Angular 2+

Workshop. Routing.

Contents

Explanation of Colors

Black color is a snippet of code that you need to fully use to create a new file, and in combination with green or red, the snippet of code that was added earlier.

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

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

```
>ng g c components/home² --spec=false --skip-import=true³
>ng g c components/about --spec=false --skip-import=true
>ng g c components/page-not-found --spec=false --skip-import=true
```

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

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

---

[1] All commands run from project root folder.

[2] Routed components don't use selector. You can leave or remove selector for such components.

[3] The meaning of the flags is following

--spec=false – don't generate unit tests files.

--skip-import – don't register components in a module.

Task 03. Routes Config

1. Create file **app/app.routing.module.ts[4]**. Add the following snippet of code to it.

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

import { AboutComponent, HomeComponent, PageNotFoundComponent } from './components';

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: PageNotFoundComponent
  }
];

export let appRouterComponents = [AboutComponent, HomeComponent, PageNotFoundComponent];

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

---

[4] According to styleguide, the name of this file is **app-routing.module.ts**, and the name of class is AppRoutingModule. But I have changed this name to **app.routing.module.ts** to place this file near the file **app.module.ts**.

Task 04. Import Routes

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

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

// 2
declarations: [
    AppComponent,
    appRouterComponents
],
imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
]

// 3
constructor(router: Router) {
    console.log('Routes: ', JSON.stringify(router.config, undefined, 2));
}
```

Task 05. <router-outlet>

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

```html
<div class="container">
    <router-outlet
            (activate)='onActivate($event)'
            (deactivate)='onDeactivate($event)'>
    </router-outlet>
    <!-- Routed views go here -->
</div>
```

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

```
onActivate($event) {
   console.log('Activated Component', $event);
}

onDeactivate($event) {
   console.log('Deactivated Component', $event);
}
```

Task 06. routerLink

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

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

// 2
<a routerLink="/about">About</a>
```

Task 07. routerLinkActive

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

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

Task 08. Task Feature Module

1. Create module **TasksModule.** Run the following command from command line:

>ng g m tasks

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

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

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

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

>ng g cl tasks/models/task.model --spec=false

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

```
export class Task {
      constructor(
            public id: number,
            public action: string,
            public priority: number,
            public estHours: number,
            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 --spec=false -m=tasks

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

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

import { Task } from './../models/task.model';

const taskList = [
  new Task(1, 'Estimate', 1, 8, 8, true),
  new Task(2, 'Create', 2, 8, 4, false),
```

```
    new Task(3, 'Deploy', 3, 8, 0, false)
];

const taskListPromise = Promise.resolve(taskList);

@Injectable()
export class TaskArrayService {

  getTasks(): Promise<Task[]> {
    return taskListPromise;
  }

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

  addTask(task: Task): void {
    taskList.push(task);
  }

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

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

  completeTask(task: Task): void {
    task.done = true;
  }
}
```

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

app>ng g c tasks/task-list --spec=false

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

```
import { Component, OnInit } from '@angular/core';

import { Task } from './../models/task.model';
import { TaskArrayService } from './../services/task-array.service';

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

  constructor(
    private taskArrayService: TaskArrayService) { }

  ngOnInit() {
    this.getTasks().catch(err => console.log(err));
```

12

```
  }

  completeTask(task: Task): void {
    this.taskArrayService.completeTask(task);
  }

  editTask(task: Task): void {

  }

  private async getTasks() {
    this.tasks = await this.taskArrayService.getTasks();
  }
}
```

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

```
<app-task
  *ngFor="let task of tasks"
  [task]="task"
  (complete)="completeTask($event)"
  (edit)="editTask($event)">
</app-task>
```

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

>ng g c tasks/task --spec=false -cd=OnPush

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

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

import { Task } from './../models/task.model';

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

  @Output() complete = new EventEmitter<Task>();
  @Output() edit = new EventEmitter<Task>();

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

  editTask() {
    this.edit.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>
```

