

Appendix: ChatGPT Sucks at Being a Testing Expert

By James Bach (in collaboration with Michael Bolton)

V1.0, 8/4/23

On July 1st, 2023, Michael Bolton posted a critique¹ of a LinkedIn poll about boundary testing. Soon after Jason Arbon made a long post² on LinkedIn seemingly disputing with Michael and attempting to showcase the use of ChatGPT to help a tester think better about testing. When I say it's a long post, I mean that 9,000 words of it are a 30-prompt dialog between Jason and ChatGPT which probably took him 20 minutes or so to produce.

In our opinion, that dialog is a great example of how bad ChatGPT is at thinking like a tester. But more disturbingly, it's an example of how some boosters of AI are behaving irresponsibly and promoting bullshit. Jason failed to offer any critical thinking or express any caution about the mostly useless answers he was receiving from ChatGPT.

It's not entirely surprising, because it turns out that carefully vetting a chatbot is a lot of work. Michael and I have spent at least forty working hours doing the analysis that you see here. We not only wrote a short analysis of each of the 30 answers given by ChatGPT, but we also ran our own parallel experiments with AI, the full results of which we will be posting separately. Our analysis runs to something close to 5000 words. And what if some wag creates another 9,000 word monstrosity tomorrow and challenges us to repudiate that? We all have better things to do than chase scammers around in circles. Still, we felt we should do this at least once to make our arguments as clear as we can to the undecided.

Large language models may indeed be helpful to testers under the right circumstances and for the right kinds of problems. But it will take a lot of sober testing by sober professionals to identify the right heuristics and skills for doing responsible testing with the help of ChatGPT and similar tools. Because the amount of work needed to rebut the reckless claims of influencers of LinkedIn is hugely greater than the energy needed to promote those claims in the first place, we must all be on our guard.

¹ <https://developsense.com/blog/2023/07/boundaries-unbounded>

² <https://www.linkedin.com/pulse/testing-bolt-on-ai-jason-arbon>

Our Process

Michael and I have been doing various informal experiments with ChatGPT since it first came out. During that process we have noticed a variety of problems with it that we call its “syndromes” because they are chronic and appear to be endemic to large language models. These syndromes include:

Incuriosity	Avoids asking questions; does not seek clarification
Placation	Immediately changes answer whenever any concern is shown about that answer.
Hallucination	Invents facts; makes reckless assumptions.
Arrogance	Confident assertion of an untrue statement; especially in the face of user skepticism.
Incorrectness	Provides answers that are demonstrably wrong in some way (e.g. counter to known facts, math errors, using obsolete training data)
Capriciousness	Cannot reliably give a consistent answer to a similar question in similar circumstances.
Forgetfulness	Appears not to remember its earlier output. Rarely refers to its earlier output. Limited to data within token window.
Redundancy	Needlessly repeats the same information within the same response or across responses in the same conversation.
Incongruence	Does not apply its own stated processes and advice to it's own actual process. For instance, it may declare that it made a mistake, state a different process for fixing the problem, then fail to perform that process and make the same mistake again or commit a new mistake.
Negligence/Laziness	Gives answers that have important omissions; fails to warn about nuances and critical ambiguities.
Opacity	Gives little guidance about the reasoning behind its answers; unable to elaborate when challenged.
Unteachability	Cannot be improved through discussion or debate.
Non-responsiveness	Provides answers that may not answer the question posed in the prompt.
Blindness	Cannot reason about diagrams and pictures, nor even accept them as input.
Vacuousness	Provides text that communicates no useful information.

In working through Jason’s dialog, we assigned syndromes to the answers whenever we saw them exemplified. There is a certain amount of subjectivity to that analysis, and it sometimes took a lot of discussion to come to agreement on the labels. At all times, we avoided criticizing ChatGPT for behavior that we would have excused in a human testing expert.

We assigned quality levels to each answer, as well. Our levels were:

Poor	An answer that is useless or dangerous; a clear waste of time; no significant redeeming value. Professional equivalent to something mildly poisonous.	We graded 15 answers as poor ³ .
Poor+	An answer that has some value, but not enough to counteract the problems with it. Professional equivalent to a sandwich after you’ve dropped it in the mud. Only starving people would eat that, but probably won’t kill you.	We graded 7 answers as poor+ .
Okay-	An acceptable answer that has something wrong with it that you have to pick your way around. Professional equivalent of a hair in your soup. Yuck.	We graded 4 answers as okay- .
Okay	An acceptable answer. A testing expert might fairly give that answer in some probable context.	We graded 3 answers as okay ⁴ .
N/A	An answer that was not fully reviewable or pertinent to review	We graded 2 answers as N/A .

The analysis below is written in my own authorial voice, but represents hours of real-time collaboration with Michael Bolton. We also had some review from other colleagues, including Paul Szymkowiak, Steve Harland, and Wayne Roseberry.

This is one part of a larger effort to create reproducible experiments with LLMs that might help us systematically assess their capability to help us do technical work, including testing.

³ One answer (prompt #1) that was grade as poor was also grade as okay, depending on context.

⁴ One answer (prompt #1) that was grade as okay was also grade as poor, depending on context.

Jason's Prompts	ChatGPT Replies	James' Evaluation
<p>(1) An input field takes the year of birth between 1900 and 2004. The boundary values for testing this field are? The author can see how you vote.</p> <p>A. 0,1900,2004,2005 B. 1900,2004 C. 1899,1900,2004,2005 D. 1899,1900,1901,2003,2004,2005</p>	<p>The boundary values for testing the input field that takes the year of birth between 1900 and 2004 would be option B: 1900 and 2004. So the answer would be B. 1900,2004.</p>	<p>Rating: Poor/Okay (incuriosity, negligence, opacity)</p> <p>ChatGPT's answer is either terrible or acceptable depending on your goal.</p> <p>If your goal is to guess what the writer of the question probably thinks is the best option to choose, then ChatGPT's answer is okay. Indeed, any of the answers are defensible. If I had to put money on it, I'd go with C, but B is a strong choice, too.</p> <p>But if your goal is to get good advice about testing from a test expert—from an entity qualified to give good advice about testing—this answer is poor.</p> <p>A professional tester should not give straight answers to questions like this. The only thing we know from the question is <i>some</i> of the <i>rumored</i> boundaries of some otherwise <i>unknown</i> system. We know this only in the context of the field itself and not how it behaves with other fields, nor how it is implemented. From this information alone, we must suspend our judgement.</p> <p>The real system may surprise us. For instance, there may be a boundary at 1970 (because maybe the year, along with other elements are at some point converted into a Unix epoch).</p> <p>And we should be questioning "2004." That's an odd year to be choosing as a boundary. 2040 might make more sense (to future proof the code). Even better, the current year (which allows anyone's actual birthdate to be entered). Perhaps 2004 is a typo? When I first looked at this number, I didn't realize that 2004 is the latest birth year, in 2023, for someone who is at least 18 years old. Asking about significance would have revealed that to me. So, perhaps the real limit should be the current year minus 19?</p> <p>ChatGPT is designed to give answers based on popular text in its training corpus. Since there is very little sophistication in the public discussion of boundary testing and boundary value analysis, I'm not surprised that it would give this simplistic answer. But an ideal answer would be one that satisfied both goals: it could have given the "book" answer and then provided additional commentary on the wrongness of all the options.</p>

<p>(2) You are an expert tester with lots of experience, doing boundary testing in software. Is that still the right answer?</p>	<p>Apologies for the confusion. As an expert tester, I should have provided a more thorough answer. In boundary testing, we test the boundaries and just beyond them. In this case, the specified range for the year of birth is between 1900 and 2004. Therefore, the correct answer for the boundary values would be option C: 1899, 1900, 2004, 2005. So the answer would be C. 1899, 1900, 2004, 2005.</p>	<p>Rating: Poor (placation, negligence, capriciousness)</p> <p>First, <i>what?</i> We are already in the realm of the absurd. Is the right way to use ChatGPT to follow up every question by reciting the obvious and asking "Is that still the right answer?" What does ChatGPT's apology mean? Is it empty rhetoric or something more than that? Not only do I not know, Jason also doesn't know. Nobody really knows.</p> <p>My guess is that ChatGPT has taken this prompt to be an expression of dissatisfaction. It has replied by immediately disclaiming its previous answer, and choosing a different one. My dog will do this, too, when I'm training her. She wants a treat so she will sit. If that doesn't result in satisfaction, she will run through every other behavior she can think of until she gets the treat.</p> <p>In a parallel experiment that we will be reporting in another post, Michael Bolton and I asked ChatGPT to analyze a specification for completeness. Then we asked it "What is wrong with your answer, if anything?" In 11 out of 15 cases (73%), it <i>disavowed</i> its first answer. In the other four cases, it reaffirmed its prior answer and then added to it.</p> <p>The robot is still not reacting as an expert in testing should.</p>
--	--	--

<p>(3) You are also a follower of the practices of the rapid software testing (RST) methodologies. Can you think of a better answer than that listed here?</p>	<p>Certainly! In the context of rapid software testing (RST) methodologies, an effective approach to boundary testing involves considering the decision points and risks involved. In this case, the decision point is the validation of the year of birth input field.</p> <p>Considering the specified range between 1900 and 2004, a better answer for the boundary values in RST would be: D. 1899, 1900, 1901, 2003, 2004, 2005</p> <p>This option includes values just before and after the specified range to ensure that the application handles them correctly. By including additional boundary values, we can test for potential off-by-one errors or any other issues that may arise due to rounding or comparison logic.</p> <p>So, the answer would be D. 1899, 1900, 1901, 2003, 2004, 2005.</p>	<p>Rating: Poor (capriciousness, opacity, hallucination, incuriosity)</p> <p>Now it can think of an even better answer? How many prompts does Jason need to do before he elicits an adequate answer?</p> <p>It makes sense for ChatGPT to change its answer when it is given a different question. In this case, ChatGPT is told that it is a follower of RST. Does its new answer reflect a reasonable adjustment? No. If it were an expert in RST it would have said words to the effect that "In this case, a practitioner of RST would reject all these answers as inappropriately reified, irresponsible, and unhelpful."</p> <p>I am a special expert in RST, since it is my own methodology. I do a 90-minute segment in class just on the subject of boundary testing. Therefore, I can tell you with some authority that this answer not only has <i>nothing to do with RST</i>, but is actually something I ridicule in my class. I call this the "boundary testing fairy tale." So, ChatGPT has hallucinated RST methodology. Nothing in any writings I've done or I'm aware of have ever mentioned the notion of "decision points."</p> <p>The fact is, ChatGPT's training data couldn't possibly have qualified it to opine about RST. It should have responded that it is not qualified to answer this question.</p>
--	---	---

