**Usage Guidelines**

Do not forward this document to any non- Infosys mail ID. Forwarding this document to a non-Infosys mail ID may lead to disciplinary action against you, including termination of employment. Contents of this material cannot be used in any other internal or external document without explicit permission from ETA\_FP\_UI\_TEAM@infosys.com

**Copyright Notice**

© 2022 Infosys Limited, Bangalore, India.

All Rights Reserved. Infosys believe the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.

Education and Research Department

Infosys Limited

Electronics City

Hosur Road

Bangalore - 561 229, India.

Tel:

91 80 852 0261-270

Fax:

91 80 852 0362

[www.infosys.com](http://www.infosys.com)

mailto: ETA\_FP\_UI\_TEAM@infosys.com

**Document Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Reviewer(s)** | **Description** |
| 1.0 | 13/05/2025 | 1.Jagadeesh Mouli Jyothula S V  2.Gandi Hima Varshini  3.Dumpala Devishree  4.Dipti Mondal  5.Subbisetti Teja | ETA\_UI | Initial Draft |

**EPIC**

**Bot Performance Analysis and Improvement** is a big data-based project focused on evaluating and optimizing chatbot performance using technologies such as **UNIX, MongoDB, PySpark, Advanced SQL and Power BI**. This project involves processing and analyzing large volumes of historical bot interaction data to generate valuable insights into how the bot performs and how users interact with it. By leveraging scalable data tools and advanced querying techniques, the objective is to enhance the bot's responsiveness, reliability, and overall user experience.

The core objective of this project is to collect and analyze historical bot interaction data to identify performance trends, usage patterns, and potential areas for improvement. Data will be gathered from various sources and processed using **PySpark**, enabling efficient cleaning and transformation of large datasets. The processed data will be stored and managed in **MongoDB**, allowing for structured querying and analysis using **Advanced SQL** to extract meaningful insights related to bot performance and user behavior.

A key part of the project is the **visualization of these insights using Power BI**. Interactive dashboards and reports will be created to present performance metrics such as response times, conversation success rates, error frequencies, and user engagement levels. These visualizations will provide stakeholders with clear, real-time views of how the bot is functioning and where improvements are needed, supporting more informed and strategic decision-making.

In summary, this project establishes a comprehensive, data-driven framework for continuously monitoring and improving chatbot performance. The expected outcomes include a detailed analysis of user interactions, performance trends, and behavior patterns, as well as intuitive Power BI dashboards that translate data into actionable insights. Ultimately, this initiative aims to optimize the chatbot’s operations and enhance the overall experience for end users.

|  |
| --- |
| **OVERALL REQUIREMENTS** |

Idea Abstract: This project aims to analyze historical performance data of a bot using PySpark Core, SQL, and MongoDB. The analysis will help identify trends, optimize bot responses, and enhance user experience.

**Objectives:**

* Collect and process historical bot interaction data.
* Analyze bot performance and user behavior using SQL.
* Identify trends and changes over time.
* Visualize key metrics and trends.
* Implement predictive analytics for future bot performance and user interactions.

**Technologies Used:**

* PySpark Core: For data processing and transformation.
* SQL: For complex data analysis and querying.
* MongoDB: For data storage and querying.
* Power BI: For data visualization and reporting.
* Python: For data manipulation and machine learning.

**Project Scope:**

* Collect historical bot interaction data from various sources.
* Process and clean data using PySpark.
* Perform data analysis using SQL.
* Store processed data in MongoDB.
* Visualize insights using Power BI.
* Implement machine learning models for predictive analytics.

**Expected Outcomes:**

* Comprehensive analysis of bot performance and user behavior.
* Interactive dashboards and reports for stakeholders.
* Insights to optimize bot responses and enhance user experience.
* Predictive models for future bot performance and user interactions.

**PRODUCT BACKLOG**

**User Story - 1**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data analyst, I want to clean and preprocess raw bot data using PySpark, so that the data is accurate, consistent, and ready for analysis. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Removal of duplicates, nulls, and corrupt records.  2. Standardization of data formats (e.g., timestamps, user IDs).  3. PySpark scripts that can scale to handle large datasets.  **Should Have:**  1. Logging of data quality issues encountered during preprocessing.  2. Reusable and modular PySpark functions for common cleaning tasks.  **Could Have:**  1.Automated detection of outliers or anomalies.  2.Summary report of cleaning results (e.g., percentage of records dropped or corrected). **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do remove duplicates and handle missing or malformed data.  2. Do ensure consistent data formats and field naming across datasets.  3. Do validate the output dataset to confirm it's ready for querying and analysis.  **Don'ts / Negative points:**  1. Don’t apply business logic or perform analytical computations at this stage.  2. Don’t bypass logging of records that fail validation checks. | |

