Project Report

On

Skill Worker Recommendation System

PG DIPLOMA IN ADVANCED COMPUTING

(DAC) C-DAC, BANGALORE.

Submitted By – **Ajinkya Sagane (200250120003)**

Bharat Sharma (200250120017)

Mohit Potode (200250120053)

Mayank Attri (200250120050)

Project Flow

1. In **Home** page there's 1 search box where user can search cities based on the service name if service is available in that particular city it'll show the list of cities and by clicking on that it'll redirect to the login page, otherwise it'll show service is not found.
2. In **Registration** page there're 2 options one is to register as customer and another as vendor. By selecting particular option it'll route to the appropriate component.
3. In **Login** page, there are 3 kinds of login customer, vendor and admin. Based on the roles it'll redirect to the different components.
4. In **customer profile** page there are 3 search boxes for state, city, skills search, based on that it'll show the list of vendor details with view option in table format.
5. By clicking on view button it'll navigate to booking page where we've to select time and date and it'll show the selected information and by clicking on hire button request will send to the vendor and Email also will be send with the information about request.
6. There's one button called update by clicking on it user can see his data and update it.
7. In **Vendor Profile** page there's one button called check request, it'll show the list of new requests vendor has got in table.
8. There's one button called update by clicking on it vendor can see his data and update it.
9. In **Admin profile** there's 2 option one for vendor and another for customer where admin can see and manage all activities.
10. Logout button will redirect to the Home page.

Scenario (Update Page)

1. In **presentation layer** there are **2 scenarios** for updates one for customer and another for vendor.
2. In the both scenarios, we required different views (HTML) for this purpose we define **1 parent component** and **2 child components**.
3. Here the parent component named as parent-update helps us to know which user is currently logged in using session (Mobile No.).
4. With the help of **mobile no**. we are making an **Ajax call** in which we're passing mobile no. as an argument, and this mobile no. is forwarded to the **service layer** of **angular**.
5. From **service layer** we're calling a **RestBase Service** name as **GetDeatilsBasedOnMobileno** which is of type **get Mapping**, this RestBase service fetch the user details based on **unique key** (mobile No.) and it'll return **object** of **Myobject** type.
6. We've already created one **class** named as MyObject which contains **2 reference** type of variables of type **customer** and **vendor**.
7. Now above RestBase service calls the **getDetailsOfUser function** where we're passing **input argument** as **mobile no**.
8. We've created 1 empty object of MyObject class & then we're calling **interfaces** of **customer** and **vendor class** name as custInterface and venInterface respectively.
9. From these interfaces we're accessing the **implementation class** methods of customer & vendor where mobile no. has passed as an argument to the both methods.
10. Now both functions will call their **JPA Repositories** to fetch the user details based on mobile no. , for this we've already written a **custom query** in both **repositories**

11. Based on the above **query** it'll set the **object** values whichever is not equal to null and return this object to the **service** of **presentation layer**.
12. If the **Ajax** call is **succeed** then it will check the **return object** is of customer or vendor type, based on that it'll store that object to their relative model class's reference variable.
13. In the **parent update view** we have written both **selector tags** for customer and vendor which will work as a **child component**. In **selector** we've called **checkuser** function and it'll set **properties** for both customer and vendor.
14. Base on this function it'll **switch** between **2 components**, it will return **true** if the **customer object** is not null or it'll return **vendor object**.
15. Based on this selected component, it will **redirect** to appropriate component and accessing the **objects** value in **child component** using **@Input** annotation.
16. After making changes in fields when user clicks on the **update button** it'll wrapped all the values in the **object** and make an **Ajax** call to **RestBase service**.
17. In **RestBase service** it'll call to appropriate **implementation** class **through interface** which implements this implementation class it'll call update function.
18. In **update** function we're passing **object** which **contains mobile no**. came from **presentation layer**, based on this it'll **update user** details in **database**.
19. If the update is **succeeded** status will **return 1** and **message** as **update successful**, and this message will return to **presentation layer**, if the status is 1 it'll redirect to customer profile view1 or if the status is 0 it'll show message as incorrect details.

Errors

1. While uploading a photo in vendor registration we tried to insert photo directly in database by using BLOB (binary large object) datatype. But later we understand that it's consuming too much space in database, and it was not a standard way to store an image in database.
2. Later we've understood that there are 2 solutions for storing the image in database.
3. One type is BLOB and another one is storing only image name in database, and store the actual image on local server.
4. When we started to implement this as we mention above, we're uploading image during vendor registration, that time we've to pass both vendor object and MultiPartFileData (Image).
5. When we implement this and tried to test it on postman, then we understood that we can't pass the object and image in one http request, it means we cannot use @RequestBody annotation with multipart file.
6. For this we had to convert vendor object into @RequestParam annotation which means we converted the object into string.
7. After this we've successfully inserted the vendor details and image using postman.
8. Later when we tried to implement this in angular (presentation) that time we faced the problem in ajax call because http protocol doesn't support directly to MultiPartFileData, for that we used the FormData which is a inbuilt class in angular.
9. By using append function which is non static function in FormData class we append the vendor object and file and send this FormData class object in http request.

What we learn?

1. In angular we can't create 2 views in 1 component, it was kind of error for us, that time we learn that we have to use parent and child relationship between components for different views, where we used @Input annotation which helps us to transfer the data from parent to child.
2. There was a scenario where we wanted to share a data between 2 component that time we had 2 options, one was we could have used parent to child another one is sharing data using service. So we're using the service approach in which we've created getter and setter function to share a data between 2 components.
3. We learn to implement the email in angular to springboot.
4. We've learn to implement forget password.
5. We've learn canActivateRoutes in angular.
6. We've also learn Wild Card search.
7. We've learn the usage of sessionStorage class and onInit function in angular component.
8. Learn to upload an Image locally on springboot server.

Blank Page