SOFE 4840 Winter 2022 - Software & Computer Security

Project Phase 3: Verification & Testing

Project Topic: Cloud Security

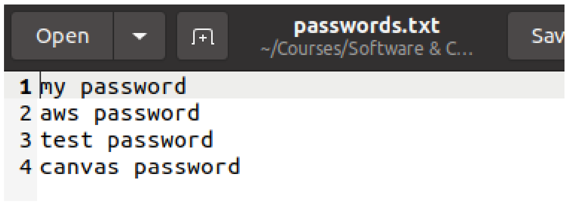
Project Group No: 16

Group Members:

Ashwin Shanmugam - 100700236
Tegveer Singh - 100730432
Mihirkumar Patel - 100702168
Esam Uddin - 100711116

Discussion and Results

Testing Data Encryption using AWS KMS

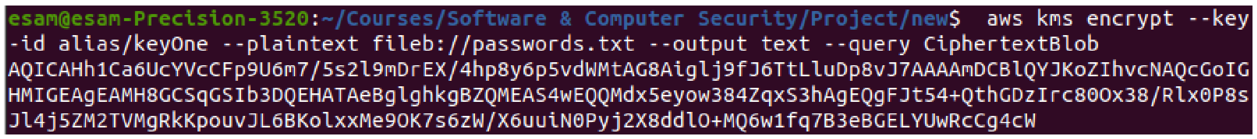


```
my password
aws password
test password
canvas password
```

Figure 1.1 - Passwords.txt data file

Data Encryption

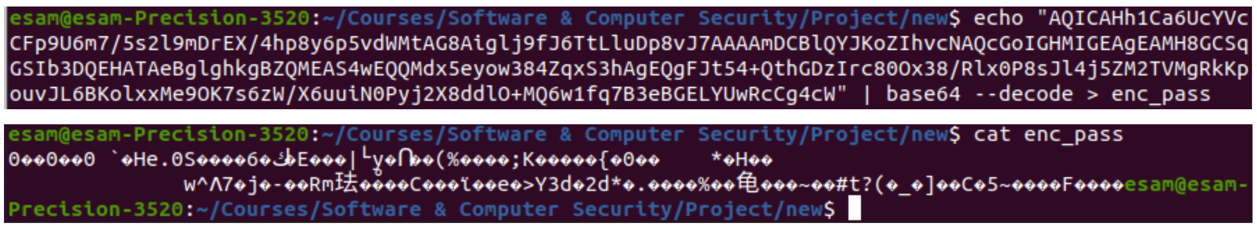
Using aws kms encrypt operation to encrypt passwords.txt file:



```
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$  aws kms encrypt --key
-id alias/keyOne --plaintext fileb://passwords.txt --output text --query CiphertextBlob
AQICAHh1Ca6UcYVcCFp9U6m7/5s2l9mDrEX/4hp8y6p5vdWMtAG8Aiglj9fJ6TtLluDp8vJ7AAAAmDCBlQYJKoZIhvcNAQcGoIG
HMIGEAgEAMH8GCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMdx5eyow384ZqxS3hAgEQgFJt54+QthGDzIrc80Ox38/Rlx0P8s
Jl4j5ZM2TVMgRkKpouvJL6BKolxxMe9OK7s6zW/X6uuiN0Pyj2X8ddlO+MQ6w1fq7B3eBGELYUwRcCg4cW
```

Figure 1.2 - AWS KMS encrypt operation

Decoding generated ciphertext and saving in enc_pass file:



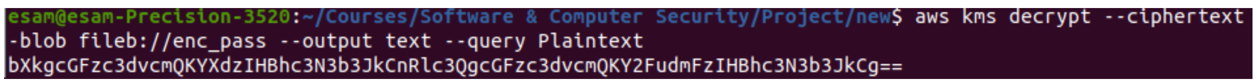
```
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ echo "AQICAHh1Ca6UcYVc
CFp9U6m7/5s2l9mDrEX/4hp8y6p5vdWMtAG8Aiglj9fJ6TtLluDp8vJ7AAAAmDCBlQYJKoZIhvcNAQcGoIGHMIGEAgEAMH8GCSq
GSIb3DQEHATAeBglghkgBZQMEAS4wEQQMdx5eyow384ZqxS3hAgEQgFJt54+QthGDzIrc80Ox38/Rlx0P8sJl4j5ZM2TVMgRkKp
ouvJL6BKolxxMe9OK7s6zW/X6uuiN0Pyj2X8ddlO+MQ6w1fq7B3eBGELYUwRcCg4cW" | base64 --decode > enc_pass
```

```
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ cat enc_pass
0��0��0 `�He.0S����6��Ԫ�E���|└y�∩��(%����;K�����{�0��     *�H��
              w^Λ7�j�-��Rm珐����C���ĭ��e�>Y3d�2d*�.����%��龟���~��#t?(�_�]��C�5~����F����esam@esam-
Precision-3520:~/Courses/Software & Computer Security/Project/new$
```

