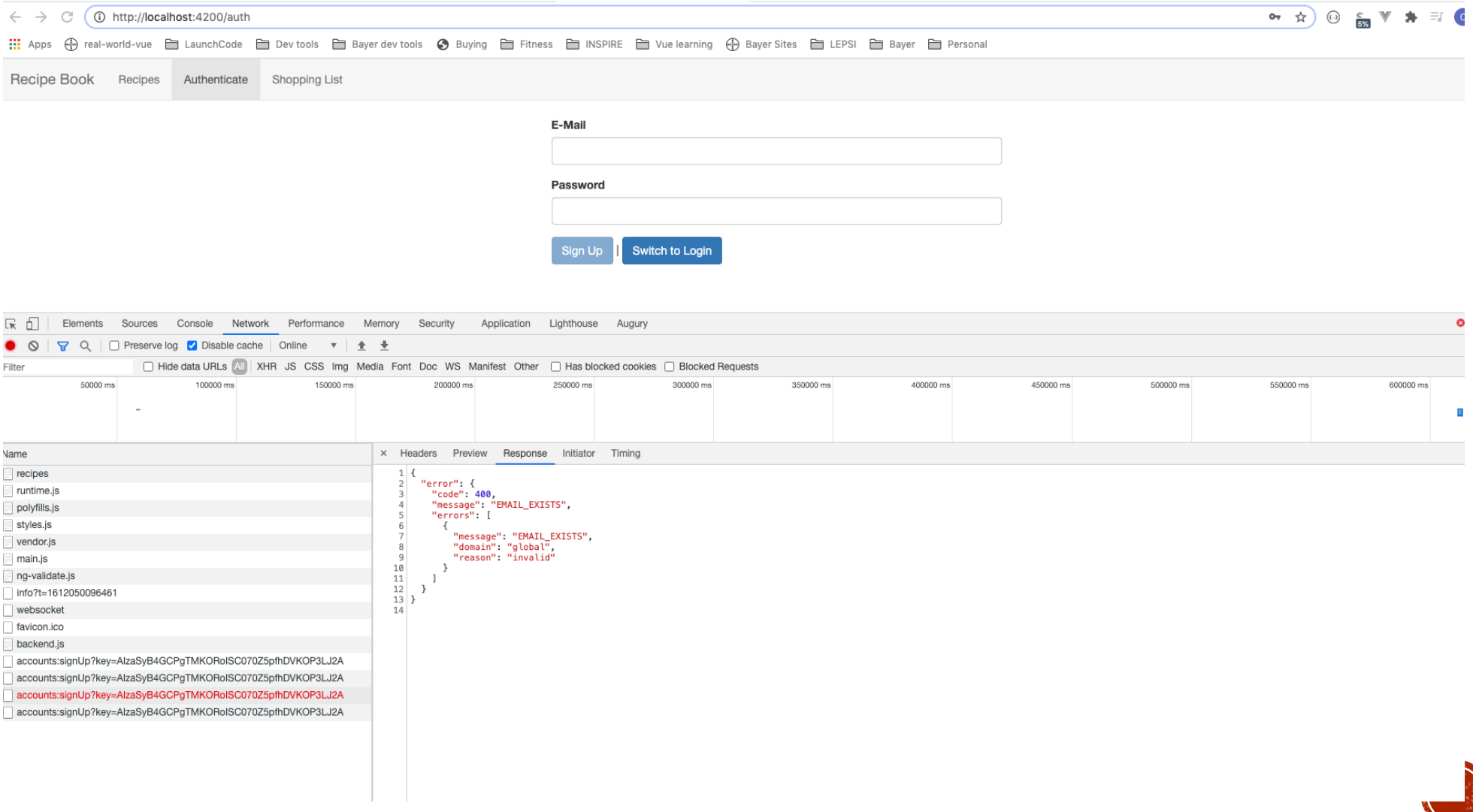
PREPARING THE SINGUP REQUEST



PREPARING THE SINGUP REQUEST



PREPARING THE SINGUP REQUEST



ADDING A LOADING SPINNER AND ERROR HANDLING

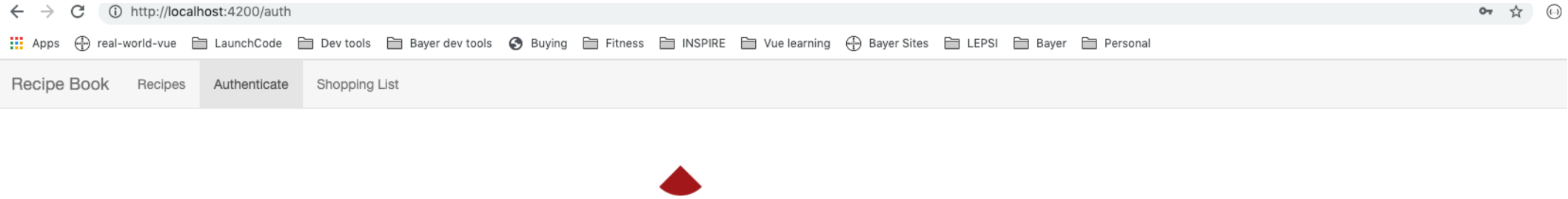
```
import { AuthService } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent {
  isLoginMode = true;
  isLoading = false;
  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;
      },
      error => {
        console.log(error);
        this.isLoading = false;
      })
    }
    // this.isLoading = false;
    // console.log(form.value);
    form.reset();
  }
}
```

