Google Cloud

End to end (E2E) testing

Google Cloud

End to end (E2E) testing

Google Cloud

End to end (E2E) testing

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Focused journey

Focused on the entire customer journey but with simulated inputs.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Focused journey

Focused on the entire customer journey but with simulated inputs.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Focused journey

Focused on the entire customer journey but with simulated inputs.

Full-stack

Encompasses all layers of the technology stack involved in the customer journey.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Focused journey

Focused on the entire customer journey but with simulated inputs.

Full-stack

Encompasses all layers of the technology stack involved in the customer journey.

Three types of E2E testing

Modular

Testing of an individual flow or component that is a subset of the entire customer journey.

Focused journey

Focused on the entire customer journey but with simulated inputs.

Full-stack

Encompasses all layers of the technology stack involved in the customer journey.

Modular testing in Conversational Agents

Definition

Modular testing focuses on individual flows within the agent in Conversational Agents. It tests the functionality and logic of specific flows without considering the entire agent's context.

Scope

- Limited to specific flows within the agent.
- It focuses on the internal logic and responses within a single flow.

Purpose

- To validate the logic and performance of individual flows.
- Ideal for testing specific features or updates in isolation.

Process

Involves testing a flow from its start to its end, including the handling of intents, fulfillment, and transitions within the flow.

Modular testing in Conversational Agents

Definition

Modular testing focuses on individual flows within the agent in Conversational Agents. It tests the functionality and logic of specific flows without considering the entire agent's context.

Scope

- Limited to specific flows within the agent.
- It focuses on the internal logic and responses within a single flow.

Purpose

- To validate the logic and performance of individual flows.
- Ideal for testing specific features or updates in isolation.

Process

Involves testing a flow from its start to its end, including the handling of intents, fulfillment, and transitions within the flow.

Modular testing in Conversational Agents

Definition

Modular testing focuses on individual flows within the agent in Conversational Agents. It tests the functionality and logic of specific flows without considering the entire agent's context.

Scope

- Limited to specific flows within the agent.
- It focuses on the internal logic and responses within a single flow.

Purpose

- To validate the logic and performance of individual flows.
- Ideal for testing specific features or updates in isolation.

Process

Involves testing a flow from its start to its end, including the handling of intents, fulfillment, and transitions within the flow.

Modular testing in Conversational Agents

Definition

Modular testing focuses on individual flows within the agent in Conversational Agents. It tests the functionality and logic of specific flows without considering the entire agent's context.

Scope

- Limited to specific flows within the agent.
- It focuses on the internal logic and responses within a single flow.

Purpose

- To validate the logic and performance of individual flows.
- Ideal for testing specific features or updates in isolation.

Process

Involves testing a flow from its start to its end, including the handling of intents, fulfillment, and transitions within the flow.

Modular testing in Conversational Agents

Advantages

- More focused and faster.
- Easier to isolate and fix issues within a specific flow.

Considerations

- Does not assess how different flows interact.
- May miss issues related to context management across flows.

Modular testing in Conversational Agents

Advantages

- More focused and faster.
- Easier to isolate and fix issues within a specific flow.

Considerations

- Does not assess how different flows interact.
- May miss issues related to context management across flows.

Modular testing in Conversational Agents

Advantages

- More focused and faster.
- Easier to isolate and fix issues within a specific flow.

Considerations

- Does not assess how different flows interact.
- May miss issues related to context management across flows.

Focused journey testing in Conversational Agents

Definition

Focused Journey tests simulate specific user paths through a system using controlled inputs and responses, isolating features for targeted testing.

Scope

These tests center on a single feature, critical process, or limited set of interactions within a user journey.

Purpose

- Provide early feedback on specific journeys, even with incomplete systems.
- Isolate problems within workflows for targeted debugging.

Process

1. Select a user journey.
2. Develop a test cases.
3. Execute tests with controlled inputs.
4. Analyze results and report findings for improvement.

Focused journey testing in Conversational Agents

Definition

Focused Journey tests simulate specific user paths through a system using controlled inputs and responses, isolating features for targeted testing.

