Data Entry and Validation

Goal

In this lab, you will:

- Create a reactive data entry form
- Add validation to make sure there is a movie title
- Add a custom validator to make sure the movie release date has a reasonable value

Your mission

In this lab, you will create an Angular reactive data entry form to allow the end user to make changes to a movie. You will add validation to make sure the movie title meets a minimum and maximum length standard. Finally, you will create a custom validator to make sure that the movie release date has a reasonable value.

Type it out by hand?

> Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now.

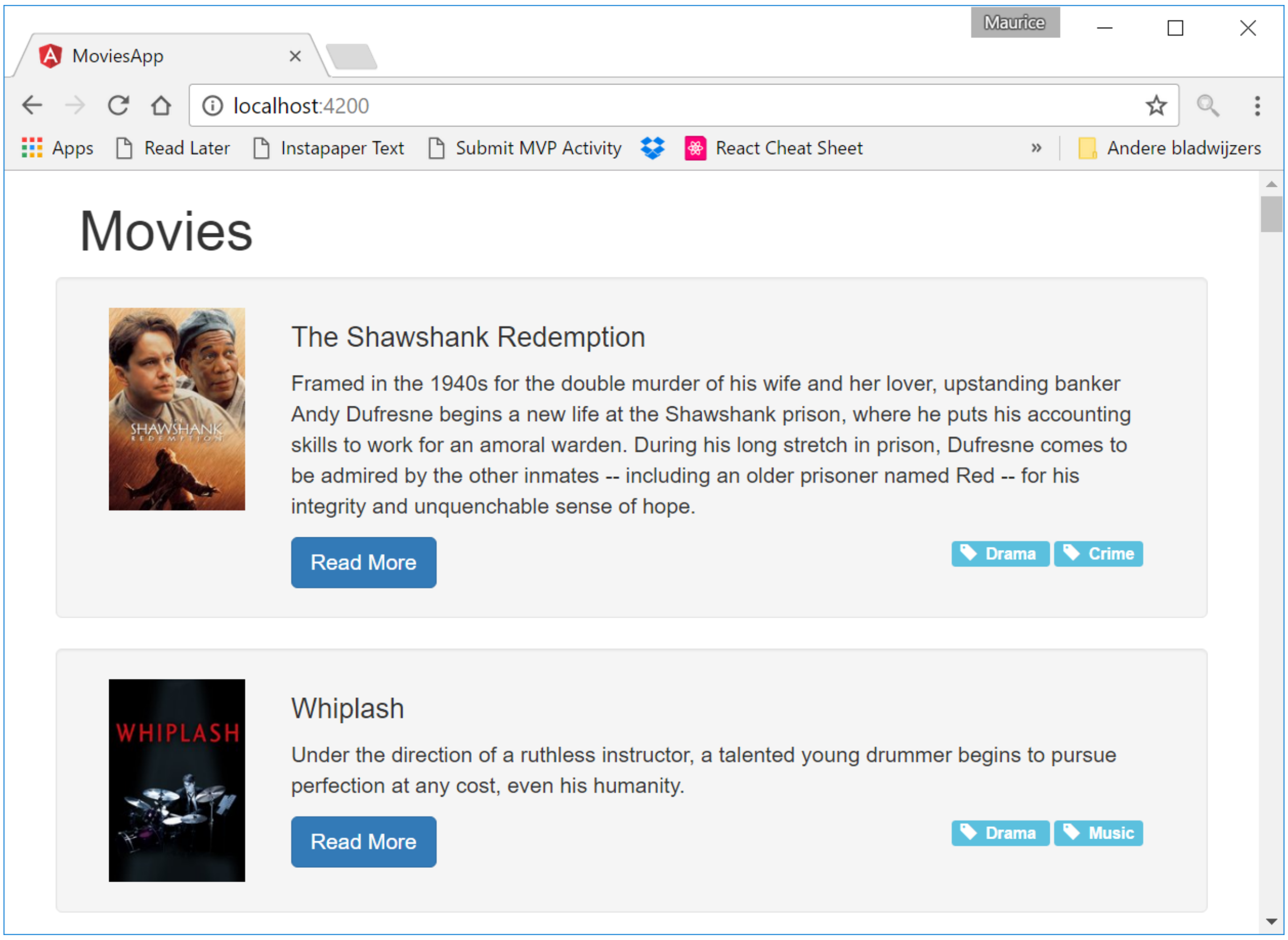
Load the list of movies using an Ajax request

In this section, you will convert the template based editor to be a reactive form.