ADDING A LOADING SPINNER AND ERROR HANDLING

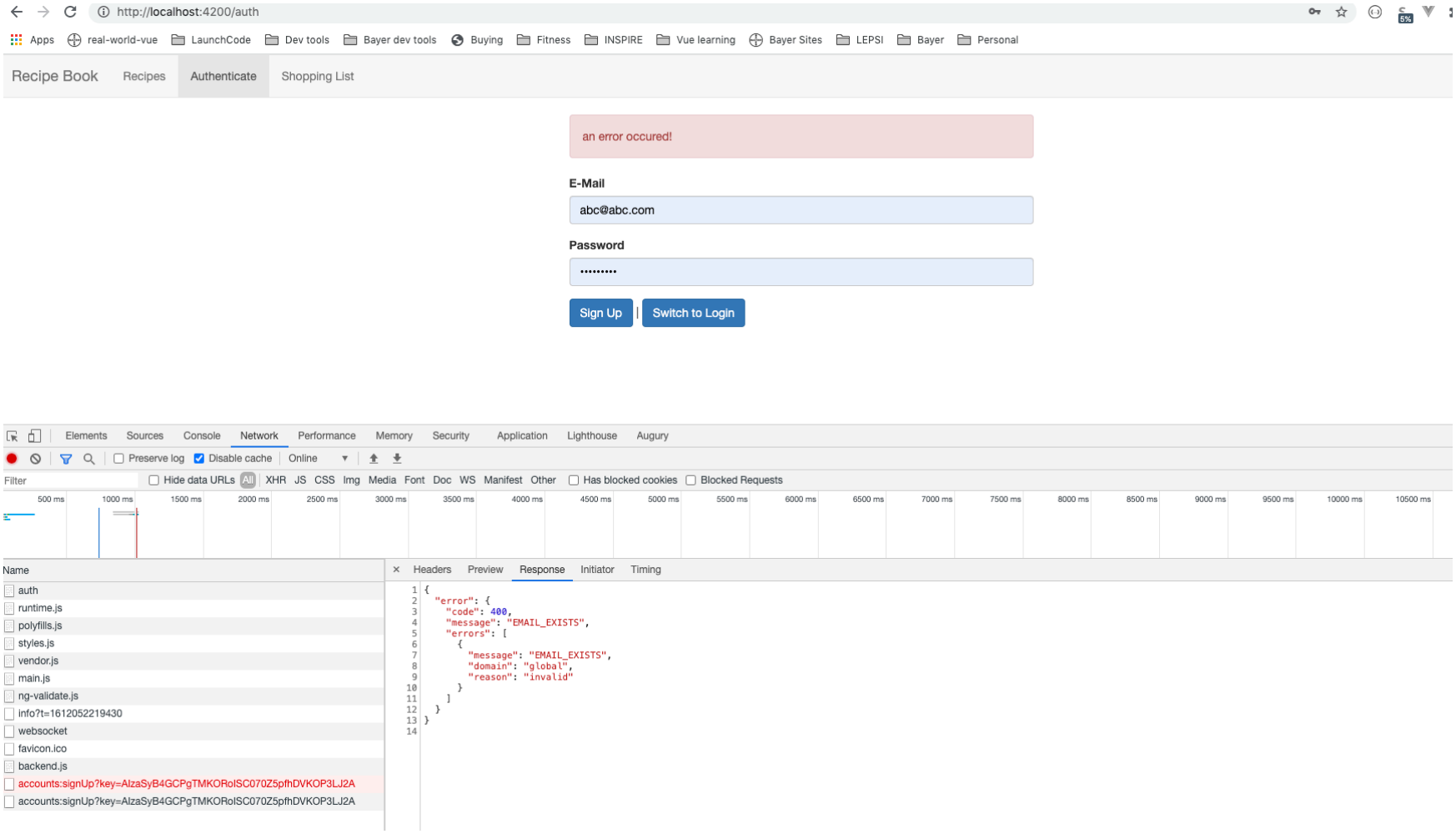
```
import { Component } from "@angular/core";

@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



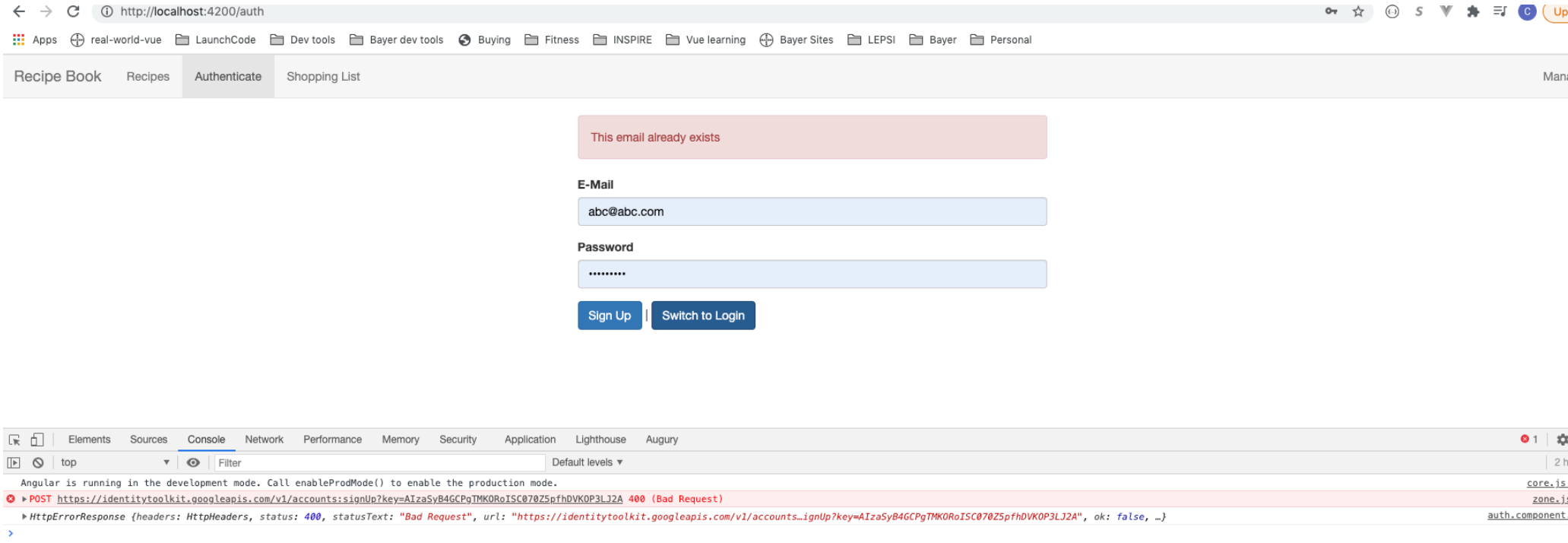
ADDING A LOADING SPINNER AND ERROR HANDLING



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

ADDING A LOADING SPINNER AND ERROR HANDLING



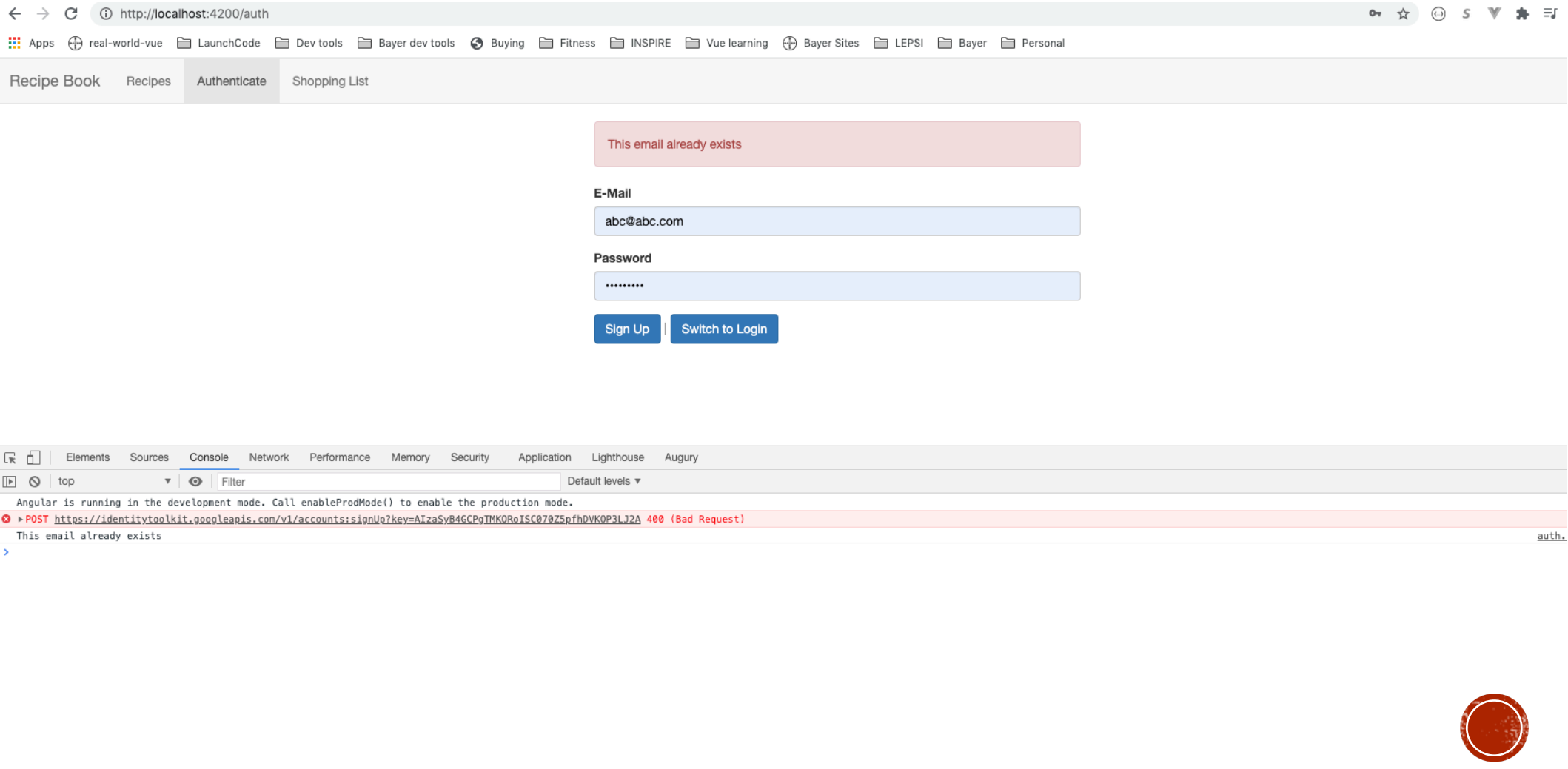
ADDING A LOADING SPINNER AND ERROR HANDLING

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(errorRes => {
            let errorMessage = 'An unknown error occured';
            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



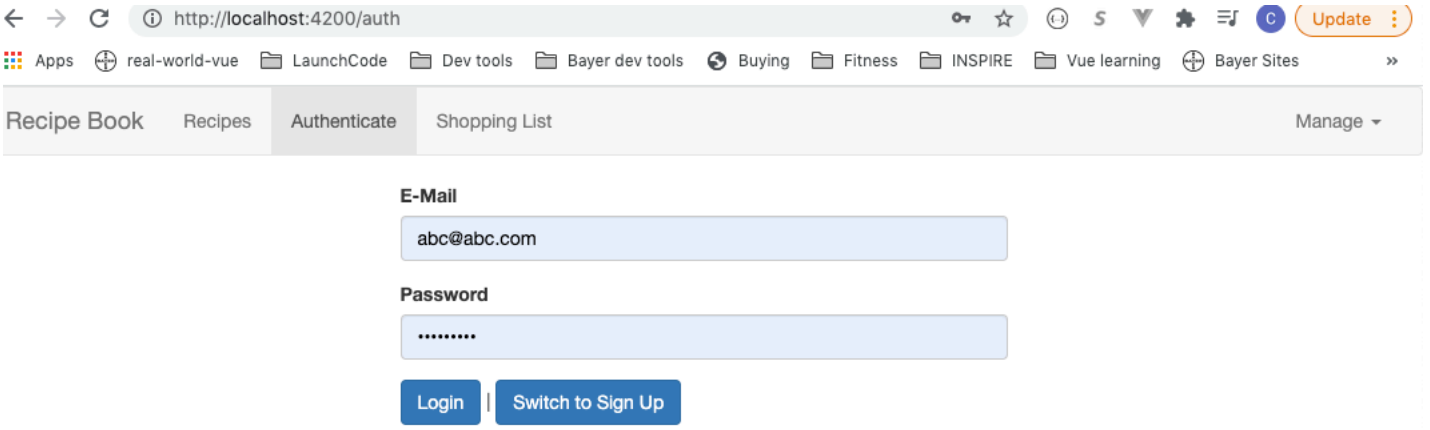
SENDING LOGIN REQUESTS

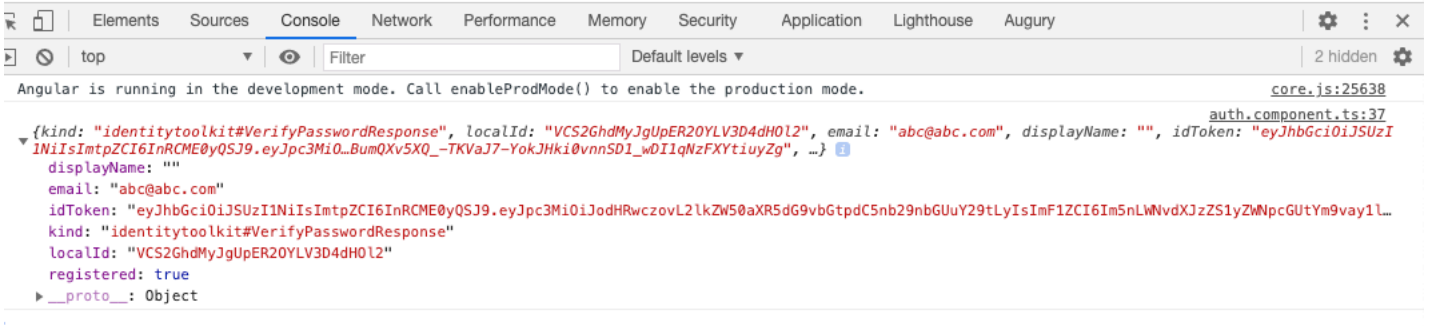
```
import { AuthService, AuthResponseData } from './auth.service';
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(errorRes => {