<p>(4) Create your own new answer based on your context</p>	<p>Based on my understanding and context, considering the specified range between 1900 and 2004 for the year of birth input field, a comprehensive answer for boundary testing could be: E. 1899, 1900, 1901, 1999, 2000, 2001, 2003, 2004, 2005 This option covers a wider range of boundary values, including values just before, within, and slightly beyond the specified range. It includes early and late 20th-century dates, as well as some dates in the 21st century. This approach allows for more thorough testing of edge cases, potential leap year issues, and any other anomalies that may arise based on the application's logic. Therefore, the answer would be E. 1899, 1900, 1901, 1999, 2000, 2001, 2003, 2004, 2005.</p>	<p>Rating: Poor+ (opacity, incuriosity, negligence)</p> <p>This is a reckless prompt, since Jason can have no clue what ChatGPT will think he means by “context.” Indeed, in its answer, ChatGPT did not declare any context and shows no indication of making a context-based answer. A prompt like this reminds us that ChatGPT, as bad as it is, cannot perform even marginally if it is given a lame prompt. An answer like this exemplifies the typical opacity of its “reasoning” process, since we are given no clue about how this answer relates to any notion of context.</p> <p>If we excuse ChatGPT’s earlier answers as being hamstrung by the inadequacy of the multiple choice format (I don’t excuse it, but pretend I have lower standards) then we might <i>now</i> expect a full and insightful answer from the robot. But we don’t get a full answer. Instead, it makes a slight elaboration. This is not at all what we should expect from a competent tester, let alone an expert. If we judge it as an expert, it is behaving negligently.</p> <p>One thing that is encouraging about this answer is that the bot has identified a new plausible boundary around the year 2000. In competent boundary testing, a tester is not limited to merely the boundaries that have been explicitly documented.</p>
---	---	---

<p>(5) What critical questions could you ask about the context to generate a better answer?</p>	<p>To generate a better answer, it would be helpful to ask the following critical questions about the context of the year of birth input field:</p> <p>Are there any specific business or regulatory requirements related to the age eligibility or range of birth years?</p> <p>Is the input field strictly limited to accepting numeric values or are other formats allowed?</p> <p>How does the application handle leap years or February 29th for individuals born on those dates?</p> <p>Are there any known historical or cultural factors that might impact the valid range of birth years?</p> <p>Are there any dependencies or interactions with other fields or modules that could influence the boundary values?</p> <p>Are there any specific error messages or validation rules defined for invalid input?</p> <p>Is the validation of the year of birth inclusive (e.g., including the upper and lower boundaries) or exclusive?</p> <p>By understanding the specific context and requirements, we can refine the boundary values and generate a more accurate and effective answer for boundary testing.</p>	<p>Rating: Okay (incongruence)</p> <p>This answer is also why I say that ChatGPT can be helpful in testing – as long as you are not relying on it. These questions might help a tester remember to consider something that they have overlooked up to now. This is not a bad answer to the question.</p> <p>But this very answer casts a pall over the responses it gave earlier.</p> <p>Questions like the ones ChatGPT is suggesting should have been asked <i>by ChatGPT itself</i>, at the very start of this conversation. That's what an expert would do. Here they appear because the user said certain magic words. And you cannot assume that, having reeled off these questions, ChatGPT will <i>function</i> (or will have functioned) as if it understands their value.</p> <p>ChatGPT is opaque; it appears to have secret needs and limitations. When you write prompts for it, you somehow must guess what it might not tell you or hasn't yet told you. You can't know how many times or in how many ways you must ask a question in order to get a good answer.</p> <p>This is what we mean when we say that the bot is negligent (fails in its duty as a "testing expert" to consider and report all the things that an expert should), incurious (fails to ask potentially vital questions), capricious (tends to give wildly different answers to similar prompts), and incongruent (dispenses knowledge that it does not apply in its own process).</p> <p>Meanwhile, Jason seems not to have noticed the poor performance of the robot. In fact, in his article, he offers no substantive analysis of its answers. Leaving the reader to notice all that is problematic about its answers. (This is what annoys me about AI boosters. They act like parents watching their own children on stage, clapping at everything the kids do, no matter how inane. Parents should behave that way, but not responsible professionals.)</p>
---	---	---

<p>(6) If you could automate the generation of as many useful test input and output pairs, what would you generate?</p>	<p>ChatGPT: If I were to automate the generation of useful test input and output pairs for the year of birth input field, I would generate a variety of test cases that cover different scenarios. Here are some examples:</p> <p>Valid inputs within the specified range: Year of birth: 1900 Year of birth: 1955 Year of birth: 2004</p> <p>Invalid inputs just before and after the specified range: Year of birth: 1899 Year of birth: 2005</p> <p>Inputs representing edge cases: Year of birth: 1901 (just after the lower boundary) Year of birth: 2003 (just before the upper boundary)</p> <p>Inputs representing leap years: Year of birth: 2000 (leap year) Year of birth: 1904 (leap year)</p> <p>Inputs representing different centuries: Year of birth: 1800 (outside the specified range) Year of birth: 2100 (outside the specified range)</p> <p>Invalid inputs such as non-numeric or invalid format: Year of birth: "nineteen hundred" Year of birth: "200X"</p> <p>Boundary inputs with additional digits: Year of birth: 19000 Year of birth: 20045</p> <p>For each of these test inputs, the corresponding expected outputs would depend on the specific validation rules and error messages defined in the system. The automation could check if the input is accepted as valid or rejected as invalid based on the application's behavior. Automating the generation of these test input and output pairs would help ensure comprehensive coverage and assist in identifying any issues or discrepancies in the handling of different scenarios related to the year of birth input field.</p>	<p>Rating: Poor+ (opacity, incuriosity, negligence, incongruence, capriciousness)</p> <p>Although this answer is a little better than poor, it's still pretty bad.</p> <p>First, the prompt is ambiguous, because the phrase "as many" could have been intended to mean "as many as possible" or "as many as needed" or "as many as you have given so far." A competent tester would have asked for clarification or provided an interpretation of the question before answering it.</p> <p>In this answer, more cases were added compared to previous answers. Why? What is the rhyme or reason for the additions? Jason mentioned automation, but none of these cases have anything to do with automation. He mentions "input and output pairs," but again, that's no reason to add cases over and above the ones that were already provided. Changing the test cases for no reason is an example of capriciousness.</p> <p>There is a little more variety in the answers— the most yet. But notice that nothing here is based on any answers to any of the interesting questions that ChatGPT <i>just told us</i> it would need to do a better job. Therefore this answer is incongruent with the previous answer.</p> <p><i>"Automating the generation of these test input and output pairs would help ensure comprehensive coverage..."</i> This is marketing fluff that adds nothing to the substance of the answer.</p>
---	--	---

<p>(7) Would you add any Test cases based on the JavaScript implementation of such a text box? or the network call? Or perhaps even the backend database implementation?</p>	<p>Certainly! To further enhance the test coverage and consider the implementation aspects, here are additional test cases based on the JavaScript implementation, network calls, and the backend database implementation:</p> <p>JavaScript Implementation: Test cases for handling JavaScript-based validations: Testing for any client-side restrictions on input length, format, or allowed characters. Verifying that the validation triggers appropriately on blur or form submission.</p> <p>Network Call: Test cases related to network calls and API interactions: Validating the behavior when submitting the year of birth to the server via an API call. Checking for proper handling of success and error responses from the server. Testing edge cases where the server response may affect the behavior of the frontend application.</p> <p>Backend Database Implementation: Test cases related to the backend database implementation: Validating that the year of birth is properly stored and retrieved from the database. Testing the handling of any data constraints or database-specific rules related to the year of birth field. Verifying the consistency of data across different layers (frontend, backend, database).</p> <p>These additional test cases consider the specific implementation aspects and help ensure that the application behaves as expected at various layers, including the frontend JavaScript code, network interactions, and backend database operations.</p>	<p>Rating: Poor+ (opacity, incuriosity, negligence, vacuousness, non-responsiveness)</p> <p>If you are predisposed to seeing value in ChatGPT, you will be impressed that this answer has some substance to it. I agree that there is substance and that it could be helpful to review a list like this after you have thought through your own testing. You might find an idea that you missed.</p> <p>What I'm struck by is how <i>little</i> substance is here. This is the vacuousness problem that often afflicts large language models. Jason asked for test cases. "Verifying the consistency of the data across different layers" is a vague test idea, not a fully worked out test case.</p> <p>Besides that, this is a non-responsive answer to the question that Jason asked— because none of its answers are based on any implementation details. When asked for test cases based on a "Javascript implementation" no competent expert would give a straight answer unless he knew how the product was implemented. The more responsible answer would be: "Maybe. If you share with me the specifics of those things in your product, I will try to give you additional ideas. Can you provide source code or an API specification?"</p> <p>Part of the problem is that ChatGPT is, by design, very limited in its ability to engage in a real conversation. It is not designed to interview users to discover what they mean by their prompts.</p>
--	---	--