1. Open the **begin/movies-app** folder of the lab in the terminal window and start the application using the following command:

> Note: Make sure not to use **ng serve** to start the application because you will also use a movies API server in this lab

```
npm start
```



Movies

The Shawshank Redemption

Framed in the 1940s for the double murder of his wife and her lover, upstanding banker Andy Dufresne begins a new life at the Shawshank prison, where he puts his accounting skills to work for an amoral warden. During his long stretch in prison, Dufresne comes to be admired by the other inmates -- including an older prisoner named Red -- for his integrity and unquenchable sense of hope.

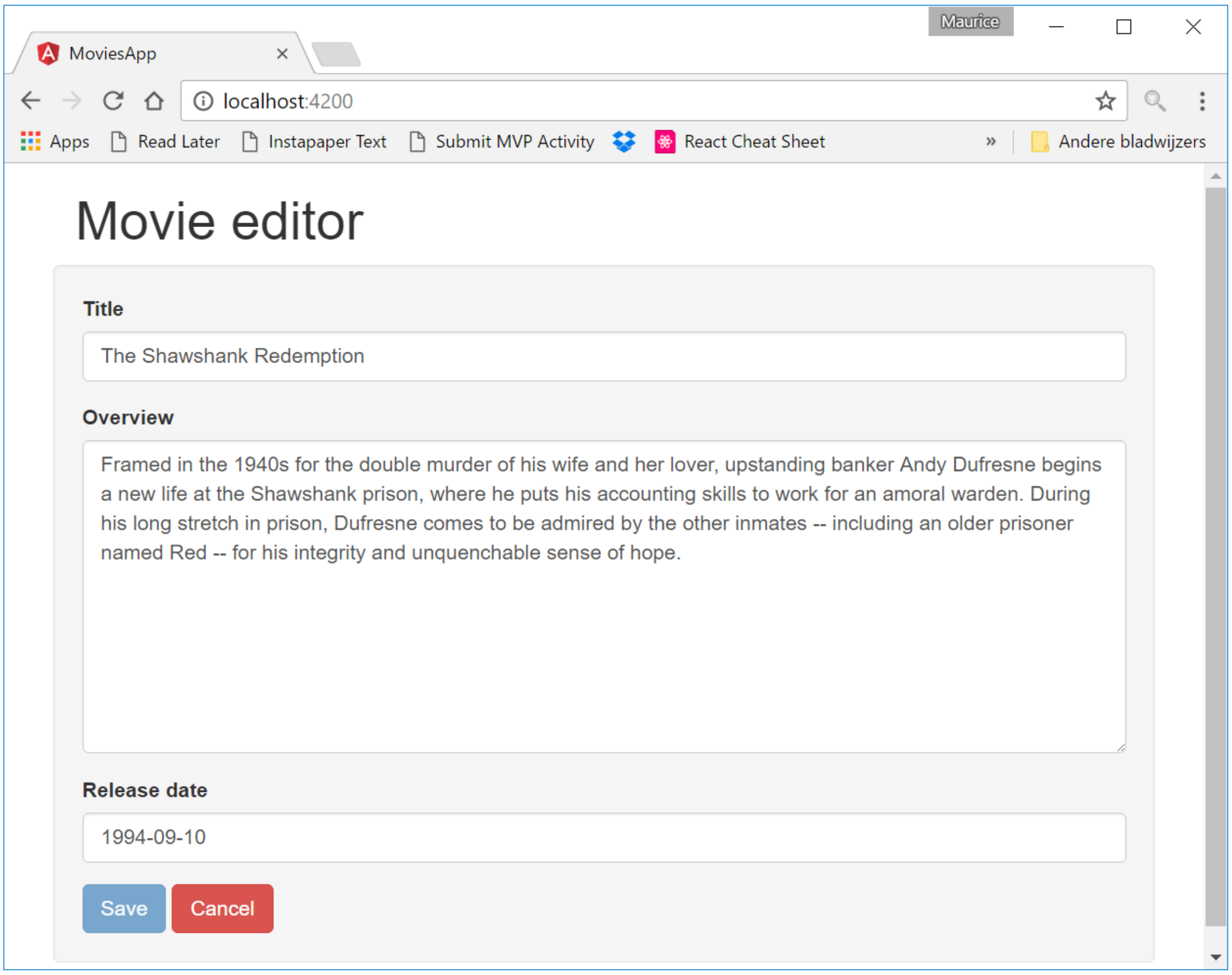
Read More | Drama | Crime

Whiplash

Under the direction of a ruthless instructor, a talented young drummer begins to pursue perfection at any cost, even his humanity.

