# Angular 2+

# Workshop. Routing.

## Contents

## Explanation of Colors

Blue color is a snippet of code that you need to fully use to create a new file.

Black color in combination with green or red, means the snippet of code that was added earlier.

Green color is the snippet of code that needs to be added.

Red color is the snippet of code that needs to be removed.

## Task 01. Basic Setup. <base> Tag.

1. Add tag <base> to the file **src/index.html**

```
<title>Angular Routing</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

## Task 02. Components

1. Create 4 blank components **HomeComponent, AboutComponent, PathNotFoundComponent.** Run the following commands in command line from project root folder[1]:

```
ng g c pages/home --skip-tests true --standalone true
ng g c pages/about --skip-tests true --standalone true
ng g c pages/path-not-found --skip-tests true --standalone true
ng g c pages/abc --skip-tests true --standalone true
```

2. Create file **app/pages/index.ts** and add the following snippet of code to it:

```
export * from './about/about.component';
export * from './home/home.component';
export * from './path-not-found/path-not-found.component';
export * from './abc/abc.component';
```

[1] All commands run from project root folder.

## Task 03. Routes Config

1. Create file **app/app-routing.module.ts**. Add the following snippet of code to it.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule, type UrlSegment, type UrlSegmentGroup, type Route, type
UrlMatchResult } from '@angular/router';

import { AboutComponent, AbcComponent, HomeComponent, PathNotFoundComponent } from './pages';

const routes: Routes = [
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'about',
    component: AboutComponent
  },
  {
    component: AbcComponent,
    matcher: (url: UrlSegment[], group: UrlSegmentGroup, route: Route): UrlMatchResult | null => {
      console.log(url, group, route);
      // один фрагмент, который включает 'abc'
      return url.length === 1 && url[0].path.includes('abc') ? ({consumed: url}) : null;
    }
  },
  {
    path: '',
    redirectTo: '/home',
    pathMatch: 'full'
  },
  {
    // The router will match this route if the URL requested
    // doesn't match any paths for routes defined in our configuration
    path: '**',
    component: PathNotFoundComponent
  }
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

## Task 04. Import Routes

1. Make changes to the **AppModule.** Use the following snippets of code:

```
// 1
import { Router } from '@angular/router';
import { AppRoutingModule } from './app-routing.module';

// 2
imports: [
    BrowserModule,
    FormsModule,
    LayoutModule,
    // MUST BE LAST
    AppRoutingModule
]

// 3
constructor(router: Router) {
    const replacer = (key: string, value: unknown): string =>
      typeof value === 'function' ? value.name : (value as string);

    console.log('Routes: ', JSON.stringify(router.config, replacer, 2));
}
```

## Task 05. <router-outlet>

1. Make changes to the **AppComponent template.** Use the following snippet of HTML:

```
<div class="container">
     <!-- #routerOutlet="outlet" - access to the properties of router-outlet directive -->
     <router-outlet
          #routerOutlet="outlet"
          (activate)='onActivate($event, routerOutlet)'
          (deactivate)='onDeactivate($event, routerOutlet)'>
     </router-outlet>
     <!-- Routed views go here -->
</div>
```

2. Make changes to the **AppComponent.** Use the following snippet of code:

```
// 1
import type { RouterOutlet } from '@angular/router';

// 2
onActivate($event: any, routerOutlet: RouterOutlet): void {
    console.log('Activated Component', $event, routerOutlet);
}

onDeactivate($event: any, routerOutlet: RouterOutlet): void {
    console.log('Deactivated Component', $event, routerOutlet);
}
```

## Task 06. routerLink

1. Make changes to the **AppComponent template.** Use the following snippet of HTML:

```
// 1
<a class="navbar-brand" href="#" routerLink="/home">Task Manager</a>

// 2
<a routerLink="/about" href="#about" >About</a>

Enter link: localhost:4200/abc123
```

## Task 07. routerLinkActive

1. Make changes to **AppComponent template.** Use the following snippet of HTML:

```
<li routerLinkActive="active" (isActiveChange)="onRouterLinkActive($event)">
      <a routerLink="/about">About</a>
</li>
```

2. Make changes to the **AppComponent.** Use the following snippet of code:

```
onRouterLinkActive($event: boolean): void {
    console.log($event);
}
```

## Task 08. Task Feature Module

1. Create modules **TasksModule and TasksRoutingModule.** Run the following command from command line:

```
ng g m tasks --routing true -m app.module
```

2. Make changes to the **TasksModule**. Use the following snippet of code:

```
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
  ],
  imports: [
    CommonModule,
    FormsModule,
    TasksRoutingModule
  ],
  providers: [
  ]
})
export class TasksModule {}
```

3. Create a **model of task**. Run the following command from command line:

```
ng g cl tasks/models/task --type model --skip-tests true
```

4. Replace the content of the class. Use the following snippet of code:

```
export class TaskModel {
  constructor(
    public id: number | null = null,
    public action: string = '',
    public priority: number = 0,
    public estHours: number = 0,
    public actHours?: number,
    public done?: boolean
  ) {
    this.actHours = actHours || 0;
    this.done = done || false;
  }
}
```

5. Create service **TaskArrayService.** Run the  the following command from command line:

```
ng g s tasks/services/task-array --skip-tests true
```

6. Replace the content of service **TaskArrayService**. Use the following snippet of code:

```
import { Injectable } from '@angular/core';
import { TaskModel } from './../models/task.model';

@Injectable({
  providedIn: 'root'
})
export class TaskArrayService {
  private taskList = [
    new TaskModel(1, 'Estimate', 1, 8, 8, true),
```

```
    new TaskModel(2, 'Create', 2, 8, 4, false),
    new TaskModel(3, 'Deploy', 3, 8, 0, false)
  ];

  getTasks(): Promise<TaskModel[]> {
    return Promise.resolve(this.taskList);;
  }

  getTask(id: NonNullable<TaskModel['id']> | string): Promise<TaskModel | undefined> {
    return this.getTasks()
      .then(tasks => tasks.find(task => task.id === +id))
      .catch(() => Promise.reject('Error in getTask method'));
  }

  createTask(task: TaskModel): void {
    this.taskList.push(task);
  }

  updateTask(task: TaskModel): void {
    const i = this.taskList.findIndex(t => t.id === task.id);

    if (i > -1) {
      this.taskList.splice(i, 1, task);
    }
  }

  deleteTask(task: TaskModel): void {
    const i = this.taskList.findIndex(t => t.id === task.id);

    if (i > -1) {
      this.taskList.splice(i, 1);
    }
  }
}
```

7. Create **TaskListComponent.** Run the following command from command line.

```
ng g c tasks/components/task-list --skip-tests true -m tasks.module
```

8. Replace the content of **TaskListComponent.** Use the following snippet of code:

```
import { Component, inject, type OnInit } from '@angular/core';
import { TaskArrayService } from './../../services/task-array.service';
import type { TaskModel } from './../../models/task.model';

@Component({
  templateUrl: './task-list.component.html',
  styleUrls: ['./task-list.component.css']
})
export class TaskListComponent implements OnInit {
  tasks!: Promise<Array<TaskModel>>;

  private taskArrayService = inject(TaskArrayService);

  ngOnInit(): void {
    this.tasks = this.taskArrayService.getTasks();
  }
```

```
  onCompleteTask(task: TaskModel): void {
    const updatedTask = { ...task, done: true };
    this.taskArrayService.updateTask(updatedTask);
  }

  onEditTask(task: TaskModel): void {}
}
```

9. Replace the content of **TaskListComponent template**. Use the following snippet of HTML:

```
<app-task
  *ngFor="let task of tasks | async"
  [task]="task"
  (completeTask)="onCompleteTask($event)"
  (editTask)="onEditTask($event)">
</app-task>
```

10. Create **TaskComponent.** Run the following command from command line:

```
ng g c tasks/components/task --skip-tests true --standalone true -c OnPush
```

11. Replace the content of **TaskComponent**. Use the following snippet of code:

```
import { ChangeDetectionStrategy, Component, EventEmitter, Input, Output } from '@angular/core';
import { CommonModule } from '@angular/common';
import { TaskModel } from '../../models/task.model';

@Component({
  selector: 'app-task',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './task.component.html',
  styleUrls: ['./task.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class TaskComponent {
  @Input({ required: true}) task!: TaskModel;

  @Output() completeTask = new EventEmitter<TaskModel>();
  @Output() editTask = new EventEmitter<TaskModel>();

  onCompleteTask(): void {
    this.completeTask.emit(this.task);
  }

  onEditTask(): void {
    this.editTask.emit(this.task);
  }

}
```

12. Replace the content of **TaskComponent template**. Use the following snippet of HTML:

```
<div class="panel panel-default">
      <div class="panel-heading">Task</div>
      <div class="panel-body">
            <ul>
                  <li>Action: {{task.action}}</li>
                  <li>Priority: {{task.priority}}</li>
                  <li>Estimate Hours: {{task.estHours}}</li>
                  <li>Actual Hours: {{task.actHours}}</li>
```

```
                <li>Done: {{task.done}}</li>
            </ul>
            <button class="btn btn-primary btn-sm"
                (click)="onCompleteTask()"
                [disabled]="task.done">
                Done
            </button>
            <button class="btn btn-warning btn-sm"
                (click)="onEditTask()">
                Edit
            </button>
        </div>
</div>
```

13. Create file **tasks/components/index.ts.** Add the following snippet of code to it:

```
export * from './task/task.component';
export * from './task-list/task-list.component';
```

14. Create file **tasks/index.ts.** Add the following snippet of code to it:

```
export * from './components';
```

15. Make changes to **TasksModule**. Use the following snippet of code:

```
// 1
import { TaskListComponent } from './components/task-list/task-list.component';
import { TaskListComponent, TaskComponent } from './components';

// 2
imports: [CommonModule, FormsModule, TasksRoutingModule, TaskComponent],
```

## Task 09. Tasks Feature Route Configuration

1. Make changes to file **tasks/tasks-routing.module.ts.** Use the following snippet of code:

```
import { type Routes, RouterModule } from '@angular/router';

import { TaskListComponent } from './components';

const routes: Routes = [
  {
    path: 'task-list',
    component: TaskListComponent
  }
];
```

## Task 10. Tasks List on Home Page

1. Make changes to **TasksRoutingModule**. Use the following snippet of code:

```
const routes: Routes = [
  {
    path: 'task-list',
    path: 'home',
    component: TaskListComponent
  }
];
```

2. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { AboutComponent, AbcComponent, HomeComponent, PathNotFoundComponent } from './pages';
{
    path: 'home',
    component: HomeComponent
  },
```

3. Make changes to file **app/pages/index.ts.** Use the following snippet of code:

```
export * from './about/about.component';
export * from './abc/abc.component';
export * from './home/home.component';
export * from './path-not-found/path-not-found.component';
```

4. Delete **HomeComponent** (folder pages/home)

## Task 11. Navigate

1. Create **TaskFormComponent.** Run the following command from command line:

**ng g c tasks/components/task-form --skip-tests true --skip-import true**

2. Replace the content of **TaskFormComponent.** Use the following snippet of code:

```
import { Component, inject, type OnInit } from '@angular/core';
import { TaskModel } from './../../models/task.model';
import { TaskArrayService } from './../../services/task-array.service';

@Component({
  templateUrl: './task-form.component.html',
  styleUrls: ['./task-form.component.css']
})
export class TaskFormComponent implements OnInit {
  task!: TaskModel;

  private taskArrayService = inject(TaskArrayService);

  ngOnInit(): void {
    this.task = new TaskModel();
  }

  onSaveTask(): void {
    const task = { ...this.task } as TaskModel;

    if (task.id) {
      this.taskArrayService.updateTask(task);
    } else {
      this.taskArrayService.createTask(task);
    }
  }

  onGoBack(): void {}
}
```

3. Replace the content of **TaskFormComponent template**. Use the following snippet of HTML:

```
<div class="panel panel-default">
    <div class="panel-heading">
        <h4 class="pannel-title">
            Task Form
        </h4>
    </div>
    <div class="panel-body">
        <form *ngIf="task" (ngSubmit)="onSaveTask()" id="task-form" #form="ngForm">
          <div class="form-group">
            <label for="action">Action</label>
            <input type="text"
                class="form-control"
                id="action" name="action"
                placeholder="Action"
                required
                [(ngModel)]="task.action">
          </div>
          <div class="form-group">
            <label for="priority">Priority</label>
            <input type="number"
```

```
                min="1" max="3"
                class="form-control"
                id="priority" name="priority"
                placeholder="Priority"
                [(ngModel)]="task.priority">
            </div>
            <div class="form-group">
              <label for="estHours">Est. Hours</label>
              <input type="number"
                  min="0"
                  step="2"
                  class="form-control"
                  id="estHours" name="estHours"
                  placeholder="Est. Hours"
                  [(ngModel)]="task.estHours">
            </div>
            <button
                type="submit"
                class="btn btn-primary"
                form="task-form"
                [disabled]="form.invalid" >Save
            </button>
            <button
                type="button"
                class="btn btn-primary"
                (click)="onGoBack()">Back
            </button>
          </form>
    </div>
</div>
```

4. Make changes to the file **tasks/components/index.ts.** Use the following snippet of code:

```
export * from './task/task.component';
export * from './task-form/task-form.component';
export * from './task-list/task-list.component';
```

5. Make changes to **TasksModule**. Use the following snippet of code:

```
// 1
import { TaskListComponent, TaskComponent, TaskFormComponent } from './components';

// 2
declarations: [
    …
    TaskFormComponent
],
```

6. Make changes to **TasksRoutingModule**. Use the following snippet of code:

```
// 1
import { TaskListComponent, TaskFormComponent } from './components';

// 2
const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent
```

```
  },
  {
    path: 'edit/:taskID',
    component: TaskFormComponent
  }
];
```

7. Make changes to **TaskListComponent.** Use the following snippet of code:

```
// 1
import { Router } from '@angular/router';

// 2
private router = inject(Router);


// 3
onEditTask(task: TaskModel): void {
    const link = ['/edit', task.id];
    this.router.navigate(link);
}
```

## Task 12. Getting the route parameter

1. Make changes to **TaskFormComponent**. Use the following snippet of code:

```
// 1
@Input() taskID!: string; // pathParam

// 2
ngOnInit(): void {
    this.task = new TaskModel();

    this.taskArrayService.getTask(this.taskID)
      .then(task => {
        this.task = task ?? {} as TaskModel;
      })
      .catch(err => console.log(err));
  }
```

2. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
RouterModule.forRoot(routes, { bindToComponentInputs: true })
```

## Task 13. Navigate Back

1. Make changes to **TaskFormComponent.** Use the following snippet of code:

```
// 1
import { Router } from '@angular/router';

// 2
private router = inject(Router);

// 3
onGoBack(): void {
    this.router.navigate(['/home']);
}

// 4
if (task.id) {
      this.taskArrayService.updateTask(task);
}
else {
      this.taskArrayService.createTask(task);
}

this.onGoBack();
```

## Task 14. Secondary Router Outlet

1. Make changes to **AppComponent template**. Use the following snippet of HTML:

```
<div class="container">
          <!-- #routerOutlet="outlet" - access to the properties of router-outlet directive -->
          <router-outlet
              #routerOutlet="outlet"
              (activate)='onActivate($event, routerOutlet)'
              (deactivate)='onDeactivate($event, routerOutlet)'>
          </router-outlet>
          <!-- Routed views go here -->
</div>

<div class="container">
     <div class="col-md-10">
          <!-- #routerOutlet="outlet" - access to the properties of router-outlet directive -->
          <router-outlet
                #routerOutlet="outlet"
                (activate)='onActivate($event, routerOutlet)'
                (deactivate)='onDeactivate($event, routerOutlet)'>
          </router-outlet>
          <!-- Routed views go here -->     </div>
     <div class="col-md-2">
          <router-outlet name="messages"></router-outlet>
     </div>
 </div>
```

2. Create **MessagesService.** Run the following command from command line