Figure 1.3 - Decoding Process

Data Decryption

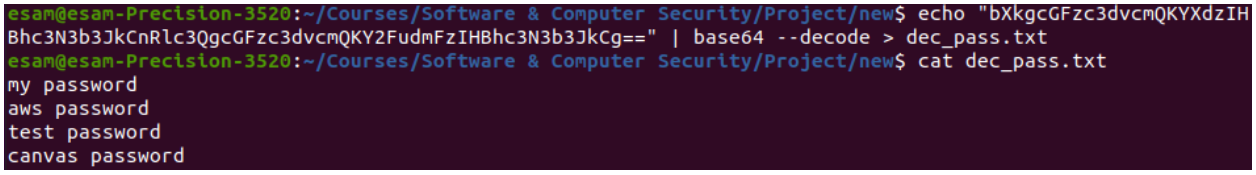
Using aws kms decrypt operation to decrypt the data file:



```
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ aws kms decrypt --ciphertext
-blob fileb://enc_pass --output text --query Plaintext
bXkgcGFzc3dvcmQKYXdzIHBhc3N3b3JkCnRlc3QgcGFzc3dvcmQKY2FudmFzIHBhc3N3b3JkCg==
```

Figure 1.4 - AWS KMS decrypt operation

Decoding decrypted ciphertext and saving in dec_pass.txt file:



```
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ echo "bXkgcGFzc3dvcmQKYXdzIH
Bhc3N3b3JkCnRlc3QgcGFzc3dvcmQKY2FudmFzIHBhc3N3b3JkCg==" | base64 --decode > dec_pass.txt
esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ cat dec_pass.txt
my password
aws password
test password
canvas password
```

Figure 1.5 - Decoding Process 2

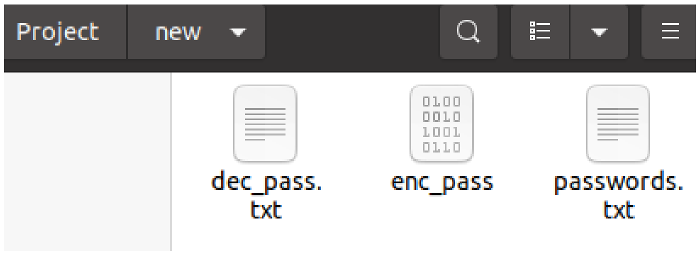


Figure 1.6 - Generated Files

Following is the general acceptance test for the encryption and decryption of all data files in our application (.txt type) using AWS KMS (Key Management Service)

—---

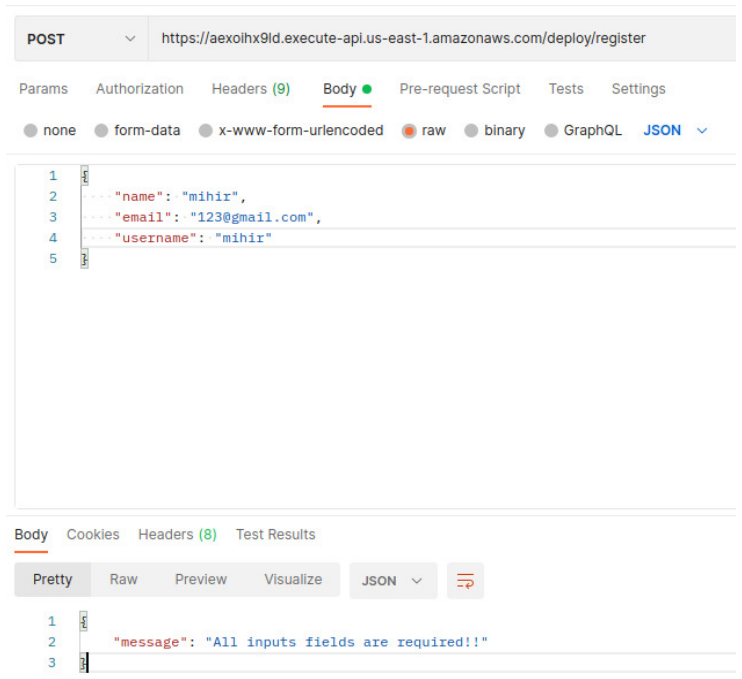
Title: Data Encryption and Decryption using AWS KMS
ID: DED-01
Preconditions: all data files saved including passwords.txt, usernames.txt
Input: encrypt operation on data files: *.txt (save encrypted files)
Output: decrypt operation on data files: *.txt (see original data files)
Process: Using the aws kms encrypt operation, generate encrypted ciphertext and decode it, store the decoded ciphertext into a file. To retrieve the encrypted file, simply perform the aws kms decrypt operation, decode the generated decrypted text.
Result: PASS

