Q. Build (or) generate of ?
→ ng build --base-href /foldersname/
→ java -jar paste.jar

Q. what is enterprise application?
It is also known as enterprise application software (EAS), is computer software used to satisfy the needs of an organization rather than individual user.
An Enterprise application (EA) is a large software system platform designed to operate in a corporate environment such as bussiness or govt.

Q. What is Desktop application?
Desktop applications are software programs run locally on computer device. They are not accessible from a browser (web-base apps like) and development on a personal computer or laptop.
ex:- MS word, MS office, skype, Chrome etc.

Q. Standalone application?
A program that does not require operating system's services to run.
A portable app, which can be run without the need for installation procedure.
ex:- Notepad, calculator

Q. Single page Application?
SPA is a web app that is presented to a user through single html page and allow user to jump different parts of system without reloading the entire page, ex:- fb, youtube, Google map, google Gmail.

Generics

BaseForm.java

```
public class BaseForm {
        public BaseDTO getDto() {}
        public <T extends BaseDTO> T initDTO(Id
        { }
        public void populate(BaseDTO bdto){}
```

BaseDTO.java

```
@Mapped Superclass
public abstract class BaseDTO implements Serializable, DropDownList, Comparable<BaseDTO>{

    public abstract String getUniqueKey();
    public abstract String getUniqueValue();
    public abstract String getUniquekey();
    public abstract String getLabel();
    public abstract LinkedHashMap<String, String>
            orderBy();
    public abstract LinkedHashMap<String, Object>
                                    Uniquekeys();
}
```

BaseDAOInt.java

```
1) public interface BaseDAOInt<T extends Base>
2) public abstract class BaseDAOImpl<T extends Base
                    implements BaseDAOInt<T>{

    @PersistenceContextext
    }
3) public interface BaseServiceInt<T extends BaseDTO>

4) public abstract class BaseServiceImpl<T extends BaseI
            D extends BaseDAO<T>>
```

Design Patterns

1) DTO design pattern :- Data processing के लिए ~~एक~~ layer से दूसरी layer पर data transfer करती है।

सारी DTO classes मे attributes define किए है और उनकी setter getter बनाई है।

2) Session Facade :- single entry point for entire app

Service layer पर Base DAO को autowire करके follow किया है।

3) DAO design pattern :- DAO (Data Access Object) class: crud operation perform किए है

4) Command design pattern - सारी api को call करने के, command आती है जिसे uri के throw navigate किया

Navigate by URL to its own controller.

5) Front Controller design patter :- It prevents any user to access application without login.

6) Bridge design pattern :- Single Interface Multiple implementation.

follows in service and DAO class.

Dependencies

Backend Spring Boot Dependencies in pom.xml

→ org.springframework.boot ⇒ Version 2.1.2 RELEASE
→ Java version ⇒ 1.8

<groupId>org.springframework boot </>
1) spring-boot-starter-actuator
2) spring-boot-devtools (scope-runtime, optional-true)
3) spring-boot-starter-data-jpa
4) spring-boot-starter-mail
5) spring-boot-starter-web
6) spring-boot-starter-test

7) mysqlr connector-java (8.0.28)
8) C3P0 (0.9.1.2)
9) jasperreports (6.11.0)
× 10) log4j (1.2.17)

actuator - print program run on console
devtools - auto save and load the application when makes changes

JWT (~~Jasson~~ JSON web Token) dependency
package :- org.springframework.boot

→ Spring-boot-starter-security
→ jjwt (io.jsonwebtoken, 0.9.1)

(JSON - JavaScript Object Notation)

devtools - reload changes in source code without Restarting Server

Jasper Report query -

→ AddUser (Save api) - Submit()
Update api &
this.serviceLocator.HttpService.post (this.api.save, this.form.data)

→ deleteMany (delete api)
this.serviceLocator.HttpService.post (this.api.deleteMany+"/" + id, this.form.data)

⇒ Preload()
this.serviceLocator.HttpService.get (this.api.preload)

Project —

Google → springinitializer → Click → maven - Java

3.1.4 → Group, Artifact, Name, Desci, Package name → Packaging .Jar, 17,

AddDep click → deeadd → Gen ctrl → downloading → .zip → show in folder → Right click → Extract → eclips open →

File → import click → Existing maven pr → Next →
Root dire → Browse → folder → select → Pom.xml click →
↓ src/main

finish → spring boot → open → springbootproj app →

app.prop → server.port = 9192

@GenericGenerator
* pkg – org. hibernate. annotations. GenericGenerator;
* used only in BaseDTO
→ It is hibernate annotation used to denote a custom generator.

@Transactional.
pkg – org. spfw. transaction. annotation. Transactional
- Transaction handling to group multiple changes
→ used in - service classes.

@Valid – Javax. validation. Valid (spring MVC annotation)
- used for method level ~~annotati~~ validation.
- used to apply Declarative validation.

@EnableWebSecurity
pkg – org. spfw. security. config. annotation web. configurati
. EnableWebSecurity.
→ to enable spring web security.

@LastModifiedDate
→ pkg – org. spfw. data annotation. LastM.
→ used to capture the user who created / Modifi
entity.

@CreatedDate – to capture the 'point in time
this happened.

Q. What is Angular ?
Angular is a open-source web application used to create single page application.

:- BackEAPI on FE :-

1) ~~Lo~~ SignIn api → loginComponent.ts

this.httpService.post(this.endpoint + "/login", this.form")

endpoint = "http://localhost:8084/Auth";

2) ForgetPasswordComponent.ts → ~~forgetpassword~~ api

this.httpService.get(this.endpoint + "/fp/" + this.form.loginId,)

3) SignUp comp.ts → SignUp

this.httpService.post(this.endpoint, this.form.data);

endpoint = "http://localhost:8084/Auth/SignUp";

4) basectl.ts → Search api

this.serviceLocator.httpservice.post(this.api.search "/" + this.form.pageNo, this.form.searchParams,

→ edit api in display()

this.serviceLocator.Httpservice.get(this.api.get + "/" + this.form.data.id)

@RequestParam -
→ used to extract query parameter from the URL.
@PathVariable - used to extract/read the value from URL
@GetMapping - used for mapping http get req. to specific handler method.
@RequestMapping (Method = RequestMethod.GET)

@PostMapping - maps the http post req. on the specific handler method.
= RequestMethod.POST.

@RestController -
→ combination of @Controller and @ResponseBody
→ class level
→ used to controllers for RESTAPIs which can return JSON
→ used to create RestApi.

@Repository
→ pkg - org.springfw.stereotype.Repository
→ used to create DAO., class level
→ indicates that the class provides mechanism for storage CRUD operations.

@Service
→ pkg - same, class level
→ indicates that the class perform business logics.

@Email -
→ pkg - javax.validation.constraints.Email
→ validates that the anotated property is valid email addr

@NotEmpty
→ The anotated element must be ~~not~~ not be null/em
→ for string value

~~used for data~~
@NotNull - the annoted element must not be null.
@Pattern - The annoted character sequence must match the specific regular expression (Java regular expression convention)
→ used on mobile No/ phone No and alternate mobile No field.

~~@Min~~ - javax.persistance Name

@PersistanceContext
→ used for entity Manager
→ an entity manager Instance is associated with persistence context, and uses JPA methods
→ It is a set of entity instance

entityManager.find()
.remove(), .persists(
.merge(), .Create Qu
.getCriteriaBuilder()

@MappedSupperClass
→ defines mapping for an abstract or non persistent supper class that is inherit by its sub-class entities.
→ used only in BaseDTO class.
@Id - specify PK of an entity
@GeneratedValue - used to automatic generate the PK value
@Entity - specifies that the class is an entity and it is mapped to a database table.
@Table - specifies database Table, with this entity is mapped.
@Column - mapped a column for a persistente property
@Lob - refers to a variable length data type for spring larger object.

@Value -
→ pkg - org.spfw.beans.factory.annotation.Value;
→ class - BaseCH, JwtTokenUtil
→ field and method level annotation
→ used to assign default values to variable (field) & method arguments.

@ModelAttribute
pkg - org.spfw.web.bind.annotation.ModelAttribute;
class Method level annotation.
class - BaseCH.
It is used to bind request parameters to a bean object and stores bean into model object.

@RequestMapping
→ pkg - same
→ class - every ctl, method level annotation
→ used to map web request to spring controller methods.

@ResponseBody
→ parameter level annotation
→ used to bind method parameter to the body of
→ It tells a controller that the object is automatically serialized into Json and passed back to the http response object.

@RequestBody -
→ annotate a method parameter with req. body, the spring framework binds the incoming http req. body to the method parameter.

@Bean :-

pkg :- org.spfw.context.annotation.Bean;
→ method level annotation.
→ It indicates that class's methods return bean, to be managed by spring container.