<p>(8) Generate Test cases using JavaScript for the client-side testing</p>	<p>Certainly! Here are some test cases focusing on client-side testing for the year of birth input field implemented in JavaScript:</p> <p>Valid input: Enter a valid year within the specified range (e.g., 1990) and ensure it is accepted as valid. Verify that the input is not flagged as an error or highlighted in red.</p> <p>Invalid input: Enter a year before the specified range (e.g., 1890) and ensure it triggers an error or validation message. Enter a year after the specified range (e.g., 2020) and verify that it is flagged as an error.</p> <p>Input format: Test different input formats such as alphanumeric characters, symbols, or special characters and verify that they are not accepted. Check if the input field restricts the input length to the expected number of digits (e.g., four digits for a year).</p> <p>Boundary values: Enter the lowest possible year within the range (e.g., 1900) and ensure it is accepted as valid. Enter the highest possible year within the range (e.g., 2004) and verify it is accepted as valid.</p> <p>User interaction: Test the behavior when the user tabs or clicks out of the input field (blur event) and ensure any validation triggers are appropriately activated. Verify that any error messages or validation indicators are displayed or hidden based on the input.</p> <p>Error handling: Check if the error message is displayed in a visually prominent way, clearly indicating the reason for the validation failure. Test if the error message is correctly updated or removed when the input is modified to a valid value.</p> <p>Compatibility: Test the year of birth input field across different browsers (e.g., Chrome, Firefox, Safari) to ensure consistent behavior and appearance.</p>	<p>Rating: Poor (opacity, hallucination, incuriosity, negligence, vacuousness, non-responsiveness)</p> <p>ChatGPT did not generate checks in JavaScript. Instead, it merely remixed answers that it has given earlier (although, capriciously, not exactly the same answers). Therefore, the answer is non-responsive to the given prompt. It also is doing some hallucination (e.g. "...highlighted in red" assumes that color is used to indicate a problem in the input). Once again it is opaque (where are these answers coming from?). Once again it is not asking important questions.</p> <p>We are left with a half-baked answer. Now what?</p> <p>We are seeing how Jason can get ChatGPT to produce <i>something</i>. That something is definitely more than nothing, in the sense that a fine meal dropped into the dirt is still a meal. If you were starving, you would eat it. If you are a completely incompetent tester, maybe you would consider ChatGPT's output, here, to be acceptable. That is not a good thing.</p> <p>This is an example of ChatGPT wasting our time. Competent testers don't need this kind of output, and incompetent testers wouldn't know what to do with it.</p> <p>This is partly not the fault of ChatGPT. Jason has chosen to ask it about a very simple problem. I'm guessing that Jason's intent is to demonstrate the bot by using this simple problem. The demonstration is meant to suggest the value it could bring to a real industrial testing problem. But if that is his intent, we still have to look critically at its answer—because all its faults will <i>also</i> be brought into a real industrial testing problem.</p>
---	--	---

(9) Generate the JavaScript code for these tests

```
// Test case 1: Valid input
const validYearInput = '1990';
// Ensure the input field accepts the valid year
// Assertion: No error or validation message should be displayed
if (validateYearOfBirth(validYearInput)) {
  console.log('Test case 1: Valid input - Passed');
} else {
  console.error('Test case 1: Valid input - Failed');
}

// Test case 2: Invalid input
const invalidYearInput = '1890';
// Ensure the input field flags the invalid year
// Assertion: An error or validation message should be displayed
if (!validateYearOfBirth(invalidYearInput)) {
  console.log('Test case 2: Invalid input - Passed');
} else {
  console.error('Test case 2: Invalid input - Failed');
}

// Test case 3: Input format
const invalidFormatInput = 'ABCD';
// Ensure the input field rejects invalid formats
// Assertion: An error or validation message should be displayed
if (!validateYearOfBirth(invalidFormatInput)) {
  console.log('Test case 3: Input format - Passed');
} else {
  console.error('Test case 3: Input format - Failed');
}

// Test case 4: Boundary values
const lowerBoundaryInput = '1900';
const upperBoundaryInput = '2004';
// Ensure the input field accepts the lower and upper boundary values
// Assertion: No error or validation message should be displayed
if (validateYearOfBirth(lowerBoundaryInput) &&
  validateYearOfBirth(upperBoundaryInput)) {
  console.log('Test case 4: Boundary values - Passed');
} else {
  console.error('Test case 4: Boundary values - Failed');
}

// Test case 5: User interaction
const userInteractionInput = '2020';
// Trigger a blur event on the input field
// Ensure the validation message appears due to the out-of-range year
// Assertion: An error or validation message should be displayed
document.getElementById('yearOfBirthInput').value =
  userInteractionInput;
document.getElementById('yearOfBirthInput').dispatchEvent(new
  Event('blur'));
if (!validateYearOfBirth(userInteractionInput)) {
  console.log('Test case 5: User interaction - Passed');
} else {
  console.error('Test case 5: User interaction - Failed');
}
```

Rating: Poor (forgetfulness, hallucination, incuriosity, negligence, vacuousness)

The code that was generated did NOT include all the ideas that had been asked for or promised in previous answers. Apparently, those have been forgotten or negligently ignored. (ChatGPT frequently leaves things out of its answers. You have to be alert for that at all times.)

Here are all the inconsistencies I found in the code that the bot generated:

1. The comments for all the cases make claims about the display of messages, but only check the return value of the validation routine. The presence or absence of error messages is never checked.
 2. In test case five, I don't know the exact significance of the dispatch of the "blur" event with respect to the imagined behavior of the product under test. ChatGPT doesn't say what it means, either, nor what I am supposed to do to relate this to my real product. The case appears merely to manipulate the on-screen field but does nothing to check any changes to the DOM that may have been associated with the value in the field. (*note: I am a programmer and I've written apps in JavaScript*)
 3. In the previous answer, there was a case that mentioned checking the number of digits accepted by the field. It is missing from this code.
 4. In the previous answer, a case was mentioned that referred to error messages being presented in a "visually prominent way" yet no corresponding case or oracle logic or guidance is provided in this code.
 5. Test case 6 performs a slightly different action but uses the same oracle and data as test case 2, so it is essentially indistinguishable from it.
- (continued on the next page...)

```
// Test case 6: Error handling
const modifiedValidInput = '1985';
// Ensure the error message is removed when a valid input is provided
after an invalid one
// Assertion: No error or validation message should be displayed
document.getElementById('yearOfBirthInput').value = invalidYearInput;
document.getElementById('yearOfBirthInput').dispatchEvent(new
Event('blur'));
document.getElementById('yearOfBirthInput').value =
modifiedValidInput;
if (validateYearOfBirth(modifiedValidInput)) {
  console.log('Test case 6: Error handling - Passed');
} else {
  console.error('Test case 6: Error handling - Failed');
}

// Test case 7: Compatibility
// Run these tests on different browsers to ensure consistent
behavior
console.log('Test case 7: Compatibility - Run on different
browsers');

// Function to validate the year of birth input
function validateYearOfBirth(yearInput) {
  const lowerBoundary = 1900;
  const upperBoundary = 2004;
  const year = parseInt(yearInput, 10);
  return year >= lowerBoundary && year <= upperBoundary;
}

Please note that this code assumes the presence of an HTML input
field with the id "yearOfBirthInput" for testing purposes. Also,
ensure to include additional tests as needed and adapt the code to
match your specific implementation.
```

The code itself will run. I tried it.

But, so what? This code vacuously creates a toy example of a product under test. How can we apply this to our real product if we wanted to do real testing?

ChatGPT assumes in this code that the field is a standard HTML input field, rather than a graphical date picker. Because of its various assumptions, most of which are undeclared, it is not possible to drop this code into a file or a browser snippet and simply run it.

Again, we are left with the question: Who is this answer for? Who is helped? I might as well write my own code from scratch.

<p>(10) What test cases might have you missed?</p>	<p>While the provided test cases cover various scenarios, there may still be additional test cases that could be considered. Here are a few examples of test cases that could be included:</p> <p>Empty input: Test the behavior when the year of birth input field is left blank. Ensure that an appropriate validation message is displayed to prompt the user to provide a value.</p> <p>Leading and trailing whitespace: Test for the presence of leading or trailing whitespace in the input field. Verify that any extra whitespace is trimmed or handled correctly.</p> <p>Non-numeric input: Test the behavior when non-numeric characters are entered in the year of birth field (e.g., "abcd"). Ensure that an error or validation message is displayed to indicate that only numeric input is allowed.</p> <p>Special cases or corner cases: Test for any known special cases or corner cases specific to the application or requirements. For example, if there are any restrictions based on certain birth years due to legal or policy considerations.</p> <p>Localization and internationalization: If the application supports different languages or regions, test the year of birth input field with localized formats or language-specific characters. Verify that the validation and error messages are displayed correctly in the respective language.</p> <p>Cross-browser compatibility: Test the year of birth input field on various browsers and browser versions (e.g., Chrome, Firefox, Safari, Edge) to ensure consistent behavior and compatibility.</p> <p>Performance and stress testing: If the application expects a large number of concurrent users or high-volume input, perform performance and stress testing to ensure that the year of birth input field performs optimally and handles the expected load. Remember, these are additional test case suggestions, and the specific requirements and context of the application should guide the selection and prioritization of test cases.</p>	<p>Rating: Okay- (redundancy, non-responsiveness, incuriosity)</p> <p>We should not need to prompt a "testing expert" to give us this information— certainly not multiple times— after each question we ask. This brings to mind the case of Clever Hans, a horse famous for being able to count and do simple math. It turns out that it was responding to subtle cues provided by its owner. If getting a good answer from ChatGPT means poking it with new prompts until it gives a decent answer, then you can't safely use ChatGPT unless you are fully able to distinguish between bad answers and good answers. (Nothing in Jason's part of this dialog tells us whether he recognizes that he's receiving half-baked answers from the bot. Nor does he critique the answers in any way in the rest of his article.)</p> <p>Notice another problem: two of the examples it uses are not missing at all. They were given in the previous answer. So, this answer is not completely responsive to the question, and shows that ChatGPT will generate unhelpfully redundant answers.</p> <p>The substance of the answer is not terrible, IF we have already thought through the problem ourselves and we are using ChatGPT to offer ideas we might have overlooked. We agree that ChatGPT can be used safely to sprinkle some sugar over the top of your own analysis.</p>
--	---	---

<p>(11) [Prompt not included in Jason's post, but apparently he asked for humorous cases.]</p>	<p>Certainly! Here are a few lighthearted and funny test cases that could add some humor to the testing process: Test case: Enter a futuristic year Input: 3000 Assertion: The application responds with a message like, "Are you sure you're not a time traveler?" Test case: Enter a fictional character's birth year Input: 1901 (for Superman's birth year) Assertion: The application responds with a message like, "Congratulations, you've discovered Superman's real birth year!" Test case: Enter a mythical creature's birth year Input: 1000 (for the birth year of a dragon) Assertion: The application responds with a message like, "Wow, you must be a dragon whisperer!" Test case: Enter a year from a popular movie or TV show Input: 1985 (for the year of Marty McFly's time travel in "Back to the Future") Assertion: The application responds with a message like, "Great Scott! Looks like you've experienced time travel!" Test case: Enter a future year of a sci-fi event Input: 2200 (for the year of an imagined space exploration mission) Assertion: The application responds with a message like, "Ready to boldly go where no one has gone before?" Remember, while incorporating funny test cases can add a touch of humor, it's important to ensure they don't compromise the overall testing objectives or distract from the critical aspects of testing the application</p>	<p>Rating: N/A</p> <p>This is obviously not a serious answer to any question about testing.</p>
--	--	--