—---

Testing AWS Lambda Functionalities

Our login page is built using multiple Amazon web services (AWS) such as Lambda, AWS CLI, API Gateway and DynamoDB. These AWS services were used to prevent security threats and vulnerabilities to cloud applications through attacks on REST APIs. In order to test the backend functionalities of our login system, the API testing application postman was used. A secure token was generated from AWS to access each API endpoint, to ensure only legitimate users can access the web pages.

If all of the input fields on the registration page are not filled out, an error message is displayed on the webpage. In the below screenshot, we can see that the field password was not mentioned by the user and an error message stating " All input fields are required " is displayed.



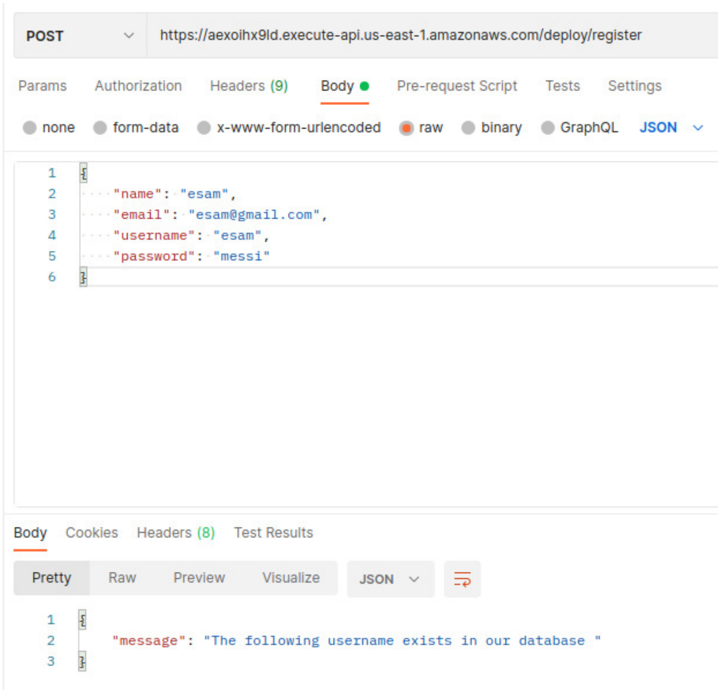
https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/register

```
{
    "name": "mihir",
    "email": "123@gmail.com",
    "username": "mihir"
}
```

```
{
    "message": "All inputs fields are required!!"
}
```

Figure 2.1 Functional test 8 ←[verification of inputs variables]

If a user tries to register with a username that already exists in the database, an error message is displayed on the webpage. In the below screenshot, we can see that the "username:esam" exists in the database and the message "The following username exists in our database" is displayed.



