**How to create a toast**

Main-nav.component.html we create a click event on About

          <li class="nav-item active">

            <a class="nav-link" (click)="showAbout()">About</a>

          </li>

On this code we navigated to main-nav.component.ts

1. We import the ToastService
2. Below the constructor after ngOnInit we create our showAbout

showAbout() {

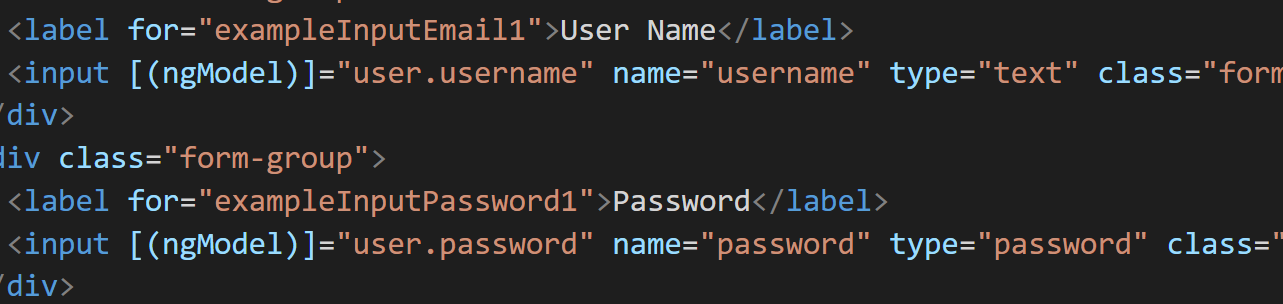
this.toastService.showToast('success', 5000, 'This application was created by Oscar Vasquez.');

**How to create login button credential capture**

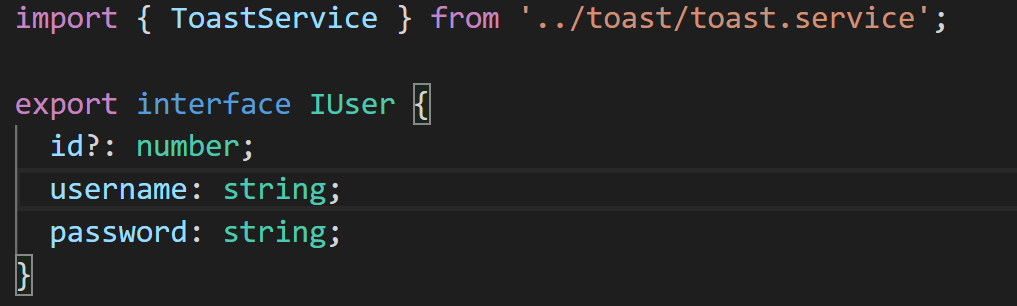
1. Locate login button and create a click event “login(creds)” creds is a user input.

<button mat-raised-button color="primary" (click)="login(user)">Login</button>

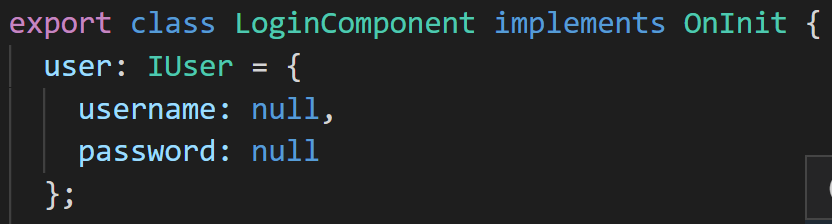
1. Next, we will go to login.component.html and find the input for username and password and capture the users input and pass it to the login as creds.
2. Create the banana in a box syntax [( )] and inside we will use ngModel and use the user component variable.



1. Next, we will create an interface



1. Then we will create the user input and add the required items from the interface



1. Then we will create the login function

  login(user: IUser) {

    console.log('from login user: ', user);

        // set the user credentials

    const presetUser = { username: 'ovasquez', password: 'selma123' };

        // log the user in

    if (user.username != null && user.password != null &&

      user.username !== '' && user.password !== '') {

      console.log('from within if statement...');

        // actually log them in

      if (user.username === presetUser.username &&

        user.password === presetUser.password) {

        // saving data to localStorage

        localStorage.setItem('user', JSON.stringify(user));

        // navigate to contacts page

        this.router.navigate(['contacts', user]);

      } else {

        // toast warraning if username or password was incorrect

        this.toastService.showToast('warning', 2000, 'Username or password incorrect!');

      }

    } else {

      console.log('Must specify credentials');

      // toast warraning if missing either username or password

      this.toastService.showToast('danger', 2000, 'Must specify credentials');

    }

  }

}

**Contact component load from a Jason files**

1. Make sure to look for \*ngFor=”let contact of contacts; let I = index”

<div class="contact-grid">

  <div class="gallery-grid">

    <mat-card class="example-card" \*ngFor="let contact of contacts; let i = index">

      <mat-card-content>

1. then we will need to load the contacts from contacts.json files. This will add the 3 contacts in the contact.json

  async ngOnInit() {

    this.contacts = await this.loadContactsFromJson();

    console.log('this.contacts from ngOninit...', this.contacts);