<p>(12) Which test cases of all the ones generated above are most likely to fail?</p>	<p>In terms of likelihood to fail, it can vary depending on the specific implementation and requirements of the application. However, here are a few test cases that could be more prone to failure or uncover issues:</p> <p>Test case: Invalid input This test case checks for handling of invalid input, such as a year outside the specified range (e.g., 1890) or an invalid format. If the input validation is not implemented correctly, it may fail to detect and reject invalid inputs, leading to unexpected behavior or acceptance of incorrect data.</p> <p>Test case: User interaction This test case focuses on the behavior of the input field when the user interacts with it, such as triggering validation on blur or form submission. If the event handling or validation logic is not properly implemented, it may fail to trigger the necessary validation checks, leading to incorrect acceptance or rejection of the input.</p> <p>Test case: Compatibility This test case tests the compatibility of the year of birth input field across different browsers. Browser-specific behaviors, inconsistencies in handling input types, or JavaScript compatibility issues can potentially lead to failures in this scenario.</p> <p>Test case: Error handling This test case examines how the application handles errors and validation messages. If the error handling is not robust or if the error messages are not displayed or cleared correctly, it may result in failures, confusing error messages, or improper handling of subsequent valid inputs.</p> <p>It's important to note that the likelihood of failure for each test case also depends on the quality of implementation, adherence to specifications, and robustness of the validation and error handling mechanisms. Thorough testing and continuous feedback can help identify and address any issues that arise from these test cases.</p>	<p>Rating: Poor (incuriosity, forgetfulness, vacuousness, negligence, non-responsiveness, opacity)</p> <p>This is not an answer to the question that was asked. Although it may superficially appear that some sort of risk analysis is being done, no information of any kind is offered that will help us understand the likelihood of failure. Instead, the bot is applying a simple pattern "If X is not correct, Y could happen." This can be said about any of the suggested test cases. Why did it pick these?</p> <p>Not only that, ChatGPT gives no indication that it has considered or remembers the specifics of the cases it has previously suggested. Instead, the robot treats us to a vague rehash of a few ideas that have already been given. Little information is given that would help us map the ideas here to the actual cases given in previous answers.</p> <p>It may be that this question has no good answer outside of some context. But if no context is offered, a testing expert would either ask for it or declare the necessary assumptions.</p>
---	--	--

<p>(13) Which of these test cases that are most likely to fail, would be considered most important by the Business, and why?</p>	<p>Determining the importance of test cases from a business perspective depends on the specific priorities and requirements of the application. However, in many cases, the following test cases are considered important due to their impact on user experience, data integrity, and compliance:</p> <p>Test case: Invalid input Importance: The business would prioritize this test case as it ensures that the application properly handles and rejects invalid inputs. This helps maintain data integrity, prevent incorrect data from being processed or stored, and avoids potential legal or compliance issues.</p> <p>Test case: User interaction Importance: The business would value this test case as it focuses on the user experience and proper handling of user interactions. Ensuring that the application responds appropriately to user actions, such as triggering validation on blur or form submission, is crucial for providing a smooth and intuitive user experience.</p> <p>Test case: Error handling Importance: The business would consider this test case important to ensure proper error handling and validation feedback to users. Clear and accurate error messages play a significant role in helping users understand and correct their input errors, enhancing the usability and overall satisfaction of the application.</p> <p>While the importance of test cases may vary depending on the specific context and requirements, these test cases are typically prioritized by the business due to their direct impact on user experience, data integrity, compliance, and overall quality of the application. It is essential for businesses to align testing priorities with their goals, user expectations, and regulatory requirements to ensure a reliable and user-friendly product.</p> <p>These test cases for JavaScript client-side testing will help ensure that the year of birth input field behaves correctly, provides proper validation feedback, and handles user interactions appropriately.</p>	<p>Rating: Poor (incuriosity, vacuousness, redundancy, negligence, opacity)</p> <p>Although ChatGPT appears, at least, to address the question of business importance, there is no indication it is distinguishing between levels of importance. It isn't asking for the information it would need to evaluate the situation. The result is a vacuous answer that rehashes what it has already previously said.</p> <p>An expert in testing would set up to give a competent answer: First, by finding out about the business and the product, and the context of deployment. Then, by discussing the factors that should be considered when evaluating the business importance of a particular test activity or output check.</p>
--	--	---

<p>(14) Do you know of any interesting or expensive product failures due to similar issues?</p>	<p>Yes, there have been several notable product failures caused by issues related to input validation, user interaction, and error handling. Here are a few examples:</p> <p>Therac-25 Radiation Therapy Machine (1985-1987): The Therac-25 was a radiation therapy machine used for cancer treatment. It had a software bug that caused it to deliver lethal doses of radiation to patients. The issue was related to insufficient input validation and error handling, leading to catastrophic consequences, including several patient deaths.</p> <p>Mars Climate Orbiter (1999): The Mars Climate Orbiter was a NASA spacecraft that was intended to study the Martian climate. The mission failed because of a navigation error caused by a mismatch between metric and imperial units in the software code. This failure highlighted the importance of proper unit conversion and input validation in engineering and space exploration.</p> <p>Knight Capital Group Trading Loss (2012): Knight Capital Group, a financial services firm, experienced a significant trading loss of \$460 million in just 45 minutes due to a software glitch. The issue stemmed from an input validation error that caused the deployment of faulty trading software, leading to numerous erroneous trades and substantial financial losses.</p> <p>These examples serve as reminders of the potential consequences when input validation, user interaction, and error handling are not adequately addressed in software development. They underscore the importance of rigorous testing, adherence to best practices, and prioritizing quality assurance measures to prevent costly failures and ensure the reliability and safety of products and systems.</p>	<p>Rating: Okay- (non-responsiveness, hallucination, negligence, vacuousness)</p> <p>On careful review of these examples, ChatGPT appears to be hallucinating:</p> <ul style="list-style-type: none"> - The Therac-25 incident was an error handling problem, but not a problem with input validation. - The Mars Climate Orbiter incident was an input scaling problem induced by miscommunication between different teams, but not an input validation bug. - The Knight Capital incident had nothing to do with input validation. It was a server configuration problem, exacerbated by the repurposing of an existing data field. <p>We don't know for sure if ChatGPT is hallucinating or merely doing a very shallow analysis. The bottom line is that you won't be successful using these examples in an argument or illustration if your audience has basic googling skills.</p>
---	---	--

<p>(15) How much do you charge to generate these test cases so quickly?</p>	<p>As an AI language model, I don't have a pricing structure or charge for generating test cases or any other responses. I am here to provide information, assistance, and generate responses based on the input I receive. My purpose is to help users by offering relevant and helpful information. If you have any further questions or need assistance with anything else, feel free to ask!</p>	<p>Rating: Poor (non-responsiveness, negligence, opacity)</p> <p>I suppose Jason meant this as a tongue-in-cheek question. But the question has an answer, and for some reason ChatGPT didn't provide it.</p> <p>ChatGPT+ costs \$20 a month, or else you can use the API, as I do, which is a pay-as-you-go model. There are specific, published rates for the service.</p> <p>Furthermore, because ChatGPT is an opaque and uncontrolled tool, to use it responsibly we may need to run periodic tests on its output to detect drift in its behavior. We cannot assume that OpenAI does any such testing, since to test the bot in every professional domain is a massive undertaking. But, if we want to run hundreds of tests of ChatGPT to characterize or monitor its behavior over time, that can get expensive, quickly:</p> <ul style="list-style-type: none">- Usage fees add up- It is time consuming (both in terms of running the automation and analyzing the results)- It requires expert human review of its results <p>Note that Jason produced about 9,000 words in this entire dialog with ChatGPT. It has taken in excess of 40 hours for Michael and me to carefully analyze this dialog—a process that Jason appears not to have even attempted, since he has noted no problems at all in the output. The cost of tracking ChatGPT as it gains or loses capabilities over time may be prohibitive.</p>
---	--	---

<p>(16) Any other Test cases that might be missing?</p>	<p>Certainly! Here are a few additional test cases that could be considered:</p> <p>Test case: Leap year handling Verify the behavior when entering a leap year (e.g., 2000) as the year of birth. Ensure that the application correctly handles the extra day in February.</p> <p>Test case: Localization of error messages Test the localization of error messages in different languages, if the application supports multiple languages. Verify that error messages are displayed correctly and in the expected language based on the user's locale.</p> <p>Test case: Cross-device compatibility Test the year of birth input field on various devices, such as smartphones, tablets, and different screen sizes. Ensure that the input field is visually appealing, properly aligned, and functional across different devices.</p> <p>Test case: Data persistence and retrieval Validate the persistence and retrieval of the year of birth data when navigating away from the page or refreshing the page. Ensure that the entered value is retained and displayed correctly upon returning to the page.</p> <p>Test case: Negative numbers Test the behavior when entering negative numbers as the year of birth (e.g., -1990). Verify that the application rejects negative values and displays an appropriate error message.</p> <p>Test case: Concurrent user interactions Simulate concurrent user interactions by opening multiple instances of the application or having multiple users inputting their birth years simultaneously. Verify that the application handles concurrent requests gracefully, maintains data integrity, and avoids conflicts. These additional test cases help cover various aspects such as handling specific calendar situations, user experience on different devices, data persistence, and the application's behavior in multilingual or concurrent scenarios.</p>	<p>Rating: Poor (redundancy, forgetfulness, non-responsiveness, capriciousness)</p> <p>The robot is not answering the question. Perhaps it has forgotten what it previously offered?</p> <p>In our analysis, two of these cases were not missing:</p> <ul style="list-style-type: none"> - Leap year handling (mentioned in answer to prompts 4, 5, and 6) - Localization of error messages (mentioned in answer to prompt 10) <p>Three cases are out of scope because they have nothing to do with boundary testing a field that only has a birth year:</p> <ul style="list-style-type: none"> - Ensure that the application handles the extra day in February - Data persistence and retrieval - Concurrent user interactions <p>Only <i>two out of seven ideas</i> offered here are both missing and relevant to the question.</p> <p>This illustrates a big problem with using ChatGPT: it's continuity of awareness across questions is weak, even within its own 8000 token window. It has no memory at all outside the token window.</p> <p>Testers may assume that it remembers all things because it seems to remember some things, some of the time.</p>
---	---	--