```html
<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)="completeTask()"
        [disabled]="task.done">
        Done
    </button>
    <button class="btn btn-warning btn-sm"
        (click)="editTask()">
        Edit
    </button>
</div>
</div>
```

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

```typescript
export * from './services/task-array.service';

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

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

```typescript
import { TaskListComponent } from './task-list/task-list.component';
import { TaskComponent } from './task/task.component';
import { TaskArrayService } from './services/task-array.service';

import { TaskListComponent, TaskComponent, TaskArrayService } from '.';
```

Task 09. Tasks Feature Route Configuration

1. Create file **tasks/tasks.routing.module.ts.** Use the following snippet of code:

```typescript
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TaskListComponent } from '.';

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

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

Task 10. Register Task Feature Routing

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

```
// 1
import { TasksRoutingModule } from './tasks.routing.module';


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

Task 11. Register Tasks Feature Module

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

```
// 1
import { TasksModule } from './tasks/tasks.module';

// 2
imports: [
    BrowserModule,
    FormsModule,
    TasksModule,
    AppRoutingModule
],
```

Task 12. 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, PageNotFoundComponent } from './components';
{
    path: 'home',
    component: HomeComponent
},

// 2
export let appRouterComponents = [AboutComponent, HomeComponent, PageNotFoundComponent];
```

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

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

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

Task 13. Navigate

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

> ng g c tasks/task-form --spec=false --skip-import=true

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

```
import { Component, OnInit } from '@angular/core';

import { Task } 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: Task;

  constructor(
    private taskArrayService: TaskArrayService,
  ) { }

  ngOnInit(): void {
    this.task = new Task(null, '', null, null);
  }

  saveTask() {
    const task = {...this.task};

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

  goBack(): 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)="saveTask()" 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
```

```html
                    [(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)="goBack()">Back
      </button>
    </form>
  </div>
</div>
```

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

```typescript
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:

```typescript
import { TaskListComponent, TaskComponent, TaskFormComponent, TaskArrayService } from '.';
```

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

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

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

```
  {
    path: 'edit/:id',
    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
editTask(task: Task) {
    const link = ['/edit', task.id];
    this.router.navigate(link);
}
```

Task 14. Getting the route parameter

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

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

// rxjs
import { switchMap } from 'rxjs/operators';

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

// 3
ngOnInit(): void {
    this.task = new Task(null, '', null, null);

    // it is not necessary to save subscription to route.paramMap
    // it handles automatically
    this.route.paramMap
      .pipe(
        switchMap((params: Params) => this.taskArrayService.getTask(+params.get('id'))))
      .subscribe(
        task => this.task = {...task},
        err => console.log(err)
      );
  }
```

Task 15. Navigate Back

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

```
// 1
import { Location } from '@angular/common';

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

// 3
goBack(): void {
    this.location.back();
  }

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

this.goBack();
```

Task 16. Secondary Router Outlet

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

```html
<div class="container">
    <router-outlet
        (activate)='onActivate($event)'
        (deactivate)='onDeactivate($event)'>
    </router-outlet>
    <!-- Routed views go here -->
</div>

<div class="container">
    <div class="col-md-10">
        <router-outlet
            (activate)='onActivate($event)'
            (deactivate)='onDeactivate($event)'>
        </router-outlet>
        <!-- Routed views go here -->
    </div>
    <div class="col-md-2">
        <router-outlet name="popup"></router-outlet>
    </div>
 </div>
```

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

```
>ng g m core -m=app --spec=false
```

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

```
import { NgModule, Optional, SkipSelf } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: []
})
export class CoreModule {
  constructor(@Optional() @SkipSelf() parentModule: CoreModule) {
    if (parentModule) {
      throw new Error(`CoreModule is already loaded. Import it in the AppModule only.`);
    }
  }
 }
```

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

```
>ng g s core/services/messages --spec=false -m=core
```

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

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

@Injectable()
```

```
export class MessagesService {
  isDisplayed = false;

  private messages: string[] = [];

  constructor() { }

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

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

  clearMessageList(): void {
    this.messages.length = 0;
  }
}
```

