Manual QA Engineer Take-Home Assignment

Name: Mohammad Ghanayem

Email: mohammad.gh454@gmail.com

Phone Number: 0528942919

Table of Contents

Part 1: Test Case Design (3 hours)

Software Test Design (STD) – VPC Creation Feature

Test Design Overview:

This document defines the test cases and test design for the VPC Creation feature. It includes positive, negative, boundary, edge, integration, and UI level tests to ensure the feature meets its functional and technical requirements.

Test Coverage Areas:

- CIDR block validation (format, range, overlap, public/private)
- VPC name validation (length, character set)
- VPC creation limits (per region)
- Region & Availability Zones handling
- DNS option behavior
- UI form validations
- Concurrency and performance limits

Test Cases:

ID	Description	Pre-conditions	Steps	Expected results	Priority	Category
(01)	Create VPC with valid CIDR block	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid CIDR block 10.0.0.0/16. 3. Submit the form.	VPC successfully created within 30 seconds.	Medium	Functional, API
(02)	Create VPC with valid name	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid VPC name. 3. Submit the form.	VPC successfully created within 30 seconds.	Medium	Functional
(03)	Create valid number of VPCs (<=5) in a region for an account	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid inputs for a VPC in one region. 3. Submit the form. 4. Do steps 2-3 again in the same region 0-4 times.	VPCs successfully created, each one within 30 seconds.	Low	Functional
(04)	Create VPC within specific availability zone region	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid availability zone region. 3. Submit the form.	VPC successfully created within the chosen availability zone region, and within 30 seconds.	Medium	Functional
(05)	Create VPC with DNS **disabled**	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. disable DNS. 3. Submit the form.	VPC successfully created with DNS disabled, and within 30 seconds.	High	Functional

(06)	Create VPC with DNS **enabled**	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. enable DNS. 3. Submit the form.	VPC successfully created with DNS enabled, and within 30 seconds, instances in this VPC can resolve domain names after creation	High	Functional
(07)	Create VPC with invalid CIDR block (larger network)	User logged in, other fields are in the right form.	1. Navigate to VPC creation form. 2. Fill invalid CIDR block 10.0.0.0/15. 3. Submit the form.	VPC failed to create with the given CIDR block, the UI should show warning error	Medium	Functional, UI, Security
(08)	Create VPC with invalid CIDR block (smaller network)	User logged in, other fields are in the right form.	1. Navigate to VPC creation form. 2. Fill invalid CIDR block 10.0.0.0/29. 3. Submit the form.	VPC failed to create with the given CIDR block, the UI should show warning error	Medium	Functional, UI, Security
(09)	Create VPC with empty name	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill invalid VPC name, empty name. 3. Submit the form.	VPC failed to create with empty name, the UI should show warning error	Low	Functional, UI, Security
(10)	Create VPC with very long name	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill invalid VPC name, 256 characters name (or more). 3. Submit the form.	VPC failed to create with this large name, the UI should show warning error	Low	Functional, UI, Security
(11)	Create 6 VPCs in the same region from the same account	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid inputs in VPC creation form. 3. Submit the form. 4. Repeat 2-3 steps 5 times or more in the same region from the same account.	The 6th VPC creation fail, the API should reject the request and the creation will fail, the UI should show message when user try to create 6th VPC.	Medium	Functional, API, UI, Security
(12)	Create VPC without specifying the availability zone region	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid inputs in the form, but without specifying the availability zone region. 3. Submit the form.	VPC creation fails, the UI should show a warning message before submission, and not allowing submission without specifying AZ region	Medium	Functional, UI
(13)	Create VPC with largest network (10.0.0.0/16)	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill CIDR block with 10.0.0.0/16. 3. Submit the form.	VPC creation successfully created within the given CIDR block within 30 seconds.	Medium	Functional, API
(14)	Create VPC with smallest network (10.0.0.0/28)	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill CIDR block with 10.0.0.0/28. 3. Submit the form.	VPC creation successfully created within the given CIDR block within 30 seconds.	Medium	Functional, API