```
ng g s core/services/messages --skip-tests true
```

3. Create file **core/index.ts.** Use the following snippet of code:

```
export * from './services/messages.service';
```

4. Replace the content of **MessagesService.** Use the following snippet of code:

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

@Injectable({
  providedIn: 'root'
})
export class MessagesService {
  isDisplayed = false;

  private messages: string[] = [];

  addMessage(message: string): void {
    const currentDate = new Date();
    this.messages.unshift(`${message} at ${currentDate.toLocaleString()}`);
  }

  getMessages(): Array<string> {
    return this.messages;
  }
}
```

5. Create **MessagesComponent.** Run the following command from command line:

```
ng g c pages/messages --skip-tests true --standalone true
```

6. Replace the content of **MessagesComponent template.** Use the following snippet of code:

```
<div class="row">
  <h4 class="col-md-10">Message Log</h4>
  <span class="col-md-2">
    <a class="btn btn-default" (click)="onClose()">x</a>
  </span>
</div>
<div>
  <textarea [(ngModel)]="message"></textarea>
</div>
<div>
  <button type="button" class="btn btn-primary" (click)="onSend()">Send</button>
</div>
<div *ngFor="let message of messagesService.getMessages()" class="message-row">
    {{message}}
</div>
```

7. Replace the content of **MessagesComponent style. Use the following snippet of CSS:**

```
.message-row {
  margin-bottom: 10px;
}
```

8. Replace the content of **MessagesComponent.** Use the following snippet of code:

```
import { Component, inject, type OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { MessagesService } from '../../core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-messages',
  standalone: true,
  imports: [CommonModule, FormsModule],
  templateUrl: './messages.component.html',
  styleUrls: ['./messages.component.css']
})
export class MessagesComponent implements OnInit {
  message = '';

  messagesService = inject(MessagesService);
  private router = inject(Router);

  ngOnInit(): void {}

  onClose(): void {
    this.router.navigate([{ outlets: { messages: null } }]);
    this.messagesService.isDisplayed = false;
  }

  onSend(): void {
    if (this.message) {
      this.messagesService.addMessage(this.message);
      this.message = '';
    }
  }
```

```
}
```

9. Make changes to **pages/index.ts.** Use the following snippet of code:

```
export * from './messages/messages.component';
```

10. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { AboutComponent, AbcComponent, MessagesComponent, PathNotFoundComponent } from './pages';

// 2
{
    path: 'messages',
    component: MessagesComponent,
    outlet: 'messages'
  },
```

11. Make changes to **AppComponent.** Use the following snippet of code:

```
// 1
import { Component, inject } from '@angular/core';
import { MessagesService } from './core';
import { Router, type RouterOutlet } from '@angular/router';

// 2
messagesService = inject(MessagesService);
private router = inject(Router)

// 3
onDisplayMessages(): void {
    this.router.navigate([{ outlets: { messages: ['messages'] } }]);
    this.messagesService.isDisplayed = true;
}
```

12. Make changes to **AppComponent template**. Use the following snippet of HTML:

```
<div class="col-md-2">
  <button class="btn btn-success"
      *ngIf="!messagesService.isDisplayed"
      (click)="onDisplayMessages()">Show Messages</button>
  <router-outlet name="messages"></router-outlet>
</div>
```

## Task 15. Users Components

1. Create **UsersModule**. Run the following command in the command line:

```
ng g m users --routing true -m app.module
```

2. Make changes to **UsersModule**. Use the following snippet of code:

```
// 1
import { FormsModule } from '@angular/forms';

// 2
  imports: [
    …,
    FormsModule,
  ]
```

3. Create a **model of user**. Run the following command from command line:

```
ng g cl users/models/user --type model --skip-tests true
```

4. Replace the content of the class. Use the following snippet of code:

```
   export class UserModel {
      constructor(
            public id: number | null,
            public firstName: string,
            public lastName: string
      ) {}
   }
```

5. Create **UserArrayService.** Run the following command from command line:

```
ng g s users/services/user-array --skip-tests true
```

6. Replace the content of **UserArrayService**. Use the following snippet of code:

```
import { Injectable } from '@angular/core';
import { EMPTY, of, throwError, catchError, switchMap, type Observable } from 'rxjs';
import { UserModel } from './../models/user.model';

@Injectable({
  providedIn: 'root'
})
export class UserArrayService {
  private userList: Array<UserModel> = [
    new UserModel(1, 'Anna', 'Borisova'),
    new UserModel(2, 'Boris', 'Vlasov'),
    new UserModel(3, 'Gennadiy', 'Dmitriev')
  ];

  users$: Observable<UserModel[]> = of(this.userList);

  getUser(id: NonNullable<UserModel['id']> | string): Observable<UserModel> {
    return this.users$.pipe(
      switchMap((users: Array<UserModel>) => {
        const user = users.find(user => user.id === +id);
        return user ? of(user) : EMPTY;
      }),
```

```
      catchError(err => throwError(() => 'Error in getUser method'))
    );
  }

  createUser(user: UserModel): void {
    this.userList.push(user);
  }

  updateUser(user: UserModel): void {
    const i = this.userList.findIndex(u => u.id === user.id);

    if (i > -1) {
      this.userList.splice(i, 1, user);
    }
  }
}
```

7. Create **UserListComponent.** Run the following command from command line:

```
ng g c users/components/user-list --skip-tests true --skip-import true
```

8. Replace the content of **UserListComponent.** Use the following snippet of code:

```
import { Component, inject, type OnInit } from '@angular/core';
import { EMPTY, type Observable, catchError } from 'rxjs';
import type { UserModel } from './../../models/user.model';
import { UserArrayService } from './../../services/user-array.service';

@Component({
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent implements OnInit {
  users$!: Observable<Array<UserModel>>;

  private userArrayService = inject(UserArrayService);

  ngOnInit(): void {
    this.users$ = this.userArrayService.users$
      .pipe(
        catchError(err => {
          console.log(err);
          return EMPTY;
        })
      );
  }

  trackByFn(index: number, user: UserModel): number | null {
    return user.id;
  }
}
```

9. Replace the content of **UserListComponent template**. Use the following snippet of HTML:

```
<app-user
  *ngFor="let user of users$ | async; trackBy: trackByFn"
  [user]="user">
</app-user>
```

10. Create **UserComponent.** Run the following command from command line:

```
ng g c users/components/user --skip-tests true --standalone: true -c OnPush
```

11. Replace the content of **UserComponent.** Use the following snippet of code:

```
import { Component, Input, ChangeDetectionStrategy } from '@angular/core';
import type { UserModel } from './../../models/user.model';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-user',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class UserComponent {
  @Input({ required: true }) user!: UserModel;

  onEditUser(): void {

  }
}
```

12. Replace the content of **UserComponent template**. Use the following snippet of HTML:

```
<div class="panel panel-default">
	<div class="panel-heading">User</div>
	<div class="panel-body">
		<ul>
			<li>FirstName: {{user.firstName}}</li>
			<li>LastName: {{user.lastName}}</li>
		</ul>
		<button class="btn btn-warning btn-sm"
			(click)="onEditUser()">
			Edit
		</button>
	</div>
</div>
```

13. Create **UserFormComponent.** Run the following command from the command line:

```
ng g c users/components/user-form --skip-tests true --skip-import true
```

14. Replace the content of **UserFormComponent.** User the following snippet of code:

```
import { Component, type OnInit, inject, DestroyRef } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Subscription } from 'rxjs';
import { UserModel } from './../../models/user.model';
import { UserArrayService } from './../../services/user-array.service';

@Component({
  templateUrl: './user-form.component.html',
  styleUrls: ['./user-form.component.css']
```

