Test Space

Test Cases : API-life-cycle-TC-scenario.xlsx

Topic - Test Group	Test Case Scenario	Condition	Expected result	Test Status (PASS\FAIL)
User Authentication and Authorization	Verify that users can succesfully login through Okta authentication	Provide valid credentials and submit	User will be redirected to the application after authentication and session is succesfully created.	
	Ensuring that unauthorized users cannot access protected resources.	Attempting to login without valid credentials	Access is denied with 401 unauthorized status and user is not redirected to the protected resource.	
	Validate the users have appropriate access based on their roles	Different roles assigned to users in Okta (RBAC)	Users with different roles can access or are denied access to specific resources based on their role assignment.	
	Verify that once a user is authenticated in Okta, they can seamlessly access other applications without re-entering credentials	User logged into one application using Okta.	User can access another application without re-entering credentials, demonstrating seamless SSO functionality.	
	Confirm that the SSO session expires after a specified period of inactivity	Active SSO Session.	Attempting to access a protected resources after timeout required re-authentication.	
	Ensure that error messages are user-friendly without revealing senstive information.	Intentionally trigger an error condition.	The error messages is displayed to the user without disclosing sensitive details.	
	Confirm that Okta integration logs capture relevant events.	User logins and logouts.	Okta logs accurately record authentication and authorization events with time stamps and user details.	
	Monitor dashboards based on threshold criteria defined	Threshold values need to be defined	Monitoring dashboard will display usage details	
	Confirm that user sessions are properly invalidated upon logout or after a certain period of inactivity.	User logged in and then logged out.	Attempting to access a protected resource after logout or session timeout requires re-authentication.	
	Ensure that revoking a users access in Okta promptly terminates active sessions.	User access revoked in Okta.	The User is immediately denied access to protected resources, and active sessions are invalidated.	
	Verify that the Okta application is correctly configured with the AWS Cognito User Pool as an identity provider.			
	Ensure that the AWS Cognito User Pool is correctly configured to trust Okta as an OpenID Connect identity provider.			
	1. User Attributes Mapping: Test Case 3.1: Check that user attributes (e.g., username, email) are correctly mapped and synchronized between Okta and AWS Cognito.	Verify that updates to user attributes in Okta reflect in AWS Cognito and vice versa.		
API Gateway Integration	Validate that only authenticated and authorized users can access APIs through the API Gateway.	Attempt to make an API request without authentication.	Access denied with a 401 unauthorized status.	
	Validate that only authenticated and authorized users can access APIs through the API Gateway.	Attempt to make an API request with valid credentials.	Succesful access with a 200 OK status.	

	Ensure that API requirement are rate-limited to prevent abuse.	API rate limit configured, on API GW/terraform sripts as a series of rapid requests.	Request beyond the rate limit are rejected with a 429 too many requests status and an appropriate error messages.	
	Ensure that API requirement are rate-limited to prevent abuse.	API rate limit configured, API GW/terraform sripts, as adjusted rate limit and retest.	Requests within the adjusted limit are succesful, requests beyond the limit are rejected.	
	Ensure that error messages are user-friendly without revealing senstive information.	Intentionally trigger an error condition.	The error messages is displayed to the user without disclosing sensitive details.	
	Ensure that API Gateway Logs are generated and contain relevant information.	API requests and actions performed.	Logs show details about API requests	
	Monitor dashboards based on threshold criteria defined	Threshold values need to be defined	Monitoring dashboard will display usage details	
WAF Integration	Confirm that the WAF protects the application from common web vulnerabilities.	Attempt to inject SQL or XSS payloads into web forms.	The WAF detects and blocks these attempts, returning a 403 forbidden status.	
	Verify that the WAF logs security events for analysis and monitoring.	WAF logging is enabled.	Events are accurately recorded in the WAF logs with relevant details.	
	Ensure that error messages are user-friendly without revealing senstive information.	Intentionally trigger an error condition.	The error messages is displayed to the user without disclosing sensitive details.	
	Ensure that WAF Logs are generated and contain relevant information.	WAF actions performed.	Logs show details on responses	
	Monitor dashboards based on threshold criteria defined	Threshold values need to be defined	Monitoring dashboard will display usage details	
Service Catalog	To be done by Sudip			
Terraform (IaC) scripts - CD	Execute terraform script to provision infrastructure on AWS	Terraform script is written to provision the infrastructure.	Infrastructure resources are created successfully without errors. Terraform state is updated with the new infrastructure details.	
	Continuous deployment trigger , push changes to the git repo	Changes are made to the terraform script and pushed to the git	CICD system detects the changes in repo and triggers a deployment.	
	Rollback mechanism, identify issues /errors during the deployment	A new terraform script is applied to changes , it leads to issues.	CICD detects the issues, trigger the rollback mechanism (auto /manual) to restore the infra in previous state.	
	Variable handling , update/modify the variables	terraform script uses the variables for configuration	Variable changes are reflected in the infrastructure. Terraform apply updates the resources with the new variable values.	
	Statefile and Secret management , implement mechanism to create respective environment	Multiple instances in different environments.	Terraform statefile is managed separately for each environment changes do not impact the state of other environments. Secrets are not exposed in plain text in scripts, and securely stored and injected during terraform execution.	
Portal Customiz ation	Ensure that Swager File can be imported and uploaded into Global Developer Portal and Published successfully. Environment : Sandbox environment	User must be logged in with admin credentials.	User must be able to import swagger file, without any error. User must be able to see API **Definition**, **Documentation** and **API Usage Plans** appear in Global Developer Portal correctly as it was documented Enterprise API Gateway. User must be able to publish API.	
	Ensure Published API is available to the Developers.	API must be published on Dev Portal. User must be logged on with "Developer Role" credentials.	Developer is able to generate API Key by subscribing to API.	
	Ensure user receives Client_id and Client_Secret after creating an App on the Global Developer Portal.	User must be logged in Global Developer Portal as a Developer.	After creating an App, user must be able to generate client_id /client_secret	

