**GotNote Project**

This project manages notes for multiple users. It performs below operations:

1. Save note
2. Update note
3. Delete single note and all notes
4. Fetch single note and all notes information

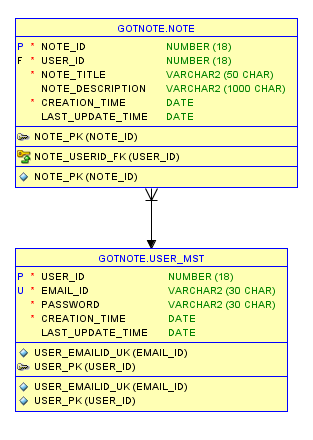
Below technologies are being utilized to develop this project

1. **Spring MVC** : To implement MVC design pattern
2. **Spring REST Web services** : To expose REST API to client
3. **Spring security** : To implement http basic authentication to protest REST API
4. **JPA :** JPA is used as an ORM tool.

Database information

**USER\_MST** : this table store user information

**NOTE** : this table store note information



One user can have multiple notes, so there is one to many relationship between these tables.

**Role of different classes**

**Entity Layer**

**Note.java** : This is JPA entity class for NOTE table.

**UserMst.java** : This is JPA entity class for USER\_MST table.

**NoteDto.java** : This class represent data transfer object for transferring data between controller and service layer.

**CustomUser.java** : This class extends spring security user object to store the authenticated user information. userId is required to perform CRUD operations on notes so this class is populated after successful authentication.

**Controller Layer**

**NoteRestController** : This is rest controller. This intercepts each REST request which is send by client.

This controller provides various methods to handle different HTTP (GET, POST, PUT, DELETE) request and perform CRUD operation on note object.

**Service Layer**

**NoteService** : This interface provide various methods for performing CRUD operations.

**NoteServiceImpl** : This is the implementation class of NoteService interface which provides implementation of CRUD methods. This class takes input from controller layer using NoteDto and calls different repository methods.

**DAO Layer**

**NoteRepository** : This extends JpaRepository so default implementation is already available. For specific functionalities different methods are added in this.

**Security Layer**

**ApplicationSecurity.java** : This class configures spring security for whole project. This enables http basic authentication to protect REST API from unauthenticated users. This uses AuthenticationProvider to authenticate user.

**CustomAuthenticationProvider** : This is the authentication provider class. This class fetches user using email id and password. If valid user exist in the database then CustomUser is populated with required information otherwise BadCredentialsException is thrown.