```
})
export class UserFormComponent implements OnInit {
  user!: UserModel;
  originalUser!: UserModel;

  private sub!: Subscription;
  private userArrayService = inject(UserArrayService);
  private route = inject(ActivatedRoute);
  private destroyRef = inject(DestroyRef);

  ngOnInit(): void {
    this.user = new UserModel(null, '', '');

    // we should recreate component because this code runs only once
    const id = this.route.snapshot.paramMap.get('userID')!;
    const observer = {
      next: (user: UserModel) => {
        this.user = { ...user };
        this.originalUser = { ...user };
      },
      error: (err: any) => console.log(err)
    };
    this.sub = this.userArrayService.getUser(id).subscribe(observer);

    // OnDestroy
    this.destroyRef.onDestroy(() => {
      console.log('OnDestroy hook of UserForm via DestroyRef');
      this.sub.unsubscribe();
    });
  }

  onSaveUser(): void {
    const user = { ...this.user };
    const method = user.id ? 'updateUser' : 'createUser';

    this.userArrayService[method](user);
  }

  onGoBack(): void {}
}
```

15. Replace the content of **UserFormComponent template**. Use the following snippet of HTML:

```
<div class="panel panel-default">
    <div class="panel-heading">
        <h4 class="pannel-title">
            User Form
        </h4>
    </div>
    <div class="panel-body">
        <form *ngIf="user" (ngSubmit)="onSaveUser()" id="user-form" #form="ngForm">
          <div class="form-group">
            <label for="firstName">First Name</label>
            <input type="text"
                class="form-control"
                id="firstName" name="firstName"
                placeholder="First Name"
                required
                [(ngModel)]="user.firstName">
```

```
          </div>
          <div class="form-group">
            <label for="lastName">Last Name</label>
            <input type="text"
                class="form-control"
                id="lastName" name="lastName"
                placeholder="Last Name"
                [(ngModel)]="user.lastName">
          </div>
         <button
            type="submit"
            class="btn btn-primary"
           [disabled]="form.invalid">Save
        </button>
        <button class="btn btn-primary"
            type="button"
            (click)="onGoBack()">Back
        </button>
        </form>
    </div>
</div>
```

1. Create file **users/components/index.ts.** Add the following snippet of code to it:

```
export * from './user/user.component';
export * from './user-form/user-form.component';
export * from './user-list/user-list.component';
```

1. Create file **users/services/index.ts.** Add the following snippet of code to it:

```
export * from './user-array.service';
```

2. Create file **users/index.ts.** Add the following snippet of code to it:

```
export * from './services';
export * from './components';
```

3. Make changes to **UsersModule**. Use the following snippet of code:

```
// 1
import { UserComponent, UserFormComponent, UserListComponent } from './components';

// 2
imports: […, UserComponent],
declarations: [UserListComponent, UserFormComponent]
```

## Task 16. Users Feature Area

1. Create **UsersComponent.** Run the following command from the command line:

**ng g c users/users --skip-tests true --flat true --skip-import true --skip-selector true**

2. Replace the content of **UsersComponent template.** Use the following snippet of HTML:

```
<h2>Users</h2>
```

## Task 17. Users Nested Routing

1. Make changes to **AppComponent template**. Use the following snippet of HTML:

```
<div>
   <ul class="nav navbar-nav">
      <li routerLinkActive="active">
         <a routerLink="/users">Users</a>
      </li>
      …
   </ul>
</div>
```

2. Make changes to **UsersComponent template**. Use the following snippet of HTML:

```
<h2>Users</h2>
<router-outlet></router-outlet>
```

3. Make changes to **UsersRoutingModule**. Use the following snippet of code:

```
// 1
import { type Routes, RouterModule } from '@angular/router';
import { UsersComponent } from './users.component';
import { UserListComponent, UserFormComponent } from './components';

const routes: Routes = [
  {
    path: 'users',
    component: UsersComponent,
    children: [
      {
        path: 'add',
        component: UserFormComponent
      },
      {
        path: 'edit/:userID',
        component: UserFormComponent,
      },
      {
        path: '',
        component: UserListComponent
      },
    ]
  }
];

export class UsersRoutingModule {
  static components = [UsersComponent, UserListComponent, UserFormComponent];
}
```

4. Make changes to **UsersModule**. Use the following snippet of code:

```
// 1
import { UserComponent, UserFormComponent, UserListComponent } from './components';

// 2
declarations: [
    UsersRoutingModule.components,
    UserListComponent, UserFormComponent,
]
```

## Task 18. Relative Navigation

1. Make changes to **UserComponent.** Use the following snippet of code:

```
// 1
import { Component, Input, ChangeDetectionStrategy, Output, EventEmitter } from '@angular/core';


// 2
@Output() editUser = new EventEmitter<UserModel>();

// 3
onEditUser(): void {
    this.editUser.emit(this.user);
}
```

2. Make changes to **UserListComponent template**. Use the following snippet of HTML:

```
<app-user
  *ngFor='let user of users$ | async ; trackBy: trackByFn'
  [user]="user"
  (editUser)="onEditUser($event)">
</app-user>
```

3. Make changes to **UserListComponent.** Use the following snippet of code:

```
// 1
import { Router, ActivatedRoute } from '@angular/router';

// 2
private router = inject(Router);
private route = inject(ActivatedRoute);

// 3
onEditUser({ id }: UserModel): void {
    const link = ['/users/edit', id];
    this.router.navigate(link);
    // or
    // const link = ['edit', id];
    // this.router.navigate(link, {relativeTo: this.route});
  }
```

4. Make changes to **UserFormComponent.** Use the following snippet of code:

```
// 1
import { ActivatedRoute, Router } from '@angular/router';

// 2
private router = inject(Router);

// 3
this.userArrayService[method](user);
this.onGoBack();

// 4
onGoBack(): void {
     this.router.navigate(['./../../'], { relativeTo: this.route});
}
```

## Task 19. Optional Parameters

1. Make changes to the method **onSaveUser** of **UserFormComponent**. Use the following snippet of code:

```
if (user.id) {
      this.router.navigate(['/users', {editedUserID: user.id}]);
} else {
      this.onGoBack();
}
this.onGoBack();
```

2. Make changes to **UserListComponent.** Use the following snippet of code:

```
// 1
import { Component, DestroyRef, inject, Input, type OnInit } from '@angular/core';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

// 2
@Input() editedUserID!: string;
private editedUser!: UserModel;
private destroyRef = inject(DestroyRef);

// 3
ngOnInit(): void {
    this.users$ = this.userArrayService.users$
      .pipe(
        catchError(err => {
          console.log(err);
          return EMPTY;
        })
      );

// listen to editedUserID => editedUser from UserFormComponent
const observer = {
   next: (user: UserModel) => {
      this.editedUser = { ...user };
      console.log(`Last time you edited user ${JSON.stringify(this.editedUser)}`);
   },
   error: (err: any) => console.log(err),
   complete: () => console.log('Complete listening to editedUser')
};

this.userArrayService
    .getUser(this.editedUserID)
    .pipe(takeUntilDestroyed(this.destroyRef))
    .subscribe(observer);
  }

// 5
isEdited({ id }: UserModel): boolean {
    if (this.editedUser) {
      return id === this.editedUser.id;
    }
    return false;
}
```

3. Make changes to **UserListComponent template.** Use the following snippet of HTML:

```
<app-user
  *ngFor='let user of users$ | async; trackBy: trackByFn'
```

```
  [user]="user"
  [class.edited]="isEdited(user)"
  (editUser)="onEditUser($event)">
</app-user>
```

4. Make changes to **UserComponent style.** Use the following snippet of CSS:

```
:host.edited > div {
  border: 2px dotted red;
}
```

## Task 20. Admin Feature Area

1. Create **AdminModule** and **AdminRoutingModule**. Run the following command from command line:

```
ng g m admin --routing true -m app.module
```

2. Create the following blank components:
    a. **AdminDashboardComponent,**
    b. **ManageTasksComponent,**
    c. **ManageUsersComponent**,
    d. **AdminComponent**

Run the following commands from command line:

```
ng g c admin/components/admin-dashboard --skip-tests true --standalone true
ng g c admin/components/manage-tasks --skip-tests true --standalone true
ng g c admin/components/manage-users --skip-tests true --standalone true
ng g c admin/admin --skip-tests true --flat true
```

3. Replace the content of **AdminComponent template.** Use the following snippet of HTML:

```
<h3>Admin</h3>
<nav>
  <ul class="nav nav-tabs">
    <li routerLinkActive="active" [routerLinkActiveOptions]="{ exact: true }">
      <a routerLink="./">Dashboard</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./tasks">Manage Tasks</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./users">Manage Users</a>
    </li>
  </ul>
</nav>
<router-outlet></router-outlet>
```

4. Create the file **admin/components/index.ts.** Use the following snippet of code:

```
export * from './admin-dashboard/admin-dashboard.component';
export * from './manage-tasks/manage-tasks.component';
export * from './manage-users/manage-users.component';
```

5. Create the file **admin/index.ts.** Use the following snippet of code:

```
export * from './components';
```

6. Replace the content of **AdminRoutingModule**. Use the following snippet of code:

```
// 1
import { type Routes, RouterModule } from '@angular/router';
import { AdminComponent } from './admin.component';
import { AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent } from './components';

// 2
const routes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
```

```
    children: [
      {
        path: '',
        children: [
          { path: 'users', component: ManageUsersComponent },
          { path: 'tasks', component: ManageTasksComponent },
          { path: '', component: AdminDashboardComponent }
        ]
      }
    ]
  }
];

// 3
export class AdminRoutingModule {
  static components = [
    AdminDashboardComponent,
    ManageTasksComponent,
    ManageUsersComponent
  ];
}
```

7. Make changes to **AdminModule**. Use the following snippet of code:

```
imports: [CommonModule, AdminRoutingModule, AdminRoutingModule.components],
```

8. Make changes to **AppComponent** template. Use the following snippet of HTML:

```
<li routerLinkActive="active">
    <a routerLink="/users">Users</a>
</li>

<li routerLinkActive="active">
    <a routerLink="/admin">Admin</a>
</li>
```

## Task 21. canActivate Guard

1. Create a **canActivateAuthGuard.** Run the following command from command line:

```
ng g g core/guards/can-activate-auth --functional true --skip-tests true --guardType CanActivate
```

2. Make changes to the **canActivateAuthGuard**. Use the following snippet of code:

```
export const canActivateAuthGuard: CanActivateFn = (route, state) => {
  console.log('CanActivate Guard is called');
  return true;
};
```

3. Make changes to the file **core/index.ts.** Use the following snippet of code:

```
export * from './guards/can-activate-auth.guard';
```

4. Make changes to the **AdminRoutingModule**. Use the following snippet of code:

```
import { canActivateAuthGuard } from './../core';

const adminRoutes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    canActivate: [canActivateAuthGuard],
    children: [
      {
        path: '',
        children: [
          { path: 'users', component: ManageUsersComponent },
          { path: 'tasks', component: ManageTasksComponent },
          { path: '', component: AdminDashboardComponent }
        ]
      }
    ]
  }
];
```

## Task 22. Auth Service

1. Create **AuthService.** Run the following command from command line:

```
ng g s core/services/auth --skip-tests true
```

2. Make changes to the file **core/index.ts**. Use the following snippet of code:

```
export * from './services/auth.service';
```

3. Replace the content of **AuthService.** Use the following snippet of code:

```
import { Injectable } from '@angular/core';
import { type Observable, of, delay, tap } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  isLoggedIn = false;
  isAdmin = false;

  // store the URL so we can redirect after logging in
  redirectUrl!: string;

  login(isAdmin: boolean = false): Observable<boolean> {
    return of(true).pipe(
      delay(1000),
      tap(val => {
        this.isLoggedIn = val;
        this.isAdmin = isAdmin;
      })
    );
  }

  logout(): void {
    this.isLoggedIn = false;
    this.isAdmin = false;
  }


  checkLogin(url: string): boolean {
    if (this.isLoggedIn) {
      return true;
    }

    // Store the attempted URL for redirecting
    this.redirectUrl = url;

    // Navigate to the login, return false
    return false;
  }
}
```

4. Make changes to the **canActivateAuthGuard.** Use the following snippet of code:

```
import { inject } from '@angular/core';
import { CanActivateFn, Router } from '@angular/router';
import { AuthService } from './../services/auth.service';
```

```
export const canActivateAuthGuard: CanActivateFn = (route, state) => {
  const authService = inject(AuthService);
  const router = inject(Router);
  const { url } = state;

  console.log('CanActivate Guard is called');
  return true;
  return authService.checkLogin(url)
      ? true
      : router.parseUrl('/login');
};
```

## Task 23. Login Component

1. Create **LoginComponent.** Run the following command from command line:

```
ng g c pages/login --skip-tests true --standalone true
```

2. Make changes to the file **pages/index.ts**. Use the following snippet of code:

```
export * from './login/login.component';
```

3. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { AboutComponent, MessagesComponent, LoginComponent, PathNotFoundComponent } from './pages';

// 2
{
    path: 'about',
    component: AboutComponent
},
{
    path: 'login',
    component: LoginComponent
},
```

4. Make changes to **LoginComponent.** Use the following snippet of code:

```
// 1
import { Component, type OnInit, inject, DestroyRef } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';
import { AuthService } from './../../../core';

// 2
message!: string;
authService = inject(AuthService);

private router = inject(Router);
private destroyRef = inject(DestroyRef);

// 3
onLogin(): void {
    this.message = 'Trying to log in ...';
    const observer = {
      next: () => {
        this.setMessage();
        if (this.authService.isLoggedIn) {
          // Get the redirect URL from our auth service
          // If no redirect has been set, use the default
          const redirect = this.authService.redirectUrl
            ? this.authService.redirectUrl
            : '/admin';
          // Redirect the user
          this.router.navigate([redirect]);
        }
      },
      error: (err: any) => console.log(err),
      complete: () => console.log('[takeUntilDestroyed] complete')    };
```

```
    this.authService
      .login()
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe(observer);
}

onLogout(): void {
    this.authService.logout();
    this.setMessage();
}

private setMessage(): void {
    this.message = 'Logged ' + (this.authService.isLoggedIn ? 'in' : 'out');
}

// 4
ngOnInit(): void {
  this.setMessage();
}
```

5. Replace the content of **LoginComponent template.** Use the following snippet of HTML:

```
<h2>LOGIN</h2>
<p>State: {{message}}</p>
<p>
  <button class="btn btn-primary" (click)="onLogout()"  *ngIf="authService.isLoggedIn; else
loginBtn">Logout</button>

  <ng-template #loginBtn>
    <button class="btn btn-primary" (click)="onLogin()" >Login</button>
  </ng-template>
</p>
```

6. Make changes to **AppComponent template.** Use the following snippet of code:

```
<li routerLinkActive="active">
     <a routerLink="/admin">Admin</a>
 </li>
<li routerLinkActive="active">
     <a routerLink="/login">Login</a>
</li>
```

## Task 24. canActivateChild Guard

1. Create a **canActivateChildAuthGuard.** Run the following command from command line:

**ng g g core/guards/can-activate-child-auth --functional true --skip-tests true --guardType CanActivateChild**

2. Replace the content of the **canActivateChildAuthGuard** with the following snippet of code:

```
import { inject } from "@angular/core";
import { CanActivateChildFn, Router } from '@angular/router';

import { AuthService } from "../services/auth.service";

export const canActivateChildAuthGuard: CanActivateChildFn = (childRoute, state) => {
  const authService = inject(AuthService);
  const router = inject(Router);
  const { url } = state;

  console.log('CanActivateChild Guard is called');
  return authService.checkLogin(url)
    ? true
    : router.parseUrl('/login');
};
```

