Stateless Session Bean - Full Marks Answer

1. Definition:

A Stateless Session Bean is an enterprise bean that does not maintain any conversational state with the client. Each method invocation is independent, making it ideal for lightweight services.

2. Use Case Example:

We create a CriticBean which returns critic messages based on a subject like movie, book, etc. This is stateless behavior as it does not depend on user sessions.

3. Interface Code (Optional):

```
package com.example.critic;

import javax.ejb.Local;

@Local
public interface CriticBeanLocal {
    String getCriticMessage(String subject);
}
```

4. Stateless Session Bean Code:

```
package com.example.critic;

import javax.ejb.Stateless;

@Stateless
public class CriticBean implements CriticBeanLocal {
    public String getCriticMessage(String subject) {
        if (subject == null || subject.trim().isEmpty()) {
            return "Hmm, I can't critique what I don't know!";
        }
        switch (subject.toLowerCase()) {
            case "movie":
                return "The storyline was weak, but the visuals were stunning!";
            case "book":
                return "A decent read, though the character development could use work.";
            case "code":
                return "Clean structure, but optimization is needed.";
            default:
                return "Interesting choice, but I expected more depth.";
        }
    }
}
```

5. Test Client (Optional):

```
package com.example.critic;

import javax.naming.Context;
import javax.naming.InitialContext;

public class CriticClient {
    public static void main(String[] args) throws Exception {
        Context ctx = new InitialContext();
        CriticBeanLocal bean = (CriticBeanLocal) ctx.lookup("java:global/YourAppName/CriticBean");
```

```
        System.out.println(bean.getCriticMessage("movie"));
    }
}
```

6. Output:

The storyline was weak, but the visuals were stunning!

7. Conclusion:

This Stateless Session Bean is suitable for generating feedback messages in a scalable, reusable way. It does not depend on user session or previous interactions, making it ideal for microservices and chatbots.