
Virtual Agents Testing Solution

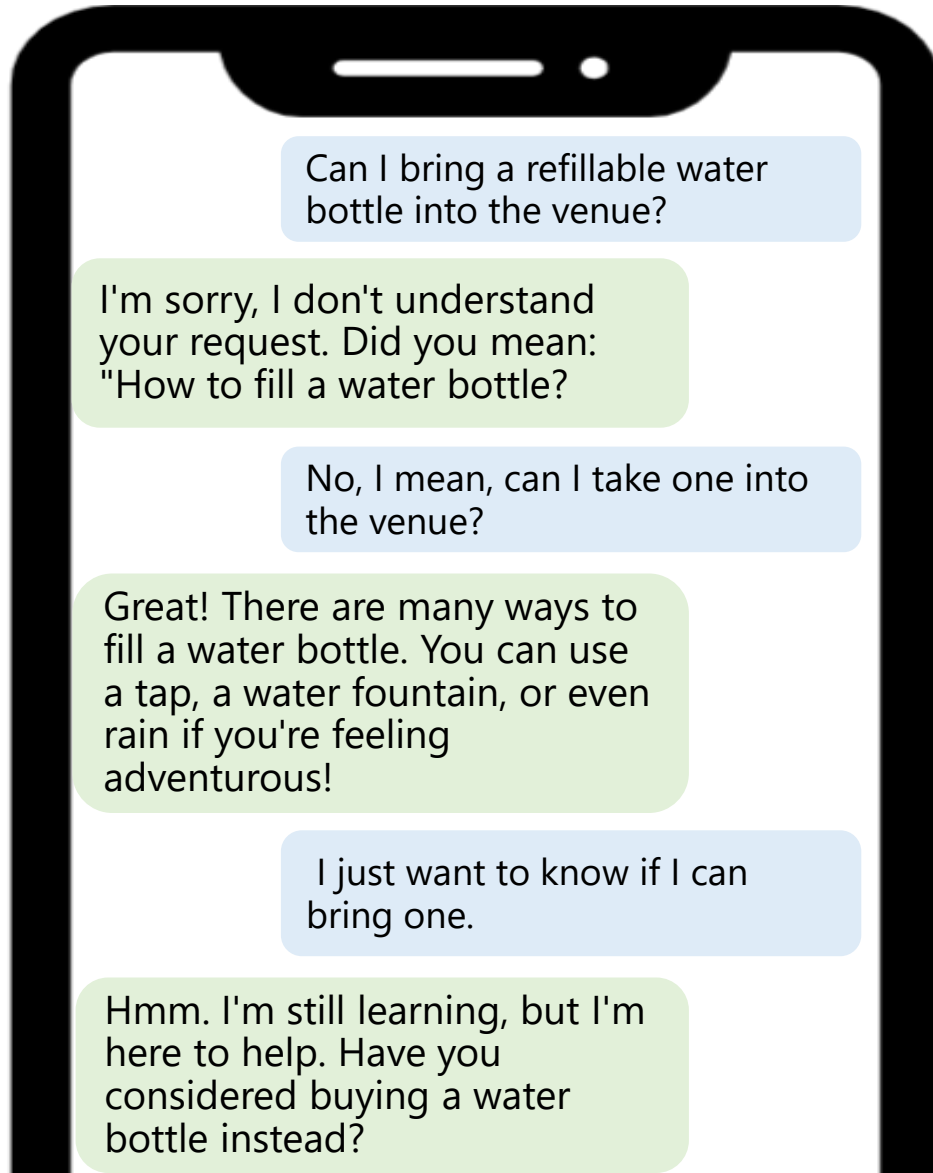
by Rahul Srivastava

(Shared with the client's full permission, this brief summary highlights a small part of my work, including consulting on the testing framework and planning development.)

