Pre-Screening Assignment for Associate QA Engineer

By Sami Mali

Date of Submission: 11th November, 2024

Task 1 : Functional Testing for Call Features.

Imagine you are testing a feature in a cloud telephony application where users can **initiate a voice call** through a web interface.

Assignment:

1. Write **5-10 test cases** covering different scenarios for initiating a voice call (positive, negative, and edge cases).
2. Specify expected outcomes for each test case.
3. Include a priority (High, Medium, Low) for each test case based on business impact and risk.

Deliverable: A document with test cases structured as Test Case ID, Description, Steps, Expected Result, and Priority.

Response:

Test cases covering different scenarios for initiating a voice call is as follow:

Test Case ID	Test Case Description	Test Preconditions	Test Steps	Expected Result	Test Priority
TC_001	Verify if the user can successfully initiate a voice call using a valid phone number.	User is logged in into the web interface, and the system has a valid phone number available for the call.	1. Log into the web interface. 2. Enter a valid phone number in the call input field. 3. Click on the "Call" button to initiate the call.	The call should be initiated successfully, and the user hears a ringing tone or receives a prompt that the call is being connected.	High
TC_002	Verify if the user can initiate a voice call using an invalid phone number format.	User is logged in into the web interface.	1. Log into the web interface. 2. Enter an invalid phone number in the call input field. 3. Click on the "Call" button.	An error message should be displayed indicating that the phone number is invalid such as "Please enter a valid phone number".	High
TC_003	Verify if the user can call with an empty phone number field	User is logged in into the web interface.	1. Log into the web interface. 2. Leave the phone number in the call input field empty. 3. Click on the "Call" button.	The system should prevent the call from being initiated and display an error message such as "Phone number is required."	High
TC_004	Verify if the system handles calls to a phone number that is not in service or disconnected.	User is logged in into the web interface and the system is configured to	1. Log into the web interface. 2. Enter a non-service phone number. 3. Click on the "Call"	The system should display an error message indicating that the number is not in service and cannot	Medium

		simulate or check against the number that are out of service.	button.	be reached such as "Call failed: Number is disconnected."	
TC_005	Verify if the system handles scenarios where the user denies microphone access when prompted by the browser during call initiation.	User is logged in into the web interface, and microphone permissions are requested during call initiation.	1. Log into the web interface. 2. Enter a valid phone number. 3. Deny the microphone permission when the browser prompts for access. 4. Click on the "Call" button.	The system should either display an error message indicating that the call cannot be completed due to missing microphone access, or disable the microphone functionality and allow the call to proceed without voice input.	Medium
TC_006	Verify if the system handles phone numbers entered with extra whitespace characters.	User is logged in into the web interface.	1. Log into the web interface. 2. Enter a phone number with extra spaces. 3. Click on the "Call" button.	The system should either remove the extra spaces automatically and initiate the call or display a warning message asking the user to remove extra spaces.	Low
TC_007	Verify if the system handles situations where there are no network connections or a weak network during the call initiation.	User is logged into the web interface, and network issues are simulated.	1. Log into the web interface. 2. Disable the network connection. 3. Enter a valid phone number. 4. Click on the "Call" button.	The system should display an error message such as "Network Error: Unable to initiate call."	High
TC_008	Verify if the system handles multiple call initiations in quick succession.	User is logged into the web interface, and a valid phone number is available for testing.	1. Log into the web interface. 2. Initiate a voice call with a valid number. 3. While the first call is active, try to initiate another voice call to a different number.	The system should either allow the user to put the current call on hold and initiate the second call or display an error message like "You are already on a call.".	High
TC_009	Verify if the system handles cases where the call is successfully initiated but disconnects after a few seconds.	User is logged into the web interface, and a valid phone number is available for testing.	1. Log into the web interface. 2. Enter a valid phone number. 3. Click on the "Call" button and wait for the call to connect. 4. Observe if the call disconnects after a few seconds.	The system should handle the disconnect gracefully, and provide an error message such as "Call dropped" or "Call ended unexpectedly."	High

TC_010	Verify if the system prevents the user from initiating a call after logging out.	User is logged in into the web interface.	1. Log into the web interface. 2. Log out of the system. 3. Try to initiate a call by entering a valid phone number	The system should prevent the call from being initiated and display a message such as "Log in to make a call."	High

Task 2 : Bug Reporting

You've discovered a bug while testing the **Call Recording** feature. When testing, you notice that recordings are intermittently unavailable in the call history, especially when calls are longer than 10 minutes.

Assignment:

1. Document the bug, including:
   ○ Bug Title
   ○ Steps to Reproduce
   ○ Expected Result
   ○ Actual Result
   ○ Severity and Priority
2. Suggest a potential impact this bug could have on business

operations.

Deliverable: A detailed bug report.