            let errorMessage = 'An unknown error occured';
            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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
```

SENDING LOGIN REQUESTS





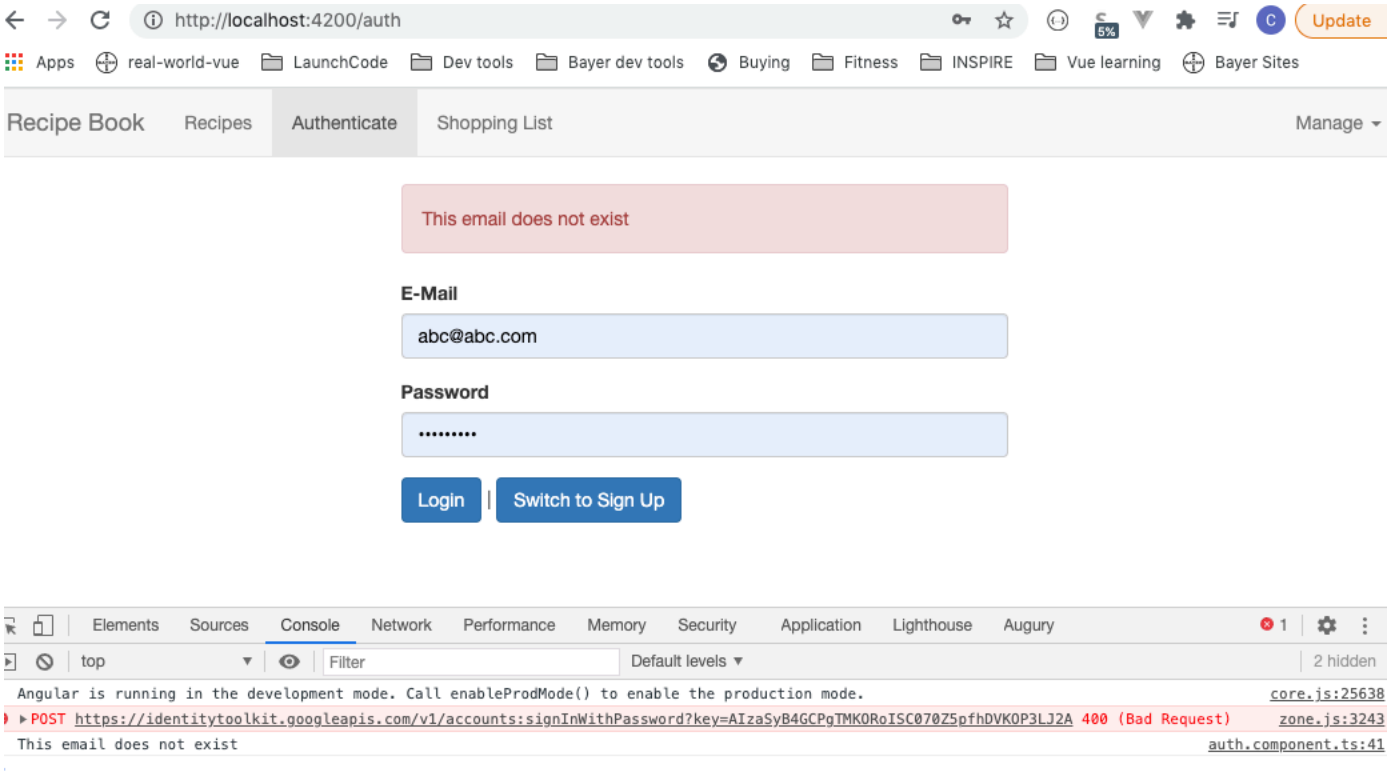
SENDING LOGIN REQUESTS

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror));
    }

    login(email: string, password: string) {
        return
this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror));
    }

    private handleEror(errorRes: HttpErrorResponse) {
        let errorMessage = 'An unknown error occured';
            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 {
  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



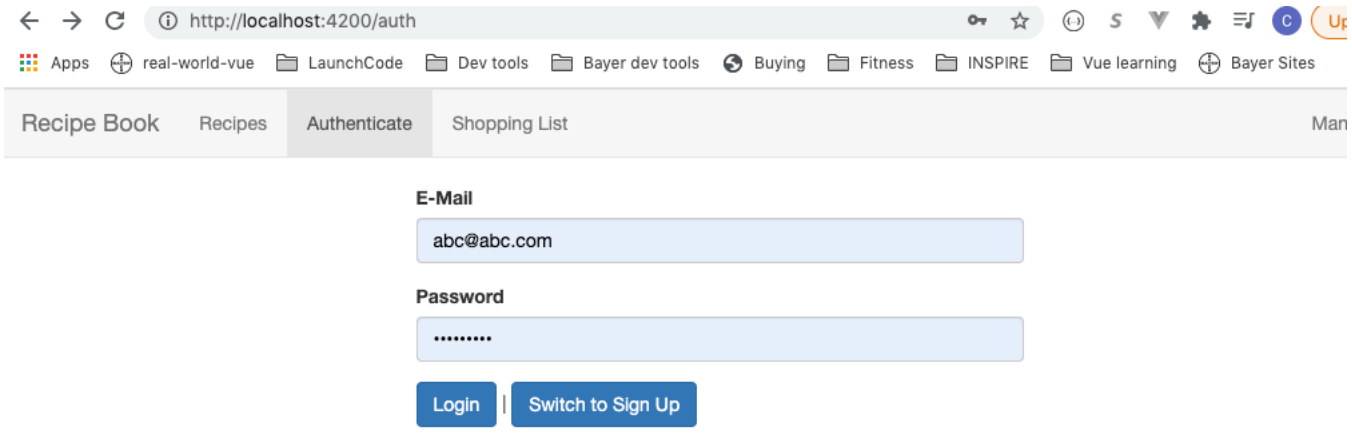
CREATING AND STORING THE USER DATA

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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 {
  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);
```