Content

1. Problem Analysis
2. Proposed Testing Framework
3. Exploring and Prioritizing Solutions
4. Feature Prioritization and Delivery Plan

Chatbot Fails



- Chatbot fails result in confusion and frustration
- This may happen due to hallucination, not understanding user's intent, repetition etc.
- Metrics like drop off rate, failure rate etc. can indicate problems with the model responses
- It is important to identify most of these issues during testing phase rather than after deployment

Problem Statement

Problem Statement

“Our current testing approach for virtual agents (VAs) reaches its limits when VA outputs are generated using Large Language Models (LLMs)”

Further Information

- Current testing framework is designed for rule-based VA outputs
- Primary issue is non-deterministic responses generated by LLMs
- Testing limit is not reached due to data volume or computational constraints

Problem Space: Identifying the root cause

Problem Statement

"Our current testing approach for virtual agents (VAs) reaches its limits when VA outputs are generated using Large Language Models (LLMs)"



As our testing framework is optimized for evaluating fixed, predictable answers but not suitable for variable, context sensitive responses

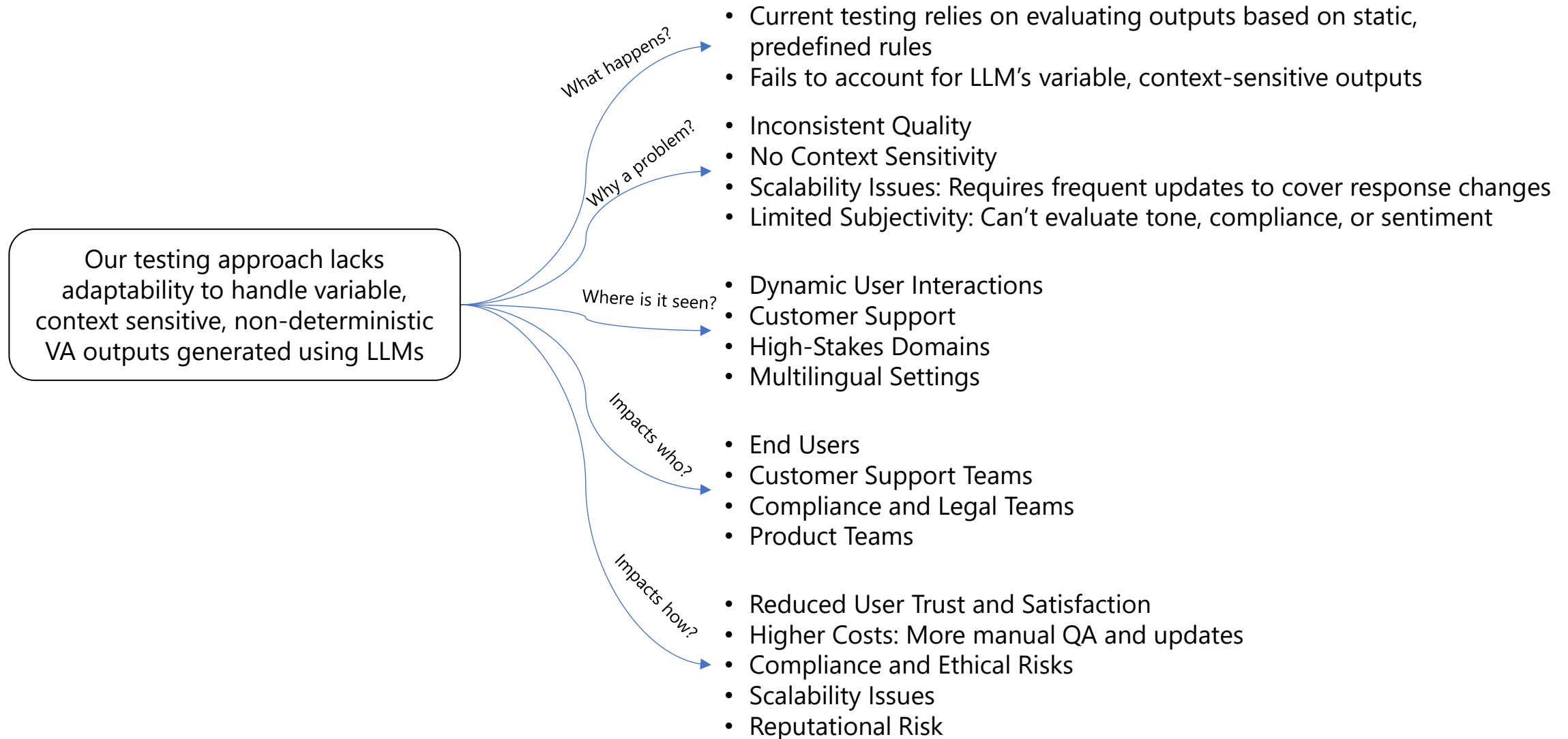


As Testing for VA outputs generated using LLMs requires adaptability to handle non-deterministic outputs.