@Component

pkg :- org.spfw.stereotype.component
→ Class को as a spring component define करता है और class की life cycle manage करता है;
→ और class को other class में autowire कर obj create करता है;
→ a class-level annotation

Email Service Imp
frontctl
JwtTokenUtil
Use - JwtRequestFilter

@Autowired :-

package :- org.spfw.beans.factory.annotation.Autowired;
→ used on - constructor, fields, setter method.
→ used for automatic dependency Injection

@ControllerAdvice -

pkg - org.spfw.web.bind.annotation.ControllerAdvice;
→ class-level
→ It handle the exception globally.
→ used in - ApplicationHandlerExceptionHandlerCtl.

@ @ExceptionHandler :-

pkg :- org.spfw.web.bind.annotation.ExceptionHandler;
→ class & Method level
→ It handles specific exception and send custom response to client.
→ Class - ApplicationExceptionHandlerCtl

Standalone Application :- A program that does not require operating system service, to run.

- A portable application which can be run without installation procedure.

Annotations and their packages :-

1) org.springframework.boot.autoconfigure.SpringBootApplication;-
   @SpringBootApplication.

2) org.springframework.beans.factory.annotation.~~autowired~~ Name;
   @Autowired
   @Value

3) org.springframework.web.bind.annotation.Name;
   @RestController
   @RequestMapping
   @RequestBody
   @ResponseBody
   @RequestParam
   @GetMapping
   @PostMapping
   @PathVariable
   @ModelAttribute
   @ControllerAdvice
   @ExceptionHandler

4) org.springframework.context.annotation.Name;
   @Bean
   @Configure.

org.springframework.data.annotation.Name;
@~~Date~~ CreatedDate
@LastModifiedDate.

6) org.springframework.transaction.annotation.Transactional;
@Transactional.

7) org.springframework.stereotype.Name;
@Component
@Service
@Repository

8) ~~org~~ javax.validation.Valid; → @Valid.

9) javax.validation.constraints.Name;
@Email
@NotNull
@NotEmpty
@Size
@Pattern
@Max
@Min

10) javax.persistance.Name;
@PersistenceContext
@MappedSuperClass
@Id - BaseDTO
@GeneratedValue
@Lob
@Entity
@Table
@Column.

11) org.hibernate.annotations.GenericGenerator;
@GenericGenerator (BaseDTO)

12) org.springframework.security.config.annotation.web.configuration.Name;
@EnableWebSecurity

13) org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
@EnableGlobalMethodSecurity

Definition of annotations :-

1) @SpringBootApplication :- Indicates a configuration class that declares one/more @Bean methods.

Combination of 3 annotations
→ @~~Enable~~ Configuration.
→ @AutoConfiguration.
→ @ComponentScan.

ये हमारी class को as a main run class mark करता है।

2) @EnableAutoConfiguration :- enables spring Boot auto-configuration mechanism.

जैसे pom.xml मे कोई dependency add करते है तो उससे related सारी चीज़े autoconfigure हो जाएगी।

ex:- mysql dependency add की तो उससे related सारी configuration automatic हो जाएगी

package:- org.springframework.boot.autoconfigure.EnableAu...

3) @ComponentScan :-

pkg:- org.springframework.context.annotation.ComponentScan;

→ It scan a package of Beans.
→ ये spring component के लिए current class का pkg scan करती है (उस pkg की class, sub pkg & thier class भी scan करती है) की कोई Bean उस pkg मे है/नहीं।

4) @Configuration (org.spw.context.annotation.configuration)

→ class level annotation
→ Bean के लिए configuration class होती है जो Spring Container द्वारा manage होती है।
→ Bean return करने वाली methods को autowire करके directly use.

Q. Search field कैसे बनाई ? (FirstName)

→ html पर

Step: 1) <input type="text" class="form-control" [(ngModel)]="
" form.searchParams.firstName" placeholder="{{'
search first name'}} | translate}}" >

Step: 2) Search button में (click) = "submit()" call
<button (click)="submit()" class="btn btn-primary" >
{{'search' | translate}} </button>

3) userlist पर BaseList(T) extends, BaseList(T) की submit() में this.search() call

4) BaseCtl की search() में Backend api hit
serviceLocator.httpService.post (-self.api.search + "/" -self.form.pageNo, -self.form.searchParams)

5) Backend पर BaseCtl में search() बनाई (common method)

6) Search() में @ReqBody में form का data get, form के obj form में hold, @PathVariable से pNo. get, int pageNo में hold.

7) form.getDto() call, T dto में hold किया

8) OrsResponse का obj बनाया, res.addData() में baseServi search() को इसमें dto, pageNo, pageSize pass

9) BaseServiceImpt की search() में BaseDAO की search() call

10) BaseDAOImpl की search() में –

CreateCriteria() में dto pass, TypedQuery के obj query में h

→ CreateCriteria() BaseDAOImpl पर ही बनाई हैं इसमें

- entityManager की getCriteriaBuilder() call, Crit. Build. के builder में hold.
- builder.createQuery() call, getDTOClass() pass की और

Q How to configure jpA ?

1→ add dependency (Pom.xml) → spring-boot-starter-data-jpA

2→ set JPA configuration parameter in application-properties

3→ In BaseDAOImpl.java class use @PersistenceContext annot. and get EntityManager instance to use JPA methods

4. JPA Methods → getCriteriaBuilder, createQuery, find, remove, persist, merge, removeall

Q AOP – Aspect Oriented Programing, used to decouple the code for spring AOP.

Q. User Context ? – it is a object which manages session of frontend and backend

Q. 5 angular events – click, change, cut, paste, copy, focus

Q Frontend Service ~~अर्थ क्यों~~ ? – for resusability, it contains business logic.

Difference b/w Spring and Spring Boot:

Spring	SpringBoot
→ Used to develop enterprise application	→ Used to develop rest APIs.
→ Spring Framework is used for Dependency Injection.	→ SpringBoot is used for Autoconfiguration.
→ Use to create loosely coupled Application	→ It helps to create a standalone Application.
→ To run Spring App, need to set the server explicitely	→ Provide embeded server like:- tomcat, Jetty etc
→ xml configuration is required.	→ No need for xml configuration.

Router href

Application ~~time~~
~~flow~~ reload
ak componen

Q. How to apply and generate JavaDoc?
→ Apply – alt + shift J
→ Generate → Create a doc folder in backend workspace
→ Go to editor (Eclipse) and click on project option option in menubar
→ click on generate javadoc → javado.exe select
→ ~~gents~~ – clean package, ~~[illegible]~~
→ ~~of~~ visibility – private
→ destination – select doc folder path → finish
→ After generate javadoc →
copy the doc folder and paste in assets folder of UI workspace

Q. Jaxa doc in tab में कैसे open हो रहा है?
⇒ UI ⇒ javadoc की link पर click event में goToLink function को call किया (navbar.html)

→ navabar.ts पर goToLink () बनाया है।
→ इसमें windows.open ('assets/doc/index.html', '_blank') set किया

Q. Exception आगयी तो क्या होगा?
→ Unchecked type of exception आगयी → not complusory to hand

→ Exception will first arises on DAO Layer then,
→ propagates to service Layer, not handle then,
→ it will propagates to controller or calling method not handle on controller then,
→ it will come to main method in startup file then it will come to JVM (if handled then transaction will commited, exception propagate → transaction will be rollback)

→ Option tag में *ngFor में let obj of form.preload.roleList value में obj.key और interpolation में obj.value pass किया।

```
<select class="form control" name="roleId" [(ngModel)]="
  form.searchParams.preload.roleId" >
<option value="undefined">{{' select Role' | translate
  }} </option>
<option *ngFor="let obj of form.preload.roleList"
    [value]="obj.key">{{ obj.value}} </option>
</select>
```

Code backend :-

```
# UserCtl
@GetMapping("/preload")
public ORSResponse preload(){

        ORSResponse res = new ORSResponse(true);
        RoleDTO dto = new RoleDTO();
        dto.setStatus(RoleDTO.ACTIVE);
        List<Dropdownlist> list = roleService.search(dt
        res.addResult("roleList", list);
        return res;
}
```

⇒ RoleServiceImp class में BaseServiceImpl को extends किया,

→ BaseServiceImp की search() चली, इसमें BaseDAOImp c search() को call

→ BaseDAOImp की search() (single parameter) चली, इसमें return cSearch(dto,0,0); pass किया।

```
public List Search(T dto){
    return search(dto, 0, 0);
}
```

Q. Preload (Role मे) कैसे लगाई और कौनसी है?
→ Role पर preload लगाई है, Dynamic preload है।
→ इसके लिए UserlistComponent ts में BaseListCH को extends किया
→ BaseListCH में constructor load हुआ ngOnInit() में this.preload() को call किया
→ BaseCH पर preload() बनाई इसमें Backend api hit की

```
preload(){
  this.serviceLocator.httpService.get(_self.api.preload
                function(res) {
    if (res.success){
      _self.form.preload = res.result;
    }else{
        _self.form.error = true;
        _self.form.message = res.result.message;
    }
  });
}
```

→ Backend पर userCH मे preload() बनाई।
→ इसमें ORSResponse का obj बनाया, RoleDTO का obj बनाया।
→ dto.setStates में (RoleDTO.ACTIVE) किया
→ roleservice की search() को call किया इसमें dto pass Dropdownlist type की List के obj में hold
→ then res.addResult में "rolelist", list का obj pass
→ return res.