Response:

Bug Title: Intermittent Unavailability of Call Recordings for Calls Exceeding 10 Minutes

Bug ID: BG-001
Assigned To: Dev11
Reported By: Sami Mali
Date and Time: 2024-11-9

Description:

Call recordings for calls exceeding 10 minutes do not consistently appear in the call history. This intermittent unavailability affects users who rely on recorded calls for compliance, quality assurance, or personal reference.

Steps to Reproduce:

1. Initiate a call using the call recording feature within the application.
2. Ensure the call duration exceeds 10 minutes.
3. After the call ends, navigate to the Call History or Call Logs section of the application.

4. Verify the availability of the call recording within the call history or logs.
5. Repeat the test several times to confirm that recordings are intermittently missing for calls longer than 10 minutes.

Expected Result:

Call recordings for calls longer than 10 minutes should be consistently available in the Call History or Call Logs without any issues.

Actual Result:

Call recordings for calls exceeding 10 minutes are intermittently missing from the Call History or Call Logs. The issue occurs sporadically, affecting the reliability of this feature for users.

Severity:

High – This issue impacts a core functionality (call recording) essential for users needing to retain recorded calls for compliance, quality assurance, or personal reference.

Priority:

High – Since this problem affects users who make extended-duration calls and rely on accurate call logs, it should be resolved urgently to maintain feature reliability and user trust.

Task 3 : API Testing for Cloud Telephony

You have been provided with an API endpoint that retrieves call history based on parameters like `user_id`, `start_date`, and `end_date`:

`GET /call-history?user_id={user_id}&start_date={start_date}&end_date={end_ date}`.

Assignment:

1. Write **3 test cases** to validate this API's functionality, considering typical and edge-case scenarios.
2. Describe how you would automate these API tests (mention the tool and approach).

Deliverable: A document with API test cases and your approach to automating these tests.

Response:

Test Case ID	Test Case Description	Test Preconditions	Test Steps	Expected Result	Test Priority
TC001	Verify if the API returns call history data when provided with valid user_id, start_date, and end_date.	Ensure that there is call data for the user within the specified date range.	1. Call the API with valid parameters: GET /call-history?user_id=12345&start_date=2024-11-01&end_date=2024-11-10 2. Verify that the API returns a 200 OK status code. 3. Ensure the response contains an array of call history records. 4. Check that the data returned is correct for the given date range and user.	The API should return a 200 OK status. The response should contain valid call history data for the user within the specified date range.	High
TC002	Verify if the API handles an invalid user_id properly	-	1. Send a GET request to /call-history?user_id=999999&start_date=2024-01-01&end_date=2024-01-31. 2. Check that no call history records are returned in the response.	- Status Code: 404 Not Found or 400 Bad Requests (based on implementation). - Response body should display an error message.	High
TC003	Date Range with Start Date After End Date	User ID 12345 exists in the	1. Send a GET request to /call-history?user_id=1234	- Status Code: 400 Bad Request.	Medium

	(Edge Case)	system.	5&start_date=2024-02-01&end_date=2024-01-01. 2. Confirm that no call history records are returned in the response.	- Response body should contain an error message like "Invalid date range".	

Automation Steps:

Test Script Creation in Postman:

- Create a collection in Postman for the /call-history endpoint.
- Define the three test cases above as individual requests within this collection, using the appropriate parameters.
- For each test case, add test scripts in the "Tests" tab to validate the expected outcomes:
- Status Code Verification: Use pm.response.to.have.status(200) or pm.response.to.have.status(400) depending on the case.
- Response Body Checks: Use assertions like pm.expect(response).to.have.property("error_message") for errors or check array length for valid responses.
- Data Validations: For valid responses, add scripts to confirm that call records fall within the specified date range.

Automation Execution with Newman:

- Export the Postman collection and run it using Newman, a CLI tool for Postman, enabling automated test runs.
- Command: newman run call-history-collection.json -e environment.json -r html
- Generate a report (in formats like HTML or JSON) to track test results and integrate it into CI/CD pipelines (e.g., Jenkins or GitLab CI) for continuous testing.

Task 4: Database Testing for Call Data Records

In cloud telephony, Call Detail Records (CDRs) are critical data stored in a database table for call tracking. The table has the following fields:

Column Name	**Data Type Description**
Call_ID	INT Unique identifier for each call
Caller_ID	INT ID of the caller
Receiver_ID	INT ID of the receiver
Call_Start_Time	DATETIME Start time of the call
Call_End_Time	DATETIME End time of the call
Status	VARCHAR Call status (e.g., 'Completed', 'Missed')

Assignment:

1. Write **3 test cases** to verify the CDRs are correctly recorded in the database. ○ Example: Verify that each call record includes a valid `Call_Start_Time` and `Call_End_Time`.
2. Write a SQL query to retrieve all completed calls made by a specific user within the last month.
3. Describe one data validation check you would perform for this table.

Deliverable: A document containing the test cases, SQL query, and data validation check.