REFLECTING THE AUTH STATE IN THE UI



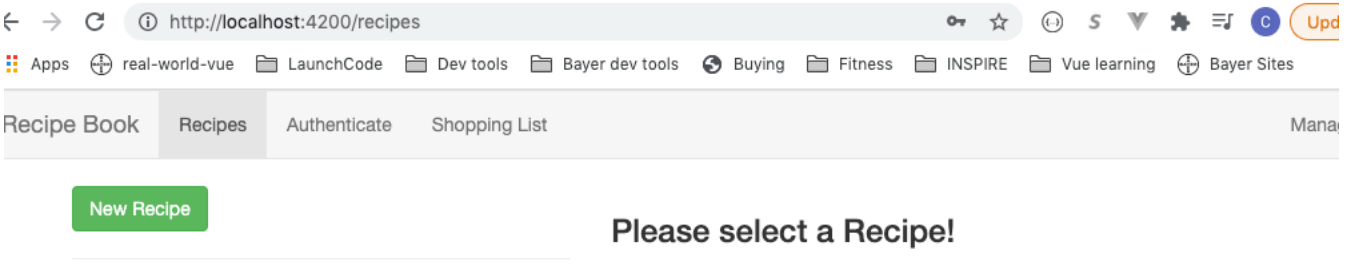
REFLECTING THE AUTH STATE IN THE UI

```
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>
```

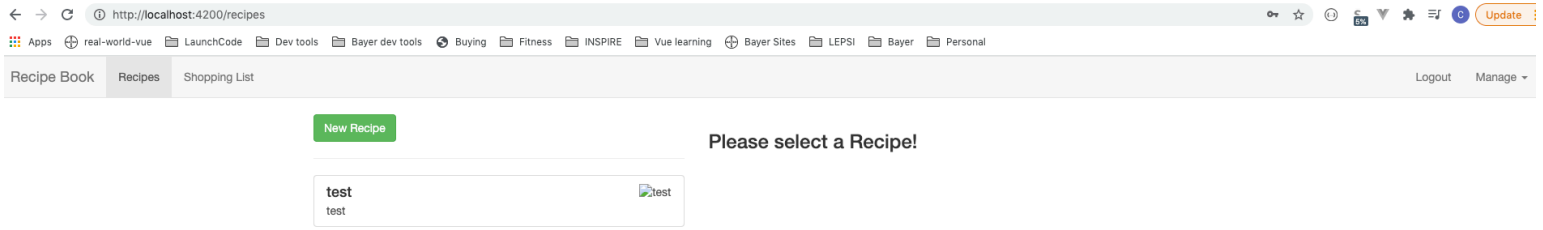
REFLECTING THE AUTH STATE IN THE UI

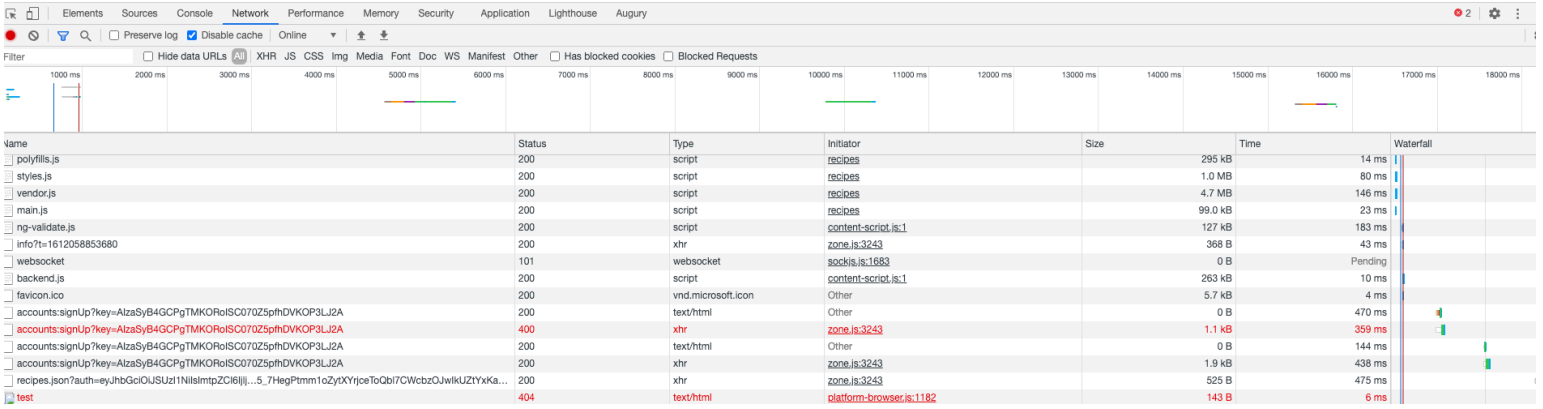


ADDING THE TOKEN TO OUTGOING REQUESTS

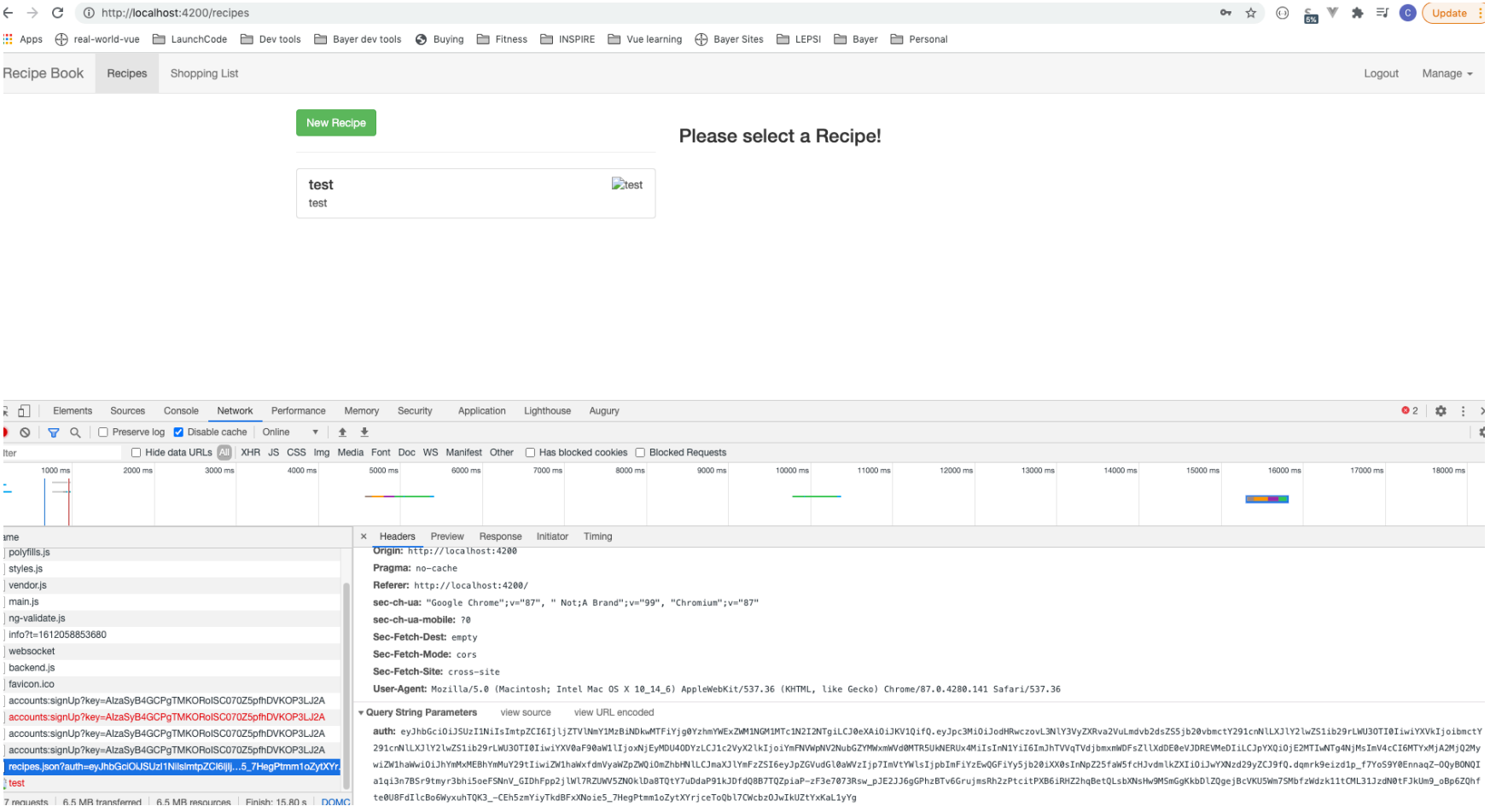
```
import { AuthService } from './../auth/auth.service';
import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { exhaustMap, map, take, tap } from 'rxjs/operators';

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-rtdb.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-rtdb.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





ADDING THE TOKEN TO OUTGOING REQUESTS



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();

    }

}
```