(15)	Create VPC with just 1 character name	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill VPC name with 1 character name. 3. Submit the form.	VPC creation successfully created with the given 1 character name, within 30 seconds.	Medium	Functional, UI
(16)	Create VPC with 255 characters name	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill VPC name with 255 characters name. 3. Submit the form.	VPC creation successfully created with a name of 255 characters, within 30 seconds.	Medium	Functional, UI
(17)	Create 5 VPCs in the same region from in the same account	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill valid inputs in VPC creation form. 3. Submit the form. 4. Repeat 2-3 steps 4 times or more in the same region from the same account.	All the VPC creations will succeed, each one within 30 seconds.	Medium	Functional, API
(18)	Create VPC with public CIDR block	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill CIDR block as 8.8.8.0/24 (or any CIDR block that is not inside the private ones). 3. Submit the form.	VPC fail as the CIDR block has public IPs.	High	Functional, UI, Security
(19)	Create VPC with overlapping CIDR block with another VPC in the same region.	User logged in, other fields are in the right form	1. Navigate to VPC creation form. 2. Fill CIDR block with overlapping CIDR block with another VPC in the region. 3. Submit the form.	VPC creation failed, API should return error code, the UI should show warning message	High	Functional, UI, API, Security
(20)	Create a VPC with valid inputs to get success message	User logged in	1. Navigate to VPC creation form. 2. Fill all fields with valid inputs. 3. Submit the form.	VPC created successfully, and a success message appears.	Low	Functional, UI
(21)	Create a VPC with at least one invalid input to get error message	User logged in, other fields filled with valid inputs.	1. Navigate to VPC creation form. 2. Fill all fields with valid inputs, except at least one field, fill it with invalid input, like CIDR block as 10.0.0.0/12. 3. Submit the form.	VPC fail to create and an error message appear.	Medium	Functional, UI
(22)	Empty Form	User logged in	1. Navigate to VPC creation form. 2. Submit the form.	The form should disable submitting without filling the required fields	Low	UI

(23)	Create VPC with name contains invalid characters	User logged in, other fields filled with valid inputs.	1. Navigate to VPC creation form. 2. Fill VPC name with a name with length of 1-255 characters, and contains ‘$’ symbol. 3. Submit the form.	VPC creation fail	High	Functional, Security
(24)	Create 6 VPCs at the same time in the same region from the same account	User logged in	1. Navigate to VPC creation form, from 6 places, like 6 tabs in the browser. 2. Fill all the fields with valid inputs for all the forms. 3. Submit the 6 forms at the same time.	VPCs creation should fail for one of the VPCs and allow creation for the other 5 VPCs	High	Functional, Performance , Security
(25)	Create 2 VPCs at the same time with the same CIDR block	User logged in, and the other field filled with valid inputs	1. Navigate to VPC creation form, from 2 places, like 2 tabs from in the browser. 2. Fill the CIDR block fields with 10.0.0.0/24 for both VPCs. 3. Submit both forms at the same time.	VPC creation should fail for one of the forms and allow creation of the other VPC, with proper error and success messages, respectively	High	Functional, Performance , Security
(26)	Paste CIDR block	User logged in	1. Navigate to VPC creation form. 2. Write the wanted CIDR block on any other software. 3. Copy it. 4. Paste it in the form. 5. Click out of the field.	CIDR block validation should work after clicking outside the CIDR field	Medium	UI, Functional

Assumptions:

- The user interface is stable.
- The backend service is available.
- No network delays and all requests are successful.

Part 2: Bug Investigation and Reporting (2 hours)

Bug Report 01

Bug ID: 01

Title: VPC Creation Sometimes Fails with CIDR block 10.0.0.0/16

Description:
VPC creation sometimes fail when creating a VPC with a CIDR block 10.0.0.0/16.
The form shows an error message said “Creation failed – please try again”, even though it sometimes works and sometimes gives an error.

Steps to reproduce:
1. Log in to the staging environment.
2. Go to the VPC creation form.
3. Enter a valid inputs in all the fields, like:
  * Name: TestVPC
  * CIDR block: 10.0.0.0/16 (must)
  * Region: us-east-1 (must)
  * DNS: enabled (optional)
4. Repeat several times.

Expected behavior:
If no existing VPC in the same region uses CIDR 10.0.0.0/16, the system should successfully create the VPC.
If the CIDR is already in use, the system should always fail immediately with a clear error.
* The system should behave consistently — not succeed sometimes and fail other times with the same input.

Actual behavior:
it sometimes works and sometimes gives an error message “Creation failed – please try again”

Environment details: Staging Environment, us-east-1 region

Severity assessment: High

Priority recommendation: High priority, should make a mechanism to check the CIDR block when filling the field.

Investigation Analysis:
The issue may be caused from a **race condition**, especially since it “seems to happen more frequently during peak hours”. It is likely that two or more users attempt to create a VPC using the same CIDR block (like 10.0.0.0/16) **concurrently**, and the backend does not handle the conflict correctly, allowing both to proceed at first, but rejecting one at last.
Moreover, unreliable network conditions may cause delays in form submission, increasing the chance of overlap between users, and getting timeout errors even though that the CIDR is not used by any other VPC.
There is also a possibility that the CIDR block had already been used by the same or a different user in the same region, but the error handling in the UI didn’t work as expected.