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

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

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

>ng g c core/components/messages --spec=false -m=core

8. Replce 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)="close()">x</a>
  </span>
</div>
<div *ngFor="let message of messagesService.getMessages(); let i=index">
  <div class="message-row">
    {{message}}
  </div>
</div>
```

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

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

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

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

import { MessagesService } from './../../services';

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

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

  ngOnInit() {
  }

  close() {
    this.router.navigate([{ outlets: { popup: null } }]);
    this.messagesService.isDisplayed = false;
  }
}
```

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

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

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

```
import { MessagesComponent } from './components/messages/messages.component';
import { MessagesService } from './services/messages.service';
```

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

```
// 1
import { MessagesComponent } from './core/components';

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

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

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

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

15. 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: {popup: ['messages']}}]">
    Show Messages
  </button>
  <router-outlet name="popup"></router-outlet>
</div>

Look at the result.
```

16. 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
displayMessages(): void {
    this.router.navigate([{ outlets: { popup: ['messages'] } }]);
    this.messagesService.isDisplayed = true;
  }
```

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

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

Task 17. Users Components

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

```
>ng g m users
```

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

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

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

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

```
>ng g cl users/models/user.model --spec=false
```

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

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

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

```
>ng g s users/services/user-array --spec=false -m=users
```

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

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

// rxjs
import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/Observable/of';
import { map, catchError } from 'rxjs/operators';

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

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

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

@Injectable()
export class UserArrayService {
  getUsers(): Observable<User[]> {
    return userListObservable;
  }

  getUser(id: number | string): Observable<User> {
    return this.getUsers()
      .pipe(
        map((users: Array<User>) => users.find(user => user.id === +id)),
        catchError(err => Observable.throw('Error in getUser method'))
      );
  }

  addUser(user: User): void {
    userList.push(user);
  }

  updateUser(user: User): 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/user-list --spec=false --skip-import

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

```
import { Component, OnInit } from '@angular/core';

// rxjs
import { Observable } from 'rxjs/Observable';
import { catchError } from 'rxjs/operators';

import { User } 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<User>>;

  constructor(
    private userArrayService: UserArrayService,
  ) { }

  ngOnInit() {
    this.users$ = this.userArrayService.getUsers();
  }
}
```

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

```html
<app-user
   *ngFor="let user of users$ | async"
   [user]="user">
</app-user>
```

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

>ng g c users/user --spec=false -cd=OnPush

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

```typescript
import { Component, Input } from '@angular/core';

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

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent {
  @Input() user: User;

  editUser() {
  }
}
```

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

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

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

app>ng g c users/user-form --spec=false --skip-import

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

```typescript
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params } from '@angular/router';

// rxjs
import { switchMap } from 'rxjs/operators';

import { User } 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: User;
  originalUser: User;

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

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

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

  saveUser() {
    const user = {...this.user};

    if (user.id) {
      this.userArrayService.updateUser(user);
    } else {
      this.userArrayService.addUser(user);
    }
    this.originalUser = {...this.user};
  }

  goBack() {
  }
}
```

15.   Make changes to **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)="saveUser()" id="user-form" #form="ngForm">
          <div class="form-group">
            <label for="firstName">First Name</label>
            <input type="text"
                class="form-control"
```

```html
              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"
              form="user-form">Save
        </button>
        <button class="btn btn-primary"
              type="button"
              (click)="goBack()">Back
        </button>
        </form>
    </div>
</div>
```

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

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

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

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

```typescript
import { UserArrayService } from './services/user-array.service';

import { UserComponent } from './user/user.component';
import { UserComponent, UserArrayService } from '.';
```

Task 18. Users Feature Area

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

>ng g c users/users --spec=false --flat --skip-import

2. Remove selector from meta data of **UsersComponent.**
3. Make changes to **UsersComponent template.** Use the following snippet of HTML:

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

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

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

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

```
// 1
import { TasksModule } from './tasks/tasks.module';
import { UsersModule } from './users/users.module';

// 2
imports: [
    BrowserModule,
    FormsModule,
    CoreModule,
    TasksModule,
    UsersModule,
    AppRoutingModule
  ],
```