Frontend → callback() मे res get, (basecom.ts) res.result को form.preload hold (userlistcom.html)
→ html पर <select> मे name = roleId, [(ngModel)] में form.searchParams.preload दिया <option> में value = undefined दिया

ProjectOrsApplication.java

```
@Override
public void addInterceptorRegistry(InterceptorRegistry
        registry){

    registry.addInterceptor(frontctl).addPathPatterns("/**"
        excludePathPatterns("/Auth/**");
}
```

Q. Date की Input field कैसे लगाएंगे?

→ user list component.html पर एक input field लेंगे।
→ input type में date, [(ngModel)] = form.searchParam.dob लेंगे

```
<input type="date" [(ngModel)]="form.searchParams.dob
placeholder="{{'Search Date' translate}}" >
```

→ Backend पर BaseDAOImpl class की getWhereClause() को userDAOImpl class में Override किया.

→ Predicate type of list का obj whereCondition बनाया
→ then check condition if (isNotNull(dto.getDob())) है तो whereCondition.add में builder.equal(qRoot.get("dob"), dto.getDob()) pass किया।
→ return whereCondition.

```
@Override
protected List<Predicate> getWhereClause(UserDto, dto,
    CriteriaBuilder builder, Root<UserDTO> qRoot){

List<Predicate> whereCondition = new ArrayList<Predicate>();
if (isNotNull(dto.getDob())){
    whereCondition.add(builder.equal(qRoot.get("dob"), dto
            getDob()));
}
return whereCondition;  }
```

Ans

Q. FrontController के लिए क्या किया? (FE)

Frontend पर HttpService class में isLogout() बनाई इसमें :-

```
isLogout() {
  let JSESSIONID = localStorage.getItem('fname');
  if((JSESSIONID == "null") && (this.router.url != "/login"
    && this.router.url != "/Auth"
    && this.router.url != "/logout"
    && this.router.url != "/forgetpassword"
    && this.router.url != "/signUp"
    && this.router.url != "/login/true"  ))} हो तो
  this.form.message = 'Your Session has been Expired !
                        please Relogin";
  this.router.navigateByUrl("/login");
}
```

login पर navigate किया और interpolation से msg को print किया।

Q. Backend पर frontctl कैसे apply किया?

↳ @Component anotation का use किया

→ frontctl class बनाई, इसे HandlerInterceptorAdapter से extends किया

→ 2 methods preHandler और postHandler को over ride किया।

Startup class (ProjectOrsApplication) में addInterceptors() को override किया

इसमें InterceptorRegistry parameter registry pass registry.addInterceptor(frontctl).addPathPatterns() में "/**" , .excludePathPattern() में "/Auth/**" pass किया।

Logout flow :-

Frontend:- Navbar.comp.html पर logout की link पर click event से logout() को call किया।

```
<a class="dropdown item" (click)="logout()">   class="fa
fa-lock></i> {{' log Out' | translate }}</a>
```

Navbar.ts में logout() बनाया, उसमें httpservice.post() में logout का path/url set किया।

("http://localhost:8084/User/logout")

Backend:- UserCtl पर logout() में ORSResponse का obj बनाया,
→ session का obj बनाया और session को invalidate() call
→ res.setSuccess(true) set किया,
→ res.addMessage में success msg set किया।
→ res को return

```
@GetMapping("/logout")
public ORSResponse LogOut(HttpServletRequest req, res) {

    ORSResponse res = new ORSResponse();
    HttpSession session = req.getSession();
    session.invalidate();
    res.setSuccess(true);
    res.addMessage("Logout Successfully");
    return res;
}
```

Frontend:- res को callback() से get किया और res.result.message को form.message में hold
- routes.navigateByUrl से login पर navigate किया
→ login.html पर interpolation से msg print

```
    baseService.update (userDTO);
}
    ORSResponse res = new ORSResponse();
    res.setSuccess (true);
    res.addResult ("imageId", imageId);
    return res;
}
```

AttachmentDTO file को read करता है।
AttachmentDTO.java class में AttachmentDTO method बनाई
जिसमे file से getOriginalFileName और contentType निकाला
@Lob (Large object Byte) और put byte type array doc
लिया
→ getDoc() and set setDoc() बनाई यहां से file को read
किया और DOC मे hold किया।

```
public AttachmentDTO (MultipartFile file){
    name = file.getOriginalFileName();
    type = file.getContentType();
    doc = file.getBytes();
}
@Lob
@Column(name="DOC")                    // contains file data
private byte[] doc;
public byte[] getDoc(){ return doc; }
public void setDoc (byte[] doc){
    this.doc = doc;
}
```

→ Respective ts पर onFileSelect() में FileList type का variable files par
→ files.item(0) को this.fileToUpload में hold.
→ Condition check fileToUpload मिलता है तो myFile() call
→ MyFile() में onSubmitProfile() को call किया।
OnSubmitProfile() में formData का obj बनाया,
formData को append() में file, fileToUpload pass किया
और this.httpService.httpClient.post() में profilepic की
api और form.data.id) को pass करके return किया।

```
onSubmitProfile(fileToUpload: file){
    const formData = new formData();
    formData.append('file', fileToUpload)};
    return this.httpclient.post("http://localhost:8084/User/
            profilePic/" + this.form.data.id)
}
```

ckend:- UserCtl की uploadPic() चली इसमें @Pathvariable से id get
long userId में hold, @RequestParam से file get किया और
file में hold (file type का data read, large data hold)

then → UserDTO userDTO = baseService.findById(userId)
AttachmentDTO ~~attachment dto~~ doc = new AttachmentDTO
(file);

```
doc.setDiscription("Profile Picture");
doc.setPath(req.getServletPath());
doc.setUserId(userId);
```

condition check -

```
if(userDTO.getImageId() != null && >0) होतो
        doc.setId(userDTO.getImageId());
}
Long imageId = attachmentService.save(doc);
if(UserDTO.getImageId == null)
        userDTO.setImageId(imageId);
```

Add User :-

Q. Add User and image upload ?

→ user.component.html पर add User button → (click) = " forward(user)

→ User component. html पर save button पर click event में onUpload() को call किया

→ User Comp.ts में onUpload() बनाया, उसमें this.submit() को call किया

→ Submit() में b.e. की save api को hit किया
this.serviceLocator.HttpService.post(this.api.save, this.form.data)

Backend :
→ usectl पर BaseCtl को extend किया,
→ BaseCtl पर save() बनाई उसमें - baseServi की add() call
→ BaseServiceImpl में baseDAO की add() call.
→ BaseDAOImpl की add() में populate() में dto' pass किया then entityManager की persist() में dto pass किया then return dto.getId() किया

→ BaseCtl की save() में res.addData में (dto.getId()) pass किया और res. return किया।

Frontend :-
→ user comp.ts में res. resw को callback() में get किया ?
→ res.result.data को form.data.id में hold
→ then interpolation में data print किया

Q. Image upload ?
→ User Comp.html में ~~inptutt~~ input type में file, name में imageId और change event पर onFileSelect() को call किया उसमें $event.target.files pass किया)

```
<input type="file" name="ImageId" (change)="onFileSelect($event.target.files)" accept="image/*" >
```

Now creatCriteria() में

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();
CriteriaQuery<T> cq = builder.createQuery(getDTOClass());
Root<T> qRoot = cq.from(getDTOClass());
cq.select(qRoot);
Typed Query<T> query = entityManager.Create Query(cq);
return query;
}
```

After search a record

```
List<predicate> wherecl~~ause~~ause = getwhereclause(dto, builder, qRoo
```

-> then UserDAOImpl में getwhereclause() (abstract method) में :-
list में obj wherecondition बनाया then -

```
List<Predicate> wherecondition = new ArrayList<Predicate>
if(!isEmptyString(dto.getFirstName())) {

    wherecondition.add(builder.like(qRoot.get("FirstName"),
                        dto.getFirstName() + "%"));
```

-> then wherecondition को return किया।

Frontend -> res get किया और usdistcomponent.html पर condition check की [(ngModel)] = "form.searchparams.firstName" से name को Interpolation से print किया

:- Search flow :-

Frontend :- UserComponent.html

UI → Search button पर click event में submit() को call किया

t-s → UserList comp.ts में BaseListComponent (ctl) को extends किया

Baselist.ts → BaseListctl की submit() में this.search() को call

→ BaseCtl की search() चली इसमें b.e. की api hit की

Base com.ts

this.serviceLocator.httpService.post(_self.api.search + "/" + _size, form.pageNo, _self.form.searchParams)

→ this.servloc.httpSer. की post() में api.search + pageNo app end किया और form.searchparams pass किया।

Backend :- BaseCtl की search() में @RequestBody से form data get किया और form के obj form में डाला then

→ @PathVariable से pageNo get किया और int pageNo में

→ baseService की search() को call किया।

→ BaseServiceImpl की search() में BaseDAO की search() चली।

→ BaseDAOImpl की search() में :-