Scope

These tests center on a single feature, critical process, or limited set of interactions within a user journey.

Purpose

- Provide early feedback on specific journeys, even with incomplete systems.
- Isolate problems within workflows for targeted debugging.

Process

1. Select a user journey.
2. Develop a test cases.
3. Execute tests with controlled inputs.
4. Analyze results and report findings for improvement.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Focused journey testing in Conversational Agents

Advantages

- Targeted feedback for critical features.
- Isolation from other system issues.
- Wider scenario coverage due to ease of setup
- Faster to execute.

Considerations

- Mock accuracy: The test's validity depends on how well your simulations replicate real-world dependencies and responses.
- Maintenance: Simulators and mocks need to be updated as the system or its dependencies evolve.

Full Stack testing in Conversational Agents

Definition

Full stack testing is the most comprehensive stage of end to end testing.
Every aspect of the agent design from intent recognition to systems integration testing is covered in this stage.

Scope

Encompasses all layers of the technology stack involved in the customer journey including integration with backend fulfillment systems.

Purpose

The final stage of testing by the agent development team to ensure the agent is ready for User Acceptance Testing by the customer and ultimately production launch.

Process

- Combine all of the test cases for all CUJ's handled by the agent.
- Add testing to ensure the function of all integrated systems such as the telephony and chat UI.

Full Stack testing in Conversational Agents

Definition

Full stack testing is the most comprehensive stage of end to end testing.
Every aspect of the agent design from intent recognition to systems integration testing is covered in this stage.

Scope

Encompasses all layers of the technology stack involved in the customer journey including integration with backend fulfillment systems.

Purpose

The final stage of testing by the agent development team to ensure the agent is ready for User Acceptance Testing by the customer and ultimately production launch.

Process

- Combine all of the test cases for all CUJ's handled by the agent.
- Add testing to ensure the function of all integrated systems such as the telephony and chat UI.

Full Stack testing in Conversational Agents

Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Full Stack testing in Conversational Agents

Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Full Stack testing in Conversational Agents

Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Full Stack testing in Conversational Agents

Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Full Stack testing in Conversational Agents

Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Full Stack testing in Conversational Agents

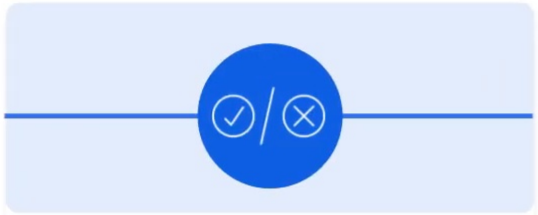
Advantages

- Improved user experience.
- Enhanced overall software quality.
- Ensures agent is ready for user acceptance testing and production launch upon customer sign off.

Considerations

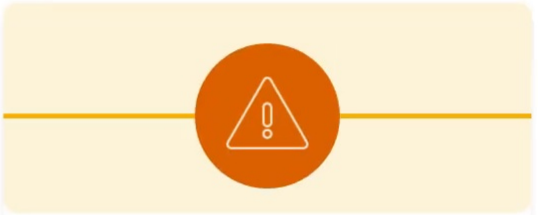
- Requires greater skillset diversity among testers.
- Can be more time-consuming than isolated testing.
- Needs careful test planning and coordination.