Task 19. Users Nested Routing

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

```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:

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

3. Create **UsersRoutingModule**. Use the following snippet of code:

```typescript
import { NgModule }  from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { UsersComponent, UserListComponent, UserFormComponent } from '.';

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

export let usersRouterComponents = [UsersComponent, UserListComponent, UserFormComponent];

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

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

```typescript
import { UsersRoutingModule, usersRouterComponents } from './users.routing.module';
```

```
imports: [
    CommonModule,
    FormsModule,
    UsersRoutingModule
],
declarations: [
    usersRouterComponents,
    UserComponent,
]
```

Task 20. Relative Navigation

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

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


// 2
@Output() edit = new EventEmitter<User>();

// 3
editUser() {
    this.edit.emit(this.user);
}
```

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

```
<app-user
  *ngFor='let user of users$ | async'
  [user]="user"
  (edit)="editUser($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
editUser(user: User) {
    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, Params, 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.addUser(user);
    }
    this.originalUser = {...this.user};
    this.goBack();

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

Task 21. Optional Parameters

1. Make changes to the method **saveUser** 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.addUser(user);
      this.goBack();
    }
this.originalUser = {...this.user};
this.goBack();
```

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

```
// 1
import { ActivatedRoute, Params } from '@angular/router';
import { catchError, switchMap } from 'rxjs/operators';


// 2
private editedUser: User;


// 3
constructor(

    …
    private route: ActivatedRoute
  ) { }

// 4
ngOnInit() {
    this.userArrayService.getUsers()
          .then(users => this.users = [...users])
          .catch(err => console.log(err));

    this.route.paramMap
      .pipe(
        switchMap((params: Params) => this.userArrayService.getUser(+params.get('editedUserID')))
      )
      .subscribe(
        (user: User) => {
          this.editedUser = {...user};
          console.log(`Last time you edited user ${JSON.stringify(this.editedUser)}`);
        },
        err => console.log(err)
      );
  }

// 5
isEdited(user: User) {
    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'
  [user]="user"
  (edit)="editUser($event)"
  [class.edited]="isEdited(user)">
</app-user>
```

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

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

Task 22. Admin Feature Area

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

```
>ng g m admin --routing=true -m=app
```

2. Rename the file **admin/admin-routing.modue.ts** to admin/admin.routing.module.ts.

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

Run the following commands from command line:

```
app> ng g c admin/admin-dashboard --spec=false --skip-import
app> ng g c admin/manage-tasks --spec=false --skip-import
app> ng g c admin/manage-users --spec=false --skip-import
app> ng g c admin/admin --spec=false --flat --skip-import
```

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

```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>
```

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

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

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

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

import { AdminComponent, AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent } from
'.';

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

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

export let adminRouterComponents = [AdminComponent, AdminDashboardComponent, ManageTasksComponent,
ManageUsersComponent];

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

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

```
// 1
import { AdminRoutingModule, adminRouterComponents } from './admin.routing.module';

// 2
declarations: [
    adminRouterComponents
],
imports: [
    CommonModule,
    AdminRoutingModule
]
```

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>
```

Step_23. canActivate Guard

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

```
>ng g g core/guards/auth --spec=false -m=core
```

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

```
canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
    console.log('CanActivateGuard is called');
    return true;
  }
```

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

```
import { AuthGuard } from './../core/guards/auth.guard';

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

Task 24. Auth Service

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

```
ng g s core/services/auth --spec=false -m=core
```

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

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

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

```
import { AuthService } from './services/auth.service';
import { AuthService, MessagesService } from './services';
```

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

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

import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';
import { delay, tap } from 'rxjs/operators';

@Injectable()
export class AuthService {
  isLoggedIn = false;

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

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

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

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

```
//
import {CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';
import { AuthService } from './../services/auth.service';

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

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

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

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

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

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

Task 25. Login Component

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

```
>ng g c core/components/login --spec=false -m=core
```