Read More | Drama | Music

2. Open the application in the browser at http://localhost:4200/. You should just see a list of movies. Each movie should have an edit page allowing you to make changes to the movie and save them.



Movie editor

Title

The Shawshank Redemption

Overview

Framed in the 1940s for the double murder of his wife and her lover, upstanding banker Andy Dufresne begins a new life at the Shawshank prison, where he puts his accounting skills to work for an amoral warden. During his long stretch in prison, Dufresne comes to be admired by the other inmates -- including an older prisoner named Red -- for his integrity and unquenchable sense of hope.

Release date

1994-09-10

Save Cancel

3. Open **app.module.ts** and replace **FormsModule** with **ReactiveFormsModule**.

```
@NgModule({
  declarations: [
    AppComponent,
    MovieEditorComponent,
    MovieListItemComponent,
    LoadingIndicatorComponent
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    HttpModule
  ],
  providers: [MoviesService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

4. Open **movie-editor.component.html** and replace the **ngModel** directive with a **formControlName** with the field name. Remove the **id** and **name** properties as they are no longer needed. Do this for all three input elements in the template.

```
<div class="form-group">
  <label for="title">Title</label>
  <input
    type="text"
    formControlName="title"
    class="form-control"
    placeholder="Title">
</div>
```

5. Remove the** #movieForm** template variable and replace it with **FormGroup** named **movieForm** to `<form>` tag

```
<form [formGroup]="movieForm" (ngSubmit)="onSubmit()">
```

6. Open **movie-editor.component.ts** and inject a **FormBuilder** service into the constructor and use it to build a **FormGroup** property named **movieForm**. Update this **FormGroup** with the move data when the AJAX request completes.

```
export class MovieEditorComponent implements OnInit {
  movie: Movie = null;
  movieForm: FormGroup;

  @Input() movieId: number;
  @Output() showList = new EventEmitter();

  constructor(private moviesService: MoviesService, private fo
rmBuilder: FormBuilder) {
  }

  ngOnInit() {
    this.movieForm = this.formBuilder.group({
      title: [],
      overview: [],
      release_date: []
    });
    this.moviesService.getMovie(this.movieId)
      .subscribe(movie => {
        this.movie = movie;
        this.movieForm.reset(movie);
    });
  }

  // Remainder of the component
```

7. Open **movie-editor.component.html** and disable the **Save** button when the form is invalid or has no changes.

```
<button type="submit" class="btn btn-primary"
        [disabled]="movieForm.pristine || movieForm.invalid">
  Save
</button>
```

8. Open **movie-editor.component.ts** and update the **onSubmit()** function in the component and combine the loaded movie with the **movieForm.value** to get a complete and updated movie. Save this movie object using the **moviesService**.

```
onSubmit() {
  const updatedMovie = {...this.movie, ...this.movieForm.valu
e};
  this.moviesService.updateMovie(updatedMovie)
    .subscribe(() => this.showList.emit());
}
```

9. Test the application to see if everything is working as expected. You should now be able to edit and save movies.

Add validation to the movie editor

In this section, you will add validation and error messages to the movie editor.

1. Open **movie-editor.component.ts** and add an **error** property to the component.

```
export class MovieEditorComponent implements OnInit {
  movie: Movie = null;
  movieForm: FormGroup;
  errors = {
    title: null,
    release_date: null
  }
```

2. Open **movie-editor.component.html** and add markup to display the error if there is one. Do this for both the **title** and **release_date** inputs.

```
<div [hidden]="!errors.title"
      class="alert alert-danger">
  {{errors.title}}
</div>
```

3. Open **movie-editor.component.ts** and import **Validators** and add the **required** validator to the **title** and **release_date** fields. Add the **maxLength** validator to the title to limit it to 100 characters. Finally, add the **pattern** validator to the **release_date** to validate that the date has a pattern of YYYY-MM-DD.