```
TypedQuery<T> query = createCriteria(dto);
if (pageSize > 0) {
    query.setFirstResult(pageNo * pageSize);
    query.setMaxResults(pageSize);

TypedQuery<T> query = entityManager.createQuery
List list = query.getResultList();
return list;
}
```

—: Edit flow :—

→ html (userlist component) पर edit की ~~link~~ button पर click
→ user ने forward() call की साथ /user/+ m.id
pass की (click)="forward('/user/' + m.id)"
→ ये user component.ts पर भेजा, यहां BaseCtl को
→ extends किया है।
→ Base(+) में भेजेंगे यहां constructor load होगा
→ 07 Constructor में servicelocator की getPathVariable() में
params['id'] को this.form.data.id में hold किया
→ ngOnInit() में condition check की if
if (this.form.data.id && this.form.data.id > 0) है तो
this.display() को call किया

→ display() में servicelocator.httpservice.get(apiget +
"/" + form.data.id) pass की

⇒ backend की api hit हुई, BaseCtl में get() बनाई
→ इसमें @PathVariable से id get की और long id में hold
→ ORSResponse का obj बनाया।
→ baseService की findById() को call किया इसमें id pass की
→ userDTO के obj dto में hold
→ condition check if (dto != null) है तो res.addData(dto) की
pass,
→ res.setSuccess(true), और res return किया।

⇒ frontend पर res को callback function से get किया
→ res.result.data को form.data.id में hold

→ usercomponent.html पर condition check *ngIf = form.data.id > 0 है तो update User show करें
और form.data से data निकाला (form.data.firstName
LastName...) और interpolation से print किया।

Backend की api hit हुई

→ UserCtl पर downloadPic () बनाई इसमे @PathVariable से id get की और userId मे hold.

→ baseservice की findById () call की userId pass, UserDTO के obj userDTO मे hold.

→ attachmentService की findById () को call किया इसमे userDTO.getImageId () pass किया और attachment DTO के obj attachmentDTO मे hold.

→ condition check if (attachmentDTO != null) मिलता है

→ response.setContentType (attachmentDTO.getType())
और OutputStream out = response.getOutputStream();
और out.write मे (attachmentDTO.getDoc())
और out.close () किया

```
@GetMapping("/profilePic/{userId}")
public @ResponseBody void detetUpadPic (@PathVariable
    Long userId, HttpServletResponse response)

UserDTO userDTO = baseService.findById (userId);
AttachmentDTO attachmentDTO = attachmentService.findById
                    (userDTO.getImageId());
if (attachmentDTO != null){

    response.setContentType(attachmentDTO.getType())
    OutputStream out = response.getOutputStream();
    Out.write(attachmentDTO.getDoc());
    out.close();
}
```

else { this.form.message = "Select Atleast One Record";
this.form.error = true;
}

→ Basecomponent.ts की deleteMany() में backend api hit
250 this.serviceLocator.httpservice.post(environment.api.delete
Many + "/" + id; this.form.data)

Backend पर deleteMany() BaseCH पर बनाई है (common meth

→ deleteMany() में @PathVariable से id को get किया String
array ids में hold then @ReqB से form का data get, form में hol
→ @RequestParam से pageNo get किया int pageNo में hold
→ res का obj बनाया, for each loop में ids को string
id में iterate किया
→ baseService की delete() को call किया, इसमें Long.pars
Long में id pass की, then
→ res.setSuccess true किया, res.addMessage में success
msg pass किया और res को return.

BaseServiceImp की delete() में BaseDAO की delete() call
BaseDAOImp की delete() में entityManager की remove()
call and dto pass.

Q Image कैसे बनाई?

↳ UserListComp.html पर condition दी है *ngIf = "m.imageId"
है तो backend की api hit की

```
<div *ngIf = " m.imageId">
<img src = " http://Localhost:8084/users/profilePic/${m.id}" >
</div>
```

next ⇒ <button (click) = " next() " [disable] = " nextlist ==0 "
type = "button" class = " btn btn-info "> {{ 'Next' |
translate }} </ button>

" Delete flow "

→ UserList.component.html पर click event मे DeleteMany() को call किया।
→ BaseListComponent.ts मे deleteMany() बनाया है।
→ इसमे checkbox.forEach() से element को iterate किया
→ condition check की if element.nativeEle.checked है तो deleteRecordList की push() मे element.nativeele.Id pass की
→ isRecordedSelected = true मिलता है तो super.deleteMany() को call किया और इसमे this.deleteRecordList मे ? pageNo को append किया और this.form.pageNo को भी append किया।

:-

```
deleteRecordList : any = [ ]
deleteMany() {
val isRecordSelected = false ;
                      :Boolean

this.checkboxes.forEach((element) => {
  if ( element.nativeElement.checked) {

  this.deleteRecordedList.push (element.nativeElemen.id);
  isRecordSelected = true ;
} });
if (isRecordSelected) {

super.deleteMany ( this.deleteRecordList + '?pageNo=' +
  this.form.pageNo )
```

Q. SelectAll का checkbox कैसे लगाया ?

⇒ UserlistComponent.html पर

```
<input type="checkbox" [(ngModel)]="isMasterSel" name="
    list_name" (change)="checkUncheckAll($event)"/>
```

→ BaseListComponent.ts में checkUncheckAll () बनाया ⇒ 46

```
checkUncheckAll (event){
    const checked = event.target.checked;
    this.checkboxes.forEach((element) => {
        element.nativeElement.checked = checked
});
```

Q. single checkbox ?

```
<input type="checkbox" id="{{ m.Id }}" name="list_name"
    (change) = "checkListUpdate()" />  value ="{{isMasterSel}}"
```

change

→ BaseList.Cts में checkListUpdate() बनाया ⇒ 82

```
checkListUpdate.()}
    this.checkboxes.forEach((element) => {
        If (element.nativeElement.checked == true){
            this checklist = this.checklist+1;
}
});
        if(this.checklist == this.form.list.length){
            this.masterSel = true;
}
```

Q. Admin का checkbox और edit link कैसे disable है

→ html पर condition दी है [disabled] = "m.roleName == 'Admi

Q. Previous button & next button कैसे disabled ?

```
<button (click) = "previous()" [disabled] = "form.pageNo==0
    class btn-btn-info>{{' previous | translate}} </button>
```

→ query.getResultList() call की List के obj मे hold
→ list को return किया

Frontend:- res को callback function से get किया
res.result.data को form.list मे hold
go → html पर ngFor से list को iterate किया -
ngFor = "let m of form.list
then m.firstName/lastname करके interpolation से pr

Backend code:-

BaseCtl.java - search()

```
@RequestMapping (value = "/search/{pageNo}")
public ORSResponse search (@RequestBody F form, @PathVariable int pageNo) {
    T dto = (T) form.getDto();
    ORSResponse res = new ORSResponse(true);
    res.addData(baseService.search(dto, pageNo, pageSize));

    return res;
}
```

# BaseServiceImp की search() मे
return baseDAO.search(dto, pageNo, pageSize);

BaseDAOImp की search() मे

```
TypedQuery<T> query = createCriteria(dto)
if (pageSize > 0){
        query.setFirstResult (pageNo * pageSize);
        query.setMaxResults (pageSize);
}
List list = query.getResultList();
return list;
}
```

jwtTokenUtil.validateToken(jwtToken, userDetails);

— : "UserList" :—

Q. List कैसे आ रही है

→ Userlist link पर click, app routing.module.ts में path में userlist दिया है & component में UserlistComponent दिया है

→ UserListComponent.ts में BaseListCtl को extends किया। Baselistct

→ वहां पर constructor load होगा & ngOnInit() चलेगी।

→ ngOnInit() में this.preload और this.search() चलेगी।

→ Search() BaseCtl पर बनाई है, BaseCtl में आएगा, search() चली, इसमें backend की api hit की → basecom.ts

this.serviceLocator.httpService.post(endpoint api search + "/" + form.pageNo को append किया और form.searchParams p

→ पर BaseCtl को extends

Backend पर UserlistCtl की search() चली।

→ इसमें (BaseCtl) पर सारी common methods बनाई है।

→ Search() में @RequestBody और form का obj form pass किया the @PathVariable से pageNo को get किया और int page में hold, then,

→ form.getDTO() को call किया, T type के dto में hold.

→ res.addData() में baseservice की search(..) को call इसमें dto, pageNo, pageSize pass किया।

→ res return किया

↳ BaseServiceImpl की search() में BaseDAO की search() call किया इसमें dto, pNo, psize pass किया और Return

↳ BaseDAOImpl की search() में :-

→ CreateCriteria() को call किया, dto pass, और TypedQ के obj query में hold.

→ then pagination perform → if (pageSize > 0){

query.setFirstResult में pageNo * pageSize pass

setMaxResults में pageSize pass किया

Authentication manager class - [illegible]

loadByUserByUserName - with this, user info load from DB authenticate

UserDetailsService