Root Cause Problem

"Our testing approach lacks adaptability to handle variable, context sensitive, non-deterministic VA outputs generated using LLMs"

Exploring the Problem



Solution Proposed : Build AI Agent Testing Framework

Salient Features

Test for Evaluation Parameters



Test for purpose built, enterprise-specific acceptance criteria

Be Specialized



Purpose-built, leveraging specific data and domain knowledge

Be Scalable



Support high-volume testing for real-world scenarios

Be Adaptive



Allow ongoing assessment and iterative improvement

Be Fine-grained



Measure performance over business-critical data slices

Test for Edge Cases



Evaluate rare or unusual scenarios ("long-tail")


Objectives Met by AI Agent Testing Framework

The framework allows us to run experiments with past and experimental data to identify if the Virtual Agents are able to pass through following criteria:

1. Provides good answers

a. Meets evaluation standards of:

- i. Accuracy
- ii. Compliance
- iii. Contextuality
- iv. Tone and Sentiment
- v. Toxicity
- vi. Fairness



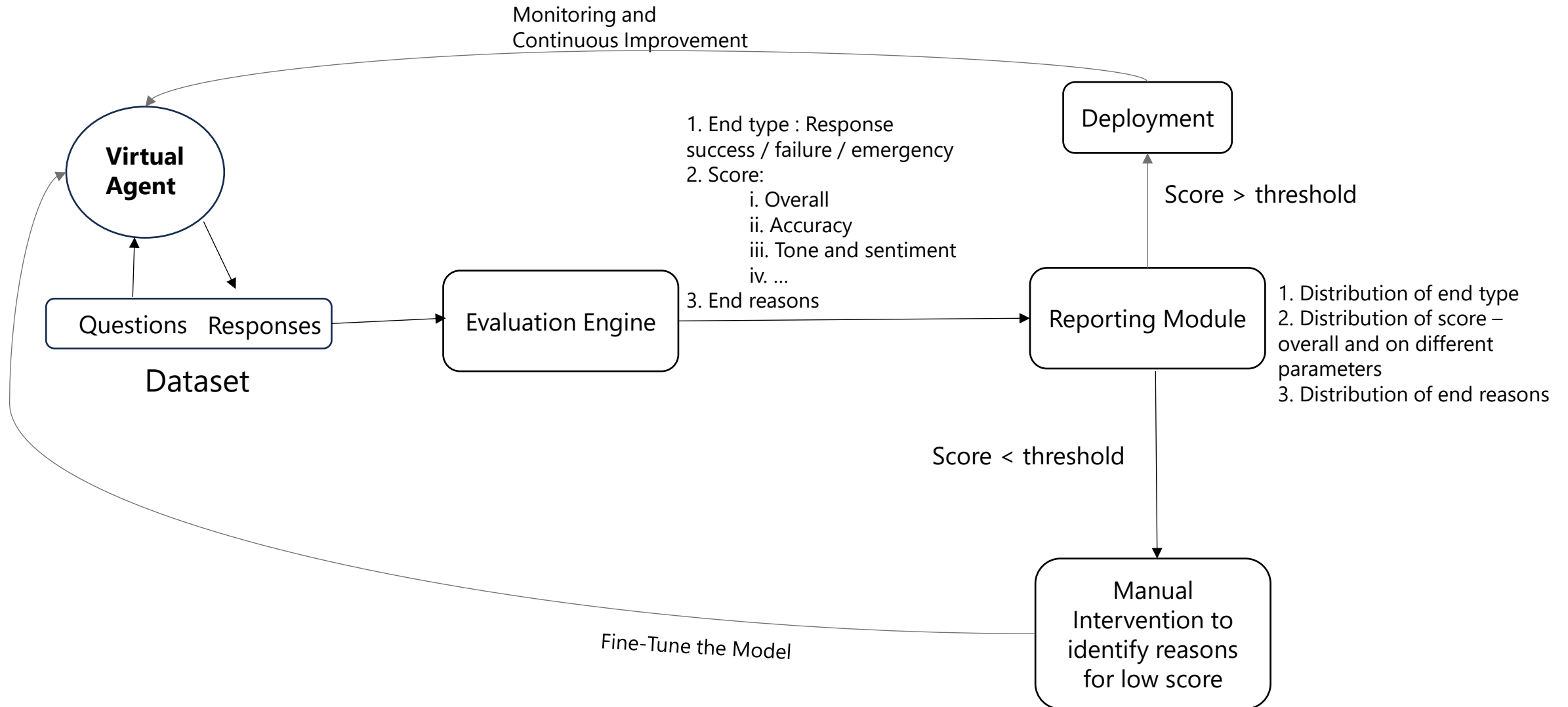
Purpose built for enterprise-specific acceptance criteria