Understanding fallback scenarios



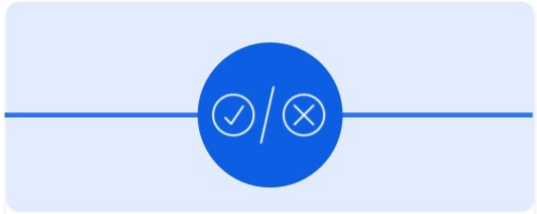
No-match

No-input

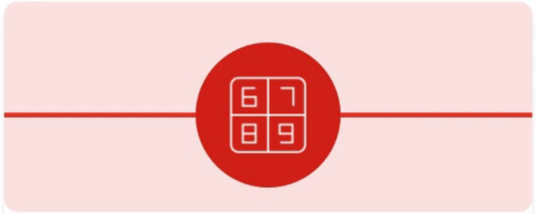


Webhook failures

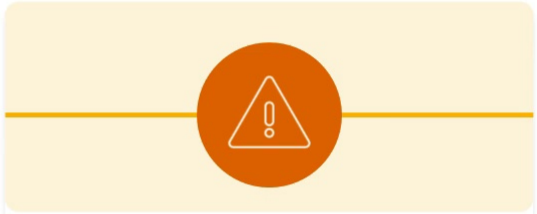
Understanding fallback scenarios



No-match

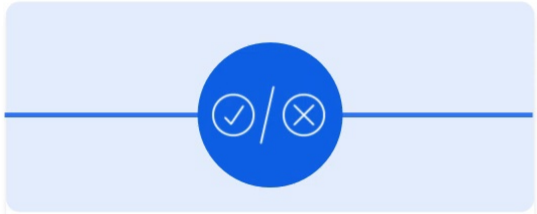


No-input

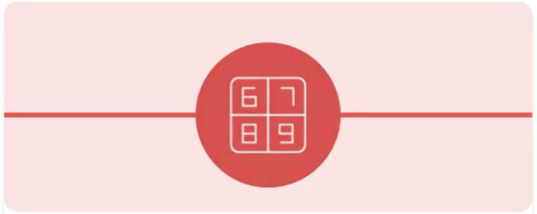


Webhook failures

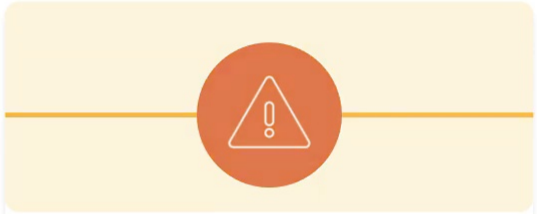
Understanding fallback scenarios



No-match

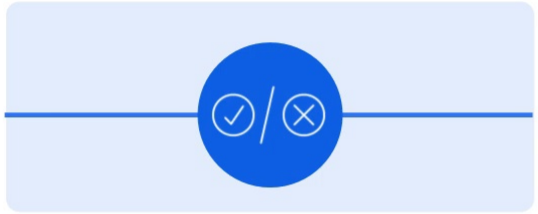


No-input



Webhook failures

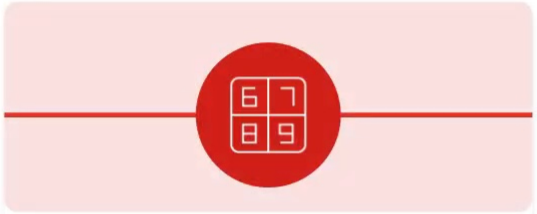
Understanding fallback scenarios



No-match

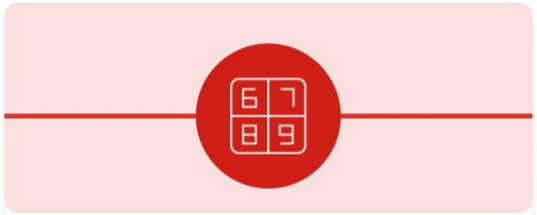
No-match scenarios occur when a user's input does not match any of the defined intents in your agent.

Understanding fallback scenarios



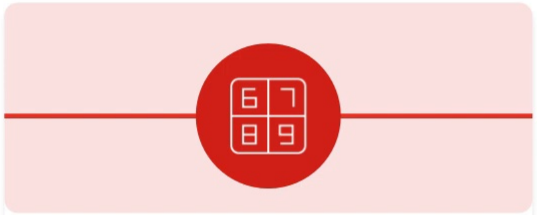
No-inputs happen when there is no user input, perhaps due to user distraction or confusion.

Understanding fallback scenarios



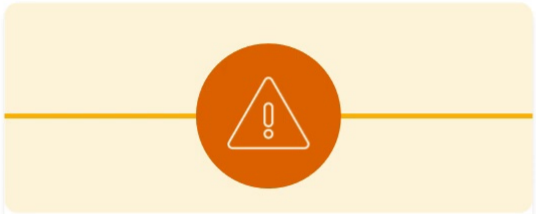
No-inputs happen when there is no user input, perhaps due to user distraction or confusion.