This will increase the risk if external applications depend on this specific CIDR block, the failure to create the VPC may break integration or expected app behavior.

Suggestions for verification testing:

- Checking for proper logic in CIDR validation.
- Reviewing error messaging and timing under concurrency.
- Verifying whether CIDR availability is re-checked on final submission.

Bug Report 02

Bug ID: 02

Title: DNS Settings Not Working

Description:
When DNS resolution is enabled during VPC creation, instances launched in VPC are unable to resolve domain names. This is against the purpose of enabling DNS resolution for VPC.

Steps to reproduce:
1. Log in to the staging environment.
2. Go to the VPC creation form.
3. Enter a valid inputs in all the fields, like:
   * Name: TestVPC
   * CIDR block: 10.0.0.0/24
   * Region: us-west-2 (must)
   * DNS: enabled (must)
4. Submit the form.
5. Launch an instance in this VPC.
6. Log in the newly created instance.
7. Try to resolve domain names using nslookup, dig, or ping google.com.

Expected behavior:
Instance **can** resolve domain names successfully, since the DNS is enabled in the VPC.

Actual behavior:
Instance **can't** resolve domain names, this happen even though the DNS is enabled in the VPC.

Environment details: Staging Environment, us-west-2 region

Severity assessment: Critical

Priority recommendation: High priority, when enabling DNS for VPC, it should work, if not, this will halt services, because of the lack of resolving domain names.

Investigation Analysis:
Potential causes might be that DNS setting in UI might not configured well with the backend.
Launching the instances might lack the proper DNS settings.
The DNS settings for each new instance might be corrupted.

This will increase the **risk** of stopping services in the instances because the instance can't resolve domain names, because some services need reaching internet and this will lead for a high downtime, this will lead to bad UX.
Check DNS configuration in instances by reaching /etc/resolv.conf file in instances.
Add logging for DNS option while creating the VPC.

This issue will affect the API requests, all the requests will fail, even if an instance try to talk to another instance, it will fail to reach it and think that it is terminated or stopped.

Bug Report 03

Bug ID: 03

Title: CIDR Validation Inconsistency

Description:
When CIDR block put as 10.0.0.0/15 in creating a VPC, the UI don’t give any warning about it, the error show just after submitting the form.

Steps to reproduce:
1. Log in to the staging/production environment.
2. Go to the VPC creation form.
3. Enter a valid inputs in all the fields, like:
   * Name: TestVPC
   * CIDR block: 10.0.0.0/15 (must)
   * Region: us-east-1 (optional)
   * DNS: enabled (optional)
4. Submit the form.

Expected behavior:
UI should raise an error for the user while filling the CIDR block field and disable the submit button to prevent the user from clicking on it.

Actual behavior:
UI accepted the CIDR block and allow the user to click on the submit button, form is submitted, then the creation failed.

Environment details: Both staging and production

Severity assessment: Medium

Priority recommendation: Medium priority, the UI should tell the user that he can’t use this CIDR block before submitting the form, to make it easy on the user, now after some seconds the user waiting for the submission to success he got a failure. Preventing invalid data in the UI improves the user experience and reduces server load.

Investigation Analysis:
Potential causes might be that some form field validate the content of the field after it loses focus (when click outside it), the user might paste the CIDR block to the CIDR field and submit the form immediately so the UI didn’t validate the field.
Moreover, the CIDR block validation might not configured well between the UI and the backend, so the validation didn’t work as expected. (like checking format x.x.x.x/x and not the range /16 - /28)
This will **affect** the security and performance of the feature, each time the user enters an invalid CIDR he will get the error message after submission proceed.
This will increase the **risk** of making the UX worse by not validating during filling fields in the UI, and lose some users.
Suggestions for verification testing:
Try entering several CIDRs like 10.0.0.0/15 (fail) 10.0.0.0/29 (fail) 10.0.0.0/16 (pass) to ensure that the UI blocks the CIDR blocks that are outside the 16-28 range, try copy 10.0.0.0/15 CIDR block from another place and paste it in CIDR block field and then submit immediately without losing focus of the field.

