Wokelo AI Assignment

Part 1 Functional Testing

Scenario -1

You are testing a generative AI-based SaaS product that provides automatic text generation and summarization functionalities.

1.1.1 Test Case Development:

Write detailed test cases for the text summarization feature. Be sure to include both positive and negative scenarios.

Positive Test Cases:

- **Short, Simple Text:** Summarize a concise text to verify accurate summarization.
- **Long, Complex Text:** Summarize a lengthy, intricate text to check for key point extraction.
- **Multiple Paragraphs:** Summarize a text with several paragraphs to ensure comprehensive coverage.
- **Technical Jargon:** Summarize a text with specialized terms to assess plain language translation.
- **Different Languages:** Summarize texts in various languages to evaluate multilingual support.
- **Diverse Writing Styles:** Summarize texts with formal and informal styles to test style preservation.

Negative Test Cases:

- **Empty Text:** Input an empty string to verify error handling.
- **Single Sentence:** Input a single sentence to assess the tool's behavior.
- **Invalid Characters:** Input text with invalid characters to check for robustness.
- **Excessively Long Text:** Input a very long text to evaluate handling of large inputs.
- **Unsupported Language:** Input text in an unsupported language to test error handling.

By executing these test cases, we can comprehensively evaluate the text summarization feature's accuracy, efficiency, and robustness.

Please Open the Sheet to see the Test Cases :

Scenario 1

1.1.2 Test Environment Setup

Describe your plan for executing the test cases you have written. Include the steps you would take from setting up the testing environment to reporting defects.

- **Install the Product:** Install the SaaS product on your computer or use the provided testing environment.
- **Prepare Test Data:** Collect a variety of text documents, including short and long articles, technical and non-technical content, and documents in different languages.

Test Case Execution

- **Input Text:** Use the product to input your test documents.
- **Check Output:** Verify the generated text and summaries:
    - Is the output accurate and relevant to the input?
    - Is the output grammatically correct and well-written?
    - Does the output match the specified style and tone?
    - Can the product handle different input lengths and complexities?
- **Record Results:** Note down the results of each test, whether it passed or failed.
- **Report Issues:** If you find any problems, report them to the development team, including:
    - A clear description of the issue
    - Steps to reproduce the issue
    - Expected and actual results

- Screenshots or other evidence

Defect Tracking and Resolution

- **Track Defects:** Keep track of the reported issues and their status.
- **Retest Fixes:** Once a fix is implemented, retest the affected areas.

Test Reporting

- **Summarize Results:** Create a report summarizing the overall test results, including pass/fail rates and any identified issues.
- **Share Report:** Share the report with the development team and other relevant stakeholders.

Additional Tips

- **Use a Test Management Tool:** Consider using a tool like TestRail or JIRA to organize your test cases and track progress.
- **Collaborate with the Development Team:** Work closely with the development team to ensure effective communication and timely resolution of issues.
- **Be Detailed in Your Reports:** Provide clear and concise reports that help the development team understand the problem.
- **Test Different Scenarios:** Try different input types and test edge cases to identify potential issues.

1.1.3 Boundary Testing:

Create test cases for boundary testing of the text generation feature. Explain your approach in identifying these boundaries.

These are the Approaches through which i have tested the ai (e:g.chatgpt)

1. URL-Based Summarization:

- Input a URL directly.
- Assess the tool's ability to extract and summarize the content.
- Consider challenges like dynamic content, complex layouts, and ad-heavy pages.

2. Image-Embedded Text:

- Input text with images.
- Evaluate the tool's focus on textual content.
- Consider potential issues with image-rich content.

3. Large Text Inputs:

- Input extremely long texts.
- Assess the tool's ability to generate concise summaries.
- Consider computational cost and contextual understanding limitations.

Additional Considerations:

- **Language Support:** Test the tool with various languages to evaluate multilingual capabilities.
- **English Correction:** Assess the tool's ability to identify and correct grammatical errors.

By systematically testing these scenarios, we can gain valuable insights into the tool's strengths and weaknesses. This knowledge can help optimize its performance and identify areas for improvement.

Please Open the Sheet to See the results of the Test Cases

Scenario 1 -Boundary testing

Scenario -2

You are testing google calendar page (https://calendar.google.com/calendar/) as a google user.

1.2.1 Test Case Development:

Write detailed test cases for the commonly used features. Be sure to include both positive and negative scenarios.