```
@Override
protected void configure (HttpSecurity httpSecurity) throws
    Exception {
        httpSecurity.csrf().disable()
    .authorizeRequests().antMatchers("/Auth/**", "/User/
    profilePic/**").permitAll().
    anyRequest().authenticated().and().exceptionHandling().a
    .sessionManagment().sessionCreationPolicy(SessionCreationPolicy.
    STATELESS)
    httpSecurity.cors();
}
```

define url, disable csrf

JwtRequestFilter.java

```
@Component
public class JwtRequestFilter extends OncePerRequest
    Filter {

    @Autowired
    private JwtUserDetailsService jwtUserDetailsService;
    @Autowired
    private JwtTokenUtil jwtTokenUtil;

    @Override
    protected void doFilterInternal (HttpServletRequest req, Res
    {
        final String requestTokenHeader = request.getHeader("
                                                Authorization");
        if (requestTokenHeader != null && requestTokenHeader.
            startsWith("Bearer")) {
        String jwtToken = requestTokenHeader.substring(7);
        String username = jwtTokenUtil.getUsernameFromToken(jwt
                                                        T
        if (username != null)
        UserDetails useDetails = this.jwtUserDetailsService.loa
            UserByUsername(username);
```

erface,
ides core
.. Token)

Q. JwtRequestFilter class में क्या किया ?

↳ JwtRequestFilter class को OncePerRequestFilter से extends

↳ इसकी doFilterInternal() को override किया, इसमे Http ServletRequest and Response के obj & filterChain का obj pass किया।

↳ then request.getHeader से Authorization pass किया और String requestTokenHeader में hold.

↳ Condition checked if (requestTokenHeader != null && requestTokenHeader.startsWith("Bearer")) है तो

↳ requestTokenHeader की substring() में 7 pass किया और string variable jwtToken में hold

↳ jwtTokenUtil class की getUsernameToken() को call कि इसमे jwtToken pass किया और String Username में hold

↳ Condition checked if (Username != null) है तो

↳ this.jwtUserDetailsService की loadUserByUsername() में username pass किया UserDetails में hold किया।

↳ jwtTokenUtil की validateToken() में jwtToken और userDetails pass करके user को validate किया।

WebsecurityConfig.java

```
@Configuration                                  (to declare Beans)
@EnableWebSecurity                              (to enable web security)
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
                                                (to override its methods & define child specific methods)

    @Autowired                                  (Interface, load user specific data)
    private UserDetailsService jwtUserDetailsService;
    @Autowired
    private JwtRequestFilter jwtRequestFilter;
```

↳ Signup ts में exit() में location.reload() किया है। जिससे page को reload कर देगा।

4 classes to generate token ⇒ com.rays.config → JwtRequestFilter, JwtTokenUtil, WebSecurityConfig
~~Controller~~ ⇒ com.rays.service → JwtUserDetailsService

Q. Token कैसे generate किया?
~~JwtUtil JwtUti~~
JwtTokenUtil की generateToken() का use किया।

Q. सबसे पहले कौनसी class चलती है और क्यों?
↳ सबसे पहले JwtRequestFilter class चलती है। web
क्योंकि → इसके लिए एक class बनाई है WebSecurityConfig

→ WebSecurityConfig class में @Configuration, @EnableWebSecurity and @EnableGlobalMethodSecurity (prePostEnabled = true) annotations का use किया है।

→ WebSecurityConfig class को WebSecurityConfigurerAdapter class में extends किया और
2 → इसकी configure() को override किया इसमें HttpSecurity का obj pass किया।
→ HttpSecurity.csrf().disable किया
• authorizeRequests().antMatchers("/Auth/**", "/User/profilePic/**") pass किया permitAll() किया और
→ • anyRequest().authenticated() का use किया।
⇒ (इससे Auth और profilepic की request पर authenticate नहीं चलेगा)

HttpSecurity.addFilterBefore() में JwtRequestFilter pass किया इसलिए सबसे पहले JwtReqF. class चलती है।

```
if (dto != null){
    res.setSuccess(false);
    res.addMessage("Login Id already exists");
    return res;
}

dto = new UserDTO();
dto.setFirstName(form.getFirstName());
dto.setStatus("Inactive");
dto.setRoleId(2L);
baseService.register(dto);
res.setSuccess(true);
res.addMessage("user has been registered");
return res;
}
```

getLastName
getLoginId
getLogin
getGender
getDob
getPhone
getPassword

RoleForm.java → pub st final int STUDENT=2

UserDAOImpl.java

```
@Override
protect void populate(UserDTO dto)
    if (dto.getRoleId()!= null && dto.getRoleId()>0){

        RoleDTO roledto = roleDao.findByPk(dto.getRoleId());
        dto.setRoleName(roleDto.getName());
    }}
```

BaseDAOImpl.java

```
public T findByPk(long pk){
    T dto = entityManager.find(getDTOclass(), pk);
    return dto;
}
```

Q. Reset button पर क्या किया है?

→ click event में exit() को call किया है।

```
<a class="btn btn-warning [routerLink]=['/signUp']
    (click)="exit()">{{'Reset' | translate}} </button>
```

↳ backend पर LoginCtl में signUp() बनाई।
↳ @RequestBody में form का data get किया. UserRegistrationForm के obj form में hold किया।
↳ UserDTO का obj dto बनाया इसमें user का सारा data set किया. dto.setFirstName etc.
↳ dto.setRoleId में 2L दिया और baseService की register() को call किया dto pass किया।
↳ res.setSuccess(true), res.addMessage("registered")
↳ return res.

UserServiceImpt की register() में add() call किया
UserServicImpl की add() में baseDAO की add() call
BaseDAO की add() में entityManage.persist() call की dto pass किया और dto.getId() return की.

11⇒ frontend पर callback() से res get किया
~~res.result.message~~ को form.message में hold किया और form.data से data get किया और interpolation से print किया।

LoginCtl

```
@PostMapping("signUp")
public ORSResponse signUp(@RequestBody @Valid UserRegistrationForm form, BindingResult bR){

    ORSResponse res = validate(bindingResult);
    if(!res.isSuccess()){
        res.addMessage("please fill following empty fields");
    return res;
}
UserDTO dto = baseService.findByLoginId(form.getLogin());
```

—: Sign Up :—

Q. SignUp page पर date की input field कैसे बनाई ?

<input type = "date" name = "dob" [(ngModel)] = "form data.dob" class = "form-control" >

opt. valid. → <span *ngIf = "form.inputerror.dob ! = null" style = color: red' > {{' dob' | translate}} </ span>

Q. Preload कौनसी है और कैसे लगाई ?

→ Gender static preload है, इसके लिए select tage में name gender, option tag में value में undefined दिया और please select लिखा है।
<option tag में value में male & female दिया

<Select name = "gender" id = "gender" [(ngModel)] = "form.data.gender" >
<option value = "undefined">{{' please Select' | translate}}<
<option value = "Male"> {{' Male' | Translate}} </option>
<option value = "Female"> {{' Female | translate}} </>
</select>

Q. SignUp flow (fill data)?

↳ 1. SignUp.c.html पर signup button में click event में submit() को call किया।
<button (click) = " submit()" class = " btn-btn-success">
{{' sign up' | translate}} </button>

↳ respective ts पर submit() में backend की api hit की -
this.http.service.post(this.endpoint , this.form.data)

Backend

LoginCH.java

```
@GetMapping("fp/{login}")
public OpsResponse forgetPassword(@PathVariable String login){

    UserDTO dto = baseService.forgetPassword(login);
    if (dto != null){
        res.setSuccess(true);
        res.addMessage("Password has been sent");
    }
    return res;
}
```

UserServiceImpl.java

```
@AutoWired
EmailServiceImpl emailservice =null;
@Override
public UserDTO forgetPassword(String login){

    emailservice.send(emailDTO);
}
```

Frontend:- forgetPassword.ts में callback function से res. get किया
res.result.message को form.message में hold

→ html पर condition check by "ngIf = "form.message = 'paswd sen
<div class = " ~~btn btn~~ success

<button type = "button" class = "close" data-dismiss="a
> × </button> {{form.message}}

Q. Flow of forget password ?

↳ pom.xml में dependency add की, app.properties में mail param. set किए

↳ login.comp.html में routerLink से /forgetpassword पर navigate किया

↳ forgetpsw.comp.html में Reset my pswd button में click event पर submit() call किया।

↳ forgetpsw.ts में sumit() के अंदर this.httpService.get() में endpoint ~~api~~ के साथ /fp और loginId append की

↳ b.e. की api hit हुई

↳ b.e. पर logintCtl में for @GetMapping से api search की then ORSResponse type की forgetPassword() बनाई

↳ इसमें @PathVariable से uri data get किया string login में hold किया।

↳ baseService की fpsw() को call किया userDTO के obj dto में hold किया

↳ UserServiceImp की forgetPassword() में emailService की send() को call किया

↳ EmailService की send() में MimeMsg का obj बनाया और emailSender की send() का use करके mail send किया।

Login.compt.html