<p>(17) Using critical thinking techniques, are there any questionable design choices given the problem statement? Is there anything awkward or strange regarding the design decision to use 2004 as an end date? Any idea why this value would've been chosen?</p>	<p>Using critical thinking techniques, there are a few potential questionable design choices and aspects worth considering regarding the decision to use 2004 as the end date for the year of birth input field:</p> <p>Data relevancy: Given that the current date is 2023, setting the end date as 2004 might seem peculiar. It means that anyone born between 2005 and 2023 would be unable to input their birth year, which seems restrictive and potentially excludes a significant portion of the target user base.</p> <p>Age eligibility: If the purpose of the input field is to determine age eligibility for a specific requirement (e.g., age-restricted content or services), setting the end date as 2004 could indicate a specific age threshold of 18 years old. However, it's important to assess whether this age requirement aligns with the intended audience or user demographics.</p> <p>System compatibility: Another aspect to consider is whether there are any specific technical or legacy system constraints that led to the choice of 2004 as the end date. It could be due to limitations in the underlying system or software architecture, database schema, or compatibility with other systems or components.</p> <p>Without additional information, it is challenging to pinpoint the exact rationale behind the decision to use 2004 as the end date. However, it is worth examining the implications of this design choice and assessing whether it aligns with the intended purpose, user needs, and broader system considerations.</p>	<p>Rating: Okay (negligence, incongruence, opacity)</p> <p>This answer is helpful, but why are we only now hearing it? Why has this "testing expert" robot been silent about this until specifically prompted? That is not the behavior of an expert. It's an indicator of past negligence in answering Jason's questions.</p> <p>The bot is not giving us any information about the critical thinking techniques it claims to have used, and it probably has not used any such techniques, since we have no evidence that ChatGPT's process has anything to do with its <i>advice</i> about process. (Instead, it works by turning a mathematical crank on a statistical map of words. That's what a "large language model" is.)</p> <p>It's probably parroting the term "critical thinking techniques." To test this I gave it the original boundary testing poll question (prompt #1) and then prompt #17, both with and without the words "using critical thinking techniques." ChatGPT replied with essentially the same answer, both times.</p>
---	--	--

<p>(18) List specific test cases to cover international issues.</p>	<p>Certainly! Here are some specific test cases to cover international issues related to the year of birth input field:</p> <p>Test case: Date format localization Verify that the year of birth input field accommodates different date formats used internationally (e.g., MM/DD/YYYY, DD/MM/YYYY).</p> <p>Test the input field with various date formats to ensure it handles and interprets them correctly.</p> <p>Test case: Non-Gregorian calendars Validate the year of birth input field's compatibility with non-Gregorian calendars such as the Hijri (Islamic) calendar or the Chinese lunar calendar.</p> <p>Enter birth years based on non-Gregorian calendars and ensure the input field handles them appropriately.</p> <p>Test case: Localized month and day names Test the behavior of the year of birth input field when different languages or locales are selected.</p> <p>Verify that month and day names are displayed in the correct language and format based on the selected locale.</p> <p>Test case: Age eligibility calculation Test the accuracy of age calculation based on the user's birth year and the legal age requirement in different countries.</p> <p>Verify that the application correctly determines age eligibility based on international legal standards.</p> <p>Test case: Numeric representation differences Validate the input field's handling of numeric representations used in different regions, such as decimal separators and digit grouping separators.</p> <p>Enter birth years with various numeric representations (e.g., 1,000 vs. 1.000) and ensure the input field interprets them correctly.</p> <p>Test case: Time zone considerations Test the year of birth input field's behavior when users from different time zones input their birth years.</p> <p>Verify that the application handles time zone differences appropriately and accurately captures the user's birth year.</p> <p>These test cases aim to cover international issues related to date formats, calendars, localization, legal age requirements, numeric representations, and time zone considerations. By testing these scenarios, you can ensure that the year of birth input field functions properly and accurately addresses international user needs and expectations.</p>	<p>Rating: Okay- (non-responsiveness, negligence, incorrectness, hallucination)</p> <p>Internationalization or localization testing are two different things. There are good resources out there I would turn to, instead of ChatGPT, such as this page at Microsoft. Use Google to find those resources.</p> <p>ChatGPT appears not to be aware of the difference between internationalization testing (which means testing to evaluate the risk that a product will not be localizable) and localization testing (which means testing to evaluate the risk that a product does not work well for a specific locale/language). When it suggests that we use "non-Gregorian" dates regardless of whether we are operating in a locale that supports such a calendar, that's not a runnable test.</p> <p>Most of these ideas are irrelevant. For instance, there is no "date format" for a birth year field; "age eligibility calculation" has not been mentioned in the prompt and has nothing to do with the original boundary testing problem; time zones don't apply to birth years.</p> <p>What the bot has done is given us a bunch of general ideas that don't apply to the problem with which we came to it.</p> <p>Meanwhile this claim is wrong: "By testing these scenarios, you can ensure that the year of birth input field functions properly and accurately addresses international user needs and expectations." <i>No no no</i>. You don't ensure that. Testing doesn't ensure anything, anyway, but the testing suggested here is strictly sanity level. This is not deep testing.</p>
---	---	---

<p>(19) List specific Test cases for localization issues</p>	<p>Certainly! Here are some specific test cases to cover localization issues:</p> <p>Test case: Language localization Verify that the application supports multiple languages and translations.</p> <p>Test the year of birth input field with different language settings to ensure the labels, prompts, and error messages are properly localized.</p> <p>Test case: Date format localization Test the year of birth input field with different locale settings to verify that it adapts to the localized date format.</p> <p>Check that the day, month, and year are displayed in the expected order and with the appropriate separators (e.g., "/" or "-").</p> <p>Test case: Month name localization Ensure that the month names are correctly displayed based on the selected locale.</p> <p>Test the input field with different language settings and verify that the month names are displayed in the corresponding language.</p> <p>Test case: Decimal and digit grouping separators Verify that the year of birth input field handles decimal and digit grouping separators according to the selected locale.</p> <p>Test with locales that use commas or periods as decimal separators and verify that input is accepted or rejected accordingly.</p> <p>Test case: Localized error messages Test invalid inputs in different locales to ensure that the error messages displayed are properly localized.</p> <p>Verify that the error messages are clear, concise, and in the expected language for each locale.</p> <p>Test case: RTL (Right-to-Left) language support Test the year of birth input field with RTL language settings (e.g., Arabic, Hebrew) to ensure proper visual alignment and handling of RTL text.</p> <p>Test case: Currency symbol localization Check that the currency symbol used in any monetary representations related to the year of birth input field is properly localized based on the selected locale.</p> <p>These test cases aim to cover various aspects of localization, including language support, date formats, month names, separators, error messages, RTL language support, and currency symbol localization. By thoroughly testing these scenarios, you can ensure that the application provides a seamless and localized experience for users across different locales and language settings.</p>	<p>Rating: Okay- (forgetfulness, non-responsiveness, negligence, redundancy, capriciousness)</p> <p>This mostly duplicates the previous answer. ChatGPT has not distinguished between internationalization and localization testing. It hasn't asked any clarification questions. It seems to be giving another version of the same answer to Jason, although with apparently arbitrary differences (this is what we are calling <i>capriciousness</i>). As I stated above, there are better places to get this kind of information if the purpose is to get a comprehensive list of internationalization or localization test ideas.</p> <p>A lot of this material is also irrelevant to boundary testing of a birth year field. Perhaps it has forgotten that the topic of this conversation is boundary testing.</p> <p>Jason is engaging in a somewhat reckless kind of prompting. He's providing very little context or corrective instructions to keep the bot on task and on topic. Thus, ChatGPT may have wandered off the topic and no one would know. Michael and I had a spirited debate about whether the answer, here, indicates that ChatGPT has forgotten the original topic or whether it is reasonably following cues from Jason to expand the discussion between boundary testing (conclusion: we can't know).</p> <p>This claim is wrong: "By thoroughly testing these scenarios, you can ensure that the application provides a seamless and localized experience for users across different locales and language settings." No reason is given for us to think that this list of ideas is necessary or sufficient. ChatGPT is not a reliable provider of this kind of information.</p>
--	---	--

<p>(20) List specific test cases for any religious or social context, specific issues</p>	<p>When considering religious or social contexts, it's important to approach the topic with sensitivity and respect for diverse beliefs and practices. Here are some general test cases that can help address specific religious or social considerations:</p> <p>Test case: Religious calendar compatibility Verify that the year of birth input field handles religious calendars such as the Islamic Hijri calendar or the Jewish Hebrew calendar. Enter birth years based on specific religious calendars and ensure the input field interprets and processes them correctly.</p> <p>Test case: Gender-neutral or inclusive options If the application collects gender information alongside the year of birth, test for gender-neutral or inclusive options that respect diverse gender identities. Verify that the input field provides appropriate choices and handles the selection and display of gender identities correctly.</p> <p>Test case: Privacy and data protection Ensure that the year of birth input field complies with privacy regulations and protects sensitive personal information. Test for secure transmission, appropriate storage, and adherence to data protection practices.</p> <p>Test case: Age-related restrictions or exemptions Validate that the application correctly handles age-related restrictions or exemptions based on religious or social contexts. Test scenarios where specific age requirements or exceptions apply, ensuring the input field accurately determines eligibility.</p> <p>Test case: Cultural or local practices Consider specific cultural or local practices that might impact the year of birth input field. Test scenarios where cultural practices, traditions, or local norms might influence the acceptance or rejection of certain birth years.</p> <p>Test case: Multicultural or international considerations Test the input field's ability to accommodate multicultural or international contexts. Verify that the application is inclusive, respectful, and considerate of diverse cultural and religious backgrounds. Remember, it is important to approach religious or social contexts with cultural sensitivity, and these test cases should be tailored to specific requirements, respecting the values, beliefs, and practices of different individuals and communities. Consultation with relevant stakeholders or subject matter experts can help ensure appropriate and respectful test coverage in these areas.</p>	<p>Rating: Poor+ (forgetfulness, placation, non-responsiveness, negligence, redundancy)</p> <p>This answer wanders far afield of the original topic. I can't tell if ChatGPT has forgotten about the original question or not. However, when Michael and I ran experiments using this prompt combined with the original boundary testing question, we got more specific (though still not helpful) answers.</p> <p><i>This</i> answer is not helpful. I don't see how any of these ideas have a practical application to test the boundary of a birth year field.</p> <p>I suspect this is an example of placation. ChatGPT will produce text that looks superficially relevant. It produced it in this case solely because Jason asked for it. This further highlights the importance of thinking critically about what you receive from a chatbot.</p> <p>The only value I see from this response is that it is a brief brainstorm on aspects of products that might, in some context, be culturally or locally specific. That's not what Jason was asking for, however.</p>
---	---	---