**User Story - 2**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data engineer, I want to store processed data in MongoDB, so that it can be efficiently queried and reused for future analysis. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**   1. Processed data must be stored in a **structured MongoDB collection** format . 2. Collections must be **indexed** on key fields like timestamps, user IDs, or session IDs for optimized performance. 3. Data storage must be **scalable and provide reliable access** to large datasets.   **Should Have:**   1. **Schema validation rules** (application-level or MongoDB native) to ensure data integrity. 2. **Automated scripts** (e.g., Python or PySpark) to insert/update data into MongoDB on a scheduled basis.   **Could Have:**   1. Automated scripts to archive or move outdated data based on retention policies. 2. Use of data partitioning to improve query performance and manage large datasets.   **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**   1. Do store only cleaned and validated data in MongoDB. 2. Do create indexes to support efficient querying and performance. 3. Do ensure MongoDB collections are designed to support analytical use cases.   **Don'ts / Negative points:**   1. Don’t store raw, malformed, or duplicate data in MongoDB. 2. Don’t hardcode connection strings or collection names—use dynamic configuration. | |

**User Story - 3**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data analyst, I want to write complex queries and aggregations on MongoDB, so that I can analyze user behavior and bot performance. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**   1. Ability to write complex queries using MongoDB’s aggregation (e.g., $match, $group, $lookup). 2. Access to indexed collections for high-performance querying. 3. Support for filtering and analyzing data based on key fields like timestamp, user type, interaction outcome.   **Should Have:**   1. Optimization of queries using MongoDB explains plans and indexing strategies. 2. Reusable and documented MongoDB Compass queries for common analysis patterns. 3. Integration with tool like Power BI dashboard for querying results.   **Could Have:**   1. Scheduled aggregation jobs for frequently used queries. 2. Query performance monitoring and benchmarking.   **Won’t Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**   1. Do ensure queries are optimized and performant, especially for large datasets. 2. Do use aggregation functions to group, filter, and join data for trend and behavior analysis. 3. Do documents commonly used queries for reusability and consistency.   **Don'ts / Negative points:**   1. Don’t run unindexed or inefficient queries that degrade performance. 2. Don’t write queries that modify or delete data unintentionally. | |

**User Story – 4**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a product owner, I want to define key performance metrics such as response time, success rate, and user drop-off points, so that we can objectively evaluate the bot's performance. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Clearly defined performance metrics including response time, success rate, and user drop-off points.  2. Agreement among stakeholders on how each metric is calculated and interpreted.  3. Metrics aligned with business goals and measurable using available data.  **Should Have:**  1. Documentation of each metric’s definition, data source, and relevance.  2. Metrics broken down by relevant dimensions.  3. Review and validation of metric definitions with the analytics and engineering teams.  **Could Have:**  1. Benchmarks or thresholds set for each metric to track performance against targets.  2. Visual representations of metrics through dashboards for easier interpretation. **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do define metrics that are clear, measurable, and based on existing data fields.  2. Do ensure all stakeholders agree on the metric definitions and how they will be used.  3. Do validate that the metrics can be derived from the available datasets in MongoDB.  **Don'ts / Negative points:**  1. Don’t define metrics that rely on assumptions or unavailable data.  2. Don’t change metric definitions frequently without stakeholder alignment.  3. Don’t include overly complex or ambiguous metrics that cannot be measured consistently. | |

**User Story – 5**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data analyst, I want to identify trends and changes in bot performance over time, so that we can understand what factors affect user satisfaction. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Ability to analyze historical data to identify trends in bot performance over time (e.g., response time, success rates, error rates).  2. Clear methodology to track changes in bot performance based on specific time intervals (daily, weekly, monthly).  3. Access to data that captures key performance indicators (KPIs) such as response time and success rate over a defined period.  **Should Have:**  1. Segmentation of performance data based on factors like user type, time of day, or interaction type.  2. Visualization tools (e.g., Power BI) to present trends over time clearly for stakeholders.  3. Correlation analysis to link performance trends with external factors (e.g., traffic volume, bot updates).  **Could have:**  1. Predictive trends or forecast data for upcoming performance expectations.  2. Detailed drilldowns for specific incidents or outliers in the performance data.  **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do analyze bot performance data over different time frames to spot trends or fluctuations.  2. Do track any significant changes in metrics like response time, user success rates, and drop-off rates over time.  3. Do create visualizations that allow stakeholders to easily identify performance shifts or patterns.  **Don'ts / Negative points:**  1. Don’t ignore smaller datasets or periods with low traffic if they can provide valuable insights.  2. Don’t make assumptions about performance trends without confirming the data with proper analysis. | |

