# Requirements for Project

**User Story-1**

* **Removal of duplicates**: Achieved using dropDuplicates () function.
* Removal of null/missing values – Handled using dropna() or fillna() via the handle\_nulls() function.
* **Corrupted record handling**: Achieved by defining a strict schema and using mode="DROPMALFORMED" while loading the CSV.
* **Standardization of timestamp formats**: Done using to\_timestamp() in standardize\_timestamp().
* **Standardization of user ID format**: Done using type casting in standardize\_column() to StringType.
* **Logging of data quality issues**: Implemented using Python's logging module with checks for null user IDs and malformed timestamps in log\_data\_quality\_issues().
* **Outlier detection with flexible threshold**: Achieved via the updated detect\_outliers() function with configurable Z-score threshold (e.g., threshold=2.5).
* **Reusable and modular functions for cleaning**: All tasks like removing duplicates, null handling, timestamp formatting, and outlier detection are implemented as modular functions.

**User story-2**

* **Store Processed Data**: The data engineer must store processed data in MongoDB as JSON documents.
* **Scalable Storage**: The data engineer must ensure the storage can handle large amounts of data and is reliable.
* **Validate Schema**: The data engineer should use schema validation to ensure data integrity.
* **Automate Data Updates**: The data engineer should set up automated scripts to insert or update data in MongoDB.
* **Remove Outdated Data**: The data engineer could implement a system to archive or remove outdated data based on retention policies.
* **Avoid Raw or Invalid Data**: The data engineer must not store raw, malformed, or duplicate data in MongoDB.

**User story-3**

* **Write Complex Queries**: The analyst must write queries using MongoDB’s aggregation pipeline (e.g., $match, $group, $lookup).
* **Filter and Analyze Data**: The analyst must filter data based on key fields like timestamp, user type, and outcome.
* **Optimize Queries**: The analyst should optimize queries to improve performance.
* **Reuse Queries**: The analyst should create reusable queries for common tasks.
* **Integrate with BI Tools**: The analyst should connect queries with BI tools like Power BI, Tableau, or Python.
* **Scheduled Queries**: The analyst could set up automatic jobs for frequently used queries.
* **Monitor Query Performance**: The analyst could track and improve query performance.
* **Avoid Slow Queries**: The analyst must not run slow or unoptimized queries.
* **Protect Data**: The analyst must not write queries that accidentally modify or delete data**.**

**User story-4**

* **Define Performance Metrics**: The product owner must define clear and measurable performance metrics such as response time, success rate, and user drop-off points.
* **Stakeholder Agreement**: The product owner must ensure all stakeholders agree on how each metric is calculated and interpreted.
* **Align with Business Goals**: The product owner must ensure metrics align with business goals and can be measured using available data.
* **Documentation of Metrics**: The product owner should document each metric’s definition, data source, and relevance for future reference.
* **Breakdown by Dimensions**: The product owner should break down metrics by relevant dimensions like user type, time range, and channel for more granular insights.
* **Review and Validation**: The product owner should collaborate with analytics and engineering teams to review and validate metric definitions.
* **Detect and Handle Outliers**: Identify abnormal response times using Z-score or standard deviation analysis.

**User story-5**

* **Analyze Historical Data**: The data analyst must be able to analyze historical data to identify trends in bot performance, such as response time, success rates, and error rates.
* **Track Changes Over Time**: The data analyst must establish a clear methodology to track changes in bot performance over time, such as daily, weekly, or monthly intervals.
* **Access to Key KPIs**: The data analyst must have access to data capturing key performance indicators like response time and success rate over a defined period.
* **Segmentation of Data**: The data analyst should segment performance data based on user type, time of day, or interaction type to gain more insights.
* **Visualization of Trends**: The data analyst should use tools like Power BI to present trends over time clearly for stakeholders.

