Mobile Application Development

LECTURE 8

Ionic Routing and Navigation

- Navigation is the process of moving from one page to another.
- In any mobile app, you will have many screens/pages, and you will have to navigate to those pages.
- Sometimes, you will need to do different tasks when the page opens. You might need pass data from one page/screen to another.

Ionic Routing and Navigation

- Ionic 4 uses the powerful routing of Angular.

- Here's an example of our routing file (app-routing.module.ts)

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then(m => m.HomePageModule)
  },
  {
    path: 'list',
    loadChildren: () => import('./list/list.module').then(m => m.ListPageModule)
  }
];
```

Ionic Routing and Navigation

- Lets go through our previous examples. We displayed a studentslist in previous lectures. Now, If I select a student, I want to display complete detail of that student.

- For that, lets create a new page in Ionic named **students.**

- Here's how the routing page looks like

Ionic Routing and Navigation

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then(m => m.HomePageModule)
  },
  {
    path: 'list',
    loadChildren: () => import('./list/list.module').then(m => m.ListPageModule)
  },
  { path: 'studentslist', loadChildren: './studentslist/studentslist.module#StudentslistPageModule' },
  { path: 'student', loadChildren: './student/student.module#StudentPageModule' }
];
```

Ionic Routing and Navigation

- Lets change the routing according to our requirements.

- this will help us use urls like

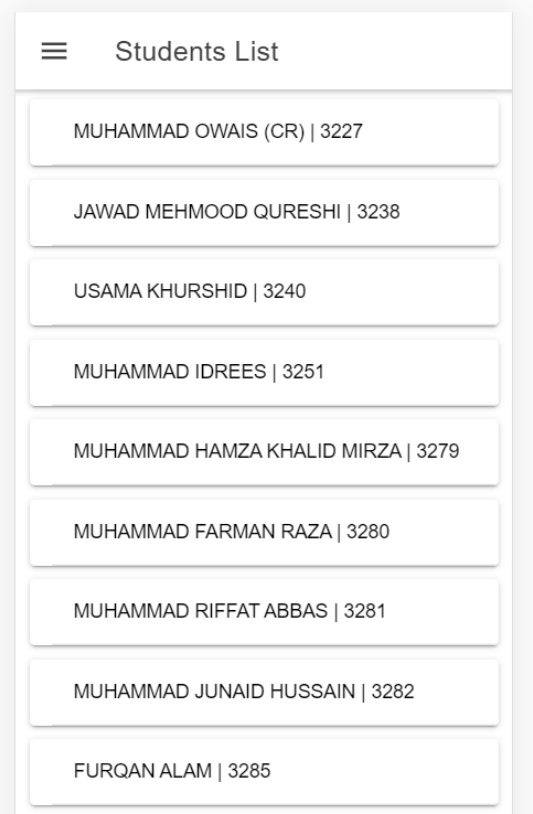
http://localhost:8100/studentslist/3198

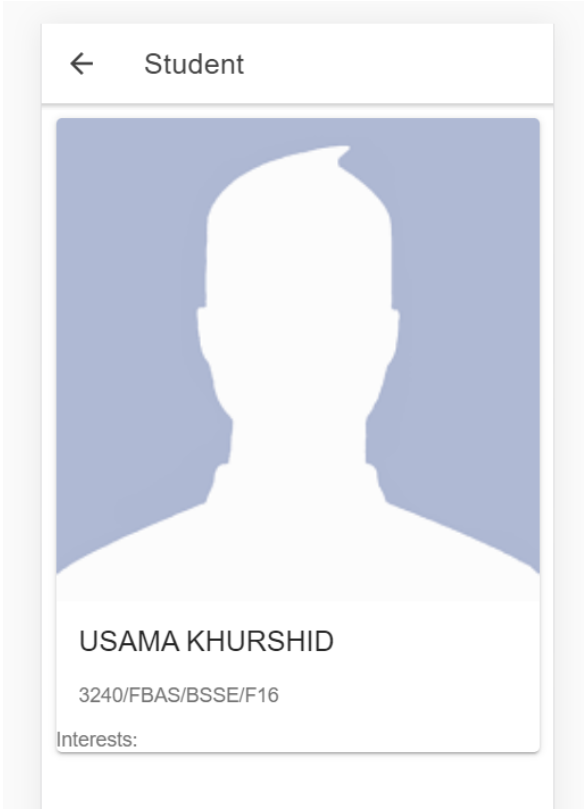
http://localhost:8100/studentslist/31989

```
{
  path: 'home',
  loadChildren: () => import('./home/home.module').then(m => m.HomePageModule)
},
{
  path: 'studentslist',
  children: [
    {
      path: '',
      loadChildren:
        './studentslist/studentslist.module#StudentslistPageModule'
    },
    {
      path: ':studentid',
      loadChildren: './student/student.module#StudentPageModule'
    }
  ]
}
];
```