**User Story-6**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data analyst, I want to run SQL queries on historical bot interaction data, so that I can extract meaningful insights and generate custom reports for business analysis. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**   1. SQL queries to fetch key performance metrics (e.g., response time, success rate). 2. Reports should be generated on a scheduled basis. 3. Reports should be exportable in common formats (e.g., PDF, CSV).   **Should Have:**   1. Ability to filter by key dimensions (e.g., date, user type). 2. Alerts for any anomalies in the report data.   **Could Have:**   1. Option to create automated email notifications for stakeholders. 2. Visual indicators for critical performance issues in the reports.   **Won’t Have:** | |
| **Acceptance Criteria:**  **Dos / Positive Points:**   1. Do ensure accurate and timely data is retrieved. 2. Do allow reports to be exported in PDF or CSV formats. 3. Do include filters for key metrics in the reports.   **Don'ts / Negative Points:**   1. Don’t include incomplete or missing data in reports. 2. Don’t generate reports without validating the underlying data first. 3. Don’t skip critical performance metrics in the reports. | |

**User Story-7**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a stakeholder, I want to view interactive dashboards in Power BI, so that I can easily interpret the bot’s performance metrics and trends. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Interactive dashboards in Power BI display key performance metrics such as response time, success rate, and user drop-off points.  2.Clear visualizations (e.g., line charts, bar charts, KPIs) to track trends and performance over time.  3.The ability to filter and drill down into specific data (e.g., by date, user segment, or performance issue).  **Should Have:**  1. Regularly updated data (daily, weekly, or monthly) to reflect the most recent performance.  2. Alerts or visual indicators on the dashboard for significant changes or issues (e.g., high failure rates or response delays).  3. The ability to export or share reports for further analysis or presentation.  **Could have:**  1.Customizable dashboards to allow stakeholders to select the metrics that matter most to them.  2. Integration with other reporting tools or platforms (e.g., email notifications, automated reports).  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1. Do create interactive and easy-to-navigate Power BI dashboards that display the most critical metrics.  2. Do ensure the dashboards are regularly updated with the most recent bot performance data and enable drill down capabilities to explore in detail.  **Don'ts / Negative points:**  1. Don’t present data in an overwhelming or overly complex format—ensure the visualizations are easy to interpret.  2. Don’t leave the dashboards static—ensure they are updated regularly and reflect current data.  3. Don’t allow users to alter the underlying data directly through the dashboard (keep data integrity intact). | |

**User Story-8**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data engineer, I want to visualize key performance metrics and trends from historical bot interaction data using python libraries so that I can uncover patterns, correlations, and anomalies that support data-driven improvements to the bot's performance. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1.Mechanism to track core metrics like **success rate**, **response time**, and **accuracy** over time.  2**.**Clean and preprocessed historical data ready for analysis.  3.Ability to **segment data** by key dimensions (e.g., region, user type).  **Should Have:**  1. Insights into **correlations between user behavior and bot performance**.  2.Integration with **python libraries** to detect anomalies or patterns.  **Could have:**  1. **Forecasting models** for future performance trends.  2.Interactive filters to **drill down by user segments or intents**.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1. **Clean and validate** historical data before applying any analysis or ML.  2.Use **timestamped data** to observe trends and patterns over time.  3. Break down metrics by **dimensions like intent, region, user type**, etc., for actionable insights.  **Don'ts / Negative points:**  1. Don’t rely solely on **overall averages** — they can hide critical variances across segments.  2. Don’t apply complex ML models without **domain validation or interpretation**.  3. Don’t mix **raw and processed data** in visualizations — maintain consistency. | |

