Author: Kristóf, Kukk

Date: January 27th, 2025

Assignment: Web Technologies I.

Cookie Dubs Website

TARTALOM

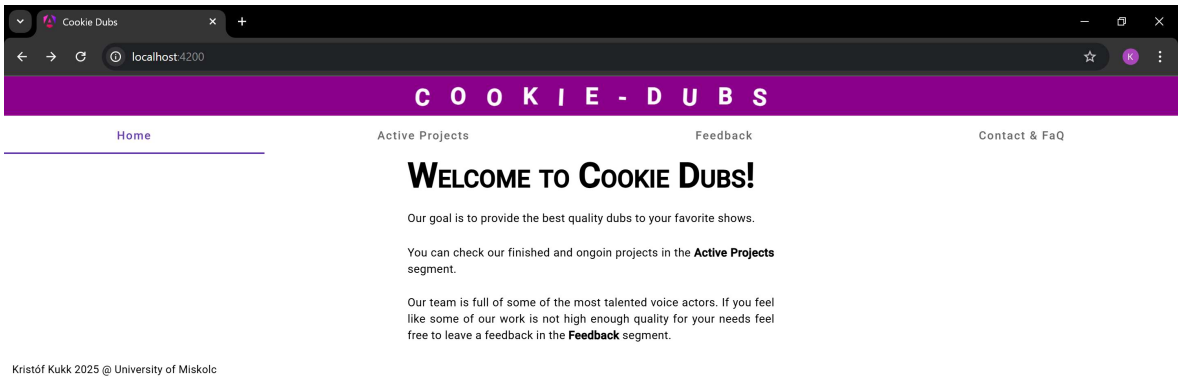
1. THE STORY OF THE WEBPAGE

My friends have been involved with creating unofficial, hobby voiceovers (in other names dubs), and I was wondering if it would be hard to do such a site for them if they decided to make it official. Currently they only hosted their works on Discord, a popular messaging site amongst my peers. This is the story baseline.

2. THE SELECTION OF FRAMEWORK

At first, I was keen on doing the whole page in plain html/css as these were the only web development resources I worked with in the past. But not long after starting I realized it will not be enough for some features I want to add to this project. So I remembered a conversation I had with the professor associated with the function, where she told me and another student about AngularJS – the tool I used on the site.

It is important to use Angular 19 when running the application! And the testing screen resolution was 2880x1800px, windows 11 zoom 200.



Home page of Cookie Dubs

3. SKELETON OF THE SITE

The website is broken up into 4 sections, **Home**, **Active Projects**, **Feedback** and **Contact & FaQ**.

3.1. HOME

The home page just contains plain text, telling the viewer about the meaning of the website.

3.2. ACTIVE PROJECTS

This is where the projects would be shown to the viewer. On the top right of the segment is a so-called pager, which allows the user to select how many projects to see at once. Currently there are twelve generic titles added, which are loaded from a typescript class, containing instances of an interface called Project. This data loaded on two different instances, one is when the pager data changes meaning, the page was changed or the items shown per page are changed, the other is when the active project segment gets loaded.

```
interface Project {
  name: string;
  finished: boolean;
  associated: string[];
  imageurl: string;
}
```

```
preload()
{
  if(document.getElementById('project-cluster') == null)
  {
    setTimeout(() => {
      this.preload();
    }, 25);
  }
  else
  {
    let event = new PageEvent();
    event.length = this.Size;
    event.pageIndex = 0;
    event.pageSize = 10;
    this.onPageEvent(event);
  }
}
```

The detection is in the form of a continuously running asynchronous check, to see if the container in which the projects are going to be in is visible or not.