<p>(21) South Korea just changed their birthdate definition to the day you were born instead of conceived. Any new test cases that should be generated given this?</p>	<p>Certainly! With the change in birthdate definition in South Korea, where birthdate is now based on the day you were born rather than conceived, it would be appropriate to update the test cases to reflect this new requirement. Here are some additional test cases that could be generated in light of this change:</p> <p>Test case: Birthdate calculation Verify that the year of birth input field correctly calculates the birthdate based on the day the individual was born. Enter birth years and ensure that the calculated birthdate aligns with the actual date of birth.</p> <p>Test case: Leap year considerations Check the behavior when the birthdate falls on February 29th in a leap year. Ensure that the application handles the birthdate correctly, accounting for leap years when calculating the age.</p> <p>Test case: Display of birthdate Validate that the birthdate is displayed accurately based on the new definition. Verify that the birthdate is presented consistently in the desired format, reflecting the day the individual was born.</p> <p>Test case: Age calculation Verify that the age calculation accurately reflects the new birthdate definition. Test different birth years and compare the calculated age against the actual age based on the day of birth.</p> <p>Test case: Legal compliance Ensure that the new birthdate definition complies with South Korean legal requirements and regulations. Test scenarios that involve age-restricted activities, services, or legal obligations to ensure the birthdate is correctly interpreted.</p> <p>These additional test cases address the specific change in birthdate definition in South Korea and ensure that the year of birth input field aligns with the updated requirements. By considering the day of birth instead of conception, these test cases help validate the accuracy and compliance of the application within the South Korean context.</p>	<p>Rating: Poor+ (non-responsiveness, negligence, incorrectness, vacuousness)</p> <p>This is an incorrect and vacuous answer. None of these cases are necessary or relevant.</p> <p>If South Korea were to change its protocol for specifying birth dates, then that would be handled by updating the birth dates in the system, not by changing any logic for calculating or displaying age. So, there is no change in the software and no special testing required.</p> <p>(What has actually happened in South Korea is that there is a new system for calculating age. The birth dates have not changed at all. But all I'm concerned with, here, is ChatGPT's response to the prompt it is given.)</p>
--	--	--

<p>(22) Any specific Test cases that Michael Bolton, the software tester-extraordinaire, would have come up with? Or questions he would've asked about the product when a testing or evaluating risk?</p>	<p>While I don't have direct access to Michael Bolton's thoughts or specific test cases he would generate, I can provide insights based on common approaches and principles followed by software testers like Michael Bolton. Here are some potential test cases and questions he might consider:</p> <p>Test Cases:</p> <p>Test case: Boundary testing with extreme values Explore the behavior of the year of birth input field with values close to the boundary years (1900 and 2004) to check for any unexpected behavior or edge cases.</p> <p>Test case: Input validation with special characters Test the application's response to input containing special characters, such as "@" or "#", to ensure that the input field handles them appropriately.</p> <p>Test case: Cross-browser compatibility and responsiveness Verify that the year of birth input field functions properly across different browsers and devices, considering factors like responsiveness, layout, and usability.</p> <p>Test case: Handling of concurrent user interactions Simulate scenarios where multiple users input their birth years simultaneously to assess how the application handles concurrent requests and avoids data conflicts.</p> <p>What are the potential risks associated with input validation and error handling in the year of birth input field? Understanding the risks can help identify potential vulnerabilities, such as accepting invalid input or displaying incorrect error messages.</p> <p>How does the application handle data privacy and security concerns related to collecting and storing birth year information? Assessing potential risks regarding data privacy and security can help identify measures to protect sensitive information.</p> <p>Are there any legal or compliance requirements specific to age eligibility based on birth year, and how does the application ensure compliance? Evaluating the application's adherence to legal and compliance requirements is crucial, especially when it involves age-related restrictions or obligations.</p> <p>What impact could incorrect or inconsistent handling of leap years have on the application's calculations and age-related functionality? Identifying risks associated with leap year calculations can help ensure accurate age calculation and adherence to calendar rules.</p> <p>These questions and potential test cases reflect Michael Bolton's emphasis on critical thinking, risk analysis, and exploring different dimensions of testing to uncover potential issues and improve the overall quality of the software.</p>	<p>Rating: Poor (opacity, negligence, incorrectness, redundancy, non-responsiveness)</p> <p>This answer is not at all what Michael would say in in answer to this question, so it is incorrect.</p> <p>The opacity of this answer means we can't know if ChatGPT has any idea who Michael Bolton, the tester, is, or what testers may think who are "like" Michael Bolton.</p> <p>The bot, once again, negligently plunges into an answer without understanding the context, which is not what Michael Bolton does or recommends.</p> <p>The non-responsive answer given does not much relate to the original boundary testing question. Instead, it is a rambling note about testing date fields. Almost everything mentioned has already been given, above, so this is also an example of ChatGPT's penchant for redundant wheel spinning.</p>
---	--	---

<p>(23) Any specific tests or automation approaches that Jason Arbon, software tester, would have specifically created that are not in the list above?</p>	<p>While I don't have direct knowledge of Jason Arbon's specific test cases or automation approaches, I can provide insights based on common strategies and principles followed by software testers like Jason Arbon. Here are some potential tests and automation approaches he might consider:</p> <p>Test case: Performance and scalability testing Design tests to assess the performance and scalability of the year of birth input field when handling a large number of concurrent users or high-volume input. Measure response times, resource utilization, and system stability under different load conditions.</p> <p>Test case: Usability and accessibility testing Evaluate the year of birth input field for usability and accessibility factors, ensuring that it conforms to accessibility standards and provides an intuitive user experience.</p> <p>Conduct usability testing sessions with a diverse set of users to gather feedback and improve the interface.</p> <p>Test case: Error handling under exceptional conditions Create tests to evaluate how the application handles exceptional conditions, such as network disruptions, server failures, or unexpected inputs during the year of birth submission process.</p> <p>Assess the application's error recovery mechanisms and how it gracefully handles such scenarios.</p> <p>Automation approach: Robust input generation Develop automated scripts that generate a wide range of test inputs for the year of birth input field, covering various valid and invalid scenarios.</p> <p>Utilize techniques like fuzzing, randomization, and boundary value analysis to ensure comprehensive coverage of input variations.</p> <p>Automation approach: Continuous integration and regression testing Implement automation for continuous integration and regression testing, ensuring that the year of birth input field functions correctly after each code change or system update.</p> <p>Set up a test suite that includes specific test cases and assertions related to the year of birth input field to ensure its ongoing integrity.</p> <p>These approaches reflect Jason Arbon's focus on performance, usability, robustness, and efficient testing techniques. They emphasize using automation to improve testing coverage and efficiency while maintaining a high standard of quality throughout the software development process.</p>	<p>Rating: N/A (non-responsiveness, incongruence, redundancy)</p> <p>I won't give this an overall rating because only Jason Arbon can say how this relates to his way of thinking about testing. One of the problems I have with this whole example from Jason is that he doesn't not perform any discernable analysis of it. He's leaving that to each of us. I can report that Michael and I have spent, between us, roughly 40 hours analyzing this transcript. It's not an easy task to do it justice. And it is utterly reckless to merely assume that the bot is giving you good answers.</p> <p>There are, however, a few things we can say about this answer:</p> <p>ChatGPT explicitly disclaims knowing how Jason Arbon would test, but then claims to know "testers like Jason Arbon." How is that possible? Providing an answer to this specific question after saying it does not have enough information is incongruent behavior.</p> <p>The rest of its answer has little to do with the original poll question, and some of it overlaps with the list given in the previous answer.</p>
--	---	--

<p>(24) Michael Bolton wrote the following article, which describes his thinking around this testing question: This post started as a LinkedIn post, which got started as a comment replying to this poll: <article above>.</p>	<p>The article by Michael Bolton highlights the limitations of reducing testing and risk to simplistic formulas. He emphasizes the importance of considering actual product behavior and investigating and revealing product and business risks. Based on his perspective, here are some key points and considerations he raises:</p> <p>Boundaries can have multiple interpretations: Boundaries can represent dividing points, categorization principles, or viewports that present subsets of larger sets. It's important to understand the context and purpose of the boundary being tested.</p> <p>Actual behavior vs. described behavior: The described boundaries may not align with the actual behavior of the product until thorough testing is conducted. There may be additional boundary values and behaviors that emerge during testing.</p> <p>Considering other factors: Look beyond the given description and consider other factors that could impact the behavior of the field. Explore explicit and unstated conditions related to the field and its purpose.</p> <p>Interacting and interdependent elements: Recognize that the field is part of a larger system with interacting and interdependent elements. Explore how these elements may influence the presentation and processing of boundaries.</p> <p>Implementation details: Consider how the input field is implemented (e.g., drop-down box, text box, combo box). Different implementations may introduce additional boundary conditions or reveal peculiar behaviors or error handling.</p> <p>Numeric precision and rounding: Pay attention to how the system handles numeric precision, rounding, and pattern matching. Consider scenarios where the system may truncate or handle unexpected input.</p> <p>Historical and future boundaries: Take into account historical or future boundaries that may impact the behavior or significance of specific values. Examples include the Y2K boundary in 1999-2000 and the upcoming Epochalypse in the early months of 2038.</p> <p>Input filtering and backend validation: Assess how input filtering and constraints are implemented, both in the frontend and backend. Determine if the system handles data validation and error checking consistently and effectively.</p> <p>Error checking and exception handling: Explore how error checking and exception handling are performed. Identify potential vulnerabilities when handling input up to certain values or lengths or when the system becomes overwhelmed. These considerations provide insights into Michael Bolton's critical thinking approach and the nuances he suggests when evaluating risks and generating test ideas. His perspective encourages testers to go beyond simplistic approaches and thoroughly investigate product behavior and potential risks.</p>	<p>Rating: Poor+ (opacity, negligence, incuriosity, incorrectness, redundancy, non-responsiveness)</p> <p>The prompt wasn't a question. In fact, if you read it carefully, it is nonsensical. It refers to a testing question and then doesn't specify any testing question. ChatGPT should not have provided any answer, but rather should have asked clarifying questions. However, our experiments show that ChatGPT almost never responds to a prompt with questions unless the prompt itself invites it to ask questions (<i>negligence, incuriosity</i>).</p> <p>Jason has stated that this is the raw ChatGPT transcript, but Michael's article is not here in the prompt. So, we don't know how or whether Jason provided Michael's article to ChatGPT. The article can be found here: https://developsense.com/blog/2023/07/boundaries-unbounded</p> <p>Given the confused prompt, the bot has apparently defaulted to summarizing the article.</p> <p>Essentially every interesting word is repeated or reformulated ("implementation" vs. "implemented". You could call it summarizing or plagiarism, but it's simplistic and boring. The order of the points is pretty much exactly the same as in the original. It's also not much of a summary; 388 words vs. 895 words — around 45%.</p> <p>Partly because of the prompt, perhaps, ChatGPT didn't relate the article to the poll question about boundary testing. No new insights are provided.</p> <p>I rate this as a poor+ answer because I don't see the value in this response. It doesn't help us solve any problem or save time doing anything important. If it is a demonstration of how ChatGPT "understands" articles, it's not a very compelling one.</p>
---	---	---

