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 Customization	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		
Lambda Function (auto-create-cert.js)	Valid certificate file and private key file	The event object contains a valid bucket name and a key that ends with self-signed-cert.pem and includes certificates/. The S3 bucket contains the certificate file and the private key file with the same name except for the extension. The certificate and the private key are valid and compatible.	The lambda function should download the certificate file and the private key file from S3, and import them to ACM successfully. The response object should have a status code of 200 and a body message of Certificate import successful.. The ACM should have a new certificate with the ARN returned by the lambda function.		
	Invalid certificate file or private key file	The event object contains a valid bucket name and a key that ends with self-signed-cert.pem and includes certificates. The S3 bucket contains the certificate file and the private key file with the same name except for the extension. The certificate or the private key is invalid or incompatible.	The lambda function should download the certificate file and the private key file from S3, and attempt to import them to ACM. The importCertificate method should throw an error. The response object should have a status code of 500 and a body message of Error importing certificate.. The ACM should not have a new certificate.		
	Missing certificate file or private key file	The event object contains a valid bucket name and a key that ends with self-signed-cert.pem and includes certificates/. The S3 bucket does not contain the certificate file or the private key file with the same name except for the extension.	The lambda function should attempt to download the certificate file or the private key file from S3. The getObject method should throw an error. The response object should have a status code of 500 and a body message of Error importing certificate.. The ACM should not have a new certificate.		
	Non-certificate file	The event object contains a valid bucket name and a key that does not end with self-signed-cert.pem or does not include certificates/. The S3 bucket contains the file with the given key.	The lambda function should ignore the file and not attempt to download or import it. The response object should have a status code of 200 and a body message of Not a certificate file. Ignoring.. The ACM should not have a new certificate.		
Lambda Function (auto-reimport-cert)	The event contains a valid certificate file with the key autoupdate/self-signed-cert.pem and the corresponding private key file with the key autoupdate/private-key.pem.	The existing ACM certificate ARN is valid and matches the region of the function.	The expected result is a successful reimport of the certificate into ACM, with a status code of 200 and a body of Certificate reimport successful.		

	The event contains a valid certificate file with the key autoupdate/self-signed-cert.pem and the corresponding private key file with the key autoupdate/private-key.pem.	The existing ACM certificate ARN is invalid or does not match the region of the function.	The expected result is an error reimporting the certificate into ACM, with a status code of 500 and a body of Error reimporting certificate.		
	The event contains a valid certificate file with the key autoupdate/self-signed-cert.pem but the corresponding private key file with the key autoupdate/private-key.pem is missing or corrupted.	The existing ACM certificate ARN is valid and matches the region of the function.	The expected result is an error reimporting the certificate into ACM, with a status code of 500 and a body of Error reimporting certificate.		
	The event contains an invalid certificate file with the key autoupdate/self-signed-cert.pem or a certificate file with a different key that does not end with self-signed-cert.pem or does not include autoupdate	The existing ACM certificate ARN is valid and matches the region of the function.	The expected result is an ignored event, with a status code of 200 and a body of Not a certificate file. Ignoring.		
Lambda Function (b2b-token-generator)	Valid Okta username and password, existing Cognito user	The event body contains a valid username, password, client_id, and client_secret. The username and password match an Okta user, and the username exists in the Cognito user pool.	The function returns a 200 status code and a JSON body with an access token and a refresh token from Cognito.		
	Valid Okta username and password, new Cognito user	The event body contains a valid username, password, client_id, and client_secret. The username and password match an Okta user, but the username does not exist in the Cognito user pool.	The function creates a new user in the Cognito user pool with the username, password, email, and email verification attributes. The function returns a 200 status code and a JSON body with an access token and a refresh token from Cognito.		
	Invalid Okta username and password	The event body contains a valid username, password, client_id, and client_secret. The username and password do not match an Okta user.	The function throws an error and returns a 401 status code and a JSON body with an error message from Okta.		
	Missing or invalid parameters in the event body	The event body is missing or has invalid values for one or more of the required parameters: username, password, client_id, or client_secret.	The function throws an error and returns a 400 status code and a JSON body with an error message indicating the missing or invalid parameter.		
Lambda Function (acm-api)	Test the lambda function with a valid ACM event	The event object contains a valid certificate ARN, source is 'aws.acm' and eventName is 'ImportCertificate'	The lambda function should call the createCustomDomain function with the domain name and certificate ARN extracted from the event object		
	Test the lambda function with an invalid ACM event	The event object does not contain a valid certificate ARN, or the source is not 'aws.acm' or the eventName is not 'ImportCertificate'	The lambda function should not call the createCustomDomain function and should return without any errors		
	Test the lambda function with a valid ACM event for an existing domain name	The event object contains a valid certificate ARN for a domain name that already exists in API Gateway	The lambda function should not call the createCustomDomain function and should log a message saying "Domain already exists: {domainName}"		
	Test the createCustomDomain function with a valid domain name and certificate ARN	The domain name and certificate ARN parameters are valid and the domain name does not exist in API Gateway	The createCustomDomain function should call the API Gateway createDomainName and createBasePathMapping methods with the correct parameters and should log the responses		
	Test the createCustomDomain function with an invalid domain name or certificate ARN	The domain name or certificate ARN parameters are invalid or the domain name already exists in API Gateway	The createCustomDomain function should handle the errors gracefully and should log them		
Lambda Function (multi-rest-api-lambda-node)	Valid route and successful API call	The event object has a valid route that matches one of the keys in the routeToEndpointMap object, and the API call returns a 200 status code and some data.	The lambda function returns an object with statusCode 200 and body as a JSON string of the response data.		
	Valid route and unsuccessful API call	The event object has a valid route that matches one of the keys in the routeToEndpointMap object, but the API call returns a non-200 status code or throws an error.	The lambda function returns an object with statusCode as the same as the response status code or 500 if there is an error, and body as a string of 'Error calling the API' or the error message.		
	Invalid route	The event object has a route that does not match any of the keys in the routeToEndpointMap object.	The lambda function returns an object with statusCode 404 and body as a string of 'Route not found'.		

