**TASK 1: Prioritization**

1. **“Forgot password” email is not being sent for the user account.**

* **Priority: Urgent**
* **Explanation:** This problem hinders users from recovering their accounts, which is essential for access and security. It significantly affects the user experience and may result in frustration and a loss of trust if not addressed quickly.

1. **Website language switcher is not working.**

* **Priority: High**
* **Explanation:** This impacts the website's accessibility and usability for users who rely on languages other than the default. Although it's not as severe as a forgotten password issue, it still greatly affects the user experience and requires timely attention.

1. **“**About-us” footer links are redirecting to the homepage.

* **Priority: Normal**
* **Explanation:** This issue impacts navigation and access to information but does not hinder users from using essential functions such as account recovery or language selection. It should be resolved within a reasonable timeframe to enhance navigation but can wait until higher-priority problems are addressed.

1. **During registration month date-of-birth dropdown box is only showing “November”.**

* **Priority: Normal**
* **Explanation:** Although this impacts the registration process, users may be able to bypass the issue by choosing a different year or day. It is a functional problem but does not entirely block registration, so it can be resolved after addressing more pressing issues.

1. **“Complete registration” button not working.**

* **Priority: High**
* **Explanation:** This issue directly hinders users from completing the registration process, a crucial step in accessing the service. It is a high priority because it prevents new users from onboarding, which could result in a loss of potential sign-ups and increased user dissatisfaction.

**TASK 2: Test Analysis**

**1. Functional Requirements**

* The verification process that voucher codes are unique, conform to defined formats, and can be generated in large quantities.
* Confirming that tokens are redeemable while validating key criteria such as usage limits, activation status and expiration dates.
* Confirming the accurate logging and reporting of voucher redemption activities.
* Evaluate the processes for activating, deactivating, expiring, and revoking vouchers.

**2. Security**

* Confirming that voucher codes are securely generated and stored.
* Authenticating safeguards against duplicate redemptions, unauthorized access, and manipulation.
* Confirming that only authorized personnel can issue, monitor, and redeem vouchers.

**3. Scalability and Performance**

* Evaluating the system's capacity to process a high volume of voucher generation and redemption requests effectively.
* Confirming that the system sustains acceptable response times even during periods of peak activity.
* Examining the performance of the database in storing, retrieving, and generating reports for vouchers.

**4. Usability**

* Assessing the simplicity, accessibility, and user-friendliness of the voucher issuance and redemption screens.
* Confirming users receive clear and informative error messages when entering invalid voucher details or attempting to redeem a voucher incorrectly.

**5. Integration**

* Testing integration with payment platforms if vouchers are tied to monetary transactions.
* Confirming compatibility with external services like Point-of-Sale (POS) systems or loyalty platforms.

**6. Accessibility**

* Testing the system across multiple devices, browsers, and screen sizes.

**7. Localization**

* Testing for multi-language support if applicable.
* Confirming support for different currencies, time zones, and formats.

**8. Backup and Recovery**

* Ensuring that the voucher data can be restored in case of system failure.
* Testing that the system’s resilience and recovery procedures for critical failures.

**9. Testing and Deployment**

* Setting up staging environments that mirror production settings.

**TASK 2: Test Analysis**

1. **What would you test about this API?**

**I would test the following facets of this API:**

* Ensuring that the API returns appropriate HTTP status codes for different request types (e.g., 200 OK for successful responses, 400 for bad requests).
* Test how fast the API responds to ensure it meets performance requirements.
* Confirming that the response body is returned in the correct format (JSON) and the expected data structure.
* If the API requires authentication, ensuring the correct handling of tokens and credentials.
* Test whether the API handles rate limiting properly.
* Test that the API returns valid and accurate cat facts as per the documentation.

1. **Why would you test this?**

* To ensure the API behaves as expected under normal and edge conditions, providing correct data to users.
* To ensure the API performs within acceptable time limits to avoid delays or interruptions for end users.
* A smooth and predictable API response leads to a better experience for developers and users consuming the service.
* Testing ensures that any required security measures are working properly and that unauthorized users are not able to access data they shouldn’t.
* Ensuring that the facts returned by the API are accurate and consistent with expectations.

1. How would you test this?

* **Set up Postman collection**: Use Postman to create a collection for the Catfact Ninja API.
* **Define test cases**: Each endpoint will have a set of test cases to check response status, response body, and response time.
* **Use assertions in Postman**: Implement test scripts within Postman to check response status codes, time, and data integrity.
* **Set up multiple environments**: If needed, define different environments for testing different versions of the API.
* **Automate test execution**: Run tests automatically using the Postman Collection Runner or integrate with CI/CD tools to run tests regularly.

1. Write three example tests for this API using Postman.

* Notepad txt File attached with source code.

1. Describe the process you went through setting up these three tests.

**Test Case 1: Successful Response**

* I chose the /fact endpoint since it's one of the main features of the API (returning a random cat fact).
* I wrote tests to ensure that the response is successful (status code 200), that the response is in JSON format, and that it contains the expected key fact in the response body.

**Test Case 2: Response Time**

* I used the same /fact endpoint to measure the API's performance.
* The test checks if the response time is less than 500ms, which is a reasonable time for a simple API to return a random fact.

**Test Case 3: Invalid Endpoint**

* I intentionally tested an invalid endpoint (/invalidendpoint) to see how the API handles errors.
* The expected outcome is a 404-status code, which is standard for a "not found" error in REST APIs.