3. Make changes to the file **core/index.ts**. Use the following snippet of code:

```
export * from './guards/can-activate-child-auth.guard';
```

4. Make changes to **AdminRoutingModule**. Use the following snippet of code:

```
// 1
import { AuthGuard, canActivateChildAuthGuard} from './../core';

// 2
{
        path: '',
        canActivateChild: [canActivateChildAuthGuard],
        children: [
          { path: 'users', component: ManageUsersComponent },
          { path: 'tasks', component: ManageTasksComponent },
          { path: '', component: AdminDashboardComponent }
        ]
}
```

## Task 25. canDeactivate Guard

1. Create **DialogService.** Run the following command from command line:

```
ng g s core/services/dialog --skip-tests true
```

2. Replace the content of **DialogService**. Use the following snippet of code:

```
import { Injectable } from '@angular/core';
import { type Observable, of } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class DialogService {
  confirm(message?: string): Observable<boolean> {
    const confirmation = window.confirm(message || 'Is it OK?');
    return of(confirmation);
  }
}
```

3. Create **interface CanComponentDeactivate.** Run the following command from command line:

```
ng g i core/interfaces/can-component-deactivate --type interface
```

4. Replace the content of **CanComponentDeactivate interface**. Use the following snippet of code:

```
import type { UrlTree } from '@angular/router';

// rxjs
import type { Observable } from 'rxjs';

export interface CanComponentDeactivate {
  canDeactivate: () =>
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree>
    | boolean
    | UrlTree;
}
```

5. Create **CanDeactivateGuard.** Run the following command from command line:

```
ng g g core/guards/can-deactivate --skip-tests true --functional true --guardType CanDeactivate
```

6. Replace the content of the **CanDeactivateGuard.** Use the following snippet of code:

```
import { CanDeactivateFn } from '@angular/router';
import { CanComponentDeactivate } from '../interfaces/can-component-deactivate.interface';

export const canDeactivateGuard: CanDeactivateFn<CanComponentDeactivate> = (component, currentRoute,
currentState, nextState) => {
  console.log('CanDeactivate Guard is called');
  return component.canDeactivate?.() ?? true;
};
```

7. Make changes to the file **core/index.ts**. Use the following snippet of code:

```
export * from './services/dialog.service';
export * from './interfaces/can-component-deactivate.interface';
```

```
export * from './guards/can-deactivate.guard';
```

8. Make changes to **UserFormComponent.** Use the following snippet of code:

```
// 1
import { ActivatedRoute, Router, type UrlTree } from '@angular/router';
import type { Observable, type Subscription } from 'rxjs';
import { DialogService } from './../../../core';
import type { CanComponentDeactivate } from './../../../core';

// 2
export class UserFormComponent implements OnInit, CanComponentDeactivate {

// 3
private onGoBackClick: boolean = false;
private dialogService = inject(DialogService);

// 4
onGoBack(): void {
    this.onGoBackClick = true;
    this.router.navigate(['./../../'], { relativeTo: this.route });
}

// 5
canDeactivate():
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree>
    | boolean
    | UrlTree {

    if (this.onGoBackClick) return true;

    const flags = (Object.keys(this.originalUser) as (keyof UserModel)[]).map(key => {
      if (this.originalUser[key] === this.user[key]) {
        return true;
      }
      return false;
    });

    if (flags.every(el => el)) {
      return true;
    }

    // Otherwise ask the user with the dialog service and return its
    // promise which resolves to true or false when the user decides
    return this.dialogService.confirm('Discard changes?');
  }
```

9. Make changes to **UsersRoutingModule**. Use the following snippet of code:

```
import { canDeactivateGuard } from './../core';

{
        path: 'edit/:userID',
        component: UserFormComponent,
        canDeactivate: [canDeactivateGuard]
}
```

## Task 26. resolve Guard

1. Create **userResolver.** Run the following command from command line:

```
ng g r users/resolvers/user --skip-tests true --functional true
```

2. Create the file **users/resolvers/index.ts**. Use the following snippet of code:

```
export * from './user.resolver';
```

3. Make changes to file **users/index.ts**. Use the following snippet of code:

```
export * from './resolvers';
```

4. Replace the content of **UserResolver.** Use the following snippet of code:

```
import { inject } from '@angular/core';
import { ResolveFn, Router } from '@angular/router';
import { catchError, EMPTY, of, switchMap, take } from 'rxjs';
import { UserModel } from '../models/user.model';
import { UserArrayService } from '../services';

export const userResolver: ResolveFn<UserModel> = (route, state) => {
  const userArrayService = inject(UserArrayService);
  const router = inject(Router);

  console.log('userResolver is called');

  if (!route.paramMap.has('userID')) {
    return of(new UserModel(null, '', ''));
  }

  const id = route.paramMap.get('userID')!;

  return userArrayService.getUser(id).pipe(
    switchMap((user: UserModel) => {
      if (user) {
        return of(user);
      } else {
        router.navigate(['/users']);
        return EMPTY;
      }
    }),
    take(1),
    catchError(() => {
      router.navigate(['/users']);
      // catchError MUST return observable
      return EMPTY;
    })
  );
};
```

5. Make changes to **UsersRoutingModule**. Use the following snippet of code:

```
// 1
import { userResolver } from './resolvers';

// 2
{
        path: 'edit/:userID',
```

```
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard],
        resolve: {
          userFromResolver: UserResolver
        }
}
```

6. Make changes to **UserFormComponent.** Use the following snippet of code:

```
// 1
import { Component, type OnInit, inject, Input, DestroyRef} from '@angular/core';
import type { Observable, Subscription } from 'rxjs';

// 2
private destroyRef = inject(DestroyRef);
private sub!: Subscription;

// 3
@Input({ required: true }) userFromResolver: UserModel = new UserModel(null, '', '');

// 4
ngOnInit(): void {
    this.user = new UserModel(null, '', '');

    // we should recreate component because this code runs only once
    const id = this.route.snapshot.paramMap.get('userID')!;
    const observer = {
      next: (user: UserModel) => {
        this.user = { ...user };
        this.originalUser = { ...user };
      },
      error: (err: any) => console.log(err)
    };
    this.sub = this.userArrayService.getUser(id).subscribe(observer);

    // OnDestroy
    this.destroyRef.onDestroy(() => {
      console.log('OnDestroy hook of UserForm via DestroyRef');
      this.sub.unsubscribe();
    });

this.user = {...this.userFromResolver};
this.originalUser = { ...this.userFromResolver };

}
```

7. Make changes to **UserFormComponent template**. Use the following snippet of HTML:

```
<form *ngIf="user" (ngSubmit)="onSaveUser()" id="user-form" #form="ngForm">
```

## Task 27. Apply Spinner

1. Create **SpinnerComponent, SpinnerService**. Use the following command in the command line:

```
ng g c widgets/spinner/spinner --skip-tests true --flat true --standalone true
ng g s widgets/spinner/spinner --skip-tests true
```

2. Crete a file **widgets/index.ts**. Use the following snippet of code:

```
export * from './spinner/spinner.component';
export * from './spinner/spinner.service';
```

3. Replace the content of **SpinnerComponent template**. Use the following snippet of HTML:

```
<div class="spinner"></div>
```

