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 module **LayoutModule.** Run the following command in the command line:

```
ng g m layout -m app.module
```

2. Create 3 blank components **HomeComponent, AboutComponent, PathNotFoundComponent.** Run the following commands in command line from project root folder[1]:

```
ng g c layout/components/home --skip-tests true --skip-import true
ng g c layout/components/about --skip-tests true --skip-import true
ng g c layout/components/path-not-found --skip-tests true --skip-import true
```

3. Create file **app/layout/components/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';
```

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

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

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

```
// 1
import { AboutComponent, HomeComponent, PathNotFoundComponent } from './components';

// 2
declarations: [HomeComponent, AboutComponent, PathNotFoundComponent]
```

[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 } from '@angular/router';

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

const routes: Routes = [
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'about',
    component: AboutComponent
  },
  {
    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: any): string =>
      typeof value === 'function' ? value.name : value;

    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>
```

Task 07. routerLinkActive

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

```
<li routerLinkActive="active">
      <a routerLink="/about">About</a>
</li>
```

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';

const 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)
];
```

```
const taskListPromise = Promise.resolve(taskList);

@Injectable({
  providedIn: 'any'
})
export class TaskArrayService {
  getTasks(): Promise<TaskModel[]> {
    return taskListPromise;
  }

  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 {
    taskList.push(task);
  }

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

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

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

    if (i > -1) {
      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 } from '@angular/core';
import { TaskArrayService } from './../../services/task-array.service';

import type { OnInit } from '@angular/core';
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>>;

  constructor(private taskArrayService: 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 -c OnPush -m tasks.module
```

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

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

@Component({
  selector: 'app-task',
  templateUrl: './task.component.html',
  styleUrls: ['./task.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class TaskComponent {
  @Input() 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:

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

Task 09. Tasks Feature Route Configuration

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

```
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, HomeComponent, PathNotFoundComponent } from './layout';
{
    path: 'home',
    component: HomeComponent
  },
```

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

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

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

```
// 1
import { AboutComponent, HomeComponent, PathNotFoundComponent } from './components';

// 2
declarations: [HomeComponent, AboutComponent, PathNotFoundComponent]
```

5. Delete **HomeComponent** (folder layout/components/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 } from '@angular/core';
import 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;

  constructor(private taskArrayService: 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
constructor(
       private router: Router,
       private taskArrayService: TaskArrayService
) { }


// 3
onEditTask(task: Task): 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
import { ActivatedRoute } from '@angular/router';
import type { ParamMap } from '@angular/router';
// rxjs
import { map, switchMap } from 'rxjs';

// 2
constructor(
    private taskArrayService: TaskArrayService,
    private route: ActivatedRoute
  ) { }

// 3
ngOnInit(): void {
    this.task = new TaskModel();

    // it is not necessary to save subscription to route.paramMap
    // when router destroys this component, it handles subscriptions automatically
    const observer = {
      next: (task: TaskModel) => (this.task = { ...task }),
      error: (err: any) => console.log(err)
    };
    this.route.paramMap
      .pipe(
        switchMap((params: ParamMap) =>
             // notes about "!"
             // params.get() returns string | null, but getTask takes string | number
             // in this case taskID is a path param and can not be null
             this.taskArrayService.getTask(params.get('taskID')!)
        ),
        // transform undefined => {}
        map(el => el ? el : {} as TaskModel)
      )
      .subscribe(observer);
  }
```

Task 13. Navigate Back

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

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

// 2
constructor(
    private taskArrayService: TaskArrayService,
    private router: Router,
    private route: ActivatedRoute
  ) { }

// 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. Make changes to **LayoutModule**. Use the following snippet of code:

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

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

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

```
ng g c layout/components/messages --skip-tests true -m layout.module
```

7. 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>
```

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

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

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

```
import { Component } from '@angular/core';
import type { OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { MessagesService } from '../../../core';

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

  constructor(
    public messagesService: MessagesService,
    private router: 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 = '';
    }
  }
}
```

10. Make changes to **layout/components/index.ts.** Use the following snippet of code:

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

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

```
import { MessagesComponent } from './layout/components/messages/messages.component';
import { AboutComponent, MessagesComponent, PathNotFoundComponent } from './components';
```

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

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

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

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

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

// 2
constructor(public messagesService: MessagesService) { }
```

