**Approach Taken:**

**Architecture:**

I have used 2-Tier architecture where I have divided entire project in Data Access Object (now referred to as DAO) and Controller layer. All the database CRUD operations are performed in DAO layer. This layer also contains a logic to connect with database.

Second layer is Service layer where I have defined all the controllers and logics related to same. Service layer will communicate with DAO layer to perform operations on database tables.

**Design Patterns Used:**

I have used below design patterns for developing this project:

1. Dependency Injection:

I have not created objects of any classes manually. I have used spring Autowired concept for dependency injection. All the classes are annotated with Component for this purpose.

1. Factory Method:

I have used factory method by creating bean factory of all classes and using these bean as per project requirements.

1. Null Object:

I have tried to avoid null objects at all points where objects are created. This is to avoid any method throwing a null pointer exception.

**Extra stuffs I would have done if I had more time:**

1. Security:

Below are security modules I would like to implement if I had more time.

* I would have implemented 2 factor authentications for login.
* I am storing password in regular text format in database. I would have used 1 way encryption method to increase security which means I would add salt (a random text) in password at random position of password and encrypt the password. Encrypted password and salt would be saved in database table. I would then use this salt for authentication every time to login but will never decrypt at any point for security point of view.
* I would have implemented authorization feature like OAuth2 for login.

1. Separate Service layer and Controller layer:

If I have had more time, I would have separated service layer and controller layer in project.

1. Separate bean classes for DAO and Service layer:

If I have had more time, I would have separated bean classes for each layer so there would not be any dependencies on each layer.

1. Well-designed UI:

If I have had more time, I would have created a well-designed UI in either Angular or Node JS.

1. More testing:

I would have performed integration testing and have had performed more unit test for methods in all layers of my project.

1. Caching:

I would have used caching for some tables in database to decrease overall calls to database.