**User Story-9**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a QA analyst, I want to analyze frequent bot failure cases and error responses, so that we can address the root causes and improve accuracy. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Access to error logs or failure response data that clearly indicate when and why the bot failed.  2. Ability to categorize errors (e.g., unrecognized intent, API failure, timeout) for detailed analysis.  3. Identification of high-frequency or recurring failure patterns for prioritization.  **Should Have:**  1. Visualization of failure trends over time (e.g., spikes in errors, time-of-day patterns).  2. Correlation of errors with specific intents, user actions, or system updates.  3. A report summarizing key failure types and suggested areas for improvement.  **Could have:**  1. Severity tagging of errors to prioritize critical issues.  2. Integration with a bug tracking system for automatic logging of common failures. **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do analyze and categorize bot failures to identify the most common and impactful issues.  2. Do document root causes where identifiable and provide recommendations for resolution.  3. Do track error rates over time to measure improvement after fixes are implemented.  **Don'ts / Negative points:**  1. Don’t rely on anecdotal evidence or assumptions—base conclusions on actual data.  2. Don’t ignore low-frequency errors if they affect critical flow or VIP users.  3. Don’t modify or clean the error data before full analysis is completed. | |

**User Story-10**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a business analyst, I want to track and analyze NPS (Net Promoter Score) feedback in Power BI, so that we can measure overall user satisfaction and loyalty post-bot interaction. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Calculate and display **NPS scores** (0–10 scale) based on user sentiment and feedback.  2.Categorize users into **Promoters**, **Passives**, and **Detractors** within Power BI.  3.Track NPS trends over time and across segments such as:   * Intent type * Region * Device type * Premium vs. non-premium users   **Should Have:**  1.Correlate NPS with key session metrics like **bot response accuracy**, **response time**, and **conversation success**.  2.Provide visual breakdowns by **user segment**, **sentiment trend**, and **feedback themes**.  **Could have:**  1. NPS prediction model (e.g., based on user interaction signals).  2.Integration with **user ID/session data** for follow-up or remarketing opportunities.  3.Export functionality to send raw or aggregated NPS data to stakeholders.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1.Do classify users correctly into **Promoter (9–10)**, **Passive (7–8)**, and **Detractor (0–6)** based on rules or ML scoring.  2.Do ensure **visual clarity** in Power BI charts showing NPS evolution and distribution.  3.Do link NPS feedback with interaction metadata (e.g., time, intent, user type).  **Don'ts / Negative points:**  1.Don’t average NPS scores—instead, calculate as **%Promoters - %Detractors**.  2.Don’t report NPS in isolation—always **contextualize with interaction data**.  3.Don’t hard-code feedback mappings—allow for periodic **retraining or rule updates**. | |

**User Story-11**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a product manager, I want insights into common user intents and behaviors, so that we can improve the bot’s design and flow accordingly. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Access to data on the most frequent user intents and actions during bot interactions.  2. Clear analysis of user behaviors, including common paths, questions, and drop-off points.  3. Identification of areas where the bot’s flow may need improvement or optimization based on user behavior patterns.  **Should Have:**  1. Categorization of user intents based on type, such as informational, transactional, or navigational.  2. Trend analysis to identify changes in user behavior over time or after bot updates.  3. Ability to segment data by user demographics or other relevant factors (e.g., device, location).  **Could have:**  1. Insights into user satisfaction related to specific intents or bot responses.  2. Automated suggestions to optimize bot’s design and flow based on behavioral trends.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1.Do provide insights into the most frequent user intents and behaviors to inform bot design improvements.  2. Do ensure the data includes key interaction points, such as common intents, conversation paths, and user drop-off points.  3. Do track trends over time to understand how user intents evolve and how they can influence bot design.  **Don'ts / Negative points:**  1. Don’t ignore low-frequency user behaviors if they represent critical interactions or edge cases.  2. Don’t make assumptions about user intents without sufficient data to back up the insights.  3. Don’t focus only on quantitative data—ensure qualitative insights (e.g., user feedback) are also considered for improvements. | |

**User Story-12**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a QA analyst, I want to monitor predicted escalation likelihood and actual escalations in Power BI, so that we can proactively identify where the bot is failing and intervene before issues grow. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Visualize **predicted escalation likelihood** per session or interaction in Power BI.  2.Track and display **actual escalations** (e.g., to live agents, unresolved conversations).  3.Correlate escalation data with **user sentiment**, **intent**, and **bot response accuracy**.  **Should Have:**  1.Highlight escalation hotspots by:   * Intent type * User segment (region, device, premium status) * Time of day/week   2.Display escalation trends over time to detect regressions or improvements.  **Could have:**  1.Power BI alerting for **threshold breaches** (e.g., when escalation rate exceeds X%).  2.ntegration with internal QA systems or Slack for escalation follow-ups.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1.Do show both **predicted** and **actual** escalations with visual alignment.  2.Do enable filtering and drill-down by **intent**, **user type**, and **feedback score**.  3.Do update Power BI dashboards with near real-time or scheduled data syncs.  **Don'ts / Negative points:**  1.Don’t rely solely on backend escalation tags—**cross-validate with sentiment or session drop-off**.  2.Don’t ignore repeated escalations tied to the same **flow** or **intent**.  3.Don’t overload dashboards with excessive raw data—focus on insight-driven design. | |