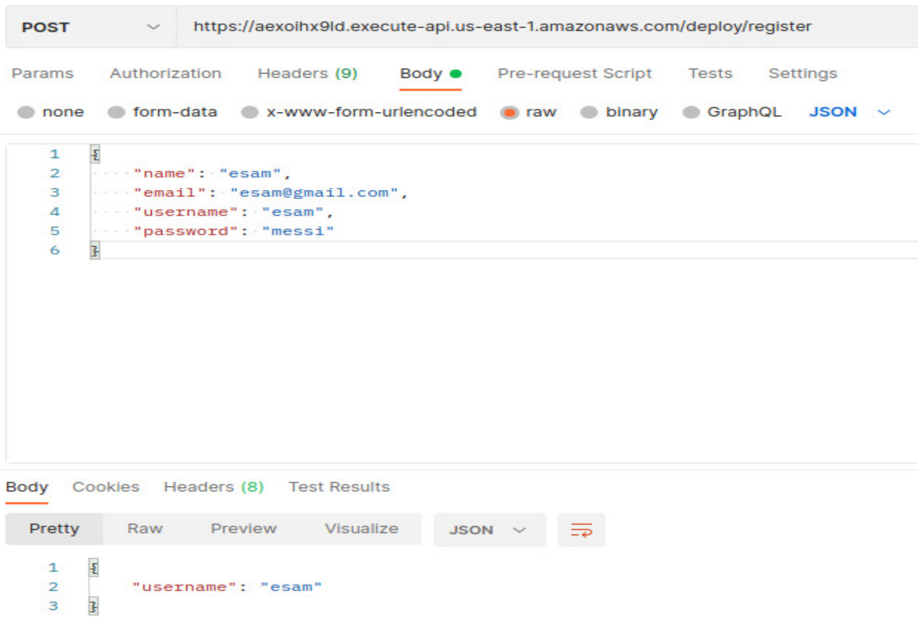
https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/register

```
{
    "name": "esam",
    "email": "esam@gmail.com",
    "username": "esam",
    "password": "messi"
}
```

```
{
    "message": "The following username exists in our database "
}
```

Figure 2.2 Functional test 9 ←[verification of data redundancy]

If all the input fields are provided by the user and there exists no same username on the database during registration, then the user will successfully be able to register and their information will be stored in the database (Dynamodb) and the username is printed in the backend to test the functionality. The below screenshot demonstrates the registration of the user "esam".



https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/register

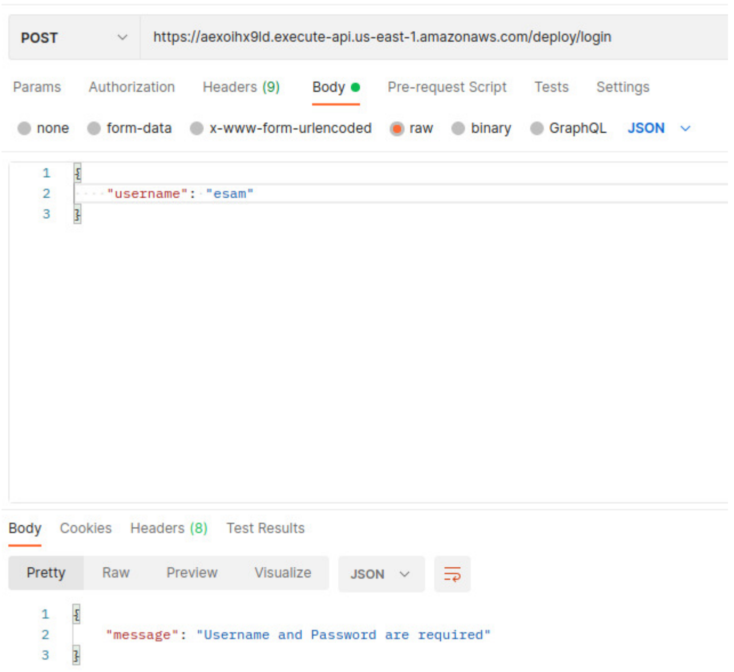
```
{
    "name": "esam",
    "email": "esam@gmail.com",
    "username": "esam",
    "password": "messi"
}
```

```
{
    "username": "esam"
}
```

Figure 2.3 Functional test 10 [Successfully registration of user]

If either username or password isn't provided by the user, the system will not be able to authenticate their identity and an error message is displayed. The below screenshot demonstrates the attempt of a user trying to login with the password field empty.