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

```
<div class="col-md-2">
  <button class="btn btn-success"
      *ngIf="!messagesService.isDisplayed"
      [routerLink]="[{outlets: {messages: ['messages']}}]">
    Show Messages
  </button>
  <router-outlet name="messages"></router-outlet>
</div>

Look at the result.
```

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

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


// 2
constructor(
```

```
    public messagesService: MessagesService,
    private router: Router
  ) { }

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

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

```
<button class="btn btn-success"
      *ngIf="!messagesService.isDisplayed"
      [routerLink]="[{outlets: {messages: ['messages']}}]">
    Show Messages
  </button>
<button class="btn btn-success"
        *ngIf="!messagesService.isDisplayed"
        (click)="onDisplayMessages()">Show Messages</button>
```

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';

// rxjs
import { EMPTY, Observable, of, throwError, switchMap, catchErro } from 'rxjs';

import { UserModel } from './../models/user.model';

const userList: Array<UserModel> = [
  new UserModel(1, 'Anna', 'Borisova'),
  new UserModel(2, 'Boris', 'Vlasov'),
  new UserModel(3, 'Gennadiy', 'Dmitriev')
];

const userListObservable: Observable<Array<UserModel>> = of(userList);

@Injectable({
  providedIn: 'any'
})
export class UserArrayService {
  users$: Observable<UserModel[]> = userListObservable;

  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 {
    userList.push(user);
  }

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

    if (i > -1) {
      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 } from '@angular/core';
import type { OnInit } from '@angular/core';
// rxjs
import { EMPTY, 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>>;

  constructor(
    private userArrayService: 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 -c OnPush --skip-import true
```

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';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class UserComponent {
  @Input() 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,  } from '@angular/core';
import type { OnInit, OnDestroy } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

// rxjs
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, OnDestroy {
  user!: UserModel;
  originalUser!: UserModel;

  private sub!: Subscription;

  constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute,
  ) { }

  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);
  }

  ngOnDestroy(): void {
    this.sub.unsubscribe();
  }

  onSaveUser(): void {
    const user = {...this.user};

    if (user.id) {
      this.userArrayService.updateUser(user);
    } else {
      this.userArrayService.createUser(user);
    }
    this.originalUser = {...this.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 } from './components';

// 2
declarations: [UserComponent],
```

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

2. Remove selector from meta data of **UsersComponent.**
3. 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 { 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:

```
// 2
declarations: [
    UsersRoutingModule.components,
    UserComponent,
]
```

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
constructor(
    private userArrayService: UserArrayService,
    private router: Router,
    private route: ActivatedRoute
  ) { }

// 3
onEditUser(user: UserModel): void {
    const link = ['/users/edit', user.id];
    this.router.navigate(link);
    // or
    // const link = ['edit', user.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
constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute,
    private router: Router
  ) { }

// 3
if (user.id) {
      this.userArrayService.updateUser(user);
```

```
} else {
      this.userArrayService.createUser(user);
}
this.originalUser = {...this.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.userArrayService.updateUser(user);
      this.router.navigate(['/users', {editedUserID: user.id}]);
    } else {
      this.userArrayService.createUser(user);
      this.onGoBack();
    }
this.originalUser = {...this.user};
this.onGoBack();
```

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

```
// 1
import type { ParamMap } from '@angular/router';
import { EMPTY, Observable, catchError, switchMap } from 'rxjs';
// 2
private editedUser!: UserModel;

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

    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)
    };
    this.route.paramMap
      .pipe(
        switchMap((params: ParamMap) => this.userArrayService.getUser(params.get('editedUserID')!))
      )
      .subscribe(observer);
  }

// 5
isEdited(user: UserModel): boolean {
    if (this.editedUser) {
      return user.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 --skip-import true
ng g c admin/components/manage-tasks --skip-tests true --skip-import true
ng g c admin/components/manage-users --skip-tests true --skip-import true
ng g c admin/admin --skip-tests true --flat true --skip-import 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 { 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 = [
    AdminComponent,
    AdminDashboardComponent,
    ManageTasksComponent,
    ManageUsersComponent
  ];
}
```

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

```
declarations: [
   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 **AuthGuard.** Run the following command from command line:

```
ng g g core/guards/auth --skip-tests true --implements CanActivate
```