4. Replace the content of **SpinnerComponent css file.** Use the following snippet of CSS:

```
.spinner {
  color: #337ab7;
  font-size: 30px;
  text-indent: -9999em;
  overflow: hidden;
  width: 1em;
  height: 1em;
  border-radius: 50%;
  margin: 72px auto;
  position: relative;
  -webkit-transform: translateZ(0);
  -ms-transform: translateZ(0);
  transform: translateZ(0);
  -webkit-animation: load6 1.7s infinite ease, round 1.7s infinite ease;
  animation: load6 1.7s infinite ease, round 1.7s infinite ease;
}
@-webkit-keyframes load6 {
  0% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
  }
  5%,
  95% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
  }
  10%,
  59% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.087em -0.825em 0 -0.42em, -0.173em -0.812em 0 -0.44em, -
0.256em -0.789em 0 -0.46em, -0.297em -0.775em 0 -0.477em;
  }
  20% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.338em -0.758em 0 -0.42em, -0.555em -0.617em 0 -0.44em, -
0.671em -0.488em 0 -0.46em, -0.749em -0.34em 0 -0.477em;
  }
  38% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.377em -0.74em 0 -0.42em, -0.645em -0.522em 0 -0.44em, -
0.775em -0.297em 0 -0.46em, -0.82em -0.09em 0 -0.477em;
  }
  100% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
```

```
  }
}
@keyframes load6 {
  0% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
  }
  5%,
  95% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
  }
  10%,
  59% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.087em -0.825em 0 -0.42em, -0.173em -0.812em 0 -0.44em, -
0.256em -0.789em 0 -0.46em, -0.297em -0.775em 0 -0.477em;
  }
  20% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.338em -0.758em 0 -0.42em, -0.555em -0.617em 0 -0.44em, -
0.671em -0.488em 0 -0.46em, -0.749em -0.34em 0 -0.477em;
  }
  38% {
    box-shadow: 0 -0.83em 0 -0.4em, -0.377em -0.74em 0 -0.42em, -0.645em -0.522em 0 -0.44em, -
0.775em -0.297em 0 -0.46em, -0.82em -0.09em 0 -0.477em;
  }
  100% {
    box-shadow: 0 -0.83em 0 -0.4em, 0 -0.83em 0 -0.42em, 0 -0.83em 0 -0.44em, 0 -0.83em 0 -0.46em, 0
-0.83em 0 -0.477em;
  }
}
@-webkit-keyframes round {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    transform: rotate(360deg);
  }
}
@keyframes round {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    transform: rotate(360deg);
  }
}
```

5. Replace the content of **SpinnerService**. Use the following snippet of code:

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

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

```
  private visible = false;

  isVisible(): boolean {
    return this.visible;
  }

  hide(): void {
    this.visible = false;
  }

  show(): void {
    this.visible = true;
  }
}
```

6. Make changes to **AppModule**. Use the following snippet of code:

```
// 1
import { SpinnerComponent } from './widgets';

// 2
imports: [
    …
    SpinnerComponent,
    AppRoutingModule
],
```

7. Make changes to **AppComponent**. Use the following snippet of code:

```
// 1
import { SpinnerService } from './widgets';

// 2
spinnerService = inject(SpinnerService);
```

8. Make changes to **AppComponent template**. Use the following snippet of HTML:

```
<div class="container">
          <app-spinner *ngIf="spinnerService.isVisible()"></app-spinner>
```

9. Make changes to **userResolver**. Use the following snippet of code:

```
// 1
import { catchError, EMPTY, of, switchMap, take, delay, finalize } from 'rxjs';
import { SpinnerService } from './../../widgets';

// 2
const userArrayService = inject(UserArrayService);
const spinner = inject(SpinnerService);
const router = inject(Router);

// 3
spinner.show();
const id = route.paramMap.get('userID')!;

// 4
return userArrayService.getUser(id).pipe(
      delay(2000),
      switchMap((user: UserModel) => {
```

```
    if (user) {
      return of(user);
    } else {
      router.navigate(['/users']);
      return EMPTY;
    }
  }),
  take(1),
  catchError(() => {
    router.navigate(['/users']);
    return EMPTY
  }),
  finalize(() => spinner.hide())
);
```

## Task 28. Query Parameters and Fragment

1. Make changes to **canActivateAuthGuard**. Use the following snippet of code:

```
// 1
import { CanActivateFn, Router, NavigationExtras } from '@angular/router';// 2

// 2
console.log('CanActivateGuard is called');

// Create a dummy session id
const sessionId = 123456789;
const navigationExtras: NavigationExtras = {
      queryParams: { sessionId },
      fragment: 'anchor'
};

return authService.checkLogin(url)
           ? true
           : router.parseUrl('/login')
if (this.authService.checkLogin(url)) {
      return true;
} else {
      this.router.navigate(['/login'], navigationExtras);
      return false;
}
```

2. Make changes to **LoginComponent.** Use the following snippet of code:

```
// 1
import { Router, type NavigationExtras } from '@angular/router';


// 2
if (this.authService.isLoggedIn) {
        const redirect = this.authService.redirectUrl
             ? this.authService.redirectUrl : '/admin';

        const navigationExtras: NavigationExtras = {
          queryParamsHandling: 'preserve',
          preserveFragment: true
        };

        // Redirect the user
        this.router.navigate([redirect], navigationExtras);
}
```

3. Make changes to **AdminDashboardComponent template**. Use the following snippet of HTML:

```
<p>Session ID: {{ sessionId | async }}</p>
<a id="anchor"></a>
<p>Token: {{ token | async }}</p>
```

4. Make changes to **AdminDashboardComponent.** Use the following snippet of code:

```
// 1
import { Component, type OnInit, Input, inject } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { type Observable, map } from 'rxjs';
```

```
// 2
@Input({transform: (value: string) => {
    return value ?? 'None';
}}) sessionId!: string;

token!: Observable<string>;

private route = inject(ActivatedRoute);

// 3
ngOnInit(): void {
    // Capture the fragment if available
    this.token = this.route.fragment.pipe(
        map(fragment => fragment || 'None')
    );
  }
```

5. Make changes to **AdminComponent template.** Use the following snippet of HTML:

```
<nav>
  <ul class="nav nav-tabs">
    <li routerLinkActive="active" [routerLinkActiveOptions]="{ exact: true }">
      <a routerLink="./" queryParamsHandling="preserve" [preserveFragment]="true">Dashboard</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./tasks" queryParamsHandling="preserve" [preserveFragment]="true">Manage
Tasks</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./users" queryParamsHandling="preserve" [preserveFragment]="true">Manage
Users</a>
    </li>
  </ul>
</nav>
```

## Task 29. Lazy-Loading Route Configuration

1. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
  {
    path: 'admin',
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)
  },
  {
    path: 'users',
    loadChildren: () => import('./users/users.module').then(m => m.UsersModule)
  },
```

2. Make changes to **AdminRoutingModule**. Use the following snippet of code:

```
const routes: Routes = [
  {
    path: 'admin',
    path: '',
    component: AdminComponent,
    canActivate: [canActivateAuthGuard],
    children: [
      …
    ]
  }
];
```

3. Make changes to **UsersRoutingModule**. Use the following snippet of code:

```
const routes: Routes = [
  {
    path: 'users',
    path: '',
    component: UsersComponent,
    children: [
      …
    ]
  }
];
```

4. Make changes to **AppModule**. Use the following snippet of code:

```
import { AdminModule } from './admin/admin.module';
import { UsersModule } from './users/users.module';

imports: [
    …
    UsersModule,
    AdminModule,
    AppRoutingModule
]
```

## Task 30. canMatch Guard

1. Create a file **core/guards/can-match-auth.guard.ts**. Use the following snippet of code:

```
import { inject } from '@angular/core';
import type { Route, UrlSegment, UrlTree } from '@angular/router';
import { type Observable } from 'rxjs';
import { AuthService } from './../services/auth.service';

export function canMatchAuthGuard(route: Route, segments: UrlSegment[]
  ): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
  const authService = inject(AuthService);
  const url = `/${route.path}`;

  console.log('CanMatch Guard is called');
  return authService.checkLogin(url);
}
```

2. Make changes to the file **core/index.ts.** Use the following snippet of code:

```
export * from './guards/can-match-auth.guard';
```

3. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { canMatchAuthGuard } from './core';

// 2
{
    path: 'admin',
    canMatch: [canMatchAuthGuard],
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)
},
{
    path: 'admin',
    redirectTo: '/login',
    pathMatch: 'full'
},
```

## Task 31. Default Preloading Strategy

1. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { type Routes, type ExtraOptions, PreloadAllModules, RouterModule, type UrlSegment, type
UrlSegmentGroup, type Route, type UrlMatchResult } from '@angular/router';