```
ngOnInit() {
  this.movieForm = this.formBuilder.group({
    title: [null, [Validators.required, Validators.maxLength(1
00)]],
    overview: [],
    release_date: [null, [Validators.required, Validators.patt
ern(/^\d\d\d\d-\d\d-\d\d$/)]]
  });

  this.moviesService.getMovie(this.movieId)
    .subscribe(movie => {
      this.movie = movie;
      this.movieForm.reset(movie);
  });
}
```

4. Create a function **validateMovie()** in the component and call this whenever a value change is detected on the **movieForm**. Also call this **validateMovie()** function after loading a movie using the **moviesService**.

```
ngOnInit() {
  this.movieForm = this.formBuilder.group({
    title: [null, [Validators.required, Validators.maxLength(1
00)]],
    overview: [],
    release_date: [null, [Validators.required, Validators.patt
ern(/^\d\d\d\d-\d\d-\d\d$/)]]
  });
  this.movieForm.valueChanges.subscribe(() => this.validateMov
ie())

  this.moviesService.getMovie(this.movieId)
    .subscribe(movie => {
      this.movie = movie;
      this.movieForm.reset(movie);
      this.validateMovie()
  });
}

validateMovie() {
}
```

5. Implement the **validateMovie()** function and set error messages as appropriate.

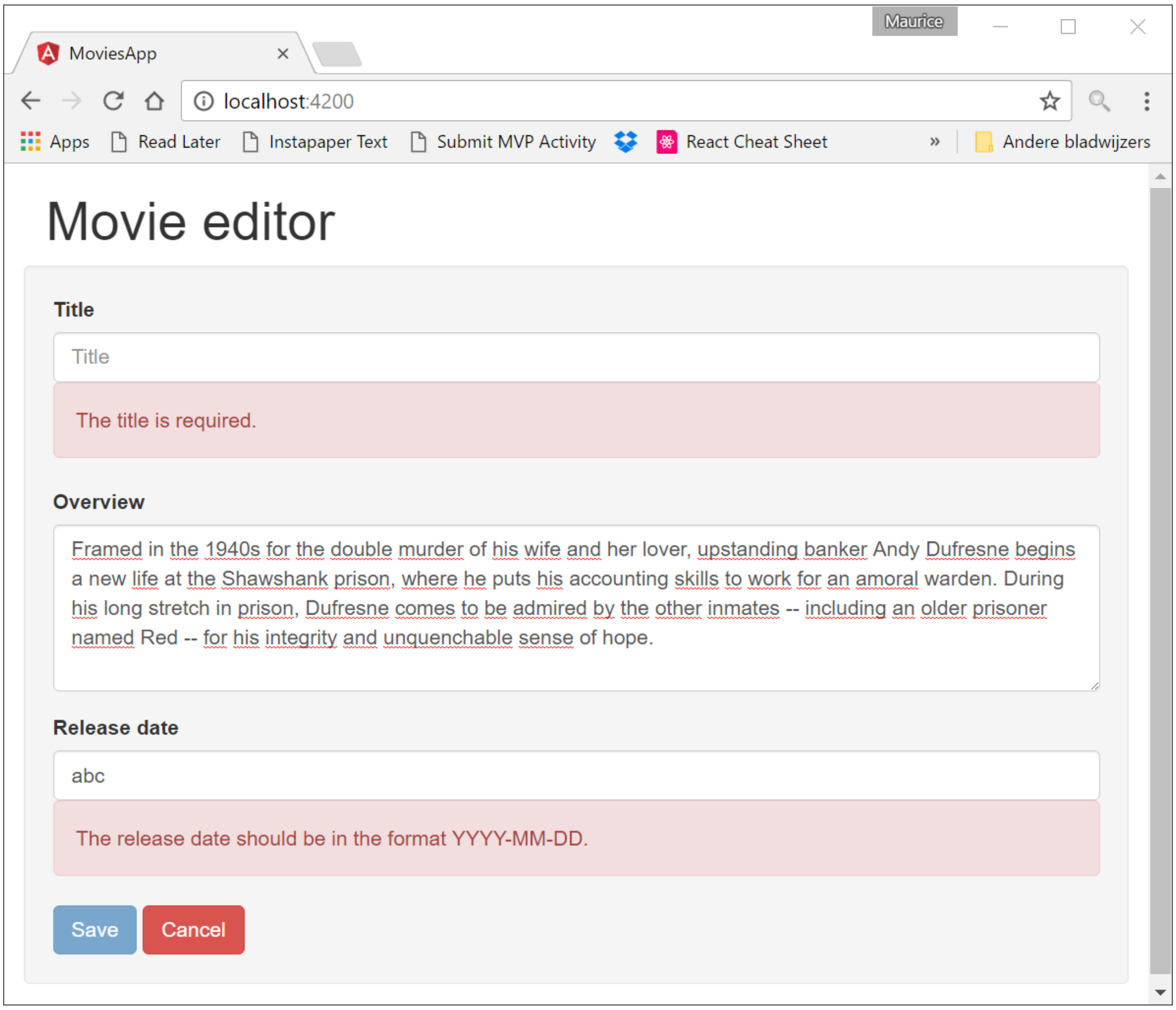
```
validateMovie() {
  this.errors.title = null;
  this.errors.release_date = null;