https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/login



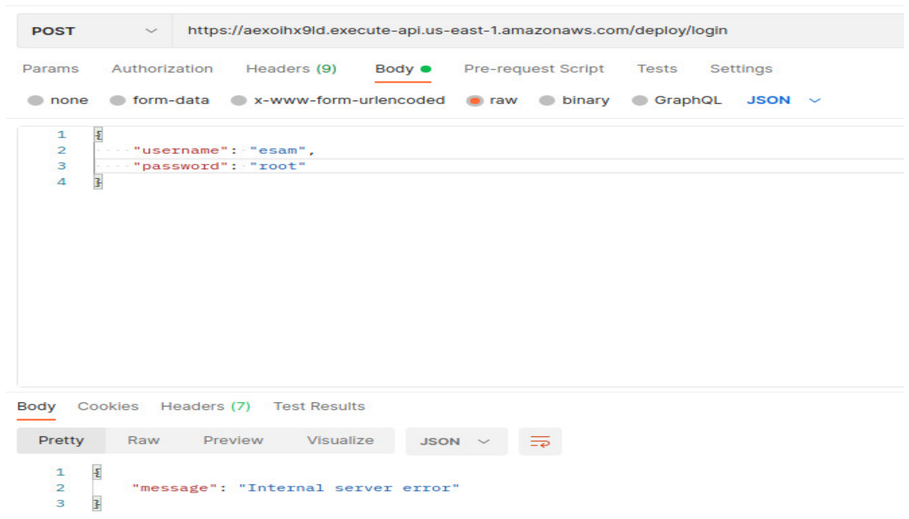
https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/login

```
{
    "username": "esam"
}
```

```
{
    "message": "Username and Password are required"
}
```

Figure 2.4 Functional test 11 [Check if both the fields are provided by user]

If both the username and password entered by the user are valid, the system should be able to log them into the system. During our test, this functionality failed and prints the following error message " Internal Server error ". The database was not able to query the user information.



```
https://aexoihx9ld.execute-api.us-east-1.amazonaws.com/deploy/login
```

```
{
    "username": "esam",
    "password": "root"
}
```

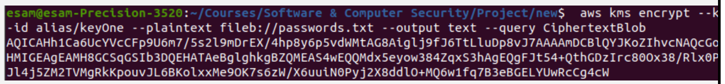
```
{
    "message": "Internal server error"
}
```

Figure 2.5 Functional test 12 [Verification of login credential]

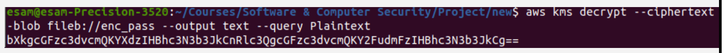
Along with the above tests, other functional tests were carried out to make sure the final project is working as per the functionalities specified in Phase 1 and Phase 2. Following is a list of functional tests that were run as part of the verification and testing process as well as a description of the degree with which they were satisfied.

<u>**Table 3.1 Functional Tests**</u>

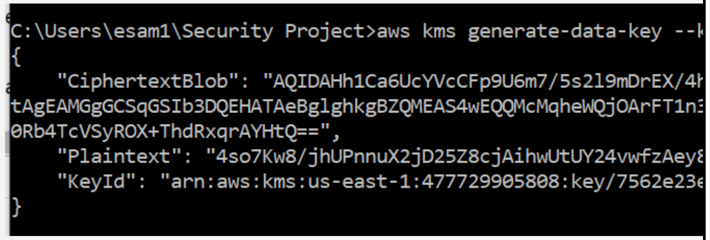
Test Title and ID	**Purpose**	**Pre conditions**	**Inputs, Output and Process**

FT-01	To check whether the data file is encrypted successfully.	Data files to be encrypted are saved in the local system.	This functional requirement was completely *satisfied* since the data file was encrypted successfully. **NOTE:** The encryption process was performed using aws kms encrypt operation as shown above. Following is a screenshot of the encryption:  `esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$  aws kms encrypt --k`
			`-id alias/keyOne --plaintext fileb://passwords.txt --output text --query CiphertextBlob`
			`AQICAHh1Ca6UcYVcCFp9U6m7/5s2l9mDrEX/4hp8y6p5vdWMtAG8Aiglj9fJ6TtLluDp8vJ7AAAAmDCBlQYJKoZIhvcNAQcGo`
			`HMIGEAgEAMH8GCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMdx5eyow384ZqxS3hAgEQgFJt54+QthGDzIrc80Ox38/Rlx0P`
			`Jl4j5ZM2TVMgRkKpouvJL6BKolxxMe9OK7s6zW/X6uuiN0Pyj2X8ddlO+MQ6w1fq7B3eBGELYUwRcCg4cW`