For the details test case and results please go through this sheet

Scenario 2

Positive Test Cases:

1. **Event Creation:**
    - Create a single-day event.
    - Create a multi-day event.
    - Create an all-day event.
    - Create a recurring event (daily, weekly, monthly, yearly).
    - Set event reminders (email, popup).
    - Add guests to an event.
2. **Event Editing:**
    - Edit event title, start/end time, location, description.
    - Change event status (busy, free, tentative).
    - Add or remove guests.
    - Modify event reminders.
3. **Event Deletion:**
    - Delete a single event.
    - Delete a recurring event instance.
    - Delete an entire recurring event series.
4. **Calendar View:**
    - Switch between day, week, month, and year views.
    - Filter events by specific calendars.
    - Search for events by keyword.
5. **Calendar Sharing:**
    - Share a calendar with specific people.
    - Set sharing permissions (view only, edit).
    - Unshare a calendar.

Negative Test Cases:

1. **Event Creation:**
    - Create an event with invalid date and time.
    - Create an event with a conflicting time slot.
    - Try to create an event with invalid guest email addresses.
2. **Event Editing:**
    - Try to edit an event that is in the past.
    - Try to edit an event that you don't have permission to edit.
3. **Event Deletion:**
    - Try to delete an event that doesn't exist.
    - Try to delete an event that you don't have permission to delete.
4. **Calendar View:**
    - Try to filter events with an invalid filter.
5. **Calendar Sharing:**
    - Try to share a calendar with an invalid email address.
    - Try to set invalid sharing permissions.

1.2.2 Test Execution Plan:

Describe your plan for executing the test cases you have written. Include the steps you would take from setting up the testing environment to reporting defects.

Test Execution Plan: Google Calendar for Students

Scenario: Testing Google Calendar functionalities as a student user.

Test Environment:

1. **Device:** Laptop/Computer or Smartphone with internet access.

2. **Browser:** Google Chrome (latest version recommended).
3. **Google Account:** A student Google account is required (e.g., school email address).

Test Execution Steps:

1. **Login:** Open Google Calendar (https://calendar.google.com/calendar/) and log in using your student Google account.
2. **Adding Class Schedule:**
    - Click on "Create" and choose "Event."
    - Enter the name of the class as the event title (e.g., "Math").
    - Set the recurring schedule for the class (e.g., "Weekly on Tuesdays and Thursdays").
    - Specify the start and end time of the class.
    - (Optional) Add location details if classes are held online or in a specific room.
    - Click "Save."
    - Repeat steps for all classes in your schedule.
3. **Adding Assignments and Exams:**
    - Click on "Create" and choose "Event" or "All-day event" depending on the deadline.
    - Enter the assignment/exam name as the title.
    - Set the due date or exam date for the event.
    - (Optional) Include details like subject, specific instructions, or link to resources.
    - Set a reminder notification for upcoming deadlines (e.g., 1 day before).
    - Click "Save."
    - Repeat steps for all assignments and exams.
4. **Sharing Calendar:**
    - (Optional) Click on the three dots next to "My Calendar" on the left menu.
    - Select "Settings and sharing."

    - Click "Add people" and enter the email address of someone you want to share your calendar with (e.g., study group partner).
    - Choose the level of access you want to grant (e.g., See only free/busy).
    - Click "Save."
5. **Verifying Functionality:**
    - Check if all events are displayed correctly in the calendar view.
    - Ensure reminders pop up for upcoming deadlines as planned.
    - Test editing and deleting events to confirm smooth operation.

Reporting Defects:

- If you encounter any issues during testing, create a detailed report. The report should include:
    - **Description of the issue:** Explain what functionality is not working as expected.
    - **Steps to reproduce:** Provide clear steps for replicating the issue.
    - **Expected outcome:** Describe the expected behavior in this scenario.
    - **Screenshots (optional):** Add screenshots if they help illustrate the problem.
- Depending on your school's system, report defects to a designated teacher, IT support, or a dedicated feedback channel.

1.2.3 Boundary Testing:

Create test cases for boundary testing of the create event feature. Explain your approach in identifying these boundaries.

For more detailed results and test case please follow the Sheet

Scenario 2 -Boundry Testing

This is the Approach

Simple Approach to Boundary Testing for Google Calendar

What is Boundary Testing? Think of boundary testing as testing the edges of a box. For Google Calendar, we want to test what happens when we create events at the very beginning or end of a day, or when we make them very short or very long.

