A.I.S.S.M.S
COLLEGE OF ENGINEERING
NO - 1, Kennedy Road, Near R.T.O Pune - 411001

Experiment - 10.

- **Aim:** Design and implement a business interface with necessary business logic for any web application using EJB.

  eg: Design and implement the web application logic for deposit and withdrawl amount transaction using EJB.

- **Theory:**

I] Introduction to Enterprise JavaBeans (EJB).

- Enterprise JavaBeans (EJB) is a serverside component based architecture for building scalable, secure, and transactional enterprise applications.

- EJB is particularly useful for applications that require business logic execution, distributed computing and database interactions.

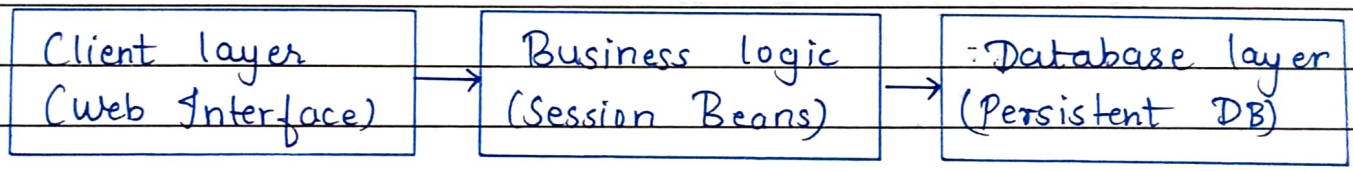
II] Why use EJB ?

For a transaction system like deposit and withdrawl, we need :-

A.I.S.S.M.S
COLLEGE OF ENGINEERING
NO - 1, Kennedy Road, Near R.T.O Pune - 411001

- Security - Transactions must be secured against unauthorized access.
- Concurrency management - Multiple users may deposit / withdraw simultaneously.
- Data Persistence: Transaction details must be stored in a data base.
- Transaction Management - To ensure consistency, transactions must follow ACID properties.

III] EJP Architecture / flow -



- Components we will use -

1. Client (JSP / Servelets / Web UI) - Allows users to input deposit / withdrawl details.

2. Session Beans (EJB business logic layer) - Implements deposit and withdrawl logic.

3. Entity Beans - (Database Handling) - Manages persistent storage for transactions.

4. JNDI (Java naming and Directory Interface) - Allows clients to locate and access EJB components.

5. Transaction Manager - Ensures deposit / withdraw transactions are automic.

* Steps to design and Implement EJB - Based application -

STEP 1: Setting up the EJB Environment.

- Install Java EE and EJB container (Wildfly, Glassfish, JBoss) etc.
- Configure database (MySQL, Oracle, etc) for storing transactions records.
- Create a JDBC connection - between EJB and database.

STEP 2: Designing the Bissiness Interface for transactions.

The interface will declare methods such as -

```
public interface BankingService {
    void deposit (double amount, int accountID);
    void withdraw (double amount, int accountID);
    double getBalance (int accountID);
}
```

STEP 03: Implementing the Business logic using Session beans.

These are two types of session beans -

1. Stateless beans (Best for transaction processing)
2. Stateful beans (Useful for maintaining user session

A.I.S.S.M.S
COLLEGE OF ENGINEERING
NO - 1, Kennedy Road, Near R.T.O Pune - 411001

48.

Business logic for -

(1.) Deposit logic -

- Check if the account exists.
- Add the deposited amount to existing balance.
- Commit the transaction to update database.

(2). Withdrawl logic -

- Verify if account exists.
- Check if balance is sufficient
- Deduct withdrawl amount.
- Commit the transaction

STEP 4: Database connectivity using entity beans.

```
@ Entity
public class Account {
        @Id
        private int accountID;
        private string accountHolderName;
        private double balance;
        // getters and setters.
}
```

STEP 5: Managing Transactions using EJB.

EJB provides automatic transaction Management using annotations.

@Transaction Attribute.

```
public void deposit (double amount, int accountID)
{
    Account acc = entityManager.find (Account.class,
                  accountID);
    acc.setBalance (acc.getBalance () + amount);
    entityManager.merge (acc);
}
```

If transaction fails system automatically rolls back to prevent data corruption.

STEP 6: Exposing EJB Service to clients

EJB functions are called from a web interface (JSP/ Servelets).

1. User enters account ID and amount in a JSP form.
2. The request is sent to a Servelet.
3. Servelet invokes EJB methods deposit() or withdraw().
4. EJB updates the database.
5. Response is sent back to user.

Conclusion: Thus we successfully implemented business logic using EJB.

* * * * * * *