FT-02	To check whether the data file is	Encrypted data files are saved in	This functional requirement was completely *satisfied* since the encrypted data file was decrypted successfully.

	decrypted successfully.	the local system.	**NOTE:** The decryption process was performed using aws kms decrypt operation as shown above. Following is a screenshot of the decryption:
			
			`esam@esam-Precision-3520:~/Courses/Software & Computer Security/Project/new$ aws kms decrypt --ciphertext`
			`-blob fileb://enc_pass --output text --query Plaintext`
			`bXkgcGFzc3dvcmQKYXdzIHBhc3N3b3JkCnRlc3QgcGFzc3dvcmQKY2FudmFzIHBhc3N3b3JkCg==`

FT-03	The original data files should be retrieved after the encryption and decryption process.	Run both aws kms encrypt and decrypt operations.	As shown in the screenshots in FT-01 and FT-02, the data file was encrypted and decrypted successfully. After the decrypt operation was performed, the original data file was retrieved.

FT-04	Generate data keys if data files have a size larger than 4KB.	There is a necessity to use data keys rather than customer managed keys (data file size is larger than 4KB)	Once the aws cli was installed in our local system, aws was configured with the appropriate access key id and the secret key. The command in the screenshot below shows that the data key is generated successfully:
			
			`C:\Users\esam1\Security Project>aws kms generate-data-key --k`
			`{`
			`    "CiphertextBlob": "AQIDAHh1Ca6UcYVcCFp9U6m7/5s2l9mDrEX/4h`
			`tAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMcMqheWQjOArFT1n3`
			`0Rb4TcVSyROX+ThdRxqrAYHtQ==",`
			`    "Plaintext": "4so7Kw8/jhUPnnuX2jD25Z8cjAihwUtUY24vwfzAey8`
			`    "KeyId": "arn:aws:kms:us-east-1:477729905808:key/7562e23e`
			`}`

FT-05	Verify that both the Key ID and the Key Alias can be used to perform aws kms operations.	Aws cli should be installed and configured in the local machine.	This requirement is satisfied because both the key id and the alias name work for aws kms operations.
FT-06	Check whether the decoded data key is encrypted	Ensure OpenSSL is installed in the local machine.	In this process, an alternative approach was taken in which the data key was encrypted using OpenSSL. It can be said that this specific requirement was somewhat satisfied because the data key was encrypted. However, the original file wasn't retrieved

	using OpenSSL.		since the decryption process failed.
FT-07	Check whether the decoded data is decrypted using OpenSSL.	Encrypted data key using OpenSSL.	An alternative approach of decryption was applied using OpenSSL. This requirement was not satisfied because this decryption method was deprecated.
FT-08	To check if inputs for all the fields are provided by the user when they register.	User has access to a page with input fields.	**Refer to Figure 1.1:** When a user completes the registration process, it is critical to verify that all input fields are completed. The data must then be reviewed to ensure that it meets the datatype criterion and that no SQL queries are inserted. The functional requirement was satisfied because it was able to display the error message in the console that not all fields were provided to the user.
FT-09	To ensure the database does not have any repeated usernames.	The database has all user info stored.	**Refer to Figure 1.2:** The functional criterion was met because the system produced an error message stating that the entered username already exists when the user submitted a login credential that already existed.
FT-10	To ensure that the user will be able to register with the system and the user data is saved in the database.	The register form takes user inputs.	**Refer to Figure 1.3:** When a user successfully authenticates, the system displays their user name in the counsel, which satisfies the criteria.
FT-11	To check if both the username and password are provided by the user when they	The users have been registered successfully in the system.	**Refer to Figure 1.4:** The requirement was met because when a user logs in without entering either the username or password, the system displays an error message to remind the user to input the credentials correctly.

	login.		
FT-12	To check if the system is able to validate the login credentials.	User info is contained in the database.	**Refer to Figure 1.5:**The requirement was met since the system should display an error message indicating that the user supplied invalid input when performing a login action with an erroneous username or password.