**User Story – 6**

* **SQL-Based Insight Extraction**: Use SQL queries to analyze historical bot interaction data, retrieving performance metrics like success rate, response time, and error frequency.
* **Scheduled Report Generation**: Set up automated scripts or tools to generate these reports on a regular schedule (e.g., daily, weekly) for ongoing monitoring**.**
* **Exportable Reports**: Allow reports to be downloaded or exported in user-friendly formats like PDF and CSV for documentation or analysis.
* **Filtering and Dimensional Analysis**: Enable filtering within SQL queries or reports by key parameters such as date ranges, user types, or interaction outcomes.
* **Anomaly Detection Alerts**: Optionally configure logic to flag or highlight unusual patterns or values in the report (e.g., a sudden drop-in success rate).
* **Email Notification System**: Provide an option for the report system to send email alerts with report attachments to stakeholders.
* **Visual Indicators for Issues**: Include visual flags or indicators in the reports (like colored cells or charts) to emphasize critical performance problems.
* **Data Validation for Reporting**: Validate underlying data before generating reports to ensure accuracy and avoid including incomplete or faulty information.
* **Coverage of Key Metrics Only**: Ensure all critical KPIs are included in each report, with no important metric missed due to filtering or query gaps.

**User Story – 7**

* **Interactive Dashboards in Power BI**: Power BI dashboards display key bot performance metrics such as response time, success rate, and drop-off points using interactive elements for easy exploration.
* **Clear Visualizations for Trend Tracking**: Use charts like line graphs, bar charts, and KPI cards to show how key metrics change over time, enabling quick interpretation of trends.
* **Filter and Drill-Down Functionality**: Dashboards allow users to filter data by date ranges, user segments, or specific performance issues, and drill down to detailed levels for in-depth analysis.
* **Scheduled Data Refresh**: Dashboards are configured to update on a regular schedule (daily, weekly, or monthly) to reflect the most current data without manual effort.
* **Visual Alerts and Indicators**: Include visual cues such as color-coded alerts or icons to highlight abnormal changes like increased failure rates or slower response times.
* **Export and Sharing Options**: Users can export dashboard reports to formats like Excel or PDF and securely share them with stakeholders for broader visibility.
* **Read-Only Access and Data Integrity**: Ensure that dashboards are view-only, preventing any direct changes to the underlying dataset while maintaining the accuracy of presented information.

**User Story – 8**

* **Preprocessed Historical Bot Data:** Ensure cleaned and timestamped bot interaction data is ready for accurate trend analysis.
* **Core Metrics Monitoring:** Track essential KPIs such as success rate, response time, and prediction accuracy over time.
* **Segmentation by Key Dimensions:** Allow data filtering by region, user type, or intent to uncover segment-specific insights.
* **Anomaly Detection with ML**: Use machine learning techniques to identify outliers or unexpected behavior patterns.
* **Trend and Pattern Recognition:** Visualize historical data to reveal long-term trends in user engagement and bot performance.
* **Behavioral Correlation Insights:** Analyze correlations between user actions and bot outcomes to inform optimization strategies.

**User Story – 9**

* **Access to Error Data**: The QA analyst must have access to error logs or failure response data that clearly show when and why the bot failed.
* **Categorize Errors**: The QA analyst must be able to categorize errors, such as unrecognized intent, API failure, or timeout, for detailed analysis.
* **Identify Recurring Failures**: The QA analyst must identify high-frequency or recurring failure patterns for prioritization of fixes.
* **Visualize Failure Trends**: The QA analyst should visualize failure trends over time, such as spikes in errors or time-of-day patterns.
* **Bug Tracking Integration**: The QA analyst could integrate error data with a bug tracking system for automatic logging of common failures.

**User Story – 10**

* **Calculate NPS Scores**: The business analyst must be able to calculate and display NPS scores on a 0–10 scale based on user sentiment and feedback.
* **Categorize Users**: The business analyst must be able to categorize users into three groups: Promoters (9–10), Passives (7–8), and Detractors (0–6).
* **Track NPS Trends**: The business analyst must be able to track NPS trends over time and across segments such as intent type, region, device type, and premium vs. non-premium users.
* **Correlate NPS with Key Metrics**: The business analyst should correlate NPS with session metrics like bot response accuracy, response time, and conversation success.
* **Provide Visual Breakdown**: The business analyst should be able to visualize NPS by user segment, sentiment trend, and feedback themes.
* **Prediction Model**: The business analyst could have the ability to implement an NPS prediction model based on user interaction signals.
* **Integration with User Data**: The business analyst could integrate NPS feedback with user ID/session data to support follow-up or remarketing opportunities.
* **Export Functionality**: The business analyst could have the ability to export raw or aggregated NPS data to stakeholders.