```
<a [routerLink] = "['/forgetpassword']" >
```

forgetpassword.html

```
<button (click)="submit() [disabled] = "!validate()">
    {{'ResetMyPassword' | translate }} </button>
```

ForgetPassword.component.ts

```
Submit() {
  this.httpservice.get(_self.endpoint + "/fp/" + this.form.loginId )
```

```
localStorage.setItem("token", tokenStr);
                     ("role", res.result.role);
                     ("fname", res.result.fname);
                     ("lname", res.result.lname);
                     ("userid", res.result.data.id);

  - self.router.navigateByUrl('dashboard');
  });
});
```

—: Forget Password :—

→ Login component.html पर forgetPassword button पर forget password ctl का path दिया है। Jo

```
<a class="btn btn warning" [routerLink]="['/forgetpassword
> {{'Forgot Password' | translate '}} </a>
```

Q. Mail कैसे send किया ?

→ pom.xml में spring-boot-starter-mail की dependecy add

→ application.properties में mail के parameters दिए है,

```
spring.mail.host = smtp.gmail.com
spring.mail.port = 587 (secure & free)
spring.mail.username = somali@gmail.com
spring.mail.password = .....
```

→ EmailServiceImpl class बनाई, इसमें send() बनाई

→ 2 predefined classes का use किया - MimeMessage and JavaMailSender.

→ Mime msg का obj बनाया

```
MimeMessage message = new emailSender.createMim
```

→ then emailSender.send का use किया और message pass

```
emailSender.send(message);
```

FrontEnd

login.component.html

```
<button (click)="signIn()">{{ 'sign in' | translate }}
<span *ngIf="inputerror.loginId != null">
    {{ 'loginId' | translate }} </span>
<b> {{ 'password' | translate }} </b>
<div *ngIf="form.message == 'Invalid Id or Password'">
    <button type="button" class="close" data-dismiss=
    "alert">&times; </button>
    {{ 'Invalid ID or Password' | translate }} </div>
```

login.component.ts

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

    endpoind = "http://localhost:8084/Auth";

signIn() {
    this.httpService.post(this.endpoint+"/login", this.form,
                                        function(res) {
    self.inputerror = res.result.inputerror;
    self.form.message = res.result.message;

    if (isTrue(res.success)){
    self.httpService.setToken(res.result.token);
    localStorage.setItem("loginId", res.result.loginId);
    let tokenStr = "Bearer" + res.result.token;
```

```
@Transactional (readOnly = true)
public UserDTO findByLoginId (String login) {
    return baseDao.findByUniquekey ("loginId", login)
}
```

~~User~~ BaseDAOImp.java

```
public T findByUniquekey (String attribute, Object val) {

    Class<T> dtoClass = getDTOClass();        return T.DTO.class
                        (abstract() of BaseDAOImp, imple in all DAOs)
    CriteriaBuilder builder = entityManager.getCriteriaBuilder();
    CriteriaQuery<T> cq = builder.createQuery(dtoClass);
    Root<T> qRoot = cq.form(dtoClass);

    Predicate condition = builder.equal(qRoot.get(attribute),
                                        val);
    cq.where(condition);
    TypeQuery<T> query = entityManager.createQuery(cq);
    List<T> list = query.getResultList();
    if (list.size()>0) {
    T dto = list.get(0);
    }

    return dto;
}
```

contain info of entity columns ← Root<T>

interface that check condition true/false return) ← Predicate

interface that begin and control execution of type queries ← TypeQuery

(select * from st_user where loginId = ?)

EntityManager - @PersistenceContext में entityMan. का instance hold होता है।

```
if (! res.setSuccess(false)
        return res;
}
UserDTO dto = baseService.authenticate(form.getLoginId(
                form.getPassword());
if (dto == null){
        res.setSuccess(false);
        res.addMessage("Invalid Id or Password");
} else{
        session.setAttribute("test", dto.getFirstName();
        res.setSuccess(true);
        res.addDto(dto);
        res.addResult("jsessionId", session.getId());
        res.addResult("jsessionId", dto.getLoginId());
        res.addResult("role", dto.getRoleName());
        res.addResult("fname", dto.getFirstName());
        res.addResult("lname", dto.getLastName());

final UserDetails userDetails = jwtService.loadByUserName
                (form.getLoginId());
final String token = jwtTokenUtil.generateToken(userDetai
res.addResult("token", token);
return res;
}
return res
}
```

LoadUserbyUN() → Locate the user base on Username.
~~return user record (never null)~~
UserDetails - provide user core info

⇒ UserServiceImpl.java

```
@Override
public UserDTO authenticate(String loginId, String password){

        UserDTO dto = findByLoginId(loginId);
}
```

Q. SignIn का flow (login successfully)

→ Frontend में signIn() से b.e. api hit की
→ Backend पर LoginCtl की authenticate() चली
→ इसमें form.getLoginId and password pass किया, dto में
→ condition check if (dto == null) false है तो else में
→ session.setAttribute में test, dto.getFirstName() pass
→ res.setSuccess (true) किया & res.addData में dto pass
→ res.addResult() में session.getId(), dto.getLoginId, RoleName(), firstName(), LastName() set किया।
→ then JwtService.loadUserByName में form.getLoginId pass किया UserDetails में hold किया।
→ JwtTokenUtil की generateToken() में userDetails pass की और string token में hold
→ res.addResult में Token दिया और res return किया

→ Frontend पर login.component में
→ res.setSuccess true check किया — 138
→ HttpService की setToken() में res.result.token pass
- localStorage.setItem में loginId, res.result.loginId
- Bearer के साथ res.result.token append किया और tokenstr variable में hold
- localStorage setItem में sara data set किया।
→ router.navigateByUrl() से dashboard पर navigate किया

Code: BackEnd (LoginCtl)

```
@PostMapping("login")
public ORSResponse login(@RequestBody @Valid LoginForm
    form BindingResult bindResult HttpSession session
    HttpServletRequest request) throws Exception {
ORSResponse res = validate(bindingResult);
```

Perform Business Validation on Login ?

login.component.html पर SignIn button, signIn() चली
backend की Api hit हुई.
backend में OneResponse type की login() बनाई इसमें
@RequestBody LoginForm का obj pass किया।
baseService की authenticate() में Log form.getLoginId()
form.getPassword() pass की और UserDTO के ob
में hold.
condition check - If (dto == null) है तो res.setSuccess
false set किया और
res.addMessage() में msg pass किया और ors return

InvalidEnd पर ts में callback function में res को get किया
res.result.message को form.message में hold किया
html पर cond. check - *ngIf = " form.message == Invalid
ID or Password" है तो interpolation में msg print

UserServiceImp में authenticate() बनाई
findByLoginId() चली, फिर findByLogin() में findBy
UniqueKey() call हुई।
इसमें "loginId", login pass किया
then authenticate() में findByLogin() में loginId pass
UserDTO dto में hold किया। यही dto Login() की
authenticate ने return किया।

ORSResponse.java
→ is a Creates a class, जिसके obj के through backend
DBT data frontend पर get करते हैं।
Define attributes - DATA, MSG, InputError, Success, Result

define Methods ⇒ addResult(key, value) - generate key for preload
addData(obj) ⇒ for Search field to add data
addMessage(obj), addInputeError(obj) ⇒ for input, Business validation

- @postMapping मे /login मिला यहां ORSResponse type की login() बनाई
- इसमे @RequestBody (form का data hold), @Valid (Validation apply करने के लिए), loginform का obj form और BindingResult का obj bindingResult pass किया।
- LoginForm, मे attributes पर @NotEmpty annot. मे message pass किये मे msg bindingResult के obj मे hold
- then LoginCH पर BaseCH को extend किया और validate() का use किया, इसमे bindRes का obj pass किया
- then cond. check if (!res.setSuccess()) है तो res return किया।
- BaseCH की validate() मे BindingResult का obj pass किया
- ORSResponse का obj बनाया then condition check - if (bindingResult.hasErrors()) है तो res.setSuccess मे false किया, map का obj बनाया errors
  map<S, S> errors = new HashMap<S, S>();
- then bindingResult की getFieldErrors() को call किया list के obj मे hold किया।
- list को for each() मे iterate किया and Lembda express मे e.getField() और e.getdefaultMessage() को errors की put() मे pass किया।
- res.addInputErrors(errors) pass किया और res return

→ FrontEnd पर login.compts मे callback function मे res को get किया
- res.result.inputerror को inputerror variable मे hold
- then html पर condition check की *ngIf = "inputerror.loginId != null" है तो interpolation के through input validation के msg print किए।

@RequestBody - form data को method parameter से bind करता
- BindingResult - interface (pre-defined), provided by Spring fr
- hasErrors() - Boolean type, return if any error.
- getFieldErrors() - Get all error associated with a field.