<u>Test Results</u>

Table 2.1 - Results of the execution of the tests

Date	**Test (**Refer to the table above for details (Table 1.1)**)**	**Priority**	**Result**	**Comments**
03/28/2022	**FT - 01**	**High**	**PASS**	The test passed as the data file was encrypted successfully.
03/30/2022	**FT - 02**	**High**	**PASS**	The test passed as the encrypted data file was decrypted successfully.
03/18/2022	**FT - 03**	**High**	**PASS**	The test passed as the original data files were retrieved after the encryption and decryption process.
03/22/2022	**FT- 04**	**Medium**	**PASS**	The test passed as the data keys were generated.
04/1/2022	**FT-05**	**High**	**PASS**	The test passed because the both

				aws kms operations work with both the key id as well as the alias name.
04/1/2022	**FT-06**	**Low**	**PASS**	In this process, an alternative approach was taken in which the data key was encrypted using OpenSSL, this test passed, however, the original file wasn't retrieved since the decryption process failed.
04/1/2022	**FT-07**	**Low**	**FAIL**	An alternative approach of decryption was applied using OpenSSL; this test failed because this method of decryption is deprecated.
04/1/2022	**FT-08**	**Medium**	**PASS**	The test passed because the webpage was able to display an error message when a user forgets to fill all the input fields.
03/22/2022	**FT-09**	**High**	**PASS**	The test passed because when a user tried to register with a username that

				already exists an error message was displayed.
03/11/2022	**FT-10**	**High**	**PASS**	The test passed because]when a user performs a registration on the web page it stores all the input field data in the database.
03/1/2022	**FT-11**	**Medium**	**PASS**	The test was successful since the webpage refused access to the system if the user did not complete all of the required input files.
03/20/2022	**FT-12**	**Medium**	**FAIL**	The test failed because the system was unable to display an error message when the user tries to enter the system without valid credentials.

Table 3.1 - Test Report

Test Report	
Test Type	**Acceptance Test**
PASS	10
FAIL	2
Total	12

Further Discussion on the Results

Overall, our implementation is performing well. Firstly, let's look at the security performance of the data encryption algorithm using AWS KMS. AWS KMS really simplifies using the keys to encrypt data across the AWS cloud. We get to select the extent to which you have access control. Plus you're able to share encrypted data between other services. In other words, it essentially provides a centralized place for key management. You're able to perform encrypt and decrypt operations by creating new keys and managing their access. Moreover, we are using KMS-protected data encryption keys to encrypt data in your local machine. These services can easily be integrated to any application as they're scalable. Therefore, one can say that this implementation performs well, the test results shown in the previous sections make it clear. Refer to the test report above to compare the number of tests that passed with those that failed.

On the other hand, as part of the authentication system, Bcrypt was used to encrypt user information such as passwords using AWS. The implementation of a secure login system using AWS services performs well since the databases are infinitely scalable because of a NoSQL database approach. The system's API endpoints were secured using Amazon API GATEWAY where the JSON web token was encrypted using SHA256 encryption.

When performing data encryption using AWS KMS, the data files were successfully encrypted using the generated data key. The same applies for the decryption of data. However, when this process was implemented using OpenSSL, there were some issues faced when decrypting the data as this command has been deprecated. This was quite unusual because the encryption was completed swiftly using OpenSSL.

The authentication system's results were not what we expected, since not all our test cases passed. The system was not able to retrieve the user information and validate their identity. An internal server error occurred as a result.

References

1. "DynamoDB API - amazon dynamodb - docs.aws.amazon.com." [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.API.html. [Accessed: 02-Apr-2022].
2. "Bcrypt," npm. [Online]. Available: https://www.npmjs.com/package/bcrypt. [Accessed: 01-Apr-2022].
3. auth0.com, "JSON web tokens introduction," JSON Web Token Introduction. [Online]. Available: https://jwt.io/introduction. [Accessed: 01-Apr-2022].
4. I. H. V. Whitehouse-Grant-Christ, "IAM," *Amazon*, 2011. [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html. [Accessed: 01-Apr-2022].
5. J. Kessler, M. Grond, and A. Schaaf, "Kognitives minimal-screening KMS," *Amazon*, 1991. [Online]. Available: https://docs.aws.amazon.com/kms/latest/developerguide/programming-encryption.html. [Accessed: 01-Apr-2022].
6. "What is the AWS encryption SDK? - AWS encryption SDK." [Online]. Available: https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html. [Accessed: 02-Apr-2022].
7. J. Kessler, M. Grond, and A. Schaaf, "Kognitives minimal-screening KMS," *Amazon*, 1991. [Online]. Available: https://aws.amazon.com/kms/. [Accessed: 01-Apr-2022].