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

```
// 1
import type {
  CanActivate,
  ActivatedRouteSnapshot,
  RouterStateSnapshot,
  UrlTree
} from '@angular/router';

// 2
canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<Boolean | UrlTree> | Promise<boolean | UrlTree> |
boolean | UrlTree {
    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/auth.guard';
```

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

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

const adminRoutes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    canActivate: [AuthGuard],
    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 { 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;
  }
}
```

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

```
// 1
import { Router } from '@angular/router';
import { AuthService } from './../services/auth.service';

// 2
constructor(
    private authService: AuthService,
    private router: Router
  ) {}

// 3
private checkLogin(url: string): boolean | UrlTree {
    if (this.authService.isLoggedIn) { return true; }

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

```
    // Navigate to the login page, return UrlTree
    return this.router.parseUrl('/login');
  }
```

5. Make changes to method **canActivate of AuthGuard.** Use the following snippet of code:

```
canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean | UrlTree>
| Promise <Boolean | UrlTree> | boolean | UrlTree{
    console.log('CanActivate Guard is called');
    const { url } = state;
    return this.checkLogin(url);
    return true;
  }
```

Task 23. Login Component

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

```
ng g c layout/components/login --skip-tests true -m layout.module
```

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

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

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

```
// 1
import { LoginComponent } from './components/login/login.component';

// 2
import { AboutComponent, LoginComponent, MessagesComponent, PathNotFoundComponent } from
'./components';
```

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

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

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

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

```
// 1
import { Component, OnInit } from '@angular/core';
import type { OnInit, OnDestroy } from '@angular/core';
import { Router } from '@angular/router';

// rxjs
import { Subject, takeUntil } from 'rxjs';

import { AuthService } from './../../../core';

// 2
export class LoginComponent implements OnInit, OnDestroy {

// 3
message!: string;
private unsubscribe: Subject<void> = new Subject();

// 4
constructor(
    public authService: AuthService,
    private router: Router
  ) { }
```

```
// 5
ngOnDestroy(): void {
    console.log('[takeUntil ngOnDestroy]');
    this.unsubscribe.complete();
}

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('[takeUntil] complete')
    };
    this.authService
      .login()
      // The TakeUntil subscribes and begins mirroring the source Observable.
      // It also monitors a second Observable that you provide.
      // If this second Observable emits an item or sends a termination notification,
      // the Observable returned by TakeUntil stops mirroring the source Observable and terminates.
      .pipe(takeUntil(this.unsubscribe))
      .subscribe(observer);
  }

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

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

// 6
ngOnInit(): void {
  this.setMessage();
}
```

6. 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>
```

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

```
// 1
import type {CanActivate, CanActivateChild, ActivatedRouteSnapshot, RouterStateSnapshot } from
'@angular/router';

// 2
export class AuthGuard implements CanActivate, CanActivateChild {
…
}

// 3
canActivateChild(next: ActivatedRouteSnapshot, state: RouterStateSnapshot) : Observable<boolean |
UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree{
       console.log('CanActivateChild Guard is called');
       const { url } = state;
      return this.checkLogin(url);
}
```

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

```
{
        path: '',
        canActivateChild: [AuthGuard],
        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';

// rxjs
import { 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 --implements CanDeactivate
```

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

```
import { Injectable } from '@angular/core';
import type { CanDeactivate, UrlTree } from '@angular/router';

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

import type { CanComponentDeactivate } from './../interfaces/can-component-deactivate.interface';

@Injectable({
  providedIn: 'root'
})
```

```
export class CanDeactivateGuard
  implements CanDeactivate<CanComponentDeactivate> {
  canDeactivate(
    component: CanComponentDeactivate
  ):
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree>
    | boolean
    | UrlTree {
    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 type { UrlTree } from '@angular/router';
import type { Observable, Subscription } from 'rxjs';
import { DialogService } from './../../../core';
import type { CanComponentDeactivate } from './../../../core';

// 2
export class UserFormComponent implements OnInit, OnDestroy, CanComponentDeactivate {

// 3
constructor(
    …
    private dialogService: DialogService
) { }

// 4
canDeactivate():
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree>
    | boolean
    | UrlTree {
    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 **UserResolveGuard.** Run the following command from command line:

```
ng g g users/guards/user-resolve --skip-tests true
```