**User Story-13**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a product owner, I want to deploy and compare alternative bot flows using A/B testing and PySpark-based success analysis, so that we can select the most effective version for users. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Randomly assign users to **Flow A or B** using User ID modulo or hashing strategy.  2.Measure and compare outcomes using:  **Conversation Success, Response Time, User Sentiment**  3.Apply **PySpark or MLlib** for significance testing (e.g., t-test or chi-square) to ensure changes are statistically valid.  **Should Have:**  1.Automatically deploy the **winning flow** if it shows **>X% improvement** over baseline KPIs.  2.Analyze outcome differences by:  **User Type, Region, Device Type**  **Could have:**  1.Support for **multivariate testing** (e.g., A1, A2, B1) to test combinations of changes.  2.Use **MLlib predictive modeling** to determine which flow a user is most likely to succeed with based on profile or behavior.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1.Do assign flows using a random method**.**  2.Do calculate statistical significance between test groups using PySpark/MLlib.  3.Do store all A/B test metadata, results, and conclusions in a **version-controlled system**.  **Don'ts / Negative points:**  1.Don’t rely on intuition to choose winning flows.  2.Don’t run tests beyond the **defined confidence interval or time window**.  3.Don’t ignore demographic or behavioral splits that might affect performance. | |

**User Story – 14**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a Data Analyst, I want to build a Power BI dashboard using advanced data transformations and DAX measures to segment users by behavior and interaction outcomes, **so that** we can generate actionable insights to improve bot performance and enhance user experience. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Capability to assess bot performance trends (e.g., response latency, conversation success) using historical data.  2.Flexible time-based analysis to monitor improvements or degradations (daily, weekly, monthly).  3.Access to consistent metrics over time for KPIs such as Response Time, Success Rate.  **Should Have:**  1.Breakdown of performance by user characteristics (Premium, Account Age, Device Type).  2.Visual analytics (via Power BI) that clearly convey performance behavior for decision-makers.  3. Ability to link changes in bot metrics with contextual factors  **Could have:**  1.Forecasting elements showing where future issues or improvements are likely.  2.Deep-dives and anomaly detection to isolate specific performance anomalies or spikes.  **Won't Have:** | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1.Build time-aware charts and trend visualizations for KPIs.  2. Use clean data transformations to ensure segmentation is accurate and insightful.  3.Clearly highlight key patterns or deviations in user interaction over defined time periods.  **Don'ts / Negative points:**  1.Avoid ignoring low-traffic time periods — they may carry trend-breaking insights.  2.Don’t rely on static reports or flat metrics — use dynamic filters and time series analysis. | |

**User Story-15**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a UX researcher, I want to integrate user feedback and sentiment scores into adaptive bot responses, so that we can resolve usability issues and improve user satisfaction in real-time. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Collect structured feedback (e.g., User Feedback, User Sentiment) and unstructured free-text feedback.  2.Use MLlib for sentiment classification on free-text feedback to validate or assign sentiment scores.  3. Adapt bot tone and response complexity in real-time based on detected sentiment (e.g., simplify for negative sentiment).  **Should Have:**  1. Store feedback-to-session mappings using fields like User ID and Intent Detected.  2.Cluster negative feedback patterns using PySpark to identify redesign needs quickly.  3.Provide a real-time sentiment dashboard for live monitoring and team insights.  **Could have:**  1.Auto-escalation trigger for sessions with persistently low sentiment (e.g., 2+ consecutive turns).  2.Personalized follow-up feedback prompts after bot improvements are deployed.  **Won't Have:** | |
| **Acceptance Criteria:  Dos / Positive points:**  1.Do track and visualize each UX or logic improvement’s business impact (quantitative + user experience).  2.Do provide both executive-friendly summaries and analyst-level drilldowns.  3. Do clearly highlight changes derived directly from user feedback or sentiment analysis.  **Don'ts / Negative points:**  1. Don’t report metrics without actionable insights or corresponding improvements.  2.Don’t include vanity metrics (e.g., generic click counts) that do not map to user success or KPIs. | |