How to do Boundary Testing for Google Calendar:

1. **Identify the Limits:**
    - **Time:** What's the earliest time you can start an event? What's the latest?
    - **Date:** What's the earliest date you can schedule an event? What's the latest?
    - **Duration:** How short can an event be? How long can it be?
2. **Test the Edges:**
    - **Time:**
        - Create an event starting at 12:00 AM (midnight).
        - Create an event ending at 11:59 PM (midnight).
        - Try creating an event with a start time before 12:00 AM or after 11:59 PM.
    - **Date:**
        - Create an event for the earliest possible date.
        - Create an event for the latest possible date.
        - Try creating an event for a date that doesn't exist.
    - **Duration:**
        - Create a very short event (e.g., 1 minute).
        - Create a very long event (e.g., 24 hours).
        - Try creating an event with a negative duration.
3. **Check the Results:**
    - Does Google Calendar accept these boundary values?
    - Does it give you an error message if you try to enter invalid values?
    - Does it correctly display and schedule the events

Part 2: User Experience Testing

2.1 Usability Testing:

Design a usability testing scenario specifically for the AI-generated outputs. Include key metrics you'd evaluate.

Usability testing ensures that AI outputs are clear, relevant, and actionable for the end-users.

Objective

Gather and use feedback from students to improve the usability of AI-generated outputs (e.g., study material summaries, chatbot responses, or recommendations).

Steps to Gather Feedback

1. **Surveys and Polls**
    - **What to Do:** Create quick surveys with a few questions.
    - **How to Share:** Use Google Forms or Microsoft Forms to distribute the survey via email, WhatsApp, or your college group chat.
    - **Example Questions:**
        - How accurate was the AI's output? (Rate 1-5)
        - Was the output easy to understand? (Yes/No)
        - What could make the AI more useful?
2. **Observation During Use**
    - **What to Do:** Watch students use the tool during a testing session.
    - **How to Collect Feedback:** Note any confusion, pauses, or repeated actions.
    - **Example:** If they keep asking the AI the same question, it might mean the output was unclear.
3. **In-App Feedback Button**
    - **What to Do:** Add a button in the app for feedback (e.g., thumbs up/down or a short text box).
    - **Why:** This allows students to share thoughts while using the tool.
4. **Quick Interviews**
    - **What to Do:** Ask students to talk about their experience with the tool.
    - **Sample Questions:**
        - What did you like most?
        - What was confusing or missing?
        - Would you recommend this tool to a friend?

Using Feedback

1. **Sort and Prioritize Feedback**
    - **Group Issues:** For example, unclear summaries, missing key points, or irrelevant suggestions.
    - **Prioritize:** Fix common or critical issues first.
2. **Make Updates**
    - Implement improvements based on feedback (e.g., make summaries shorter or add more examples).
3. **Re-Test with Students**
    - Share the updated tool with students and gather feedback again to see if it's better.

Tools for Feedback Collection

- **Google Forms:** For surveys.
- **Trello or Notion:** To organize and track feedback.
- **Discord/WhatsApp Groups:** For ongoing discussions.

Continuous Improvement Plan

1. **Weekly Check-ins:** Ask a few students for feedback weekly.
2. **Share Updates:** Let students know how their feedback has improved the tool.
3. **Create a Feedback Loop:** Encourage them to report any new issues or suggestions regularly.

Part 3: Reporting and Documentation

3.1 Bug Reporting:

Illustrate how you would document and report a critical bug found during testing. Include all relevant details like steps to reproduce, severity, priority, and possible fix.

I have taken a basic example of testing an AI-SAAS Platform: Rytr.me to showcase my testing skills. And here's how I will be documenting and reporting it.

Bug Reporting - Google Sheets.pdf

3.2 Test Documentation:

Template a concise test summary report for the stakeholders highlighting the testing

outcomes, critical issues, and next steps.

Test Summary Report.pdf

Part 4: Scenario-Based Questions

4.1 Risk Assessment:

Assume the AI service often generates incorrect summaries under specific circumstances.
How would you identify and prioritize the risk areas to focus your testing efforts?

This report addresses a critical issue in an AI-based SaaS platform, where summaries for current or dynamic topics often include inaccuracies or fabrications. This impacts user trust and the platform's credibility, especially for professionals relying on precise information. The report identifies key risk areas, prioritizes testing on time-sensitive queries, and outlines a mitigation plan to improve training data, implement validation mechanisms, and enhance overall accuracy. It aims to guide stakeholders in resolving these challenges effectively.

Risk Assessment.pdf

Submitted By : AMAN KUMAR
Enrollment No. : 230064
Email :aman.k23csai@nst.rishihood.edu.in
Reference : Chatgpt , Rytr.me , Google Calendar , Marker.io , Copy.ai