Ionic Routing and Navigation

- Here is what it would look like (link to code)





Ionic Routing and Navigation

- Routing helps us manage complexity in big applications. It helps us think better in terms of the app we're going to build.

- How did we navigate to one page to another from the List?

- **RouterLink**

RouterLink

- RouterLink is a directive that helps you navigate to a page from the html. Here, we are telling it to take us to studentslist/:id whenever its clicked. **[property Binding]**

```
<ion-card *ngFor="let item of students">
  <ion-item [routerLink]="['/studentslist', item.id]">
    <ion-card-content>
      <ion-label>{{ item?.name }} | {{ item?.id.substring(0,4) }}
      </ion-label>
    </ion-card-content>
  </ion-item>
</ion-card>
```

Ionic Routing and Navigation

- There are also other ways you can navigate to another page. You can do it from .TS / component file as well.

- Angular provides us two methods (**navigate** and **navigateByUrl**).

```
<ion-card *ngFor="let item of students">
  <ion-item (click)="changeUrl(item)">
    <ion-card-content>
      <ion-label>{{ item?.name }} | {{ item?.id.substring(0,4) }}
      </ion-label>
    </ion-card-content>
  </ion-item>
</ion-card>      You, an hour ago • set up lecture 8
```

Ionic Routing and Navigation

- Lets create a **changeUrl()** function in component. **navigateByUrl** takes a string while **navigate**

takes an Array. To use this.router, we need to import Router and inject in constructor.

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

constructor(private *router*: Router) {}

```
changeUrl(user) {
  const id = user.id.substring(0, 4);
  const url = `studentslist/${id}`;

  this.router.navigateByUrl(url);

  // or

  // this.router.navigate([url]);
}
```

“Cal Newport is an author of 6 books including “Deep Work” and “So Good They Can’t Ignore You”, He has a PhD in CS from MIT, and is an associate professor of computer science at Georgetown University.

He has to work on his research, write books, take classes etc. And after 5p.m, he doesnot uses the computer anymore. How does he manage it?

The point is, if we look at successful people, we should always find out what we can learn from them.

Router-Outlet

- The Router-Outlet is a directive exported by RouterModule and acts as a placeholder that indicates to the router where it needs to insert the matched component(s) or the page or the module.

- For each Routing file or module, if it has children or child components, it will need to have a router-outlet or ion-router-outlet to tell them where to load the component.

Ionic Routing and Navigation

- Main ion-router-outlet in the app.component.html file. Router Outlet basically tells angular where in the html to load the child components of the route.

```
<ion-app>
  <ion-split-pane>
    <ion-menu type="overlay">
      <ion-header>
        <ion-toolbar>
          <ion-title>Menu</ion-title>
        </ion-toolbar>
      </ion-header>
      <ion-content>
        <ion-list>
          <ion-menu-toggle auto-hide="false" *ngFor="let p of appPages">
            <ion-item [routerDirection]="'root'" [routerLink]="[p.url]">
              <ion-icon slot="start" [name]="p.icon"></ion-icon>
              <ion-label>
                {{p.title}}
              </ion-label>
            </ion-item>
          </ion-menu-toggle>
        </ion-list>
      </ion-content>
    </ion-menu>
    <ion-router-outlet main></ion-router-outlet>
  </ion-split-pane>
</ion-app>
```

Route Guards

- A route guard is a feature of the Angular Router that allows developers to run some logic when a route is requested, and based on that logic, it allows or denies the user access to the route. It's commonly used to check if a user is logged in and has the authorization before he can access a page.

- Route guards enables you to allow or disallow access to your specific application routes

Route Guards

- Here, we are protecting a path (admin), in **isLoginGuard**, we have a custom logic which checks if the user is logged in or not. If not, then it won't open that route.

```
{
  path: 'admin',
  canActivate: [isLoginGuard],
  loadChildren: './admin/admin.module#AdminModule'
},
```