b. Is Specialized

Objectives Met by AI Agent Testing Framework

2. Has correct technical behavior
 - a. Stores the right analytics
 - b. Scores the response correctly
 - c. Captures the exit messages for monitoring
3. Identifies missing information
4. Has good user experience

Virtual Agent Testing Framework



Exploring Solutions for “Evaluation Engine”

Solution	Explanation	Pros	Cons
OSS Benchmark Testing	Use open-source benchmarks to evaluate accuracy and consistency on basic, objective metrics.	<ul style="list-style-type: none">• Cost-effective and straightforward• Establishes a general performance baseline	<ul style="list-style-type: none">• Limited to objective, factual checks• Lacks nuance in context, tone, or compliance
LLM as a Judge	Utilize another LLM to evaluate responses based on subjective criteria like tone, context, and sentiment	<ul style="list-style-type: none">• Adds flexibility for nuanced evaluations• Scalable with custom prompts for specific criteria	<ul style="list-style-type: none">• Prone to inconsistency; may replicate model biases• Can be subjective without strict guidelines
Quality Model with Labeling Functions (LFs)	Build a quality model that uses LFs based on human annotations and LLM feedback to automatically assess responses	<ul style="list-style-type: none">• Provides a scalable, self-learning evaluation model• Aggregates multiple feedback types for consistent scoring	<ul style="list-style-type: none">• Requires ongoing LF tuning and maintenance• Complex to implement and refine over time

Weighted Criteria Matrix

Criteria	Weight	OSS Benchmark Testing	LLM-as-a-Judge	Quality Model with Labeling Functions (LFs)
Multi-Criteria Evaluation Capability	5	1 (5)	3 (15)	5 (25)
Domain Expertise Integration	3	1 (3)	3 (9)	5 (15)
Customization and Flexibility	4	2 (8)	3 (12)	5 (20)
Confidence Scoring	4	0 (0)	4 (16)	5 (20)
Error Analysis	4	2 (8)	3 (12)	5 (20)
Self-Learning	2	0 (0)	2 (4)	5 (10)
Scalability	5	3 (15)	4 (20)	5 (25)
Fine-Grained Insights	2	1 (2)	2 (4)	5 (10)
Implementation Time	5	5 (25)	3 (15)	1 (5)
Technical Feasibility	5	5 (25)	4 (20)	2 (10)
Total Score		91	127	160

Dependency Check

Evaluation Engine	Dependencies
OSS Benchmarking	None
LLM-as-a-Judge	None
Quality Model	LLM-as-a-Judge

Though the Quality Model scored highest in the Weighted Criteria Matrix, its development is dependent on LLM-as-a-Judge.

Therefore, it makes sense to create an MVP using LLM-as-a-Judge as the evaluation engine.

Post MVP development, the needs should be reassessed to decide whether to proceed with the development of Quality Model or not.

Core Features of the Testing Application (High Level)

1. Input Interface
 - a) Accepts chat transcripts of VA outputs for evaluation
 - b) Sends it to the evaluation engine for analysis
2. Evaluation Engine
 - a) Scoring mechanism based on evaluation criteria
 - b) Classification mechanism by end type (success, failure, emergency exit)
 - c) Classification mechanism by end reasons
3. Basic Reporting Module
 - a) Aggregate distribution of end types
 - b) Aggregate distribution and average for evaluation criteria
 - c) Aggregate distribution of end reasons