Understanding fallback scenarios



No-inputs happen when there is no user input, perhaps due to user distraction or confusion.

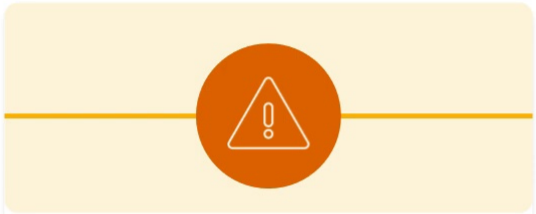
Understanding fallback scenarios



Webhook failures

Webhook failures occur when a call to an external webhook results in an error or timeout.

Understanding fallback scenarios



Webhook failures

Webhook failures occur when a call to an external webhook results in an error or timeout.

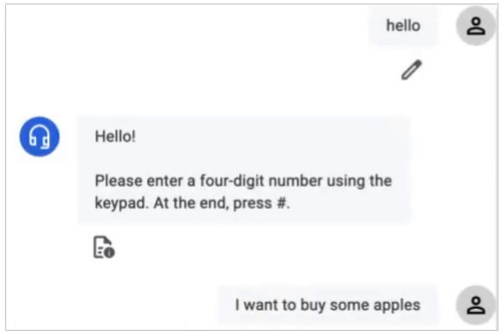
Fallback scenarios are key to creating a resilient and user-friendly conversational agent.

Fallback scenarios are key to creating a resilient and user-friendly conversational agent.

Testing no-match scenarios

01

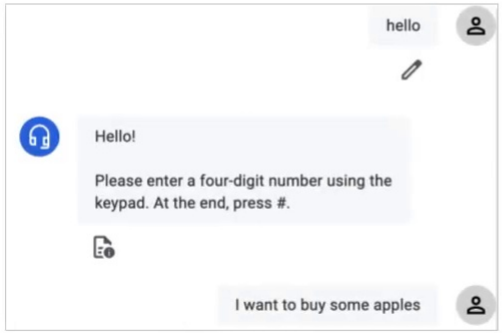
Creating scenarios where user inputs are intentionally out of scope.



Testing no-match scenarios

01

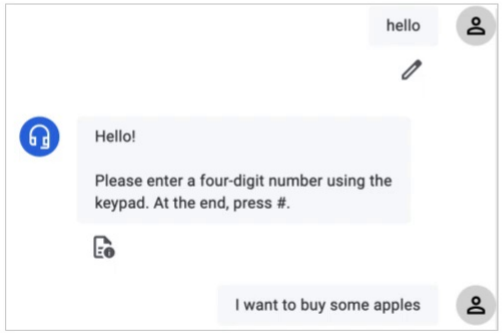
Creating scenarios where user inputs are intentionally out of scope.



Testing no-match scenarios

01

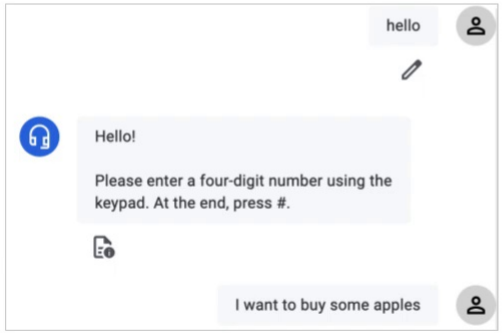
Creating scenarios where user inputs are intentionally out of scope.



Testing no-match scenarios

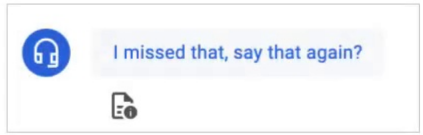
01

Creating scenarios where user inputs are intentionally out of scope.