[illegible] 2 files [illegible] nu son are
en.json & दोनों files में required msg key value
bracer के अंदर pass किए (Json formate)

Q. Language कैसे change ?
→ navbar.comp.html में select tag में change event में changeLocale function को call किया and locale value pass किया and locale id ली है, option tag में value में locale दिया
- then respective ts में changeLocale function में locale लिया
- localStorage.setItem में key value pair में locale दिया
- this.translate.use () में localStorage.getItem में locale key pass की।

Q. english by default कैसे आ रही है ?
- navbar.ts में constructor में Translate service को inject किया
- condition check की if (localStorage.getItem("locale")! = null) है तो translate.~~setItem~~ setDefaultLange में localStorage.getItem में locale pass किया।
- और else में translate.setDefaultLang में en. दिया है।

LocalStorage - BrowserModule को import करते है localstorage से item get करने के लिए।

:- Login Page :-

Q. Perform input validation ?
→ login.comp.html में sign in button बनाया, click event में SignIn() को call किया।
→ then respective ts में signI() function में this.httpServi की post() में this.endpoint के साथ /Login और this.form pass किया।
⇒ (this keyword used to call httpservice of same page)
endpoint = "http://localhost:8084/Auth"

→ Backend पर गये restAPI check की @RestController बने,
→ @RequestMapping में path search किया, /Auth मिला

Q. Navbar कैसे बनाया ?- class = Navbar navbar expand lg.

Q. Imag को link कैसे बनाया ?
<a class = "nav-link" की है
routerlink मे dashboard का path दिया
img tag के src attribute मे image का path दिया।

```
<a class = " nav-link " [routerLink] = "['/dashboard']"
  <img src = "assets/files/custom.png" > </a>
```

Q. ये select language कैसे आ रहा है और language कैसे change हो रही है ?
→ Select tag का use किया इसमे option tag मे value दी है en and hi and interpolation के through select Language print किया और translate pipe का use किया जिससे lang. change हो रही है।

→ translate pipe कहां से आया ?

→ Language change करने के लिए क्या किया ?

command चलाई → npm install @ngx-translate/core
npm install @ngx-translate/http-loader

- by this, package enable.
- app.module.ts में imports के array मे translate module को import किया

```
imports: [
  TranslateModule.forRoot({
    loader: { provide: TranslateLoader,
              useFactory: myhttploader,
              deps: [HttpClient]
    }
```

Q. URI में # कैसे आ रहा है और क्यूं use किया ?

→ app.routing.module.ts में @NgModule decorator में imports के array में RouterModule की forRoot() में useHash true किया है within curly braces.

```
@NgModule({
    imports: [RouterModule.forRoot(route, {useHash:true})]
    exports: [RouterModule]
})
```

enable routing

→ # सारे url को map करने के लिए use किया है।

Q. Dashboard कैसे open हो रहा है ?

→ app.routing.module.ts में path black set किया, redirectTo में dashboard दिया और pathMatch में full किया है।

```
const routes: Routes = [
    { path: '',
      redirectTo: 'dashboard',
      pathMatch: 'full' },
]
```

Q. Dashboard पर background image कैसे लगाई ?

→ background attribute में url में image का path दिया है।

Q. localhost:4200/#/dashboard

localhost - system IP address ; 4200 - server Port No.

#/dashboard - uri

- सारे particular urls को map करता है

=> app.routing.module.ts में routingmodule में hashtrue किया है

```
    this.httpservice = hs;
    this.router = r;
    this.endpoint = ep;
}

getPathVariable (route: ActivatedRoute, callback) {
    route.params.subscribe (params => {
        callback (params)
    });
}

forward (page) {
    this.router.navigateByUrl (page);
}
}
```

Q. project run करते है तो सबसे पहले कौनसी file/page load होता है? ⇒ index.html

→ सबसे पहले main.ts file run होती है।

→ इस file में app module component को bootstrap किया है

→ then app.module.ts में जाएगा यहां app.component bootstrap किया

→ then app.component.ts में selector में app-root दिया है और इसे index.html में use किया है thats why index.html सबसे पहले run होगा।

Q. Bootstrap कैसे apply किया?

⇒ Index.html में libraries add की है -

i) bootstrap.min.css (4.3.1)

ii) bootstrap.min.js (4.3.1)

iii) jquery.min.js (3.3.1)

iv) font-awesome.min.css (4.7.0)

endpoint service मे SERVER_URL नाम से एक variable लिया, इसमे backend का URL दिया है।
और login के बाद वाले सारे components (user, college, course, faculty, marksheet, subject, student Timetable, faspes report) को SERVER_URL के साथ concate किया है उसी नाम से जिस नाम से इन component की routing की है।

ServiceLocator :- It is a design pattern, used to locate and access service within your application without explicitly inject them in every component
It is a singleton service that acts as a container for all other services.
Benefit - makes the code more maintainable & modular

इस service के constructor मे सारी other services को inject किया है।
2 methods use
- getPathVariable - uri मे value read करने के लिए
- forward() - next page (url) पर जाने के लिए

Servicelocator.service.ts

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

    httpService = null;
    router = null;
    endpoint = null;

    constructor( private hs: HttpService, private r: Router,
                 private ep: EndpointService ) {
```

HttpInterceptor interface provides to methods :-

1) intercept () :- used to intercept (catch) http request and transfer (change/modify) it before sending req to the server.

2) handle () :- used to handle the HTTP response and transfer it before returning it to the application.

EndpointService :- is a service that is responsible for making HTTP request to a specific API endpoint and retrieving data from an API endpoint.

endpoint-service.service.ts

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

export class EndpointServiceService {

  constructor () { }
  public SERVER_URL = "http://localhost:8084";
  public MESSAGE = this.SERVER_URL + "/Message";
  public USER = this.SERVER_URL + "/User";
  (Role, college, course, Markshoot, TimeTable, Student, sub
  faculty, Jasper Report)
}
```

endpoint service used to get backend URL

```
return this.httpClient.post(endpoint, bean).subscribe(
    => {
        callback(data);
    },
    error => {
        console.log('OBS Error - ', error
    });
}
```

* HttpClient Module -

It is a core module in angular that enable communication between the application and a ser using HTTP request and response.

→ provides API for making requests to a server handling the response.

→

→ provides several methods for making requests - GE POST, PUT, PATCH, DELETE.

→ support features like - interceptor (which allow you modify request and response.)

(HttpClient module, easy to make Http requests, h errors and transfor response data according application's needs).

\# HttpInterceptor :- यह angular का interface है जो Http requests and response intercept करने के लिए व करता है (means req and resp को किसी भी ser पर भेजने या server से receive करने के पहले modify / handle करता है।)

→ It is used to implements features - adding authorizat headers, caching response, logging req & resp, error handli

HttpService

http-service service.ts

→ इस class में Token को set and get किया है।

→ Constructor में router and HttpClient को inject किया

→ get() and post methods बनाई है जो respectively http client की get() and post () return कर रही है,

→ logout () बनाई।

```
@Injectable ()
export class HttpServiceService {

setToken (token) {
  this . token = localStorage . getItem ('token');
}
getToken () {
  return localstorage . getItem ('token');
}
Constructor (private router: Router , private HttpClient: HttpCli
{
}
get (endpoint , callback) {
  if (this . islogout ()) {
    console. log ('inside logout()');
    return true ;
}
return this . HttpClient . get (endpoint) . subscribe ((data
  => {
      callback (data);
  });
}

post (endpoint , bean , callback) {
  if (this . islogout()) { return true ;
}
```

Q. Next request कैसे handle की?
next.handle(req) को return किया (Auth service.ts में)

Q. Auth Service को कैसे use किया?
→ Auth service.ts में provided In 'root' किया है। जिससे auth service class को कहीं भी use कर सकते हैं।

Q. BackEnd API कैसे hit की?

→ app module.ts में HttpClientModule को import किया
→ then एक service class बनाई by command -HttpServic
→ में manually 2 methods बनाई get & post
→ जो HttpClient की predefined methods get and pos return कर रही है।
→ So HttpClient की get & post or HttpService का use करके back end की API hit की that is called AJAX calling.
[AJAX - Asynchronous Java Script and xml]

Q. How to configure DCP / How to make connection with Database?

→ add dependencies in pom.xml
→ Springboot-starter-data-jpa
→ mysql-connector-java

→ Configure database & jpa in configuration file (application.properties)

return next.handler(req);

Header में क्यूं set करते हैं request को ?
bcoz header में set होने वाली सारी चीजें accept हो जाए (token)

Token कैसे generate किया
Backend से → JwtTokenUtil class की generateToken() से

Component
public class JwtTokenUtil implements Serializable {

# generate token for User
public String generateToken(UserDetails userDetails) {
    map<S, O> claims = new HashMap();
    return doGenerateToken(claims, userDetails.getUserName());
}

# Validate Token
public Boolean validateToken(String token, UserDetails ud) {
    final String username = getUsernamefromToken(token);
    return (username.equals(userDetails.getUserName())
        && ! isTokenExpire(toke));
}

Q.1 Frontend के लिए क्या किया है? / कौन-कौन सी service बनाई है

→ Auth service
→ http service
→ service locator
→ endpoint service

Q.2 Auth service क्या काम करती है?