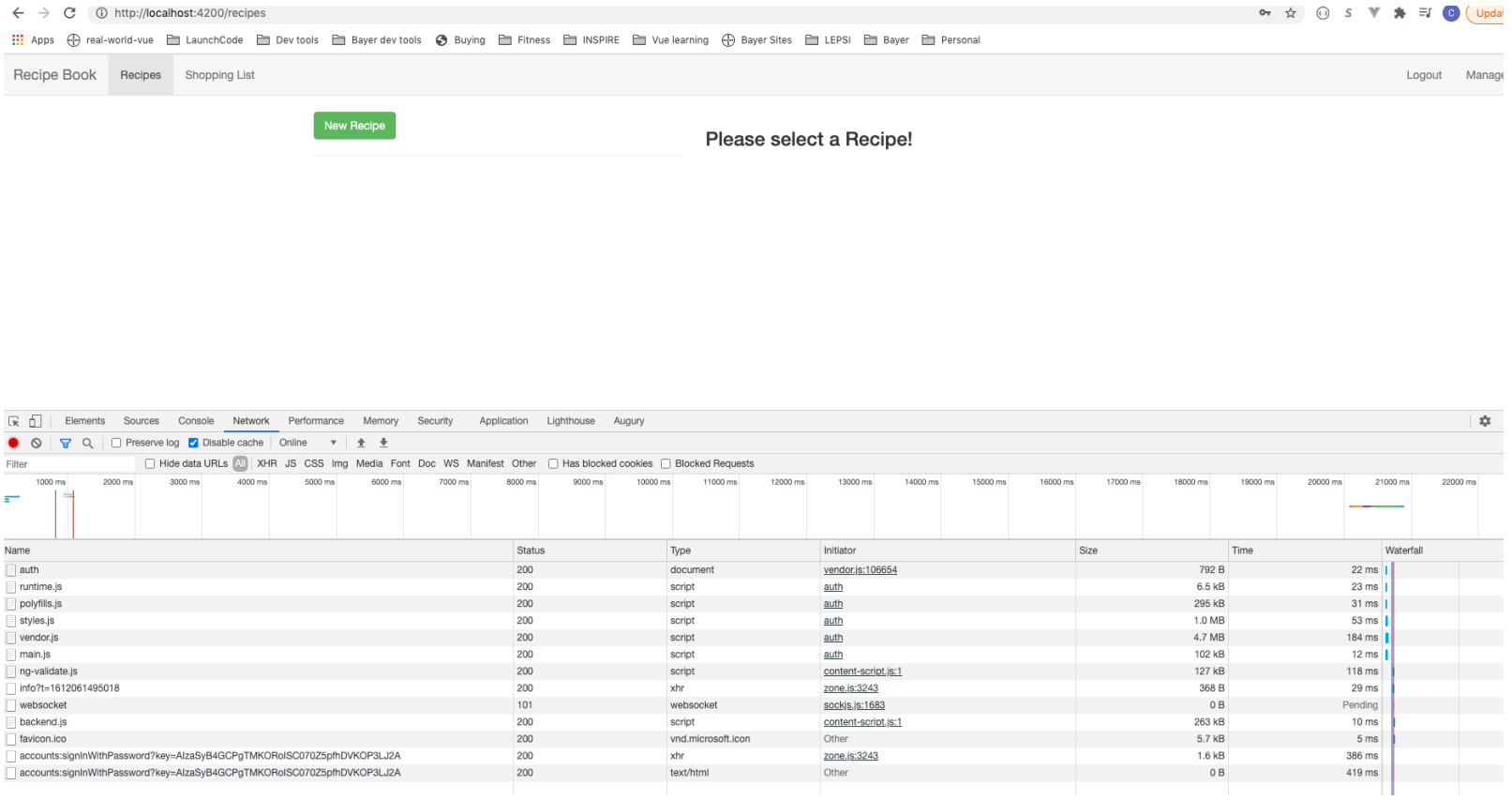
ADDING THE TOKEN WITH AN INTERCEPTOR

```
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-rtdb.firebaseio.com/recipes.json',
        recipes
      )
      .subscribe(response => {
        console.log(response);
      });
  }

  fetchRecipes() {
    return this.http
      .get<Recipe[]>(
        'https://ng-course-recipe-book-e7924-default-rtdb.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



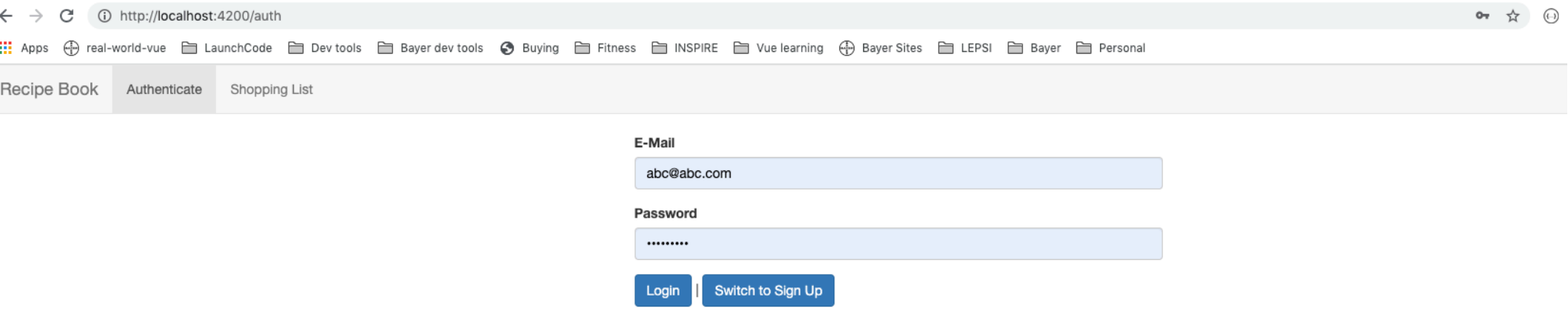
ADDING LOGOUT

```
<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"><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>
```

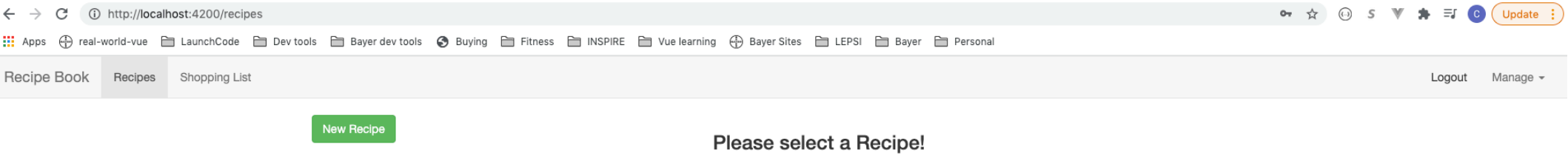
ADDING LOGOUT

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), tap(resData => {
            this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
```