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

```
export * from './user-resolve.guard';
```

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

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

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

```
import { Injectable } from '@angular/core';
import { Router,  } from '@angular/router';
import type { Resolve, ActivatedRouteSnapshot } from '@angular/router';
// rxjs
import { Observable, of, EMPTY, catchError, take, switchMap } from 'rxjs';

import { UserModel } from './../models/user.model';
import { UserArrayService } from './../services/user-array.service';

@Injectable({
  providedIn: 'any'
})
export class UserResolveGuard implements Resolve<UserModel> {
  constructor(
    private userArrayService: UserArrayService,
    private router: Router
  ) {}

  resolve(route: ActivatedRouteSnapshot): Observable<UserModel> {
    console.log('UserResolve Guard is called');

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

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

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

```
}
```

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

```
// 1
import { UserResolveGuard } from './guards';

// 2
{
        path: 'edit/:userID',
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard],
        resolve: {
          user: UserResolveGuard
        }
}
```

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

```
// 1
import type { OnInit, OnDestroy } from '@angular/core';
import type { Observable, Subscription } from 'rxjs';
import { map } from 'rxjs';

// 2
export class UserFormComponent implements OnInit, OnDestroy, CanComponentDeactivate {

// 3
private sub: Subscription;


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

    this.route.data.pipe(map(data => data.user)).subscribe((user: UserModel) => {
      this.user = { ...user };
      this.originalUser = { ...user };
    });

    // 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);

}

// 5
ngOnDestroy(): void {
    this.sub.unsubscribe();
}
```

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

```
ng g m widgets/spinner
ng g c widgets/spinner/spinner --skip-tests true --flat true --export true -m spinner.module
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. Make changes to **SpinnerModule**. Use the following snippet of code:

```
// 1
import { ModuleWithProviders, NgModule } from '@angular/core';
import { SpinnerService } from './spinner.service';

// 2
static forRoot(): ModuleWithProviders<SpinnerModule> {
    return {
      ngModule: SpinnerModule,
      providers: [SpinnerService]
    };
  }
```

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

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

5. 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);
  }
}
```

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

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

@Injectable()
export class SpinnerService {
  private visible = false;

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

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

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

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

```
// 1
import { SpinnerModule } from './widgets/spinner/spinner.module';

// 2
imports: [
    …
    SpinnerModule.forRoot(),
    AppRoutingModule
],
```

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

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

// 2
constructor(
    …
    public spinnerService: SpinnerService
  ) { }
```

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

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

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

```
// 1
```

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

// 2
constructor(
    …
    private spinner: SpinnerService
  ) {}

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

// 4
return this.userArrayService.getUser(id).pipe(
      delay(2000),
      switchMap((user: UserModel) => {
        if (user) {
          return of(user);
        } else {
          this.router.navigate(['/users']);
          return EMPTY;
        }
      }),
      take(1),
      catchError(() => {
        this.router.navigate(['/users']);
        return EMPTY
      }),
      finalize(() => this.spinner.hide())
    );
```

Task 28. Query Parameters and Fragment

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

```
// 1
import type {
  CanActivate,
  CanActivateChild,
  ActivatedRouteSnapshot,
  NavigationExtras,
  RouterStateSnapshot,
  UrlTree
} from '@angular/router';
// 2
private checkLogin(url: string): boolean | UrlTree {
    if (this.authService.isLoggedIn) { return true; }

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

    // Create a dummy session id
    const sessionId = 123456789;

    const navigationExtras: NavigationExtras = {
      queryParams: { sessionId },
      fragment: 'anchor'
    };

    // Navigate to the login page with extras
    this.router.navigate(['/login'], navigationExtras);
    return false;
    return this.router.parseUrl('/login');
  }
```

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

```
// 1
import 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, OnInit } from '@angular/core';
import type { OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

// rxjs
import { Observable, map } from 'rxjs';

// 2
sessionId!: Observable<string>;
token!: Observable<string>;

// 3
constructor(
    private route: ActivatedRoute
  ) { }

// 4
ngOnInit(): void {
    // Capture the session ID if available
    this.sessionId = this.route
      .queryParamMap
      .pipe(
        map(params => params.get('sessionId') || 'None')
      );