Part 3: End-to-End Testing Strategy (2 hours)

Test Strategy

Testing Approach Overview:

We will perform end-to-end manual testing on a 3-tier web application deployed within a custom VPC with proper network isolation and security controls.
The infrastructure includes:

- Public Subnet (10.0.1.0/24) for Load Balancer
- Private Subnet (10.0.2.0/24) for Application Tier
- Database Subnet (10.0.3.0/24) for RDS MySQL
- Internet Gateway
- NAT Gateway
- Security groups with appropriate rules
- ALB in public subnet
- 2 Application instances in private subnet
- RDS MySQL database in database subnet

Testing will include:

- Functional validation of VPC and subnet settings
- Connectivity and network isolation
- Security group configuration
- DNS and internet access
- Application availability via Load Balancer
- Rollback (if necessary)

Test Objectives and Scope:

Objectives:

- Ensure all network components are created correctly within the VPC
- Validate end-to-end communication between the 3 tiers
- Ensure that security groups rules work as intended
- Ensure that the infrastructure supports high availability and DNS resolution

In Scope:

- VPC, Subnets, Route Tables, IGW, NAT Gateway
- Instances, ALB, RDS
- Network tests (ping, SSH, curl, DB connection)

Out of Scope:

- Application logic or database schema validation
- Load on the server from the application
- Autoscaling behavior

Key Testing Scenarios:

Scenario	Description
VPC created with correct CIDR	Confirm VPC CIDR is 10.0.0.0/16
Subnet CIDRs valid and non-overlapping	Ensure public, private, and DB subnets are inside VPC
Internet access from public subnet	Test outbound access and ALB connectivity
NAT Gateway enables outbound access from private subnet	Confirm private EC2s can for example "curl google.com" (private instances can reach internet)
DB subnet has no internet access	Confirm no outbound access from DB layer
Security groups allow access and connection between ALB, app, and database	ALB can reach app servers, app can reach DB, etc.
ALB routes traffic correctly	Web traffic reach the wanted instances
Instances resolve domain names	DNS resolution functional inside all subnets
Rollback works	Manually terminate components and repeat templates

Risk Assessment:

Risk	Possibility	Impact	Check
Misconfigured route tables	Medium	High	Validate using traceroute/ping
Too much open security groups	High	Critical	Test with public IP and port scans
NAT misconfigured	Medium	High	Test private instances outbound traffic
ALB not forwarding traffic	Low	Medium	Deploy test app and test HTTP routing
DNS resolution failure	Low	High	Use nslookup and /etc/resolv.conf in EC2

Success Criteria:

All infrastructure components are deployed with correct configuration, public subnet accessible from the internet, private instances accessible only via ALB and NAT and can access internet not otherwise, DB tier is isolated from the internet and reachable from app tier.
End-to-end communication (HTTP to App to DB) works as expected, no security or routing misconfigurations found.

Detailed Test Plan

Test Execution Plan:

Step	Action	Validation Checkpoint	Expected Result
(01)	Create a VPC with CIDR 10.0.0.0/16	Check VPC exists and correct CIDR shown	VPC created with proper CIDR
(02)	Create 3 subnets: 10.0.1.0/24 (public), 10.0.2.0/24 (private), 10.0.3.0/24 (DB)	List subnets and their CIDRs	All 3 subnets exist under VPC and don’t overlap
(03)	Attach Internet Gateway to the VPC	Check IGW is attached	IGW status: attached
(04)	Add route to 0.0.0.0/0 via IGW in public subnet's route table	View route table in console	Route appears and valid
(05)	Create a NAT Gateway in public subnet	Check NAT Gateway status	NAT is active
(06)	Add route to 0.0.0.0/0 via NAT in private subnet’s route table	Route table for private subnet	NAT route appears correctly
(07)	Launch 2 instances in private subnet without public IP	Attempt direct SSH, test curl google.com via NAT	Direct SSH fails, curl works in NAT
(08)	Launch RDS MySQL in DB subnet	Check connection from app servers	RDS is only accessible from private instances
(09)	Create security groups: ALB can reach App and App can reach DB	View security groups rules	Only necessary ports open
(10)	Deploy Application Load Balancer in public subnet	Access public DNS in browser	App is accessible via HTTP
(11)	Test the connection from app to DB	App writes/reads from DB	DB reachable by app only
(12)	Run nslookup or dig google.com from private instances	Check DNS resolution	Domain names resolve
(13)	Delete an instance and rerun deployment	Components reassign correctly	Infrastructure recovers successfully

Rollback Plan (in case of failure):

Step	Rollback Action
Instance not reachable	Terminate and re-deploy instance
NAT Gateway fails	Delete and recreate NAT Gateway
DB not accessible	Check security group rules, restart RDS
DNS not working	Recheck VPC DNS settings and /etc/resolv.conf
Infrastructure misconfigured	Reapply IaC template (e.g., CloudFormation, Terraform)

Testing Considerations

Network Connectivity Between Tiers

Objective:

Ensure each tier (Public, Private, DB) is correctly networked with intended access paths and blocked where access is not allowed.

