**Revise**

**1. Hibernate Architecture**

**2. Create Maven based standalone hibernate project**

**Refer - Regarding Maven.txt**

3. Edit DB name n password , from hibernate.cfg.xml

4. Create HibernateUtils , to build SessionFactory

**5. Refer to E-R of Blogs management system**

5.1 users table

columns - id , first name , last name, email ,password , dob , registration amount,role,image

We are going to confirm auto table creation

POJO ---> Table approach

Create POJO class : User

**5.2 POJO Annotations**

**Package : javax.persistence | jakarta.persistence**

**@Entity : Mandatory : cls level**

**@Id : Mandatory : field level or property (getter) : PK**

**Optional annotation for further customization :** @Table(name="tbl\_name) : to specify table name n more

@GeneratedValue : to tell hibernate to auto generate ids

auto / identity(auto incr : Mysql) / table / sequence(oracle)

eg : @Id => PK

@GeneratedValue(strategy=GenarationType.IDENTITY) => auto increment

@Column(name,unique,nullable,insertable,updatable,length,columnDefinition="double(8,2)") : for specifying col details

@Transient : Skipped from persistence(no col will be generated in DB table)

@Temporal : java.util.Date , Calendar , GregorianCalendar

LocalDate(date) ,LocalTime(time) , LocalDateTime (timestamp/datetime) : no temporal anno.

@Lob : BLOB(byte[]) n CLOB(char[]) : saving / restoring large bin /char data to/from DB

@Enumerated (EnumType.STRING): enum (def : ordinal : int)

6. **Add <mapping class="F.Q POJO class name"/> in hibernate.cfg.xml**

**7. Create DAO i/f & write its implementation class**

**Hibernate based DAO implementation class**

**7.1 No data members ,constructor , cleanup**

**7.2 Directly add CRUD methods.**

**Steps in CRUD methods**

**1. Get hib session from SF**

**API of org.hibernate.SessionFactory**

**public Session openSession() throws HibernateException**

**OR**

**public Session getCurrentSession() throws HibernateException**

**2. Begin a Transaction**

**API of Session**

**public Transaction beginTransaction()throws HibernateException**

**3. try {**

**perform CRUD using Session API (eg : save/get/persist/update/delete/JPQL...)**

**commit the tx.**

**} catch(RuntimeException e)**

**{**

**roll back tx.**

**re throw the exc to caller**

**} finally {**

**close session --destroys L1 cache , pooled out db cn rets to the pool.**

**}**

**4 Refer to Hibernate Session API**

**(hibernate api-docs & readme : hibernate session api)**

**5. Create main(..) based Tester & test the application.**