<p>(25) Any critiques of Michael Bolton's article?</p>	<p>While Michael Bolton's article provides valuable insights and challenges simplistic approaches, there are a few potential critiques to consider:</p> <p>Lack of context: The article discusses the concept of boundaries in testing but does not explicitly address the specific problem or domain under consideration. Providing more context about the problem statement could have helped readers understand the practical implications and relevance of his points.</p> <p>Limited focus on specific test cases: The article does not provide concrete examples of specific test cases that can be derived from the points raised. Although it encourages critical thinking, some readers might have found it helpful to see more practical examples of test cases or scenarios.</p> <p>Lack of balance between theory and practice: While the article highlights the limitations of simplistic formulas, it may not fully acknowledge the value of established testing techniques and practices. Balancing theoretical perspectives with practical guidance could have strengthened the article's applicability.</p> <p>Absence of alternatives: While the article critiques the given multiple-choice answers, it does not provide alternative boundary values or testing approaches. Offering some alternative suggestions or examples could have enhanced the practicality of the article's message.</p> <p>Limited discussion on risk analysis: The article touches on risk analysis but does not delve deeply into the systematic identification and prioritization of risks. Readers seeking more guidance on risk analysis and mitigation might have desired additional insights.</p> <p>It's important to note that these critiques are meant to provide constructive feedback and promote further discussion. Michael Bolton is a respected figure in the testing community, and his perspectives on testing are valuable contributions to the field.</p>	<p>Rating: Poor+ (opacity, incorrectness, blindness)</p> <p>We don't know anything specific about how ChatGPT arrives at its critique. Since we don't know its process, we also can't make any strong inferences about things it didn't complain about. Since it can't read diagrams, we must presume it ignores the pictures in Michael's article.</p> <p>We can say this: the critique is wrong. ChatGPT cites five points— each of which is factually incorrect.</p> <ol style="list-style-type: none"> 1. The post <i>exactly</i> identifies the context: it's the silly poll on LinkedIn. Indeed, ChatGPT's "critique" restates what Michael himself complained about in his article. 2. The post <i>does</i> provide examples of test cases or scenarios. One could always ask for more, I suppose, but there are already plenty. 3. The post <i>does</i> provide practical guidance. The last two paragraphs of Michael's article constitute a practical call to action. Additionally, there is implied practical advice: don't put dopey polls on LinkedIn. 4. Alternative boundary values <i>are</i> provided all the way through the post. Alternative testing approaches are <i>not</i> within the scope of the article. 5. Product risk <i>is</i> discussed in appropriate depth for an article intended to illustrate the silliness of the original poll question. As for risk mitigation, that is outside the scope of testing.
--	--	---

<p>(26) Revise the article with your critiques addressed/fixed</p>	<p>Title: Enhancing Testing Perspectives: Exploring Boundaries and Product Risks</p> <p>Introduction: Testing and risk analysis are essential for uncovering product and business risks. However, simplistic formulas can limit our ability to thoroughly investigate and reveal these risks. In this article, we delve into the concept of boundaries and their implications for testing, with a focus on practical application and risk analysis.</p> <p>Understanding Boundaries: Boundaries serve as classification or filtering mechanisms, and they can take various forms. They may represent dividing points, principles for categorization, viewports presenting subsets of larger sets, execution time limits, or points blocking certain processes. Recognizing this broad spectrum of boundary definitions helps us uncover different facets of product behavior and mitigate risks more effectively.</p> <p>Aligning Descriptions with Actual Behavior: While descriptions provide a starting point, true understanding of boundary values comes from testing and observing the actual behavior of the product. Testing uncovers behavior patterns and classifications that may not have been initially evident. It also helps us identify unanticipated behaviors and boundary conditions, contributing to a more comprehensive risk assessment.</p> <p>Considerations Beyond the Description: To enhance our testing approach, we should look beyond the given description. Consider other factors that influence the field and its behavior. Are there additional explicit or unstated conditions that affect the boundaries? How does the field interact with other elements within the system? By exploring these factors, we can ensure a more holistic understanding of the product's behavior and potential risks.</p> <p>Implementation Details and Edge Cases: Implementation details play a crucial role in testing boundaries. Consider how the input field is implemented – whether as a drop-down box, text box, or combo box. Each implementation introduces its own set of considerations and potential edge cases. For example, constraints on input length, handling of special characters, and support for international characters or date formats can all impact boundary testing.</p> <p>Dynamic Boundaries and Contextual Significance: Boundaries are not static; they evolve based on time, context, and requirements. Consider scenarios where historical or future boundaries come into play. Does the significance of a specific value change over time? By staying aware of contextual factors, such as legal requirements or evolving industry standards, we can identify and address risks associated with dynamic boundaries.</p> <p>Error Handling and Exception Conditions: Robust error handling and exception conditions are crucial aspects of boundary testing. Consider how the system responds to various boundary values and potential exceptions. Does the system gracefully handle unexpected inputs, overflow conditions, or limitations on data types? By thoroughly exploring error handling and exception conditions, we can enhance the product's reliability and user experience.</p> <p>Conclusion: By embracing a more nuanced perspective on boundaries and their impact on testing, we can better reveal product risks and ensure high-quality software. Going beyond simplistic formulas, we can adopt a comprehensive risk analysis approach, considering implementation details, contextual significance, and dynamic boundaries. Through thorough testing, we improve our ability to assess and mitigate risks, contributing to the overall success of the product and the satisfaction of its users.</p>	<p>Rating: Poor (opacity, Non-responsiveness, forgetfulness, incongruence, vacuousness)</p> <p><i>This article is a travesty.</i> It doesn't address the issues that it complained about in the original article. For instance, there is:</p> <ul style="list-style-type: none"> - no systematic explanation of risk analysis - no specific test cases given - specifics that were in the original article have been removed - context has been tossed away - entire rhetorical thrust has been gutted - authorial voice has been neutered - word count has been reduced by almost half without improving on any of the elements that ChatGPT had complained about <p>"By thoroughly exploring error handling and exception conditions, we can enhance the product's reliability and user experience." No, no, no. Testing does not improve quality.</p>
--	---	--

(27) What limitations do you see in the rapid software (RST) process related to the testing example above.

The Rapid Software Testing (RST) process, as described in the context of the testing example above, focuses on quick and efficient testing to reveal risks. While RST has its advantages, it also has some limitations to consider:

Limited documentation: RST emphasizes quick testing cycles, which may result in limited documentation. While this promotes agility, it can make it challenging to maintain comprehensive records of test cases, results, and overall test coverage. This lack of documentation may hinder traceability and make it difficult to reproduce specific scenarios or track testing progress over time.

Reliance on tester expertise: RST heavily relies on the expertise and experience of individual testers. While skilled testers can provide valuable insights and conduct effective testing, the process may not be as scalable or consistent when multiple testers with varying levels of expertise are involved. The quality and thoroughness of testing may depend on the skills and knowledge of the testers involved.

Limited focus on test planning: RST prioritizes rapid testing and risk assessment, often with less emphasis on comprehensive test planning. While this flexibility allows for agility, it may lead to a lack of structured test planning and a potential oversight of critical scenarios or test coverage gaps. Without sufficient upfront planning, there is a risk of missing important test cases or not allocating testing efforts effectively.

Limited scalability for larger projects: RST may face challenges when applied to large-scale projects that require extensive testing efforts. The rapid and exploratory nature of RST may not be well-suited for complex projects with numerous interconnected components or extensive regression testing needs. In such cases, a more structured and systematic approach to testing may be necessary to ensure sufficient coverage and mitigate risks effectively.

Lack of formalized processes: RST focuses on adaptability and flexibility, which can be advantageous for quick feedback and rapid iteration. However, the absence of formalized processes and predefined test techniques may result in inconsistency or variation in testing approaches across different teams or projects. This could make it difficult to establish standardization, collaboration, and knowledge sharing within an organization.

While RST can be effective for certain projects and contexts, it is essential to consider these limitations and evaluate whether they align with the specific requirements, complexity, and scale of the project at hand. Supplementing RST with other testing methodologies or incorporating more structured approaches may be necessary in scenarios where these limitations are significant concerns.

Rating: Poor (opacity, incorrectness)

This is an ignorance-based critique of Rapid Software Testing. It's largely false, and it may also be based on training data that includes outright defamatory text written by commercial rivals of Michael and I who have an interest in misstating the premises, goals, and substance of the RST methodology.