    // 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: [AuthGuard],
    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. canLoad Guard

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

```
// 1
import type {
  CanActivate, CanActivateChild, CanLoad, Router, Route,
  ActivatedRouteSnapshot, RouterStateSnapshot, NavigationExtras, UrlTree, UrlSegment
} from '@angular/router';


// 2
export class AuthGuard implements CanActivate, CanActivateChild, CanLoad {

// 3
canLoad(route: Route, segments: UrlSegment[] ): Observable<boolean | UrlTree> | Promise<boolean |
UrlTree> | boolean | UrlTree {
    console.log('CanLoad Guard is called');
    const url = `/${route.path}`;
    return this.checkLogin(url);
}
```

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

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

// 2
  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)
  },
```

Task 31. Default Preloading Strategy

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

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


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

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

Task 32. Custom Preloading Strategy

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

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

// rxjs
import { 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 {RouterModule, PreloadAllModules } from '@angular/router';
import { AuthGuard, CustomPreloadingStrategyService } from './core';
// 2
const extraOptions: ExtraOptions = {
  preloadingStrategy: PreloadAllModules CustomPreloadingStrategyService,
  // 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 type { OnInit } from '@angular/core';
import { MessagesService, CustomPreloadingStrategyService } from './core';

// 2
```

```
export class AppComponent implements OnInit {

// 3
constructor(
    …
    private preloadingStrategy: CustomPreloadingStrategyService
) {}

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

Task 33. Router Events and Title Service

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

```
const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent,
    data: { title: 'About' }
  },
  {
    path: 'login',
    component: LoginComponent,
    data: { title: 'Login' }
  },
  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule),
    data: { title: 'Admin' }
  },
  {
    path: 'users',
    loadChildren: () => import('./users/users.module').then(m => m.UsersModule),
    data: {
      preload: true,
      title: 'Users'
    }
  },
  {
    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,
    data: { title: 'Page Not Found' }
  }
];
```

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

```
const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent,
    data: { title: 'Task Manager'}
  },
  {
    path: 'edit/:taskID',
    component: TaskFormComponent
  }
];
```

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

```
// 1
import type { OnInit, OnDestroy } from '@angular/core';
```

```
import { Title } from '@angular/platform-browser';
import { Router, NavigationEnd, NavigationStart } from '@angular/router';

// rxjs
import { Subscription, filter, map, switchMap } from 'rxjs';

// 2
export class AppComponent implements OnInit, OnDestroy

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

  constructor(
    …
    private titleService: Title
  ) { }

private setPageTitles(): void {
    this.sub.navigationEnd = this.router.events
      .pipe(

        filter(event => event instanceof NavigationEnd),
        map(() => this.router.routerState.root),
        map(route => {
          while (route.firstChild) {
            route = route.firstChild;
          }
          return route;
        }),
        filter(route => route.outlet === 'primary'),
        switchMap(route => route.data)
      )
      .subscribe(
         data => this.titleService.setTitle(data.title)
      );
 }

 ngOnInit(): viod {
    …
    this.setPageTitles();
  }

  ngOnDestroy(): void {
    this.sub.navigationEnd.unsubscribe();
  }
```

Look to page titles.

Another way:

```
// 4
onActivate($event: any, routerOutlet: RouterOutlet): void {
    console.log('Activated Component', $event, routerOutlet);
    // another way to set titles
    this.titleService.setTitle(routerOutlet.activatedRouteData.title);
}

// 5
ngOnInit(): void {
```

```
    …
    // this.setPageTitles();
  }

ngOnDestroy(): void {
    // this.sub.navigationEnd.unsubscribe();
}

// 6
private setMessageServiceOnRefresh(): void {
    this.sub.navigationStart = this.router.events
      .pipe(filter(event => event instanceof NavigationStart))
      .subscribe(event => {
        this.messagesService.isDisplayed = (event as NavigationStart).url.includes('messages:');
      });

}

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

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

Task 34. 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: { title: 'Task Manager'}
    data: {
      title: 'Task Manager',
      meta: metaTags

    }
  },
];
```

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

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

// 2
constructor(
    …
    private metaService: Meta
  ) { }

// 3
this.titleService.setTitle(routerOutlet.activatedRouteData.title);
this.metaService.addTags(routerOutlet.activatedRouteData.meta);
```