  const titleErrors = this.movieForm.controls['title'].errors;
  if (titleErrors) {
    if (titleErrors.required) {
      this.errors.title = 'The title is required.'
    } else if (titleErrors.maxlength) {
      this.errors.release_date = `The maximum title length is
${titleErrors.maxlength.requiredLength}`
    }
  }

  const releaseErrors =
this.movieForm.controls['release_date'].errors;
  if (releaseErrors) {
    if (releaseErrors.required) {
      this.errors.release_date = 'The release date is require
d.'
```

```
    } else if (releaseErrors.pattern) {
      this.errors.release_date = `The release date should be i
n the format YYYY-MM-DD`
    }
  }
}
```

6. Test the application to see if the validation is working as expected. With an invalid input you should see the relevant error message and the **Save** button should be disabled.



Movie editor

Title

Title

The title is required.

Overview

Framed in the 1940s for the double murder of his wife and her lover, upstanding banker Andy Dufresne begins a new life at the Shawshank prison, where he puts his accounting skills to work for an amoral warden. During his long stretch in prison, Dufresne comes to be admired by the other inmates -- including an older prisoner named Red -- for his integrity and unquenchable sense of hope.

Release date

abc

The release date should be in the format YYYY-MM-DD.

Create a custom validator for the release date

In this section, you will add a custom validator to make sure that the release date is actually a valid and reasonable date.

1. Add a new file **release-date.validator.ts** to the **movie-editor** folder.
2. Create and export a new validator function named **releaseDateValidator()**.

```
import { ValidatorFn, AbstractControl } from '@angular/forms';

export const releaseDateValidator: ValidatorFn = (control: Abs
tractControl) => {
    return null;
}
```

3. Add logic to parse the input controls date string and check if the date is valid, after 1891, when the movie camera was invented and not in the future.

```
export const releaseDateValidator: ValidatorFn = (control: Abs
tractControl) => {
    const dateNr = Date.parse(control.value);
    if (!dateNr) {
        return {
            releaseDate: {
                parseError: true
            }
        }
    }
    const date = new Date(dateNr);
    if (date.getFullYear() < 1891) {
        return {
            releaseDate: {
                beforeCamera: true
            }
        }
    }
    if (date > new Date()) {
        return {
            releaseDate: {
                future: true
            }
        }
    }