02

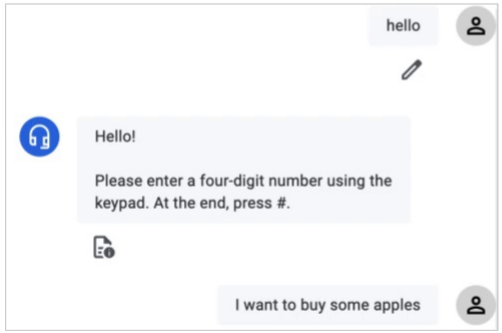
Defining expected agent responses or prompts for these inputs.



Testing no-match scenarios

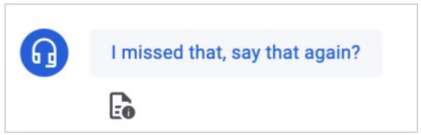
01

Creating scenarios where user inputs are intentionally out of scope.



02

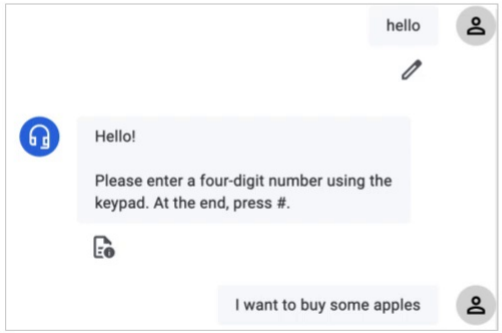
Defining expected agent responses or prompts for these inputs.



Testing no-match scenarios

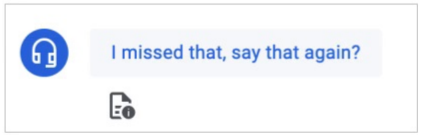
01

Creating scenarios where user inputs are intentionally out of scope.



02

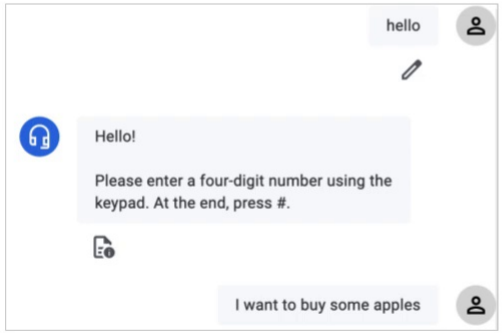
Defining expected agent responses or prompts for these inputs.



Testing no-match scenarios

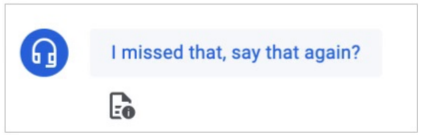
01

Creating scenarios where user inputs are intentionally out of scope.



02

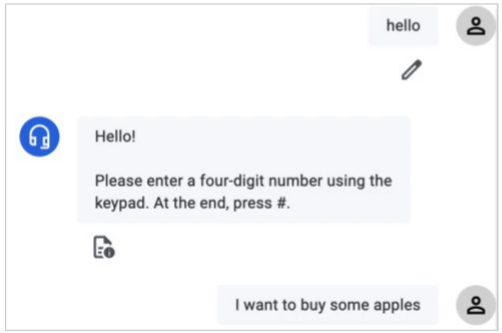
Defining expected agent responses or prompts for these inputs.



Testing no-match scenarios

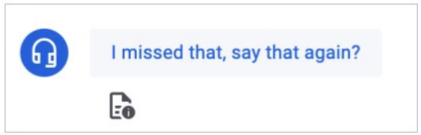
01

Creating scenarios where user inputs are intentionally out of scope.