	Ensure user receives an API Key after subscribing to the API. Ensure the DynamoDB table contains the generated API Key and is associated with Client_Id(Customer_id)	User must be logged in Global Developer Portal as a Developer. User must subscribe to the API.	User receives API Key. And on the DynamoDB table, the generated API Key is associated with Client_Id(Customer_id)	
	Ensure that user is able to receive access_token/JWT.	User must be able to provide the right API_Key, Client_id and Client_Secret to the authorization endpoint.	User receives access_token/JWT.	
	Ensure that user is able to invoke an API using Postman/curl	User must provide the API Key and access_token/JWT	User receives the successful API response.	
	Ensure API Throttling (Rate limit and Burst limit) is not breached.	User must invoke API with more concurrency specified in Rate/Burst limit.	User receives rate_limit_reached error.	
	Ensure API Quota is not breached.	User must be able to invoke at rate above the specified quota e.g. Requests per time period (minute/hour/day/month)	User receives quota_reached error.	
API Security for AWS API	To be done by Omer Approach 1) Current APIGEE Security Assessment. Identify the checklist and the gap, present it as document 2) Implement the security measurements /guidelines w.r.t. to the new AWS API GW solution 3) Share the gaps again on the ones that cannot be applied to the new solution (w.r.t. the old solution) and submit a assessment report			
API gateway integration with micro gateway	Request Comes to API Enterprise gateway	HTTPS request is properly configured and send to AWS API gateway	API gateway successfully receives the HTTPS request	
	Request Comes to API Enterprise gateway	API key is expired , revoked, or incorrect	API gateway returns a 403 Forbidden response	
	AWS API gateway having Cognito JWT authorizer to validate the HTTPS request	Token has exceeded its expiration time	API Gateway returns a 401 Unauthorized response, indicating the expired token	
	AWS API gateway having Cognito JWT authorizer to validate the HTTPS request	Insufficient scope in the token for the requested API	API Gateway returns a 403 Forbidden response, indicating the user lacks the necessary permissions	
	AWS API gateway having Cognito JWT authorizer to validate the HTTPS request	Token is valid, and the user has the necessary permissions	API Gateway successfully authorizes the HTTPS request, allowing it to proceed	
	Post AWS API gateway request validation integrated lambda function is triggered	Issues with Lambda function logic or execution	API Gateway returns a 500 Internal Server Error response	
	Post AWS API gateway request validation integrated lambda function is triggered	All processing steps in the Lambda function are successful	API Gateway returns the expected response, indicating successful processing	
	AWS lambda function has private REST API endpoint URL for request forwarding	Proper configuration of the Lambda function with the private REST API endpoint	Lambda function successfully forwards requests to the private REST API	
	AWS lambda function has private REST API endpoint URL for request forwarding	Private API Gateway is down or misconfigured	Lambda function encounters a connection error or receives an error response from the API, indicating a failure in request forwarding	
	Lambda function successfully receives the output from the AWS Rest API	Output is correctly forwarded to the Lambda function	Lambda function processes the output and returns a success code 200 to AWS API Gateway	