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

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

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

```
// 1
import { MessagesComponent, LoginComponent } from './core/components';

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

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

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

// 2
message: string;

// 3
constructor(
    public authService: AuthService,
    public router: Router
  ) { }

// 4
private setMessage() {
    this.message = 'Logged ' + (this.authService.isLoggedIn ? 'in' : 'out');
  }

  login() {
    this.message = 'Trying to log in ...';
    this.authService.login().subscribe(() => {
      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]);
      }
    });
  }
```

```
  logout() {
    this.authService.logout();
    this.setMessage();
  }
```

```
// 5
ngOnInit() {
  this.setMessage();
}
```

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

```
<h2>LOGIN</h2>
<p>State: {{message}}</p>
<p>
  <button (click)="login()"  *ngIf="!authService.isLoggedIn">Login</button>
  <button (click)="logout()" *ngIf="authService.isLoggedIn">Logout</button>
</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 26. canActivateChild Guard

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

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

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

// 3
canActivateChild(next: ActivatedRouteSnapshot, state: RouterStateSnapshot) : Observable<boolean> |
Promise<boolean> | boolean {
      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 27. canDeactivate Guard

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

>ng g m shared -m=app

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

>ng g s shared/services/dialog --spec=false -m=shared

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

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

@Injectable()
export class DialogService {

  confirm(message?: string): Promise<boolean> {
    return new Promise<boolean>(resolve => {
      resolve(window.confirm(message || 'Is it OK?'));
    });
  }
}
```

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

>ng g i shared/interfaces/can-component-deactivate.interface

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

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

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

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

>ng g g shared/guards/can-deactivate --spec=false -m=shared

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

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

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

@Injectable()
export class CanDeactivateGuard implements CanDeactivate<CanComponentDeactivate> {
  canDeactivate(component: CanComponentDeactivate) {
    return component.canDeactivate();
  }
}
```

8. Create file **shared/index.ts**. Use the following snippet of code:

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

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

```
// 1
import { Observable }    from 'rxjs/Observable';
import { DialogService, CanComponentDeactivate } from './../../shared';

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

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

// 4
canDeactivate(): Observable<boolean> | Promise<boolean> | boolean {
    const flags = Object.keys(this.originalUser).map(key => {
      if (this.originalUser[key] === this.user[key]) {
        return true;
      }
      return false;
    });

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

    return this.dialogService.confirm('Discard changes?');
}
```

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

```
import { CanDeactivateGuard } from './../shared';

{
        path: 'edit/:id',
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard]
}
```

Task 28. resolve Guard

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

>ng g g users/guards/user-resolve --spec=false -m=users

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

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

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

```
import { UserComponent, UserArrayService, UserResolveGuard } from '.';
import { UserResolveGuard } from './guards/user-resolve.guard';
```

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

```
import { Injectable } from '@angular/core';
import { Router, Resolve, ActivatedRouteSnapshot } from '@angular/router';

// rxjs
import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';
import { catchError } from 'rxjs/operators';

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

@Injectable()
export class UserResolveGuard implements Resolve<User> {

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

  resolve(route: ActivatedRouteSnapshot): Observable<User> | null {
    console.log('UserResolve Guard is called');
    const id = +route.paramMap.get('userID');

    if (id) {
      return this.userArrayService.getUser(id)
      .pipe(
        catchError(() => {
          this.router.navigate(['/users']);
          return of(null);
        })
      );
    } else {
      return of(new User(null, '', ''));
    }
  }
}
```

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

```
// 1
import { UserResolveGuard } from './../guards/user-resolve.guard';
import { UsersComponent, UserListComponent, UserFormComponent, UserResolveGuard } from '.';
```

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

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

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

    this.route.data.subscribe(data => {
      this.user = {...data.user};
      this.originalUser = {...data.user};
    });

    const id = +this.route.snapshot.paramMap.get('id');
      this.userArrayService.getUser(id)
        .then(user => {
          this.user = Object.assign({}, user);
          this.originalUser = Object.assign({}, user);
        })
        .catch(err => console.log(err));

}
```