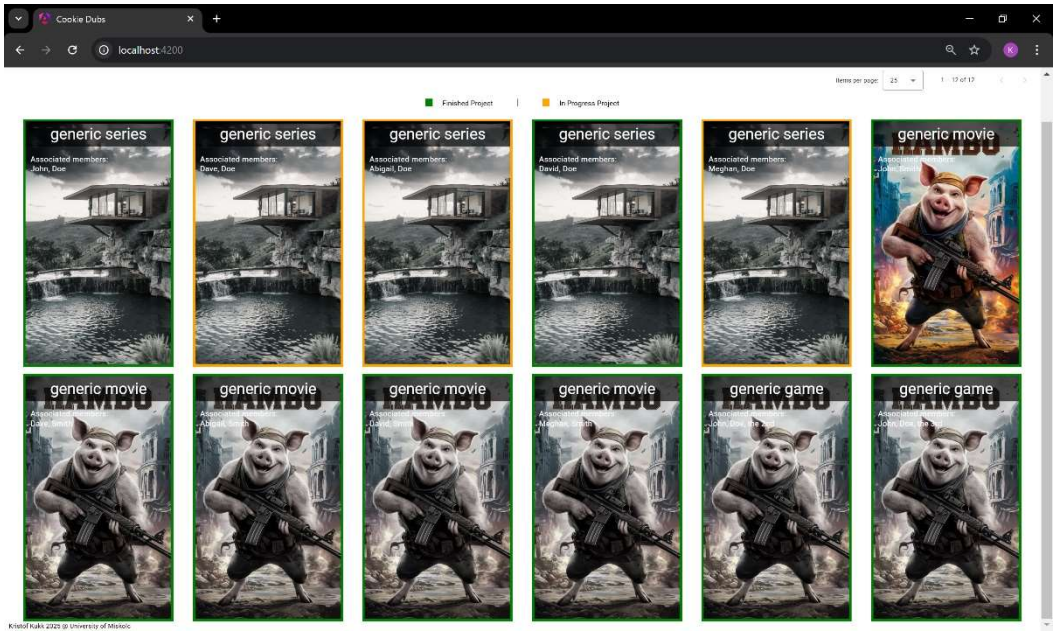


Image of the Active Projects tab, when the viewer selects to see all twelve projects, zoomed out to fit on the screen.

3.3. FEEDBACK

The feedback component has an angular element as its base, the **Stepper** component. It consists of four pages, **Basic Information**, **More Information**, **Optional Information**, **Finishing Steps**. All of them except the last one have a separate form inside of them, that is because of how Angular manages validation inside of input fields and inside the stepper.

```
export class FeedbackComponent {

  private _formBuilder = inject(FormBuilder);

  readonly firstName = new FormControl('', [ Validators.required, Validators.minLength(3) ]);
  readonly lastName = new FormControl('', [ Validators.required, Validators.minLength(3) ]);
  readonly email = new FormControl('', [ Validators.required, Validators.email ]);
  readonly gender = new FormControl('', [ Validators.required ])

  readonly message = new FormControl('', [ Validators.required, Validators.minLength(10) ]);
  readonly dateofbirth = new FormControl('', [ Validators.required ]);

  readonly favcolor = new FormControl('');
  readonly notify = new FormControl('');

  basicFormGroup = this._formBuilder.group({
    firstname: this.firstName,
    lastname: this.lastName,
    email: this.email,
    gender: this.gender
  });
  moreFormGroup = this._formBuilder.group({
    message: this.message,
    dateofbirth: this.dateofbirth
  });
  optionalFormGroup = this._formBuilder.group({
    favcolor: this.favcolor,
    notify: this.notify,
  });
```

Validation is done the following way: each step containing a form has a step control attribute set, these represent ‘form control’ groups, which are located inside the class of the component. Form Controls have so called Validators inside of them, this is where the validation part happens. It is done automatically by Angular, when set for the correct input.