Response:

Test Case ID	Test Case Description	Test Preconditions	Test Steps	Expected Result	Test Priority
TC001	1. Verify that each call record includes a valid Call_Start_Time and Call_End_Time	Calls are logged in the CDR table with populated start and end times.	1. Query the CDR table to select Call_Start_Time and Call_End_Time for each record. 2. Verify that these fields are not NULL and are valid datetime values.	- Call_Start_Time and Call_End_Time fields should contain valid datetime values. - No NULL values in these fields.	High
TC002	2. Verify the status field is correctly set for completed and missed calls	Calls in the CDR table should have varied statuses, including "Completed"	1. Query the CDR table to select records where Status is either "Completed" or "Missed". 2. Confirm that Status accurately reflects the	- All completed calls should have Status as "Completed". - All missed calls should have Status as "Missed".	High

		and "Missed".	actual call outcome.		
TC003	3. Verify duration calculation for each call	Calls in the CDR table with populated Call_Start_Time and Call_End_Time.	1. Query the CDR table and calculate the duration as TIMESTAMPDIFF(SECOND, Call_Start_Time, Call_End_Time). 2. Ensure the calculated duration is non-negative.	- The calculated duration should be non-negative for each call. - Duration should be zero or positive, not negative.	Medium

2. SQL Query

```
SELECT Call_ID, Caller_ID, Receiver_ID, Call_Start_Time, Call_End_Time,
Status
FROM CDR_Table
WHERE Caller_ID = {user_id}
  AND Status = 'Completed'
  AND Call_Start_Time BETWEEN DATE_SUB(CURDATE(), INTERVAL 1
MONTH) AND CURDATE();
```

3. Data Validation Check

One key data validation check for the CDR table is to validate that Call_End_Time is always greater than or equal to Call_Start_Time. This ensures that recorded call durations are accurate and meaningful.

SQL Query for Validation Check:

```
SELECT Call_ID
FROM CDR_Table
WHERE Call_End_Time < Call_Start_Time;
```

Task 5: Exploratory Testing

You are given access to the **Call Analytics Dashboard** in the cloud telephony application. This dashboard shows metrics like total call duration, number of completed calls, and missed calls.

Assignment:

1. Describe **3 areas** you would explore and test on this dashboard.
2. List **3 observations or issues** you might look for during exploratory testing of the analytics dashboard.

Deliverable: A summary of areas to explore and a list of potential observations

Response:

Areas to explore:

Area to Explore	Description	Testing Actions
1. Data Accuracy and Consistency	Ensure that metrics such as total call duration, completed calls, and missed calls are accurate and match actual data.	- Verify that total metrics align with the CDR data. - Test data accuracy across different date ranges (daily, weekly, monthly).
2. Real-time Data Refresh and Timeliness	Check that the dashboard reflects data updates in real time or near real time when calls are completed or missed.	- Verify the dashboard updates promptly for new call data. - Test sync speed when switching between different time filters or refreshing data.
3. Filter and Time Range Functionality	Validate that filters for predefined (Today, Last Week) and custom date ranges display accurate data.	- Test predefined time ranges to confirm accurate data filtering. - Test custom date ranges, including edge cases like overlapping months.

Observations or issues to look for:

Potential Observation/Issue	Description	Testing Actions
Incorrect Data Aggregation	Ensure that total call duration and call counts (completed and missed) are accurately aggregated and match the underlying CDR data.	- Compare dashboard metrics with raw data from the database. - Verify that aggregated data matches across different date ranges.
Performance Issues	Check for lags or slow response times on the dashboard, especially with large datasets or while updating real-time metrics.	- Test with varying dataset sizes to monitor dashboard response time. - Observe performance during real-time data updates.

UI/UX Issues	Look for UI elements with unclear labels, misaligned components, or unresponsive filters that could lead to user confusion.	- Check all labels for clarity and accuracy. - Verify alignment and responsiveness of UI elements across different screen sizes.

Task 6: Scenario-Based Logical Reasoning (Bonus)

The cloud telephony product you're testing has a **Voicemail** feature. Users should receive a notification when a voicemail is left.

Assignment:

1. Describe **two test cases** for this feature, focusing on notification accuracy and timeliness.
2. Identify **one potential edge case** and **one potential security test** you would

consider.

Deliverable: Short responses for each point.

Response:

1.

Test Case	Description	Expected Result
Notification Accuracy	Test the system by leaving a voicemail and verifying that the user receives a correct notification with the caller's ID and timestamp.	User should receive a notification with the correct caller's ID, timestamp, and voicemail details.
Notification Timeliness	Test how quickly the notification is sent after a voicemail is left.	User should receive the notification within a few seconds after the voicemail is left.

2.

Test Case Type	Description
Edge Case	Test what happens when a voicemail is left while the user's phone is in "Do Not Disturb" mode or has no network connectivity.
Security Test	Test the system's behavior when a user's voicemail inbox is accessed by an unauthorized person.