	AWS API Gateway successfully receives a success code 200 from the Lambda function	Lambda function communicates success to AWS API Gateway	AWS API Gateway forwards the success response to the client with the output from the AWS Rest API	
	AWS Rest API encounters an error during its execution	API execution fails for any reason	AWS API Gateway receives an error response (status code 500) from the Lambda function	
API Test Cases	API JSON file contains a valid server URL	Server URL is correctly specified in the API JSON file	API sends requests to the correct server endpoint	
	API JSON file contains valid paths for different API operations	Paths are correctly defined for various API operations (e.g., GET, POST)	API performs the intended operations based on the specified paths	
	API JSON file defines a 200 Success response	200 Success response is correctly defined in the API JSON file	API returns a 200 OK response upon successful execution of the request	
	API JSON file defines a 400 Bad Request response adjustment	Appropriate adjustments are made in the JSON file to simulate a 400 Bad Request scenario	API returns a 400 Bad Request response when triggered with the adjusted request	
	API JSON file defines a 500 Internal Server Error adjustment	Appropriate adjustments are made in the JSON file to simulate a 500 Internal Server Error scenario	API returns a 500 Internal Server Error response when triggered with the adjusted request	
	API JSON file includes address and user details for a specific operation	JSON file specifies address and user details required for a specific API operation	API successfully processes the request using the provided address and user details	
	API JSON file includes OAuth2 token for authorization	OAuth2 token is correctly included in the JSON file for authentication	API successfully authorizes requests using the provided OAuth2 token	
	API JSON file does not contain a valid server URL	Server URL is missing or incorrectly specified in the API JSON file	API fails to connect to the server, and an error response or message is expected	
	API JSON file contains an invalid path for an API operation	Path for a specific operation is missing or incorrectly specified	API returns a 404 Not Found response or a similar error, indicating that the specified path is not valid	
	Okta user requester has a valid ipRange	ipRange is valid	Request is accepted	
	Okta user requester has an invalid ipRange	ipRange is not valid	Receive a 403 status error response	
	Okta user requester has valid ipRestrictions	ipRestrictions is valid	Request is accepted	
	Okta user requester has invalid ipRestrictions	ipRestrictions is not valid	Receive a 403 status error response	
	Okta user requester has a valid WeekdayStarttime and WeekdayEndtime	Request time is within the specified range	Request is accepted	
	Okta user requester has an invalid WeekdayStarttime and WeekdayEndtime	Request time is outside the specified range	Receive a 403 status error response	
	Okta user requester has valid WeekendStarttime and WeekendEndtime	Request time is within the specified range	Request is accepted	
	Okta user requester has invalid WeekendStarttime and WeekendEndtime	Request time is outside the specified range	Receive a 403 status error response	

1) Okta Integration with WAF and API-GW

Test Cases

1. Configuration and Setup:

Test Case 1.1: Verify that the Okta application is correctly configured with the AWS Cognito User Pool as an identity provider.

Test Case 1.2: Ensure that the AWS Cognito User Pool is correctly configured to trust Okta as an OpenID Connect identity provider.

1. User Authentication:

Test Case 2.1: Verify that users can log in successfully through the Okta interface.

Test Case 2.2: Ensure that users authenticated through Okta can access AWS resources secured by AWS Cognito.

1. User Attributes Mapping:

Test Case 3.1: Check that user attributes (e.g., username, email) are correctly mapped and synchronized between Okta and AWS Cognito.

Test Case 3.2: Verify that updates to user attributes in Okta reflect in AWS Cognito and vice versa.

1. Security:
   Test Case 4.1: Ensure that the communication between Okta and AWS Cognito is secure by using HTTPS.

Test Case 4.2: Verify that the OAuth tokens exchanged during the authentication process are securely transmitted and validated.

1. Multi-Factor Authentication (MFA):

Test Case 5.1: Test MFA integration, ensuring that users are prompted for MFA when logging in through Okta.

Test Case 5.2: Verify that AWS Cognito enforces MFA if configured for certain user roles or security requirements.

1. Session Management:

Test Case 6.1: Check that user sessions are managed correctly, including session expiration and refresh token handling.

Test Case 6.2: Verify that users are prompted to re-authenticate when their sessions expire.

1. Error Handling:

Test Case 7.1: Test error scenarios, such as incorrect Okta or Cognito configuration, and verify that meaningful error messages are displayed to users.

Test Case 7.2: Ensure that appropriate error codes and messages are logged for troubleshooting purposes.

1. Authorization and Access Control:

Test Case 8.1: Verify that AWS Cognito correctly enforces access control policies based on the user's roles and attributes.

Test Case 8.2: Test scenarios where a user authenticated through Okta attempts to access resources for which they do not have permission in AWS Cognito.

1. Cross-Browser and Cross-Device Testing:

Test Case 9.1: Ensure that the integration works seamlessly across different web browsers.

Test Case 9.2: Verify that the authentication flow is responsive and user-friendly on various devices.

1. Logging and Monitoring:

Test Case 10.1: Confirm that relevant events and logs are generated in both Okta and AWS Cognito for auditing and monitoring purposes.

Test Case 10.2: Verify that administrators can access logs and reports to troubleshoot and monitor the integration.

1. User Provisioning and Deprovisioning:

Test Case 11.1: Check that creating a user in Okta results in the user being provisioned in AWS Cognito.

Test Case 11.2: Verify that deprovisioning a user in Okta results in the user being appropriately deactivated or removed from AWS Cognito.

2) API-GW Integration with MicroGateway

3) Service Catalog

4) CD :: Terraform & CloudFormation (Design, Implementation)

5) API Security Assessment Document (link)