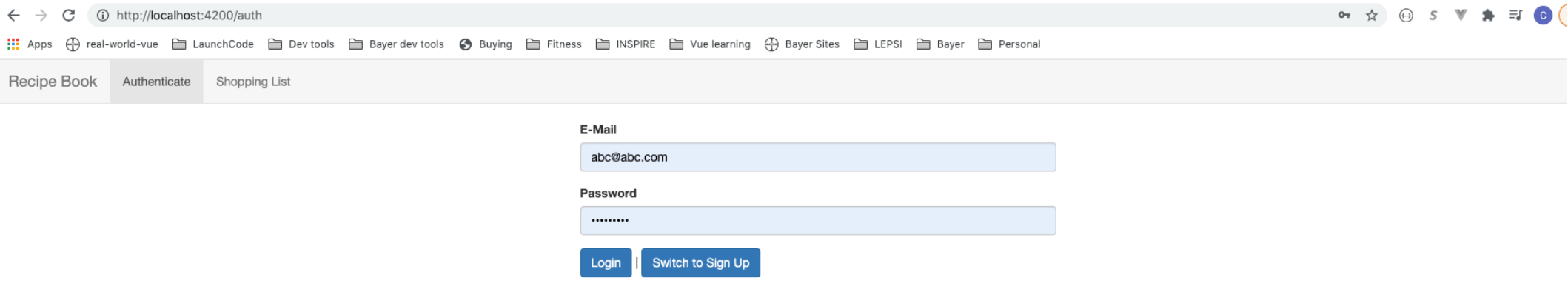
ADDING LOGOUT



ADDING LOGOUT



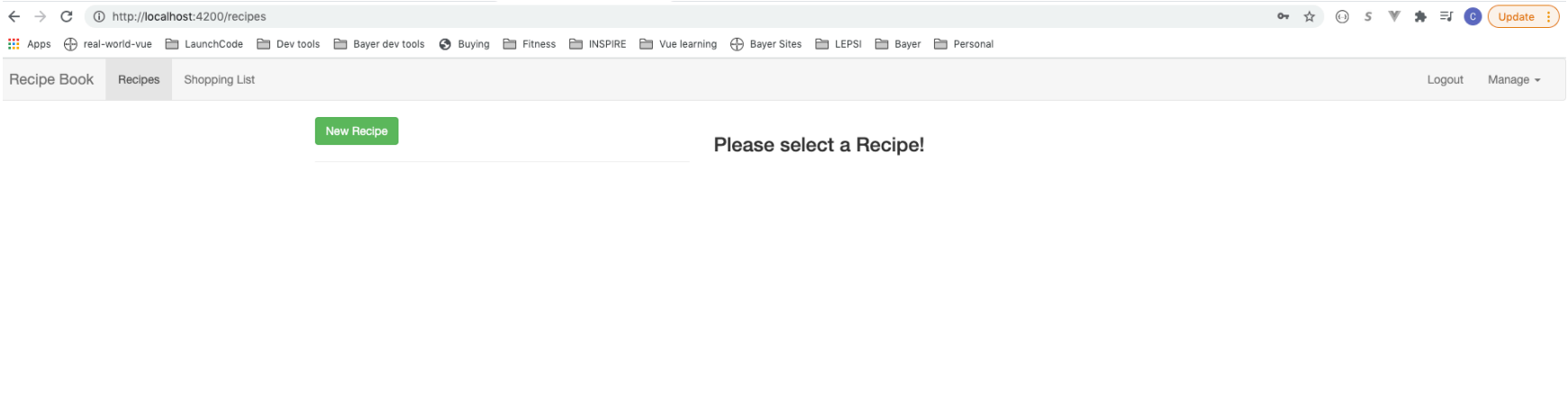
ADDING LOGOUT

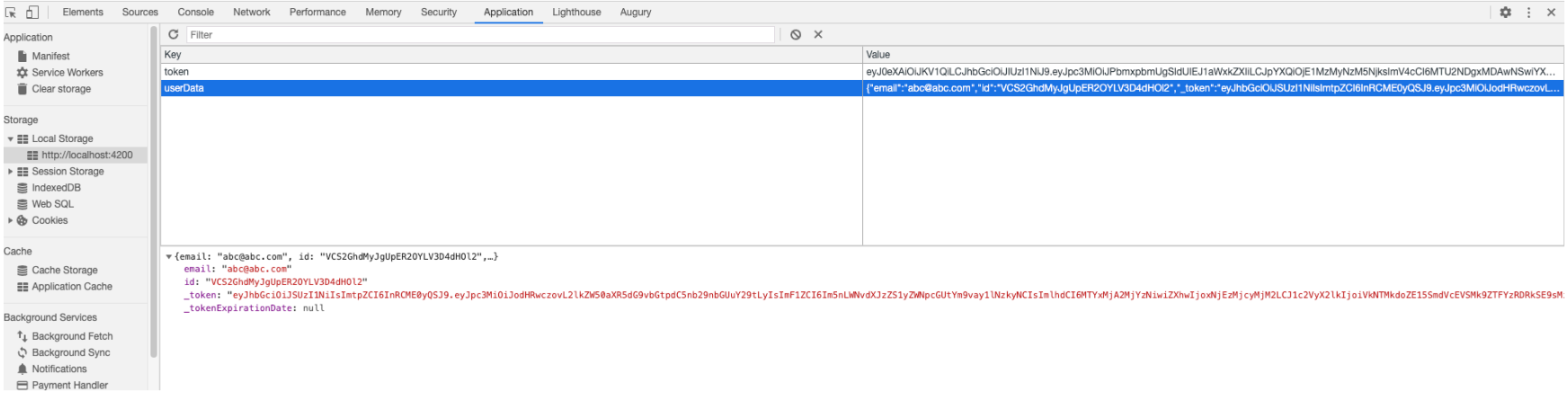


ADDING AUTO LOGIN

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), tap(resData => {
            this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
```

ADDING AUTO LOGIN





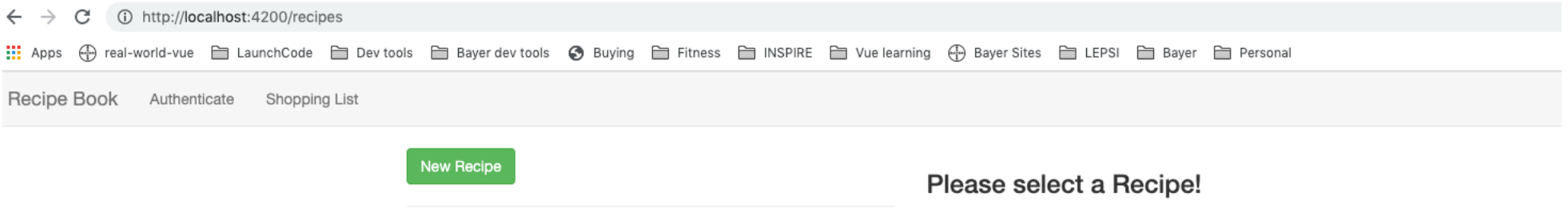
ADDING AUTO LOGIN

```
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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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



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=AIzaSyB4GCPgT
MKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), 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/accounts:signInWithPassword?key=A
IzaSyB4GCPgTMKORoISC070Z5pfhDVKOP3LJ2A',
        {
            email: email,
            password: password
        }).pipe(catchError(this.handleEror), tap(resData => {
```

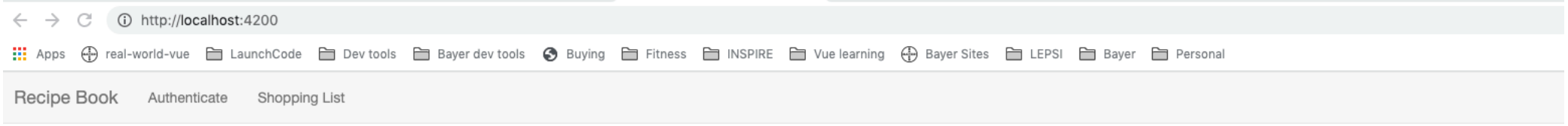
ADDING AUTH GUARD

```
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 isAuth = !!user;
            if(isAuth) return true;
            return this.router.createUrlTree(['/auth']);
        }))
        // tap(isAuth => {
        //     if(!isAuth) {
        //         this.router.navigate(['/auth']);
        //     }
        // }
        // ));
    }