Features List of the Testing Application with Prioritization

1. Test Data Creation	
1.1 Gather chat transcripts	Must have
1.2 Pre-process data to remove noise - manual	Must have
1.2.1 Data cleaning	
1.3 Pre-process data to remove noise - automation	Good to have
1.3.1 Data cleaning	
1.3.2 Standardization	
1.3.3 Tokenization	
1.3.4 Noise removal	
1.3.5 Anonymization	
2. Input Interface	
2.1 Bulk Upload Functionality	Must have
2.2 API Integration for Input with random data slicing	Good to have
2.3 API Integration for Input with specific data slicing	Good to have
3. Evaluation Engine	
3.1 Model selection, training and fine tuning for LLM as a judge	Must Have
3.2 Scoring Mechanism	Must Have
3.2.1 Evaluation Criteria Framework	
3.2.2 Scoring Implementation	
3.2.3 Threshold Management	
3.3 Classification Mechanism by End Type	Must Have
3.3.1 End-Type Classification Logic	
3.3.2 Customizable End-Type Definitions	
3.4 Identifying End Reasons	Good to Have
3.4.1 Pattern identification for identification of end reasons	
3.4.1.1 Sentiment analysis	Priority 1
3.4.1.2 Statement analysis	Priority 1
3.4.1.3 Word analysis	Priority 2
3.4.1.4 Engagement analysis	Priority 2

4. Reporting Module	
4.1 Set up Dashboard for Key metrics	Must have
4.2 End-Type Reporting	Must have
4.3 Evaluation Criteria Reporting	Must have
4.4 Manual tagging to highlight reasons for low score	Must have
4.4 Exportable Reports	Good to have
4.5 Notifications to highlight high attention issues	Good to have
4.6 Real time monitoring interface	Good to have