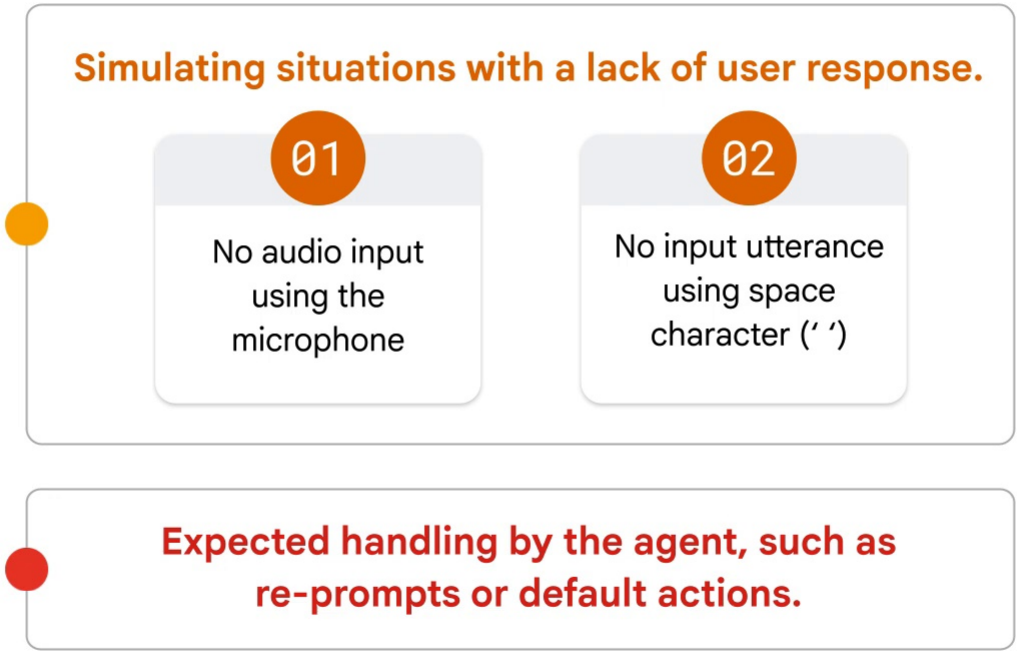


02

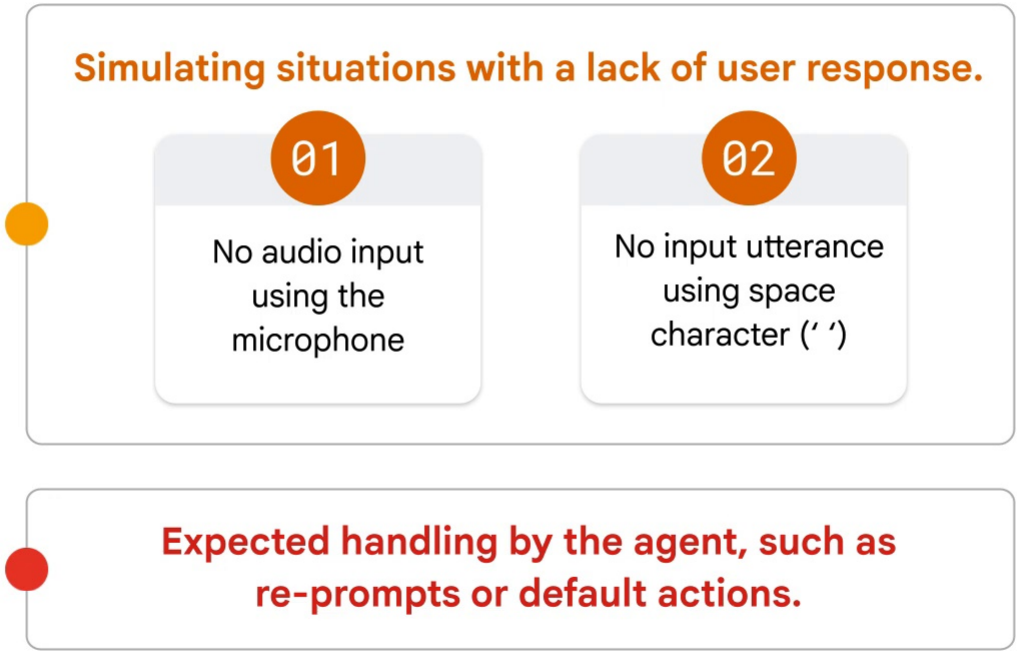
Defining expected agent responses or prompts for these inputs.