Task 29. Query Parameters and Fragment

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

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


// 2
private checkLogin(url: string): boolean {
    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: { 'session_id': sessionId },
      fragment: 'anchor'
    };

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

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

```
// 1
import { Router, 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 { ActivatedRoute } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import { map } from 'rxjs/operators';

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

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

// 4
ngOnInit() {
    // Capture the session ID if available
    this.sessionId = this.route
      .queryParamMap
      .pipe(
        map(params => params.get('session_id') || '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>Dashboard</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./tasks" queryParamsHandling="preserve" preserveFragment>Manage Tasks</a>
    </li>
    <li routerLinkActive="active">
      <a routerLink="./users" queryParamsHandling="preserve" preserveFragment>Manage Users</a>
    </li>
  </ul>
</nav>
```

Task 30. Lazy-Loading Route Configuration

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

```
{
  path: 'admin',
  loadChildren: 'app/admin/admin.module#AdminModule'
},
{
  path: 'users',
  loadChildren: 'app/users/users.module#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 31. canLoad Guard

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

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


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

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

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

```
export * from './components';
export * from './guards/auth.guard';
export * from './services';
```

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

```
// 1
import { MessagesComponent, LoginComponent, AuthGuard } from './core/components';

// 2
  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: 'app/admin/admin.module#AdminModule'
  },
```

Task 32. Default Preloading Strategy

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

```
// 1
import { Routes, RouterModule, PreloadAllModules, 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 33. Custom Preloading Strategy

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

```
{
    path: 'users',
    loadChildren: 'app/users/users.module#UsersModule',
    data: { preload: true }
},
```

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

>ng g s core/services/custom-preloading-strategy --spec=false -m=core

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

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

// rxjs
import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';

@Injectable()
export class CustomPreloadingStrategyService implements PreloadingStrategy {
  private preloadedModules: string[] = [];

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

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

```
import { CustomPreloadingStrategyService } from './services/custom-preloading-strategy.service';
import { AuthService, MessagesService, CustomPreloadingStrategyService } from './services';
```

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

```
// 1
import { Routes, RouterModule, PreloadAllModules, ExtraOptions } from '@angular/router';
import { MessagesComponent, LoginComponent, AuthGuard, CustomPreloadingStrategyService } from
'./core';

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

Task 34. 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: 'app/admin/admin.module#AdminModule',
    data: { title: 'Admin' }
  },
  {
    path: 'users',
    loadChildren: 'app/users/users.module#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: PageNotFoundComponent,
    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/:id',
    component: TaskFormComponent
  }
];
```

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

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

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

// rxjs
import { Subscription } from 'rxjs/Subscription';

// 2
export class AppComponent implements OnInit, OnDestroy

// 3
private sub: Subscription;

  constructor(
    public messagesService: MessagesService,
    private titleService: Title,
    private router: Router
  ) { }

  ngOnInit() {
    this.setPageTitles();
  }

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

private setPageTitles() {
    this.sub = 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'])
      );
  }
```

Task 35. Meta Service (Meta only available in 4.X)

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

```
const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent,
    data: { title: 'Task Manager'}
    data: {
      title: 'Task Manager',
      meta: [{
        name: 'description',
        content: 'Task Manager Application. This is an ASP application'
      },
      {
        name: 'keywords',
        content: 'Angular 4 tutorial, SPA Application, Routing'
      }]
    }
  },
];
```

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

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

// 2
constructor(
    public messagesService: MessagesService,
    private titleService: Title,
    private metaService: Meta,
    private router: Router
  ) { }

// 3
Rename method setPageTitles to setPageTitlesAndMeta

// 4
ngOnInit() {
  this.setPageTitles();
  this.setPageTitlesAndMeta();
}

// 5
.subscribe(
  data => this.titleService.setTitle(data['title'])
  data => {
      this.titleService.setTitle(data['title']);
      this.metaService.addTags(data['meta']);
  }
);
```