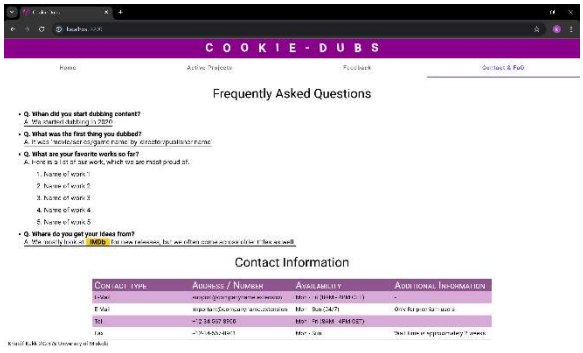
```
<mat-form-field appearance="outline">
    <mat-label>Last Name</mat-label>
    <input matInput required [formControl]="lastName"
    @if (lastName.invalid) {
        <mat-error>{{getErrorMessage('lastname')}}</ma
    }
</mat-form-field>
<mat-form-field appearance="outline">
    <mat-label>E-mail Address</mat-label>
    <input matInput required [formControl]="email" nam
    @if (email.invalid) {
        <mat-error>{{getErrorMessage('email')}}</mat-e
    }
</mat-form-field>
<label id="example-radio-group-label">Gender *:</label
<mat-radio-group [formControl]="gender">
    <mat-radio-button value="male" name="gender">Male<
    <mat-radio-button value="female" name="gender">Fem
    @if (gender.invalid) {
        <mat-error>{{getErrorMessage('gender')}}</mat-
    }
</mat-radio-group>
<button mat-raised-button matStepperNext>Next</button>
```

```
64    getErrorMessage(controlName : string)
65    {
66      let control = this.basicFormGroup.get(controlName);
67      if(control == null) control = this.moreFormGroup.ge
68      if(control == null) return '';
69      if (control.hasError('required')) {
70        return 'This field must be filled!';
71      } else if (control.hasError('email')) {
72        return 'Not a valid email address!';
73      }
74      else if (control.hasError('minlength'))
75      {
76        return 'Value is not long enough!'
77      } else {
78        return '';
79      }
80    }
81
82    hasInvalidMembers()
83    {
84      return this.basicFormGroup.invalid || this.moreForm
85    }
86  }
```

In the typescript, we can get an error message according to these validators, and in the template html, we can check with if-s if the control is invalid.

3.4. CONTACT

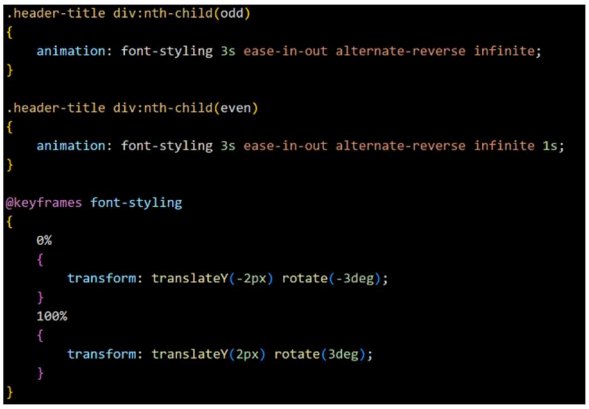
In the contact component there is a table for contact information and a list of possible questions with possible answers. There is also an example link shown on this page.



4. MISCELLANEOUS INFORMATION

4.1. ANIMATION

For the “title” of the site, the Cookie Dubs text is animated so it looks like the letters are flying. It is done with using a separate div for each letter, and for a parent div, there is a given class, so the **nth-child(odd)** and **nth-child(even)** are set an animation which rotates and moves each letter. For ‘even’ children this animation is offset by 1 second.



```
.header-title div:nth-child(odd)
{
    animation: font-styling 3s ease-in-out alternate-reverse infinite;
}

.header-title div:nth-child(even)
{
    animation: font-styling 3s ease-in-out alternate-reverse infinite 1s;
}

@keyframes font-styling
{
    0%
    {
        transform: translateY(-2px) rotate(-3deg);
    }
    100%
    {
        transform: translateY(2px) rotate(3deg);
    }
}
```

4.2. RESOURCES USED

AngularJS: https://angular.dev/

Angular Materials: https://material.angular.io/

IMDb: https://www.imdb.com/

Thank You for reading!