Key Tests:

Public subnet can reach the internet via Internet Gateway.
Private subnet can reach the internet via NAT Gateway.
Private subnet cannot be accessed directly from the internet.
App servers in private subnet can connect to RDS in DB subnet (via port 3306).
Verify no unintended access (e.g., DB subnet can't access public subnet).

Security Isolation Verification

Objective:

Confirm that security groups, subnets enforce strict access control between tiers.

Key Tests:

ALB only allows HTTP (port 80) or HTTPS (port 443).
App tier instances only accept traffic from the ALB's security group.
DB tier only accepts MySQL (3306) from app tier security group.
SSH access restricted to specific IPs (not open to 0.0.0.0/0).

Performance and Scalability aspects

Objective:

Ensure the infrastructure can handle growth and load as expected.

Key Tests:

VPC supports multiple subnets and instances without route conflicts.
NAT Gateway and ALB function reliably under concurrent connections.
DNS resolution works under pressure.
Validate that ALB routes traffic evenly across app tier servers (load balancing).

Integration Points with Other Services

Objective:

Ensure the infrastructure integrates well with key features commonly used in multi-tier apps.

Key Tests:

Logs/Alarms: Ensure metrics (e.g., CPU, latency) and logs are collected from instances/RDS/ALB.
DNS: If used, ensure domain resolution is correctly mapped to ALB or instances.
Auto Scaling (if implemented): Verify health checks, new instance joins the tier and is reachable.

Part 4: API Testing Approach (1.5 hours)

API Test Scenarios:

Test Case ID	Description	Category
(01)	Create a new VPC with valid data (POST /v1/vpcs)	CRUD operations testing
(02)	Retrieve list of all VPCs (GET /v1/vpcs)	CRUD operations testing
(03)	Get a specific VPC by ID (GET /v1/vpcs/{vpc_id})	CRUD operations testing
(04)	Update VPC setting like VPC name or DNS settings (PUT /v1/vpcs/{vpc_id})	CRUD operations testing
(05)	Delete a VPC by ID (DELETE /v1/vpcs/{vpc_id})	CRUD operations testing
(06)	Attempt to delete a non-existent VPC and confirm 404	CRUD operations testing
(07)	Try creating a VPC with invalid CIDR block (e.g., 10.0.0.0/15)	Input validation
(08)	Try creating a VPC with name longer than 255 characters	Input validation
(09)	Try creating a VPC with invalid region code	Input validation
(10)	Submit request without required fields (e.g., no name or CIDR)	Input validation
(11)	Access API with valid API key/token, should success	Authentication and authorization
(12)	Access API with invalid or expired token, should return code 401 Unauthorized	Authentication and authorization
(13)	Attempt to update or delete VPC owned by another account, should return code 403 Forbidden	Authentication and authorization
(14)	Submit a malformed JSON request, should return code 400 Bad Request	Error handling
(15)	Request non-existent VPC, should return code 404 Not Found	Error handling
(16)	Server unavailable, should return code 500 Internal Server Error	Error handling
(17)	Send 100 requests in a short time, should return code 429 Too Many Requests	Rate limiting
(18)	Verify proper Retry-After header is returned when rate limit exceeded	Rate limiting
(19)	Send 5 concurrent POST /vpcs requests with same CIDR, should make a conflict	Concurrent request handling
(20)	Send multiple DELETE requests for same VPC	Concurrent request handling

Test Implementation Plan:

Recommended Tools:

- Postman for manual REST API testing
- JMeter for load and concurrency testing
- Python + Requests for scripting

Test Data Setup:

- Predefined valid regions: us-east-1, us-west-2
- Valid CIDRs: 10.0.0.0/16, 192.168.1.0/24
- Invalid CIDRs: 10.0.0.0/15, 300.0.0.0/16
- VPC names:
  - Valid: My-VPC_01
  - Invalid: #*invalid*# , 300 characters name

Expected Response Codes:

Endpoint	Scenario	Expected Code
POST /vpcs	Valid	201 Created
POST /vpcs	Invalid input	400 Bad Request
GET /vpcs/{id}	Not found	404 Not Found
PUT /vpcs/{id}	Unauthorized	403 Forbidden
DELETE /vpcs/{id}	Valid	204 No Content
Any of the requests above	Unauthenticated	401 Unauthorized

Performance Baseline Suggestions:

- Response time for GET and POST ≤ 1 second
- POST /vpcs under concurrent load should still complete within 30 seconds
- No memory leaks or high CPU usage under 100 concurrent users

Any assumptions made:

- When there is no explicit mentioning of any other AWS services, that means that it’s not supported, like autoscaling.
- There is no constraints on regions.