**PROJECT REPORT**

**Project Name : Student Management System**

Cource : **PG-DAC**

**Team Members :**

* Jubin Joy 200250120037
* Patil Harshdeep Gopal 200250120061
* Chhangani Pallavi Kamalkishor 200250120022
* Kondaveeti Annapurna Chowdary 200250120042
* Nutan BAlkrishna Choudhari 200250120060

**Project Details :**

* Student Management System provides the platform to the students where they can get all information related their appearance in the collage.
* It has three log-in types having different functions to do-
  + ADMIN
  + FACULTY
  + STUDENT
* ADMIN : Functions admin can do are,
  + Add User
  + View User
  + Add Notification
  + View Notification
* FACULTY : Functions faculty can do are,
  + View Notification
  + Add Assignment
  + View Assignment
  + Add Attendence
  + View Attendence
* STUDENT : Functions student can do are,
  + View Notification
  + View Assignment
  + View Attendence

**Login Scenario :**

* Presentation layer,
  + It executes login.component.html
  + When we enter email and password and click on LogIn button it will go to login.component.ts
  + You can inject a service into a component, giving the component access to that service class, use the @Injectable() decorator to provide the metadata that allows Angular to inject it into a component as a dependency.
  + When we get user from Backend, control will go to admin/faculty/student page , according to method login()
* Service layer,
  + We uses the UserService Interface for the service layer.
  + We will implement this UserService in UserServImpl class to pass parameters to DAO layer.
  + It will return the User to the controller given by DAO.
* DAO layer,
  + The service layers delegates the actual verification and database call to the DAO layer.
  + The DAO class is responsible for two concepts. Encapsulating the details of the persistence layer and provide a CRUD interface for a single entity.
  + We are using UserRepository interface which extends JpaRepository which  allows us to access and persist data between Java object/ class and relational database.
  + And returns the User to service layer.

**Scenario where we got Struck :**

* Bidirectional One-to-Many Relationship 🡪
  + @JsonIgnore is used at field level to mark a property or list of properties to be ignored.
  + @JsonView annotation can be used to include/exclude a property during the serialization and deserialization process dynamically. We need to configure an ObjectMapper class to include the type of view used for writing a JSON from Java object using the writerWithView() method.
  + **@JsonManagedReference** and **@JsonBackReference annotations** can be used to create a JSON structure in a **bidirectional**way. The **@JsonManagedReference** annotation is a **forward reference** that includes during the serialization process whereas **@JsonBackReference**annotation is a **backreference**that omits during the serialization process.
* Constraints voilataion problem 🡪

The CascadeType.MERGE is inherited from the CascadeType.ALL setting, so we only have to merge the Post entity and the associated PostDetails is merged.

**Learnings during Project :**

* Sending Email using javamail API --

There are following three steps to send email using JavaMail. They are as follows:

* Get the session object that stores all the information of host like host name, username, password etc.
* compose the message
* send the message
* Upload and download Files –

In this section, we are going to use Spring Boot to build a backend API that exposes three REST endpoints:

* POST /upload to upload one single file.
* POST /upload to upload multiple files.
* GET /files to download a file.
* Angular -- Angular is a platform and framework for building single-page client applications using HTML and TypeScript.
  + Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
  + Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.