Lambda Function (OKTA-COGNITO-PRE-SIGNUP-DYNAMO)	Valid event with user attributes	The event object has a request property and a userAttributes property with valid values for region, userPoolId, userName, clientId, name, givenName, familyName, and email.	The lambda function should store the user attributes in the DynamoDB table with the specified table name, and return the event object as the output. No errors should be thrown or logged.		
	Invalid event without user attributes	The event object does not have a request property or a userAttributes property.	The lambda function should throw an Error with the message "Invalid Cognito event structure" and log the error message "Cognito event structure is not as expected.".		
	Invalid event with empty user attributes	The event object has a request property and a userAttributes property, but the userAttributes property is an empty object or has missing or invalid values for some of the attributes.	The lambda function should store the user attributes in the DynamoDB table with the specified table name, using the default values of "Unknown" for any missing or invalid attributes, and return the event object as the output. No errors should be thrown or logged.		
	DynamoDB putItem failure	The event object has a request property and a userAttributes property with valid values, but the DynamoDB putItem operation fails due to some reason (e.g., network error, table not found, etc.).	The lambda function should throw an Error with the message "Error storing user details in DynamoDB: {error.message}" and log the error message with the same format, where {error. message} is the message from the DynamoDB error object.		
Portal Customization (Admin Users)	User Registration process	New User Registrations	Fill registration form with valid user data.	User account created successfully, confirmation message displayed.	
		Duplicate credentials(Email/ username)	Register with an email that already that exist in the system.	Prompt for a unique email or error message indicating the email in use /exist.	
		Password strength Validation	Register with weak or invalid password character combination.	Prompt to enter a stronger password or specific password strength requirements.	
	User Login	Successful user login	When enter valid Credentials.	Successful login and redirection to the user dashboard.	
		Invalid Login Attempts	Invalid Credentials multiple times.	Lockout after a defined number of attempts or display of appropriate error messages.	
		Invalid user name	User is not present/User name spell incorrect.	shows error message	
		Remember me functionality	Check the "Remember Me" feature	Successful session persistence across browser restarts.	
		Swagger Docs Import	Import Swagger/Open API documentation file in "yaml" format into the system. If it is not "yaml" then error message it shows.	Successful import of API specification with proper parsing and documentation display.	
		API Endpoint Testing	Test each API endpoint documented in Swagger for functionality and accuracy.	Successful execution of each endpoint as per the documented specifications.	
		API Publishing	Publish an API using the imported Swagger documentation.	API published and available for external consumption with proper documentation accessible.	
		Authorization and authentication testing	Test API endpoints that require authentication or authorization.	Successful access with valid tokens /credentials denied access with invalid or expired tokens.	
Portal Customization (B2C users)	**User Registration Process**	Valid Registration	Enter all required fields with valid data(name, email, password, company details, select country, check box for terms and conditions, news letter, CAPTCHA.)	Verify successful registration confirmation message	
			If the user fill the required details incorrect	User will get error message	
		Unique email validation	Attempt registration with an email already existing the system.	Confirm system prompts for a unique email address.	
			If User enters email id incorrect format missing '@' or domain	Verify the system prompts for a valid email format.	
		Password strength	Test with various passwords strengths(eg. alphanumeric ,special characters)	Verify that a strong password is accepted.	