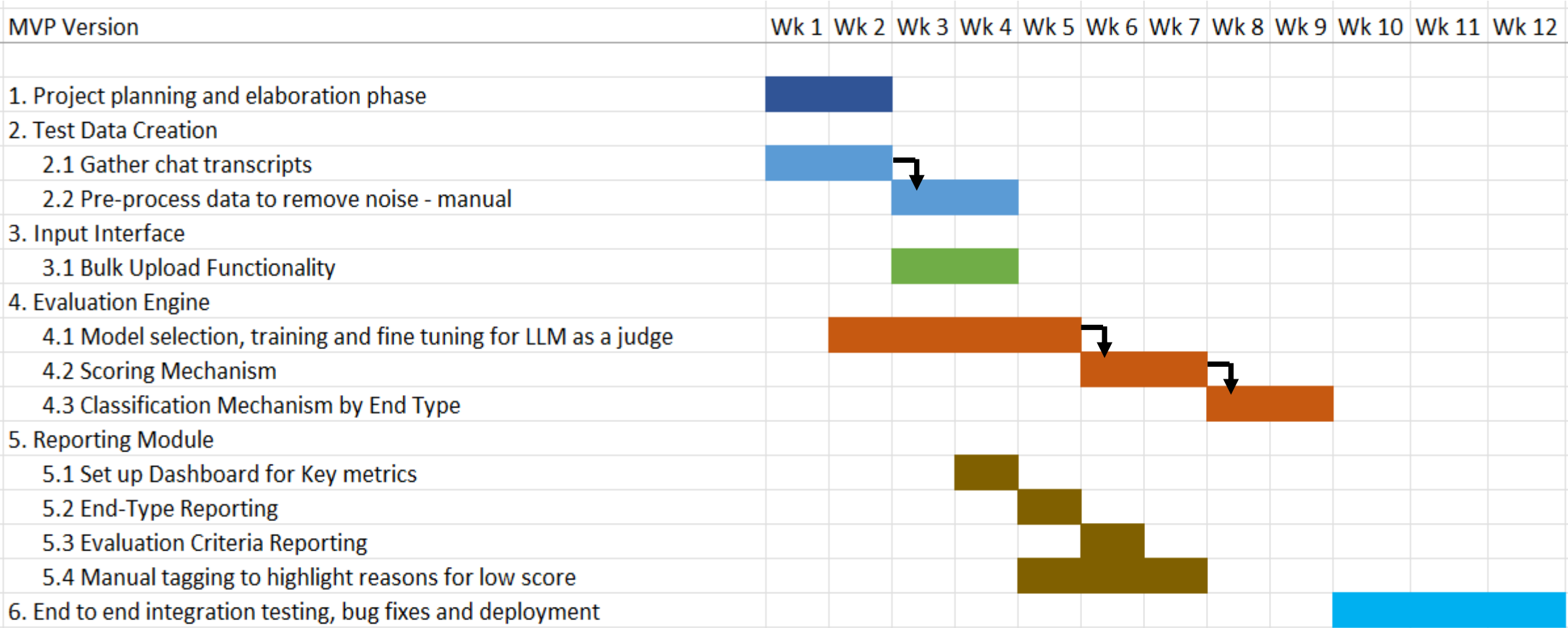
Timeline for MVP and Roadmap

MVP Version	
1. Project planning and elaboration phase	Week 1-2
2. Test Data Creation	
2.1 Gather chat transcripts	Week 1-2
2.2 Pre-process data to remove noise - manual	Week 3-4
3. Input Interface	
3.1 Bulk Upload Functionality	Week 3-4
4. Evaluation Engine	
4.1 Model selection, training and fine tuning for LLM as a judge	Week 2-5
4.2 Scoring Mechanism	Week 6-7
4.3 Classification Mechanism by End Type	Week 8-9
5. Reporting Module	
5.1 Set up Dashboard for Key metrics	Week 4
5.2 End-Type Reporting	Week 5
5.3 Evaluation Criteria Reporting	Week 6
5.4 Manual tagging to highlight reasons for low score	Week 5-7
6. End to end integration testing, bug fixes and deployment	Week 10-12

5. Future Enhancements	
5.1 Evaluation engine enhancement	
5.1.1 Pattern identification for identification of end reasons	
5.1.1.1 Sentiment analysis	
5.1.1.2 Statement analysis	
5.1.1.3 Word analysis	
5.1.1.4 Engagement analysis	
5.2 Pre-process data to remove noise - automation	
5.2.1 Data cleaning	
5.2.2 Standardization	
5.2.3 Tokenization	
5.2.4 Noise removal	
5.2.5 Anonymization	
5.3 Input interface enhancement	
5.3.1 API Integration for Input with random data slicing	
5.3.2 API Integration for Input with specific data slicing	
5.4. Reporting Module enhancement	
5.4.1 Exportable Reports	
5.4.2 Notifications to highlight high attention issues	
5.4.3 Real time monitoring interface	

Roadmap Item
Quarter 1 - MVP
Test data creation
Input interface
Evaluation Engine
Reporting Module
Quarter 2
Evaluation engine enhancement
Pre-process data to remove noise - automation
Input interface enhancement
Quarter 3
Reporting Module enhancement

Delivery Plan



Sample User Story and Acceptance Criteria

User Story

US 14: Chat Transcripts Bulk Upload

"As a QA user, I want to upload chat transcripts of VA's output in bulk so that the evaluation engine can analyze the VA's performance efficiently against predefined evaluation criteria."

Acceptance Criteria

1. Verify that the interface has a component where user can drag and drop multiple files.
2. Verify that there is also an upload component that allows for selecting and uploading multiple files using a file selector.
3. Verify that the acceptable file formats are JSON and CSV.
4. Verify that there is real-time progress update during the upload process, showing percentage of completion.
5. Verify that the system provides a toaster message summarizing number of files successfully uploaded and number of files that failed to be uploaded.
6. Verify that the list of files successfully uploaded is displayed below this upload component with an option to delete the already uploaded file.
7. Verify that clicking on delete option for one file will bring up a popup asking for confirmation before deleting the file. The popup will have "Yes" and "No" buttons. Clicking on button "Yes" will successfully delete the file and "No" will not process the delete request and close the popup.

Acceptance Criteria (cont...)

Acceptance Criteria

8. Verify that there is a pagination to structure the list of files so that user sees 30 files per page.
9. Verify that there is a link "Download the list of failed files", that will download an excel file with list of file names that failed uploading, along with reasons of failure like file format not support, or individual files exceeded size limit of 20 MB.
10. Verify that there is a "Send for Evaluation" button that sends uploaded files to the evaluation engine for processing.
11. Verify that there is a "Delete All" button to delete all uploaded files with a single click.
12. Verify that clicking on "Delete All" button will bring up a popup asking for confirmation before deleting all uploaded file. The popup will have "Yes" and "No" buttons. Clicking on button "Yes" will successfully delete all files and "No" will not process the delete request and close the popup.

Thank You!