}
```

DYNAMIC COMPONENTS

Components created runtime conditionally.

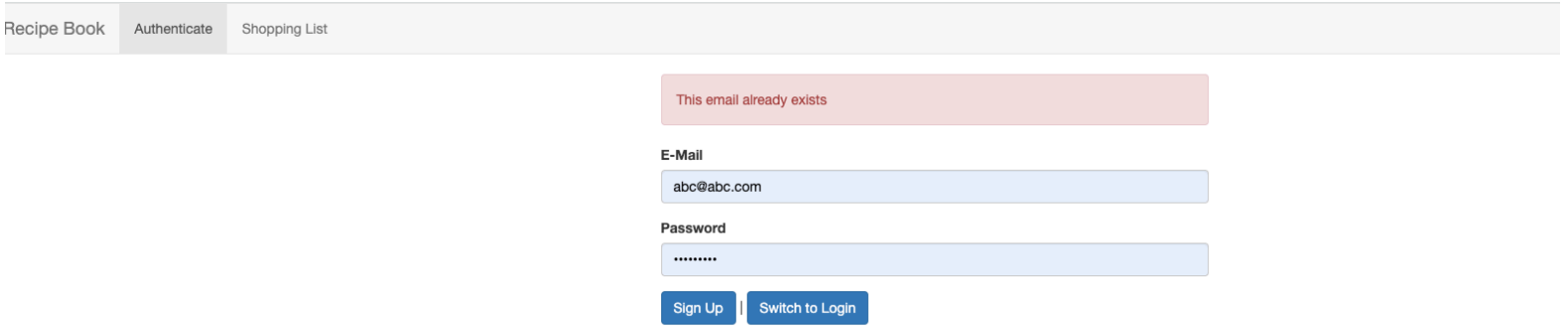
For example, on Error.

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



ADDING AN ALERT MODAL COMPONENT

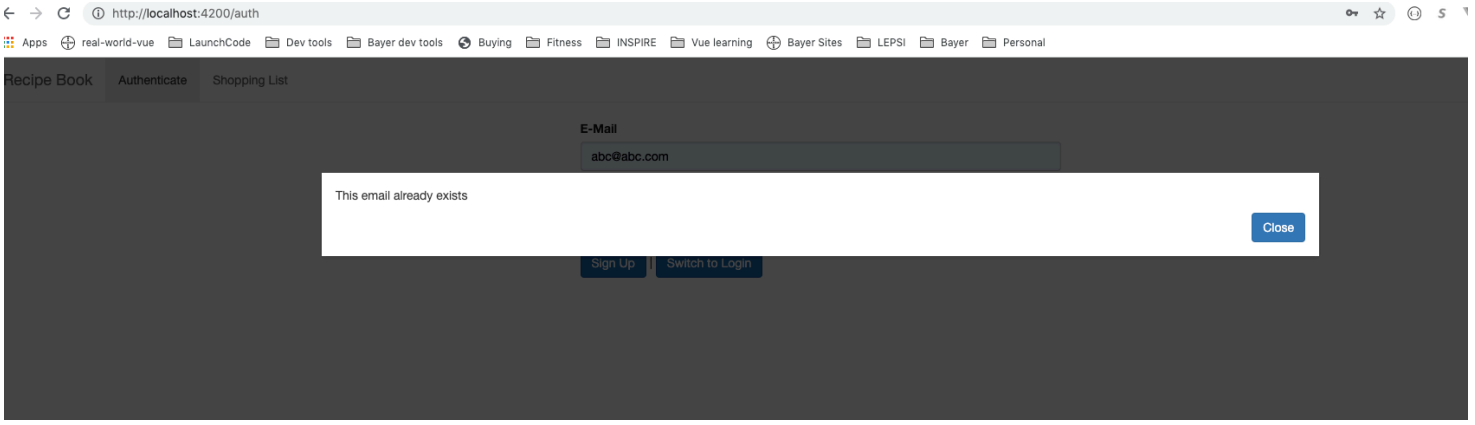
```
import { Component, Input } from "@angular/core";

@Component({
    selector:'app-alert',
    templateUrl: './alert.compononent.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

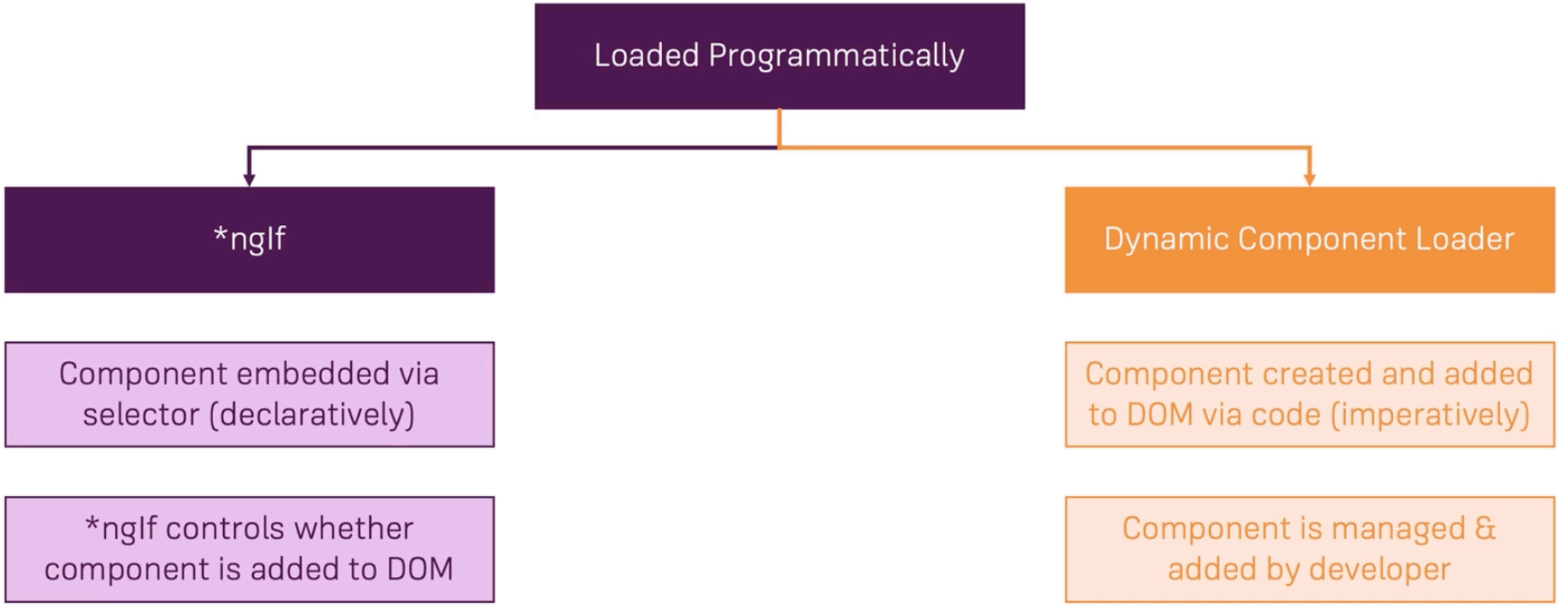


ADDING AN ALERT MODAL COMPONENT

```
.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();
    }

}
```