// 2
const extraOptions: ExtraOptions = {
  preloadingStrategy: PreloadAllModules,
  bindToComponentInputs: true,
  enableTracing: true // Makes the router log all its internal events to the console.
};

// 3
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { bindToComponentInputs: true}extraOptions)
  ]
})
```

## Task 32. Custom Preloading Strategy

1. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
{
    path: 'admin',
    canMatch: [canMatchAuthGuard],
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule),
    data: { preload: false }
},
{
    path: 'users',
    loadChildren: () => import('./users/users.module').then(m => m.UsersModule),
    data: { preload: true }
},
```

2. Create **CustomPreloadingStrategyService.** Run the following command from command line:

**ng g s core/services/custom-preloading-strategy --skip-tests true**

3. Replace the content of **CustomPreloadingStrategyService**. Use the following snippet of code:

```
import { Injectable } from '@angular/core';
import { PreloadingStrategy, type Route } from '@angular/router';
import { type Observable, EMPTY } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class CustomPreloadingStrategyService implements PreloadingStrategy {
  public preloadedModules: string[] = [];

  preload(route: Route, load: () => Observable<any>): Observable<any> {
    if (route.data?.['preload'] && route.path) {
      this.preloadedModules.push(route.path);
      return load();
    } else {
      return EMPTY;
    }
  }
}
```

4. Make changes to the file **core/index.ts**. Use the following snippet of code:

```
export * from './services/custom-preloading-strategy.service';
```

5. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
// 1
import { type Routes, type ExtraOptions, PreloadAllModules, RouterModule, type UrlSegment, type
UrlSegmentGroup, type Route, type UrlMatchResult } from '@angular/router';
import { canMatchAuthGuard, CustomPreloadingStrategyService } from './core';
// 2
const extraOptions: ExtraOptions = {
  preloadingStrategy: PreloadAllModules CustomPreloadingStrategyService,
  bindToComponentInputs: true,
  // enableTracing: true // Makes the router log all its internal events to the console.
};
```

6. Make changes to **AppComponent**. Use the following snippet of code:

```
// 1
import { Component, inject, type OnInit } from '@angular/core';
import { MessagesService, CustomPreloadingStrategyService } from './core';

// 2
export class AppComponent implements OnInit {

// 3
private preloadingStrategy = inject(CustomPreloadingStrategyService);

// 4
ngOnInit(): void {
    console.log(`Preloading Modules: `, this.preloadingStrategy.preloadedModules);
}
```

## Task 33. Page Titles & Router Events

1. Make changes to **AppRoutingModule**. Use the following snippet of code:

```
const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent,
    title: 'About'
  },
  {
    path: 'login',
    component: LoginComponent,
    title: 'Login'
  },
  {
    path: 'admin',
    canMatch: [canMatchAuthGuard],
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule),
    data: { preload: false },
    title: 'Admin'
  },
  {
    path: 'users',
    loadChildren: () => import('./users/users.module').then(m => m.UsersModule),
    title: 'Users',
    data: {
      preload: true
    }
  },
  {
    path: '',
    redirectTo: '/home',
    pathMatch: 'full'
  },
  {
    // The router will match this route if the URL requested
    // doesn't match any paths for routes defined in our configuration
    path: '**',
    component: PathNotFoundComponent,
    title: 'Page Not Found'
  }
];
```

2. Make changes to **AppComponent.** Use the following snippet of code:

```
// 1
import { Component, type OnInit, type OnDestroy } from '@angular/core';
import { Router, type RouterOutlet, NavigationStart, type Event } from '@angular/router';
import { type Subscription, filter } from 'rxjs';

// 2
export class AppComponent implements OnInit, OnDestroy

// 3
private sub: { [key: string]: Subscription } = {};
```

Look at the page titles.

```
// 6
```

```
private setMessageServiceOnRefresh(): void {
    this.sub['navigationStart'] = this.router.events
      .pipe(filter((event: Event) => event instanceof NavigationStart))
      .subscribe((event: Event) => {
        this.messagesService.isDisplayed = (event as NavigationStart).url.includes('messages:');
      });

}

// 7
ngOnInit(): void {
    …
    this.setMessageServiceOnRefresh();
}

// 8
       ngOnDestroy(): void {
           this.sub['navigationStart'].unsubscribe();
       }
```

## Task 34. TitleStrategy

1. Create file **core/services/page-title-strategy.service.ts**. Use the following snippet of code:

```
import { Injectable } from "@angular/core";
import { Title } from "@angular/platform-browser";
import { type RouterStateSnapshot, TitleStrategy } from "@angular/router";

@Injectable()
export class PageTitleStrategy extends TitleStrategy {
  constructor(private titleService: Title) {
    super();
  }

  override updateTitle(routerState: RouterStateSnapshot) {
    const title = this.buildTitle(routerState);

    if (title !== undefined) {
      this.titleService.setTitle(`Task Manager - ${title}`);
    } else {
      this.titleService.setTitle(`Task Manager - Home`);
    }
  }
}
```

2. Make changes to **AppModule.** Use the following snippet of code:

```
// 1
import { Router, TitleStrategy } from '@angular/router';
import { PageTitleStrategy } from './core/services/page-title-strategy.service';

// 2
providers: [
    …
    { provide: TitleStrategy, useClass: PageTitleStrategy }
],
```

Look at the page titles.

## Task 35. Dynamic Page Titles

1. Create a **editUserPageTitleResolver**. Run the following command from command line:

```
ng g r users/resolvers/edit-user-page-title --skip-tests true --functional true
```

2. Replace the content of the **editUserPageTitleResolver** with the following snippet of code:

```
import { inject } from '@angular/core';
import { ResolveFn } from '@angular/router';
import { catchError, of, switchMap, take } from 'rxjs';
import { UserModel } from '../models/user.model';
import { UserArrayService } from '../services';

export const editUserPageTitleResolver: ResolveFn<string> = (route, state) => {
  const userArrayService = inject(UserArrayService);

  const defaultPageTitle = 'Edit User';

  if (!route.paramMap.has('userID')) {
    return of(defaultPageTitle);
  }

  const id = route.paramMap.get('userID')!;

  return userArrayService.getUser(id).pipe(
    switchMap((user: UserModel) => {
      if (user) {
        return of(`${defaultPageTitle}: ${user.firstName} ${user.lastName}`);
      } else {
        return of(defaultPageTitle);
      }
    }),
    take(1),
    catchError(() => {
      return of(defaultPageTitle);
    })
  );

};
```

3. Make changes to file **users/resolvers/index.ts.** Use the following snippet of code:

```
export * from './edit-user-page-title.resolver';
```

4. Make changes to file **UsersRoutingModule.** Use the following snippet of code:

```
// 1
import { userResolver, editUserPageTitleResolver } from './resolvers';

// 2
{
        path: 'edit/:userID',
        component: UserFormComponent,
        title: editUserPageTitleResolver,
        canDeactivate: [canDeactivateGuard],
        resolve: {
```

```
          user: UserResolver
        }
},
```

Look at the page titles for the edit user page.

## Task 36. Meta Service

1. Make changes to **TasksRoutingModule**. Use the following snippet of code:

```
// 1
import type { MetaDefinition } from '@angular/platform-browser';

// 2
const metaTags: Array<MetaDefinition> = [
  {
    name: 'description',
    content: 'Task Manager Application. This is SPA'
  },
  {
    name: 'keywords',
    content: 'Angular tutorial, SPA, Routing'
  }
];

const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent,
    data: {
       meta: metaTags
    }
  },
];
```

2. Make changes to **AppComponent**. Use the following snippet of code:

```
// 1
import { Meta } from '@angular/platform-browser';

// 2
private metaService = inject(Meta);

// 3
onActivate($event: any, routerOutlet: RouterOutlet): void {
    console.log('Activated Component', $event, routerOutlet);
    this.metaService.addTags(routerOutlet.activatedRouteData['meta']);
}
```