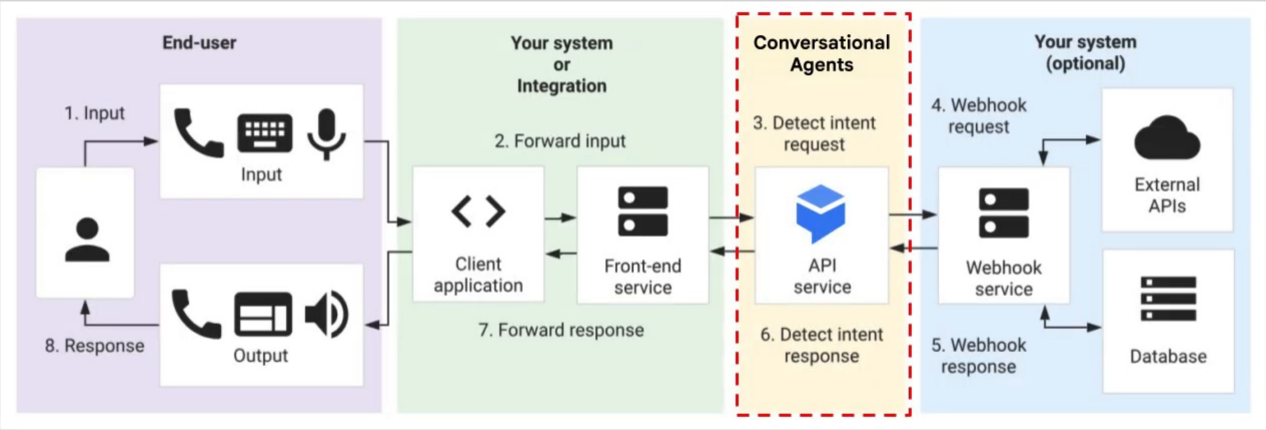
Testing no-input scenarios



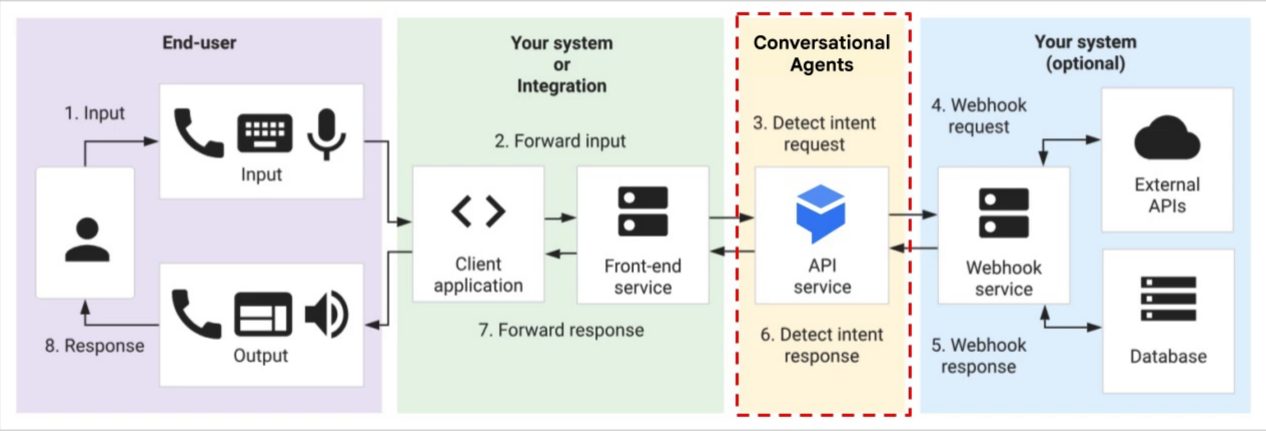
Testing no-input scenarios



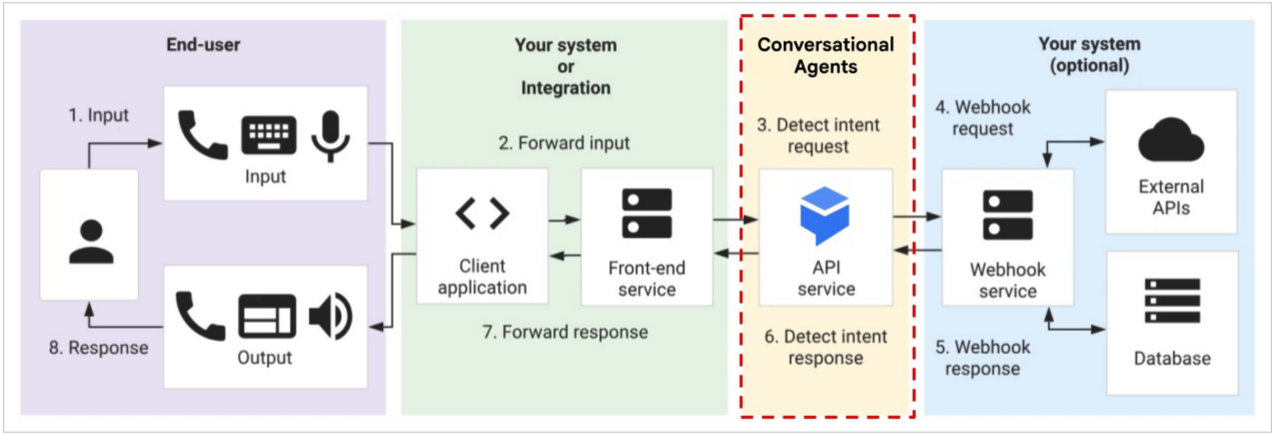
Testing webhook failure scenarios



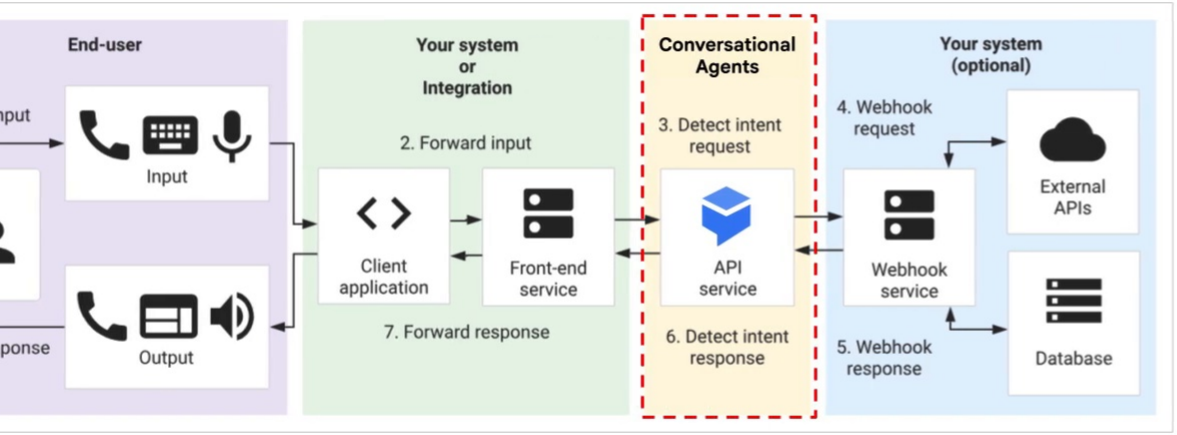
Testing webhook failure scenarios



Testing webhook failure scenarios



hook failure scenarios



E2E Testing

When intro
existing ro
the flow to
correctly o

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

NLU modifications

For NLU updates (intents, etc.), craft 10-15 test cases with variations in how the updated intent is triggered.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

NLU modifications

For NLU updates (intents, etc.), craft 10-15 test cases with variations in how the updated intent is triggered.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

NLU modifications

For NLU updates (intents, etc.), craft 10-15 test cases with variations in how the updated intent is triggered.

E2E Testing: Best practices

New page updates

When introducing a new page, test all existing routes and transitions within the flow to ensure they still function correctly on the new page.

Error handling

Always include test cases that specifically cover "No-match" and "No-input" scenarios to verify how the system handles unexpected situations.

NLU modifications

For NLU updates (intents, etc.), craft 10-15 test cases with variations in how the updated intent is triggered.

Multiple access paths

If a new page has several access points, create test cases covering each path to ensure all routes to the page work flawlessly.


Google Cloud