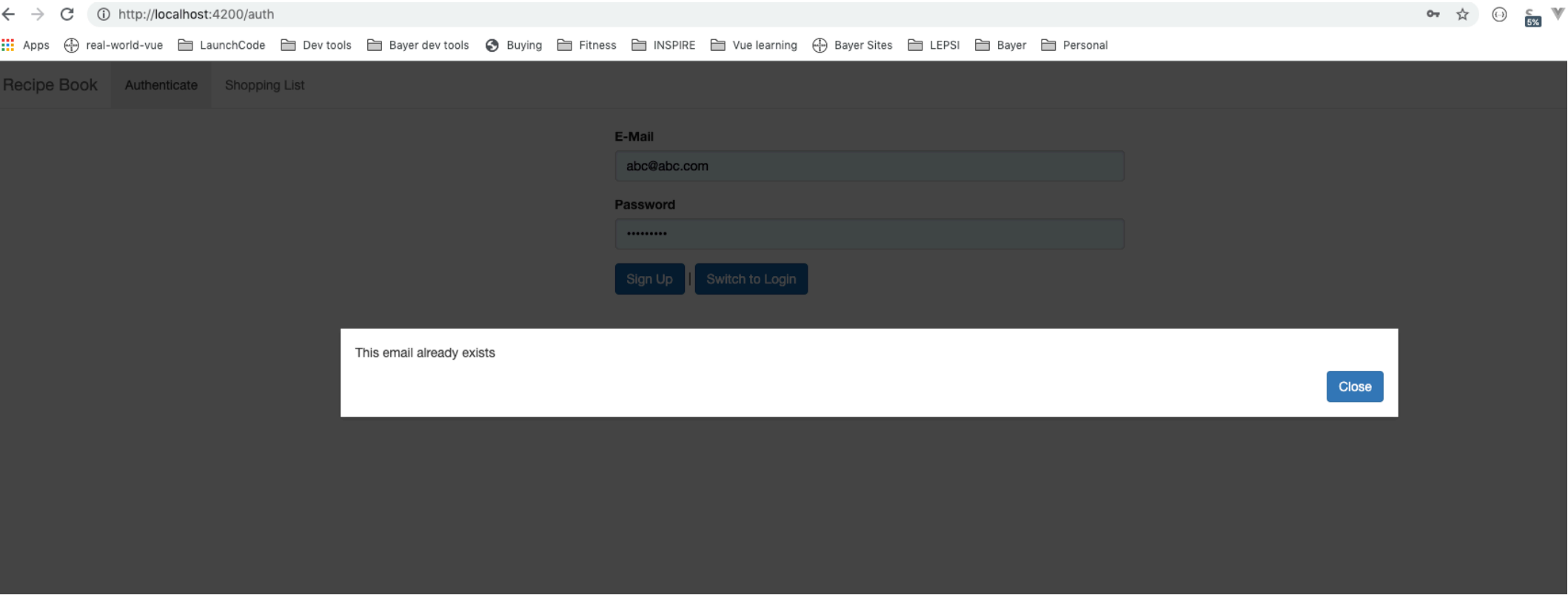
UNDERSTANDING THE DIFFERENT APPROACHES

```
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);
```

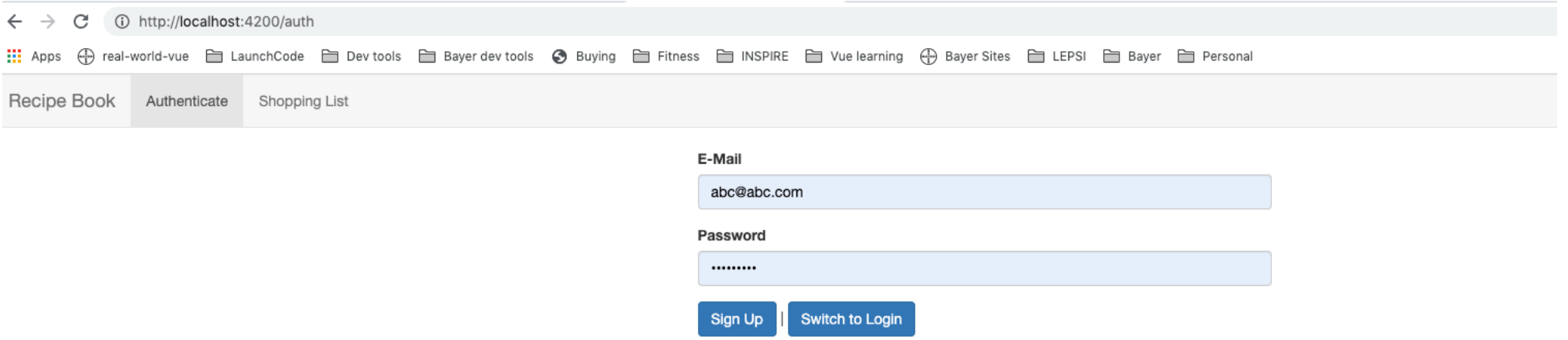
UNDERSTANDING THE DIFFERENT APPROACHES

```
<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>
```

UNDERSTANDING THE DIFFERENT APPROACHES



UNDERSTANDING THE DIFFERENT APPROACHES



PROGRAMMATIC CREATION

```
import { Directive, ViewContainerRef } from "@angular/core";

@Directive({
    selector: '[appPlaceholder'
})
export class PlaceholderDirective {
    constructor(public viewContainerRef: ViewContainerRef) {

    }

}
```

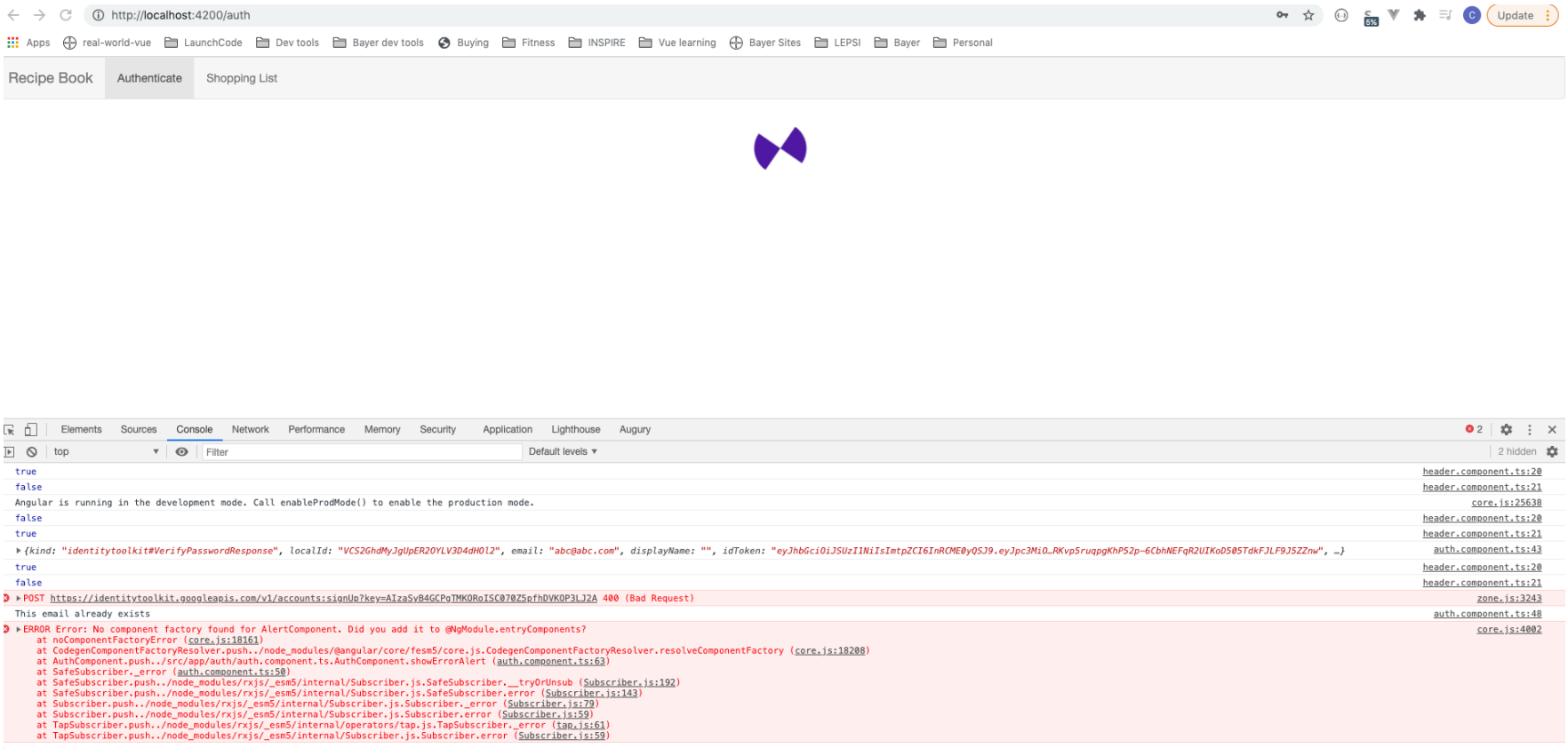
PROGRAMMATIC CREATION

```
<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>
```

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 {
  isLoginMode = true;
  isLoading = false;
  error: string = null;
  @ViewChild(PlaceholderDirective, null) alertHost: PlaceholderDirective;

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

  }

  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 => {
```

PROGRAMMATIC CREATION



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-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,
    AlertComponent,
    PlaceholderDirective
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
```

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 {
  isLoginMode = 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.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 => {
```

QUESTIONS?

STUDIO

Course Project – Unit Testing