  }

  async loadContactsFromJson() {

    const contacts = await this.http.get('assets/contacts.json').toPromise();

    return contacts.json();

  }

1. Then we will create the banana in a box syntax [(ngModel )] =”contact.id” for all variables in the contact.json file.

<label for="id">ID:</label>

<input [(ngModel)]="contact.id" type="text" class="form-control" placeholder="ID" aria-label="id" aria-describedby="basic-addon1">

1. Then we will go to the contacts.json file and add the owed field to the 3 contact

  {

    "id": 3,

    "firstName": "Joe",

    "lastName": "Biden",

    "email": "jbiden@yahoo.com",

    "phone": "(559) 111-2121",

    "editing": false,

    "owed":1000

  }

Stopped video at 40:48

**How to add a new contact**

1. In the contacts.component.ts we will create a interface called IContacts

export interface Icontact {

  id?: number;

  firstName: string;

  lastName: string;

  email: string;

  owed: number;

  phone: string;

}

1. Then we will create the addcontact function.

  addContact() {

    const contact: Icontact = {

      id: null,

      firstName: null,

      lastName: null,

      email: null,

      owed: null,

      phone: null

    };

    this.contacts.unshift(contact);

  }

1. In the contacts array we will change it from any to IContacts

 contacts: Array<Icontact> = [];

  constructor(

    private http: Http,

    private activatedRoute: ActivatedRoute,

    private router: Router,

    private toastService: ToastService

1. Then we add a click event to the add contact button

<button mat-raised-button color="primary" (click)="addContact()">Add Contact</button>

1. Then we will allow add contacts function to save to local storage.

    localStorage.setItem('contacts', JSON.stringify(this.contacts));

1. Then we will go and modify the ngOnInit to be able to save the new contact to local storage.

  async ngOnInit() {

    const contacts = JSON.parse(localStorage.getItem('contacts'));

    if (contacts && contacts.length > 0) {

      this.contacts = contacts;

    } else {

      this.contacts = await this.loadContactsFromJson();

    }

1. Then we will create a saveToLocalStorage function and replace the the existing local storage code to this.saveToLocalStorage();

    this.contacts.unshift(contact);

    this.saveToLocalStorage();

  }

  deleteContact(index: number) {

    this.contacts.splice(index, 1);

    this.saveToLocalStorage();

  }

  saveToLocalStorage() {

    localStorage.setItem('contacts', JSON.stringify(this.contacts));

  }

**How to wire the delete button**

1. Create the delete function and also copyed the local storage code and added it.

  deleteContact(index: number) {

    this.contacts.splice(index, 1);

    localStorage.setItem('contacts', JSON.stringify(this.contacts));

<button mat-raised-button color="warn" (click)="deleteContact(i)"

**How to wire the load contact button**

  async ngOnInit() {

    this.contacts = await this.loadContacts();

  }

  async loadContacts() {

    let contacts = JSON.parse(localStorage.getItem('contacts'));

    if (contacts && contacts.length > 0) {

      // this.contacts = contacts;

    } else {

      contacts = await this.loadContactsFromJson();

    }

    console.log('this.contacts from ngOninit...', this.contacts);

    this.contacts = contacts;

    return contacts;

  }

**How to make the finalize and calculation function**

  finalize() {

    this.calculate();

    const data = this.calculate();

    // this is code to do route to home page

    this.router.navigate(['home', data]);

  }

  calculate() {

    let owed = 0;

    for (let i = 0; i < this.contacts.length; i++) {

      // console.log('i--->', i, "this.contacts[i]", this.contacts[i]);

      owed += this.contacts[i].owed;

      // console.log('owed---->', owed);

    }

    // the math to calculate the total with tax

    return {

      numberOfContacts: this.contacts.length,

      subTotal: owed,

      taxAmount: owed \* .10,

      total: owed + (owed \* .10)

    };

**How to pass the data to home page and interpolate the data to the tables.**

  ngOnInit() {

    this.activatedRoute.params.subscribe((sss) => {

      this.data = sss;

      console.log('this.data from home........', this.data);

    });

<td scope="row">{{data.numberOfContacts}}</td>

        <td>{{data.subTotal}}</td>

        <td>{{data.taxAmount}}</td>

        <td>{{data.total}}</td>

**Have the logout button clear local storage and logout.**

1. Create a click event for the logout button

 <button mat-raised-button color="accent" (click)="logout()">Logout</button>

1. In main-nav.components.ts we will import the router

import { Router } from '@angular/router';

1. Then add the router code into the constructor

constructor(private toastService: ToastService, private router: Router) { }

1. Then we create the logout function

  logout() {

    localStorage.setItem('user', JSON.stringify([]));

    this.router.navigate(['login']);

  }

**Setup the search input**

1. First make and ngModel on the search input

<input [(ngModel)]="params" class="form-control mr-sm-2" type="search"

1. Create the search function

  search(params: string) {

    console.log('from serch.... params', params);

    this.contacts = this.contacts.filter((contact: Icontact) => {

      return contact.firstName.toLocaleLowerCase() === params.toLocaleLowerCase();

    });

  }

}