Encryption - Decryption

1. AES Encryption Utility

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class AESUtil {
    private static final String ALGORITHM = "AES";
    private static final String SECRET_KEY = "1234567890123456";
    //this SECRET_KEY should be of 16 chars long and need to be mentioned in
yml file;
    public static String encrypt(String input) {
        try {
            SecretKeySpec key = new SecretKeySpec(SECRET_KEY.getBytes(),
ALGORITHM);
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, key);
            byte[] encrypted = cipher.doFinal(input.getBytes());
            return Base64.getEncoder().encodeToString(encrypted);
        } catch (Exception e) {
            throw new RuntimeException("Encryption error", e);
        }
    }
}
```

2. Creating `@Encrypt` Annotation

```
import java.lang.annotation.*;
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Encrypt {

}
```

3. Custom Deserializer

```
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import java.io.IOException;

public class EncryptedStringDeserializer extends JsonDeserializer<String> {
    @Override
    public String deserialize(JsonParser p, DeserializationContext ctxt)
throws   IOException {
    String originalValue = p.getValueAsString();
    return AESUtil.encrypt(originalValue);
    }
}
```

**4. Using the Deserializer in Model

```
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
public class UserDTO {
private Long id;
private String firstName;
private String lastName;

@JsonDeserialize(using = EncryptedStringDeserializer.class)
@Encrypt
private String mobileNumber;
private String email;
private String gender;
private String password;

}
```

OUTPUTS

1. Json used :```

```
{
  "firstName": "Aditya",
  "lastName": "Prasad",
  "mobileNo": "9876543210",
  "email": "aditya@example.com",
  "gender": "Male",
```

```
  "password": "mypassword"
}
```

2. Data stored in db

    ```
    id: 302
    email: aditya@example.com
    first_name: Aditya
    gender: Male
    last_name: Prasad
    mobile_no: LAIoetV8c1heVprIoUvmZA==
    password: $2a$10$1XvVhQSwRA0jEMLiTBv5V.xbcT3EZBQ7P0qsp0.Stg5mQSkmMrLRe
    ```

3. After decryption
```json
{
    [
      {
        "id": 302,
        "firstName": "Aditya",
        "lastName": "Prasad",
        "mobileNo": "9876543210",
        "email": "aditya@example.com",
        "gender": "Male",
        "password":
"$2a$10$1XvVhQSwRA0jEMLiTBv5V.xbcT3EZBQ7P0qsp0.Stg5mQSkmMrLRe",
        "trips": []
      }
    ]
}
```