			If user enters the password without the eg, alphanumeric, special characters) then user will get error message.	User will get error message and password will not generate.	
		Validation messages	Leave required fields blank and attempt registration.	Validate appropriate error messages for each missing field.	
		Confirmation Email	Register and verify if a confirmation email is sent to the provided email address.	Check the email for correctness and completeness	
		Weak password validation	Try to register with a weak password(eg. less than minimum required character)	Ensure the system prompts for a stronger password.	
		Duplicate registration	Attempt to register with the same details immediately after a successful registration.	Verify the system prevents duplicate registrations.	
	Login user	User opens Development portal in web browser	Web page will for development portal	There is option start using data click on that and click on Get started.	
		Enter email address ,enter last name, enter first name, Enter mobile name, Company details, select country, check box for terms and condition and new letter, CAPTCHA	If the credentials are correct user will be created.	User will be created	
			If the credentials are in correct it will show error message.	Shows error message.	
		We will get sign In page where we can enter user name and password remember me and check box and Sign.	If the credentials are correct then user will login	Login page or user page will open	
			If the login credentials are incorrect we get error message.	Validation error message show.	
		After login successful go to my app page and click on my app button.	User need to add App and check the API to create App name.	New App is created.	
			If the user will not select any of the API then it shows error message.	App will not create.	
		After app creation user will able to see client id and client secrete.	Newly app created with client id and client secrete.	New app is listed on the app page	
		After click on authorize button modal popup ,User name, Password, Client credentials location ,Apps, Client ID, Client _secrete , check box for user and user scope	After fill the required details click on authorize button then secession will be started for the user.	Secession started for user.	
			If the filled details are incorrect user will get error message.	Secession will not started.	
Portal customization (B2Busers)		Valid username and password	Enter a valid username and password.	User should be successfully logged in and directed to the dashboard /homepage.	
		Case sensitive-Username and password	Use correct username and password with incorrect cases(eg. uppercase username /password if its case sensitive).	Verify if the system is case -sensitive. It should reject incorrect cases and prompt for the correct ones.	
		Incorrect user name	Enter a valid password with an incorrect username.	System should display an error message for an invalid user name.	
		Incorrect password	Enter a valid user name with an incorrect password.	System should display an error message for an incorrect password.	
		Empty username and password fields	Leave both username and password fields empty.	System should prompt for both fields as required.	
		Empty password field	Enter a valid username and leave the password field empty	System should prompt for the password as required.	
		Multiple Login Attempts-Account lockout.	Enter incorrect credentials multiple times (exceeding the allowed limit. if any).	Verify the account gets locked after the specified number or failed attempts.	
		Password Complexity Requirements.	Enter a password that doesn't meet the complexity Requirements(eg. Length, special characters, etc.)	System should enforce password complexity and prompt the user for a password that meets the criteria.	
		Remember Me Functionality.	Log in with the "Remember Me" option checked.	Verify the system keeps the user logged IN even after closing and reopening the browser.	
		Session Timeout.	Log in and wait for the session timeout duration.	Verify if the system automatically logs the user out after the specified period of inactivity.	
		Multi-factor Authentication(MFA).	Enable MFA and log in with correct credentials	System should prompt for the second factor authentication(eg. OTP, authorization app)and grant access upon successful verification.	
		After login successful go to my app page and click on my app button.	User need to add App and check the API to create App name.	New App is created.	

			If the user will not select any of the API then it shows error message.	App will not create.	
		After app creation user will able to see client id and client secrete.	Newly app created with client id and client secrete.	New app is listed on the app page	
		After click on authorize button modal popup, Username, Password, Client creditionals location, Apps ,Client ID, Client secrete, check box for user and user scope	After fill the required details click on authorize button then secession will be started for the user.	Secession started for user.	
			If the filled details are incorrect user will get error message.	Secession will not started.	
		Ensure that Swager File can be imported and uploaded into Global Developer Portal and Published successfully.	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.	
	Admin Portal Customization	Environment : Sandbox environment			
		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.	

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)