    return null;
}
```

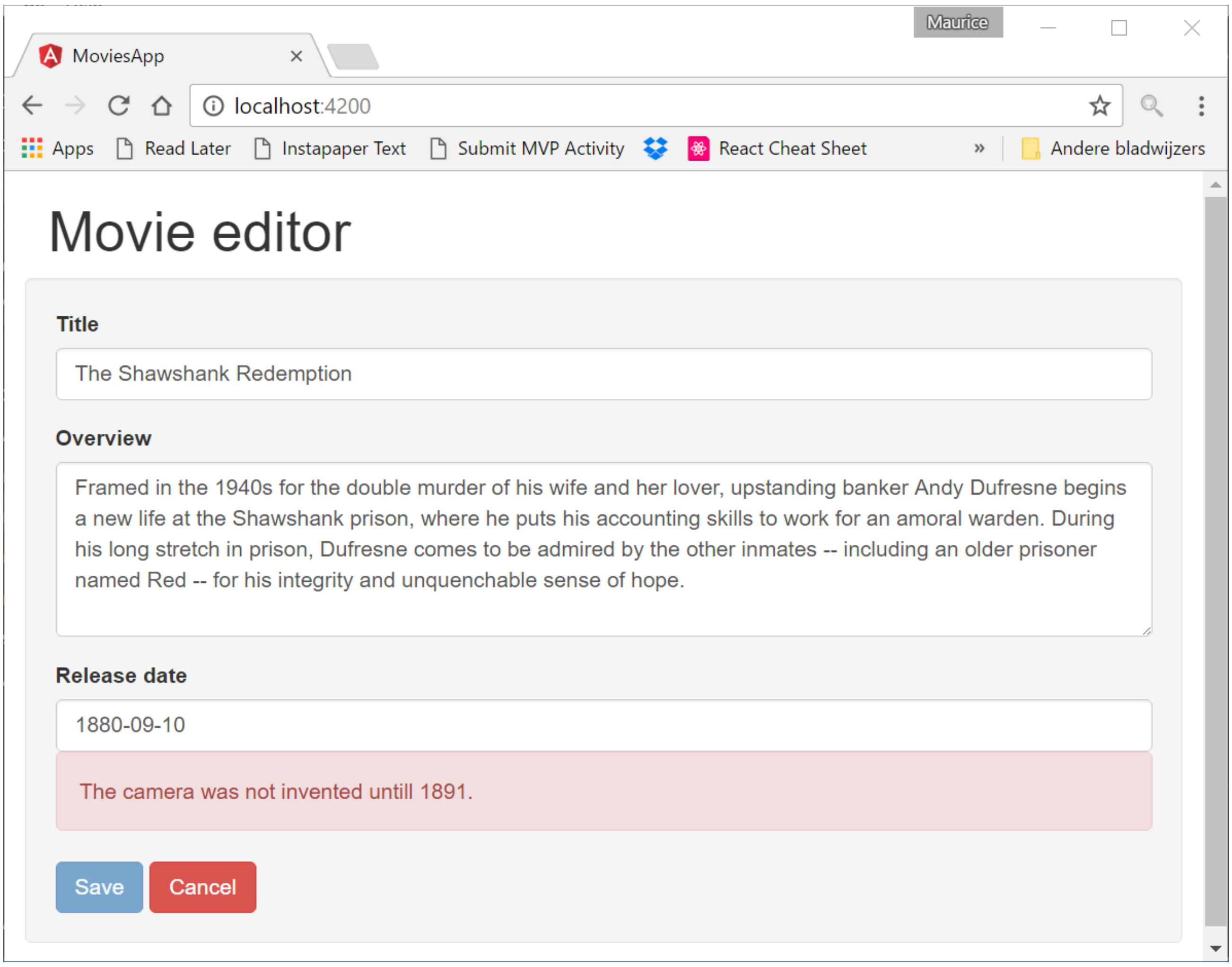
4. Open the **movie-editor.component.ts** and import the **releaseDateValidator**. Add it to the **release_date** validators collection.

```
ngOnInit() {
  this.movieForm = this.formBuilder.group({
    title: [null, [Validators.required, Validators.maxLength(1
00)]],
    overview: [],
    release_date: [null, [Validators.required, Validators.patt
ern(/^\d\d\d\d-\d\d-\d\d$/), releaseDateValidator]]
  });

  // Remaining code
}
```

5. Extend the **validateMovie()** function to include relevant error messages for the errors from the **releaseDateValidator**.

```
validateMovie() {
  // Remaining code
  const releaseErrors =
this.movieForm.controls['release_date'].errors;
  if (releaseErrors) {
    if (releaseErrors.required) {
      this.errors.release_date = 'The release date is require
d.'
    } else if (releaseErrors.pattern) {
      this.errors.release_date = 'The release date should be i
n the format YYYY-MM-DD.'
    } else if (releaseErrors.releaseDate) {
      if (releaseErrors.releaseDate.parseError) {
        this.errors.release_date = 'Invalid date.'
      } else if (releaseErrors.releaseDate.beforeCamera) {
        this.errors.release_date = 'The camera was not invente
d untill 1891.'
      } else if (releaseErrors.releaseDate.future) {
        this.errors.release_date = "The movie can't be release
d in the future."
      }
    }
    }
}
```



Movie editor

Title

The Shawshank Redemption

Overview

Framed in the 1940s for the double murder of his wife and her lover, upstanding banker Andy Dufresne begins a new life at the Shawshank prison, where he puts his accounting skills to work for an amoral warden. During his long stretch in prison, Dufresne comes to be admired by the other inmates -- including an older prisoner named Red -- for his integrity and unquenchable sense of hope.

Release date

1880-09-10

The camera was not invented untill 1891.

6. Test the application to see if the release date validation is working as expected.

Solution

The solution can be found in **complete** folder