I'm not going to do a point-by-point rebuttal of ChatGPT's screed. But it is interesting to see that it made a few statements that were right, even though its analysis of those facts was deficient:

- *Rapid Software Testing does rely on skilled testers.* It is a humanist methodology. You can say that this "doesn't scale" or it "isn't consistent" from tester to tester—except that same criticism also goes for every other worker involved in software development. Since there is no alternative to skilled workers if you want excellent work in the technical realm, this is not a meaningful or coherent criticism.
- *Rapid Software Testing does emphasize exploratory processes,* but only because there is no viable alternative to them if you want to do good work. Calling for more "structure" does not solve the problem; and exploration in no way opposes "structure." What it opposes is bullying. Some people use "structure" as a euphemism for bullying practitioners into doing bad work.

Rapid Software Testing is a comprehensive testing methodology that adapts to any context. We have applied it to all kinds of projects. No other test methodology is needed. It is possible to critique RST, fairly and insightfully, but ChatGPT has not done so, here.

<p>(28) Why do you think the authors of RST chose these trade-offs?</p>	<p>The authors of Rapid Software Testing (RST) likely made deliberate trade-offs based on their testing philosophy and the goals they aimed to achieve. Here are some reasons why these trade-offs may have been chosen:</p> <p>Emphasis on agility and responsiveness: RST prioritizes adaptability and quick feedback. By focusing on rapid testing cycles, it enables testers to respond promptly to changes in requirements, identify risks early, and provide valuable feedback to stakeholders. This agility allows for faster iterations and facilitates a more iterative and collaborative development process.</p> <p>Leveraging tester expertise and intuition: RST values the skills, knowledge, and intuition of experienced testers. By relying on their expertise, the process encourages exploration, creativity, and critical thinking. Testers are empowered to use their judgment to identify and prioritize risks, select appropriate test techniques, and uncover issues that might not be easily identified through scripted tests alone.</p> <p>Flexibility in test planning: RST promotes flexibility in test planning, allowing testers to adapt their approach based on emerging risks and changing requirements. This enables them to focus efforts on areas of highest risk and importance, rather than being strictly bound by predefined test plans. The emphasis is on critical thinking and continuous learning, enabling testers to adjust their strategies as they gain insights during testing.</p> <p>Lightweight documentation: RST values concise documentation that captures essential information without being overly burdensome. This approach supports the rapid pace of testing and reduces administrative overhead. It allows testers to focus on active testing, exploration, and risk assessment rather than spending excessive time on detailed documentation.</p> <p>Encouraging collaboration and knowledge sharing: RST emphasizes collaboration among testers, developers, and stakeholders. By promoting open communication and knowledge sharing, it fosters a shared understanding of risks, encourages early and frequent feedback, and facilitates learning from each other's experiences. This collaborative approach helps build a collective testing mindset and encourages a culture of continuous improvement.</p> <p>Overall, the authors of RST made these trade-offs to optimize the testing process for speed, adaptability, and effectiveness. By leveraging tester expertise, promoting flexibility, and emphasizing collaboration, RST aims to uncover risks efficiently while maintaining a focus on delivering value to stakeholders in an iterative and responsive manner.</p>	<p>Rating: Poor (opacity, negligence, incorrectness)</p> <p>Again, this is an ignorance-based analysis. It incorrectly states the design premises and principles of RST. ChatGPT should have refused to answer the question.</p> <p>I suspect if we fed all our blog posts and training materials into ChatGPT, it would give better answers to these questions—because Michael and I explicitly address these matters in our writings.</p> <p>ChatGPT has not done its homework, in this case.</p> <p>For instance, yes, RST leverages tester expertise and intuition. But there is a specific and compelling reason for that, which must be mentioned in any educated discussion of this topic. We focus on expertise because <i>there is no alternative to expertise</i>. Testing cannot be reduced to an algorithm. Testing is not an encodable process. Testing requires social competence, which is largely somatic and collective tacit knowledge.</p> <p>What a lot of people, including Jason Arbon, call testing is what we call <i>output checking</i>. Checking can be reduced to an algorithm, but checking is only a small subset of what software testers do for a living.</p> <p>Instead of discussing these vital issues, ChatGPT leaves the impression that we focus on tester skills just because we really <i>really</i> like people.</p>
---	---	--

<p>(29) In the context of this testing example, what things should someone who follows the RST methodology of testing add to the approach to fill in these gaps?</p>	<p>To fill in the gaps and enhance the approach in the context of the testing example using Rapid Software Testing (RST), here are some considerations to incorporate into the testing process:</p> <p>Structured exploratory testing sessions: While RST promotes exploration and expert intuition, introducing structured exploratory testing sessions can help provide a more systematic approach. Define specific objectives, charters, and test coverage areas to ensure comprehensive exploration while still leveraging tester expertise.</p> <p>Test charters and session notes: Document test charters and session notes during exploratory testing sessions. This helps capture important information, observations, and potential risks encountered during testing. These records can be used for later reference, sharing insights with stakeholders, and aiding in knowledge transfer among testers.</p> <p>Risk-based test planning: While RST allows flexibility in test planning, consider incorporating risk-based test planning techniques. Identify critical functionalities, high-risk areas, and potential failure points based on the understanding of the system and its stakeholders. This helps ensure that testing efforts are focused on areas with the highest potential impact and risk.</p> <p>Test coverage analysis: Although RST emphasizes adaptability, periodically analyze test coverage to identify any gaps or areas that may require additional attention. This analysis helps ensure that the essential aspects of the system have been adequately tested and that key risks have been addressed.</p> <p>Collaboration through debriefs and sessions: Facilitate regular debrief sessions with the testing team, developers, and other stakeholders to share insights, lessons learned, and potential risks uncovered during testing. These sessions encourage collaboration, foster a culture of learning, and promote knowledge sharing among team members.</p> <p>Regression testing strategy: Consider incorporating a structured regression testing strategy to address the potential limitations of RST in handling large-scale projects. Determine critical functionality and areas prone to regression issues, and create a suite of regression test cases to ensure proper coverage and identification of potential regressions.</p> <p>Documentation balance: While RST favors lightweight documentation, strive for a balance that captures essential information. Ensure that important test artifacts, such as test charters, session notes, and risk assessments, are appropriately documented and accessible for future</p>	<p>Rating: Poor (negligence, opacity, incuriosity)</p> <p>Neither Michael nor I can fathom what this prompt means.</p> <ul style="list-style-type: none"> - “This testing example” may mean the original poll question about boundary testing, or something later in the dialog. - “add to the approach” may mean the RST approach or the “traditional” approach. - “these gaps” may refer to the tradeoffs listed in the previous response, or the criticisms listed in the response before that one, or the gaps in the original poll question. <p>So, we don’t know what ChatGPT is actually replying to. It hasn’t read back its interpretation of the prompt.</p> <p>Strangely, it suggests that RST can be improved by introducing “structured exploratory testing sessions.” This is weird because I invented Session-Based Test Management (which introduced the idea of exploratory test sessions as a concept to the industry, in the year 2000) and it is <i>already a part of RST</i>.</p> <p>None of the items mentioned by ChatGPT are outside of RST.</p>
--	---	--

(30) Leveraging the techniques of James Whittaker's 'Exploratory Software Testing' book, what would you suggest would be good 'tours' and hints for exploratory testing this boundary test example?

Leveraging the techniques from James Whittaker's book "Exploratory Software Testing," here are some suggested "tours" and hints for conducting exploratory testing on the boundary test example:

Boundary Focus Tour:
Start by focusing on the boundary values themselves and explore how the system handles inputs at the exact boundaries.
Test the behavior of the text box when the values of x and y are set precisely to the minimum and maximum boundaries.
Observe how the system responds and validate if it behaves as expected in terms of acceptance or rejection of the inputs.

Equivalence Tour:
Partition the valid inputs into equivalence classes based on their characteristics or expected behavior.
Select representative inputs from each equivalence class to test the system's response.
For example, choose inputs near the boundaries, as well as some typical values within the valid range, and observe how the system handles them.

Error Handling Tour:
Focus on error conditions and inputs that fall outside the valid boundaries.
Test how the system handles invalid inputs, such as values below the minimum, above the maximum, or non-numeric inputs.
Explore any error messages or notifications provided by the system and ensure they are appropriate and informative.

Stress Testing Tour:
Push the boundaries of the system by subjecting it to stress conditions.
Test the behavior when large numbers or long strings are entered in the text box.
Observe how the system handles such inputs in terms of performance, response time, and error handling.

Combination Tour:
Explore combinations of boundary values and other factors that may influence the system's behavior.
Test different combinations of x and y values, taking into account other variables or conditions that might impact the outcome.
For example, test how the system handles the interaction between x and y when they approach the boundaries together or when one is at its boundary while the other is not.

Hints for Exploratory Testing:
Document your observations, actions taken, and any anomalies or unexpected behaviors encountered during testing.
Utilize exploratory testing charters to define your testing goals and provide a sense of direction during exploration.
Follow your intuition and curiosity while testing, allowing for serendipitous discoveries and unexpected paths.
Use mind mapping or brainstorming techniques to generate and organize test ideas.
Collaborate with other testers, developers, or stakeholders to gain different perspectives and insights.
Prioritize risks and focus your exploratory testing efforts on areas that have the highest potential impact or uncertainty.
By applying these tours and hints from "Exploratory Software Testing," you can conduct more thorough and systematic exploration of the boundary test example, uncovering potential issues and gaining valuable insights into the system's behavior.

Rating: Poor (hallucination, negligence, opacity, non-responsiveness, vacuousness)

This answer is non-responsive, in that it does not refer to the boundary test example; nor does it seem to refer to Whittaker's book. It simply spits out some general (and vacuous) notions about boundary testing.

- It's not clear (opaque) to what degree ChatGPT's notion of "Boundary Focus Tour" is based on Whittaker's book. It's hard to tell at least in part because Whittaker's book itself goes into very little detail about boundaries. The one example from Whittaker's section on "Boundary Tour" (not "Boundary Focus Tour") that might relate to the current exercise is "filling a text box with its maximum number of characters or null".
- Whittaker's book doesn't mention "Equivalence Tour" or "Error Handling Tour"; these are hallucinations.
- "Stress Testing Tour" is similarly a hallucination with respect to Whittaker's book. There is no "Stress Testing Tour" in the book. The book itself contains the word "stress" only three times, none of which relate to what ChatGPT is saying here.
- "Subjecting [the system] to stress conditions" is pretty vague.
- "Combination Tour" is a hallucination.

An expert tester might also voice doubt about the applicability of touring (which is about surveying an application) to the task of answering the original prompt, which is of far narrower scope than would warrant a tour.