**User Story-11**

* **Access to Frequent User Intents**: Provide visibility into the most common intents and user actions during bot interactions by analyzing structured intent data from historical logs.
* **Behavioral Flow Analysis**: Identify and visualize user behaviors, including common conversation paths, frequently asked questions, and typical drop-off points within the bot flow.
* **Bot Flow Optimization Insights**: Use behavior patterns to detect confusing or inefficient parts of the bot's design, suggesting where flow improvements may be needed.
* **Intent Categorization**: Organize user intents into defined categories (e.g., informational, transactional, navigational) to enable structured analysis and prioritization**.**
* **Trend Analysis of Intent Usage:** Track how user behavior and intent frequencies change over time or after bot updates to assess impact and monitor shifting needs.
* **Segmentation by User Attributes**: Allow for segmentation of behavioral data by user demographics, device types, locations, or other relevant dimensions to uncover targeted insights.
* **Satisfaction-Based Insights**: Optionally analyze satisfaction data (e.g., feedback ratings) linked to specific intents to understand where users feel the bot is helping or falling short.

**User Story – 12**

* **Visualize Predicted Escalation Likelihood**: The QA analyst must be able to visualize predicted escalation likelihood per session or interaction in Power BI.
* **Track Actual Escalations**: The QA analyst must track and display actual escalations (e.g., to live agents, unresolved conversations).
* **Correlate Escalation Data**: The QA analyst must correlate escalation data with user sentiment, intent, and bot response accuracy.
* **Highlight Escalation Hotspots**: The QA analyst should highlight escalation hotspots by intent type, user segment (region, device, premium status), and time of day/week.
* **Display Escalation Trends**: The QA analyst should display escalation trends over time to detect regressions or improvements.
* **Power BI Alerts**: The QA analyst could set up Power BI alerts for threshold breaches (e.g., when escalation rate exceeds X%).
* **Integration for Follow-ups**: The QA analyst could integrate with internal QA systems or Slack for escalation follow-ups.

**User Story – 13**

* **Randomly Assign Users to Flows**: The product owner must be able to randomly assign users to Flow A or Flow B using a User ID modulo or hashing strategy.
* **Measure and Compare Outcomes**: The product owner must measure and compare outcomes based on conversation success, response time, and user sentiment.
* **Apply Significance Testing**: The product owner must use PySpark or MLlib for statistical significance testing (e.g., t-test or chi-square) to ensure that changes are statistically valid.
* **Automatically Deploy Winning Flow**: The product owner should automatically deploy the winning flow if it shows >X% improvement over baseline KPIs.
* **Analyze Outcome Differences**: The product owner should analyze outcome differences by user type, region, and device type.
* **Support Multivariate Testing**: The product owner could support multivariate testing (e.g., A1, A2, B1) to test combinations of changes.
* **Use Predictive Modeling**: The product owner could use MLlib predictive modeling to determine which flow a user is most likely to succeed with based on profile or behavior.

**User Story – 15**

* **Collect User Feedback**: The UX researcher must collect structured feedback (e.g., user feedback, sentiment scores) and unstructured free-text feedback.
* **Sentiment Classification with MLlib**: The UX researcher must use MLlib for sentiment classification on free-text feedback to validate or assign sentiment scores.
* **Adapt Bot Responses in Real-Time**: The UX researcher must adapt the bot's tone and response complexity in real-time based on detected sentiment (e.g., simplify responses for negative sentiment).
* **Store Feedback-to-Session Mappings**: The UX researcher should store feedback-to-session mappings using fields like User ID and Intent Detected.
* **Cluster Negative Feedback Patterns**: The UX researcher should cluster negative feedback patterns using PySpark to identify redesign needs quickly.
* **Provide Real-Time Sentiment Dashboard**: The UX researcher should provide a real-time sentiment dashboard for live monitoring and team insights