→ Intercepts every request

Q.3 Auth service कैसे बनाई?

→ app.module.ts में providers की array में →

```
providers : [
    { provide : HTTP_INTERCEPTOR , useClass : AuthService
                multi : true
    },
```

Q. Auth service every request पर कैसे चलती है?

multi : true किया है।

Q. Code of AuthService →

```
@Injectable({
        providedIn : 'root' (← this class can be used in other class
})
export class AuthService implements HttpInterceptor {

constructor( private http: HttpServiceService) { }
intercept(req ( HttpRequest<any>, next: HttpHandler) {

if (localStorage.getItem('fname') && localStorage.getItem
                ('token')) {
    req = req.clone({
            setHeaders : { "withCredentials" : "true",
                           "name" : "Sonali",
            Authorization : this.http.getToken()
```

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

// creates a module that contains all directives
↑ given routes, router service

app.module.ts

It is configuration file of our project.
Components, Routers, Services, modules are imported and configured in this file.

Q. root component of your application?

→ app.component.ts file, contains root component class

```
@Component {
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent { }
```

Q. Change Front end port number?

temporary → ng serve --port 4202

parmanent → angular.json file में serve की array में key value pair में

```
"serve": {
    "options": { "port": "4201" },
```

services which are related to the application

cmd > ng g m mymodule
↳ mymodule.module.ts

```
@NgModule ({
    declarations : [ ],  ← Components
    imports : [
                         ← modules
    ],
    providers : [ ],     ← services
    bootstrap : [ ]      ← components
```

Q. Root folder of angular project ?

app → contains 6 files → app.routing.module.ts
→ app.component.css
→ app.component.html
→ app.module.ts
→ app.component.ts
→ app.component.spec.ts

app.routing.module.ts

In this file, we defines routes for our components

```
Const routes : Routes = [
    { path : ' ',
      redirectTo : 'dashboard',
      pathMatch : 'full'
    },
    { path : 'dashboard',
      Component : DashboardComponent
    },
```

Routes ⇒ represents a route configu. for Router service

Q. Change port no. of b.e
In Application properties
serve.port = 8080

npm - Node Package Manager
online repository for open-source Node.js packages
CLI is installed using NPM

CLI - Command Line Interface
used to generate component, proj and run the project

— : FrontEnd : —

Q. which Architecture used in front end ?
Angular - Component - service architecture
(internally it follows MVC)

Q. How to create angular project ?
→ node.js install
→ then create a folder, open it in VS code
→ then run cmd ⇒
> node -v
> npm -v
> npm install -g @angular/cli
[npm install -g @angular/cli@latest

Q. why we use Angular?
To create single page Application
Q. How? → through <router-outlet> in app.component.html

Q. How create Angular Project in FrontEnd?

>ng new projectname (>ng new angular10)
>ng generate component user ← create component
>ng g s servicename ← Service
>ng g m modulname ← module

A → is a web app that is presented to a user through single html and allows user to navigate diff. part without reload-

```
cors.allowedOrigins("http://localhost:4200");
cors.allowedHeader("*");
cors.allowCredentials(true);
}
};
return w;
}
}
```

=> @SpringBootApplication - indicates a configuration class that declares one or more Bean methods (pkg-org.sf.boot.autoconfigure)

It is combination of
- @Configuration
- @EnableAutoConfiguration
- @ComponentScan

WebMvcConfigurer - Interface, part of spring MVC framework in Java. It is used to configuring various aspects of spring MVC web app such as adding custom interceptors, resource handlers, message converters, view resolvers etc.

CORS - Cross Origin Resource Sharing to configure cors behaviour for your applications RestfulAPIs or to specify allowed origin, methods, headers etc.

addCorsMappings() - allows to define CORS config

CorsRegistry - is a class of spring framework that allows to configure CORS setting for your web app.

spring.mail.properties.mail.smtp.auth = true
spring.properties.mail.smtp.starttls.enable = true

Backend ने project को create किया - IoInitializer से (online)

Q. How to deploy web application on external web server?

startup class extends by SpringBootServletInitializer

Startup class → ProjectOrsApplication

```
package com.ways;
imports ...

@SpringBootApplication
public class ProjectOrsApplication extends springBootServlet
                                        Initializer {
@Autowired
private Environment env;

@Autowired
FrontCtl frontCtl;

public static void main(String[] args) {
  SpringApplication.run(ProjectOrsApplication.class, args);
}
```

a class, used to launch a spring app from java main method

```
@Bean
public WebMVcConfigurer corsConfigurer() {

  WebMvcConfigurer w = new Web... ();

  public void addCorsMappings(CorsRegistry registry)
    CorsRegistration cors = registration
```

Q. Application.properties ?
Spring Boot keeps application configurable parameter into application.properties file (in root folder)
Parameters ⇒
1) server.port = 8084
2) DCP
3) Pagination ⇒ page.size = 5
4) Email
5) JPA

DCP properties -
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/ors?allowPublicKeyRetrieval=true & useSSL = true
spring.datasource.username = root
. password = root
Jwt.secret = RaysTech
[SSL = Secure socket language]

JPA properties -
spring.jpa.show-sql = ~~true~~ false
spring.jpa.properties.hibernate.format-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDB53Dialect
(Default dialect)

Email -
Spring.mail.host = smtp.gmail.com
spring.mail.port = 587
. username = Sonali
. password =

Ronali Nvantika Yadav
Rupali
Patel

JSonWeb Token
JWT

Project 10 JSON → JavaScript Object Notation

BackEnd
Spring Boot – used to create RestfulAPIs
– microservices (communicate b/w 2 server and api

Q.1 What is RESTful API ?
Representational State Transfer API (RESTful API) is an architectural style for designing networked application.

RESTful APIs provide stateless and flexible approach to build web services to communicate between different server over standard protocol (interact) (HTTP)

API - Application Programming Interface

Q. which architecture is used ?
Spring-Boot – Layer Architecture
↓
4 Layers

Presentation layer	Business layer	Data Access layer	Integration L
↓	↓	↓	↓
DTO design pattern	Bridge design P.	DTO	DTO
Front Controll design	DTO design pat.	Bridge	Builde
command design pat.	Session Faced	DAO desi	
↓			
navigates by URL calls specific controller)			

Q. which module is applied at each layer ?
Presentation Layer – Angular
Business Layer – Spring AOP
Data Access Layer – Spring DAO, Spring ORM
Integration layer – Spring mail, Spring security

- CriteriaQuery के obj cq में hold.
- cq.from() में getDTOClass() pass की Root के obj qRoot में hold and cq.select में qRoot pass किया
- getWhereClause() call से dto, builder का obj, qRoot pass, list<predicate> type की के obj whereClause में hold.
- getWhereClause BaseDAOImpl पर abstract() बनाई, इसे userDAOImpl में Override किया
  यहां list का obj whereCondition बनाया and condition check की -

```
if(!EmptyString(dto.getFirstName()))
  whereCondition.add(builder.like(qRoot.get("firstName
      , dto.getFirstName()+"%"));
}

if(!EmptyString(dto.getRoleName()))
  whC.add(b.like(qRoot.get("roleName"), dto.getRoleNam
          ()+"%"));
}

if(!isNotNull(dto.getDob()))
  whereCond.add(builder.equal(qRoot.get("dob"), dto.
          getDob()));
}
```

- whereCondition को return किया
- → list के obj whereClause में hold then
- → eq.where(whereClause.toArray(New Predicate[whereClause.size()])); pass किया।
- → entityManage की createQuery() में cq pass, TypedQuery obj query में hold, query return

11) Frontend में callback() में res को get और res.result data को form.List में hold

12) html पर ngModel से form.searchParams.firN. get और pri.

Q. List alphabetical order में कैसे आ रही है?

→ BaseDAOImpl की createCriteria() में getOrderByClause List<order> type की method बनाई।

→ इसमें LinkedHashMap का obj map बनाया और इसमें dto.orderBy() को hold

→ List<order> का obj orderBys बनाया

→ map.forEach() में key, value pass की, condition check if (value.equal("asc")) है तो orderBys.add(builder.asc(qRoot.get(key))); और else में orderBys.add में builder.desc() में qRoot.get(key)) pass किया,

→ orderBys को return.

→ then createCriteria() में getOrderByClause() call, dto, builder and qRoot pass orderBy obj में hold eg. orderBy() में orderBys.toArray() में new order[orderBys.size()])) pass

Code:-

```
Protected List<order> getOrderByClause(T dto, CriteriaBuilder
        builder, Root<T> qRoot) {

LinkedHashMap<String, String> map = dto.orderBy();
List<order> orderBys = new ArrayList<order>();
map.forEach(key, value) -> {
        if (value.equals("asc")) {
            orderBys.add(builder.asc(qRoot.get(key)));
} else
    {
        orderBys.add(builder.desc(qRoot.get(key)));
}
return orderBys;
}
```