Control automation

Author: Banele Mdluli

February 2026

Version tracking

Version	Approvers	Approval status	Comments	Date
1.0.0	Banele Mdluli	Pending	compiling documentation	February 2026

1 Summary

Control automation is a project to automate control performance. The automation will cover all processes associated with control performance. This includes frequent data extraction, logic implementation, and exception recording. Beyond control performance, an automation management console will be created to interact with various automation parts.

2 Introduction

Control performance is a crucial BAU (Business As Usual) activity aimed at mitigating risks associated with business processes that could result in financial loss. A control is a mechanism used to mitigate financial risk in a business through manual, semi-automated, or automated procedures. These procedures are performed using various tools tailored for risk mitigation. Control performance is the process of observing, recording and analysing control outcomes. Governance provides guidance on how control performance is done based on audit requirements and senior management expectations.

3 Problem statement

The current control performance method requires human resources to perform. In most cases, an analyst is required to obeserve, record and analyse the control output. Additionally, some control outcomes can require secondary analysis, whereby further analysis is done. There are three phases that make-up control performance: obeserve, record and analyse control outcome. The observe and record phase are repetitive, manually intensive and rarely change. While analysis will differ depending on the recorded outcome.

Control performance dependency on human resources results in a cost in operational expenditure. The OpEx can be reduced using automation for observing and recording control outcomes and using an A.I agent to analyse an outcome.

4 Objectives

The project objectives are targets that must be reached for the project to be considered a success. These objectives are set to reduce the cost associated with control performance and enabling better re-allocation of resources for more critical activities. The objectives are as follows:

- To create a platform that extracts data from a database using a predefined logic (control logic)
- To create a platform that records the information in a control outcome database.
- To create a platform that analysis the data that was recorded in a control outcome database saves it in the control outcome database.

5 Requirements

5.1 Functional requirements

- Control outcome monitoring and recording requirement
  - Extract data from a database using a predefined logic written in SQL
  - Calculate aggregations such as record count, total transaction value, record count per category (where applicable) and dispersion. Store the results in a database.

  - Create visuals based on the output and take a snapshot, save it in the database.

- Control outcome analysis requirement
  - Provide broader analysis (overall) of the recorded control outcome using AI agents.
  - A user should to interact with the AI agent.
  - Log in page

5.2 Non-Function requirement

- Database and data engineering tools
  - Superbase to store raw data
  - Azure data factory to trigger pipelines, based on event, schedule or tumbling window.
  - Databricks to create externally managed tables for control performance.
  - Postgres to store Control outcome results and interaction logs
- A.I agent activities
  - Review record control outcomes
  - Provide users with response within 2 mins.
  - Admitt when info is not available.

5.3 Technical requirement

- A.I agent framework (**To be confirmed**)
- API dev framework
  - Fast API and documentation OpenAPI (Swagger)
- Frontend
  - React
- Environment packaging
  - docker
- Deployement platform, code repo and CI/CD
  - Azure
  - Github
  - Github actions (switch to Azure Dev Ops later)

6 Design (Architecture)

The section focuses on showing how each component of the project interacts with the other. It is divided into two parts, HLD and LLD.

6.1 HLD (High level design)

HLD will show how each unit of component interact with another another unit. The HLD will be divide into following groups Data storage, Cloud services, Communication tool and User interface.

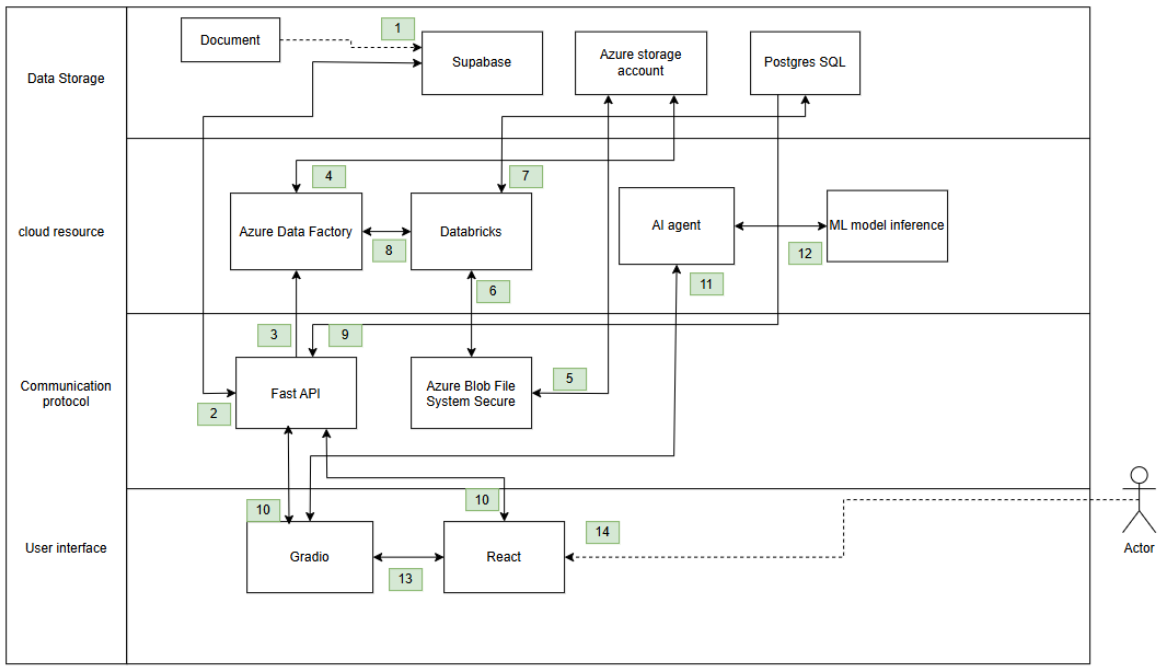

Figure 1: HLD for control automation

Figure 1 shows the various unit components that are required for the project and how they interact. The project design is a multi-platform project. Multi-platform design was selected due to its real-world application resemblance.

6.2 LLD (Low level design)

The low level design shows the structure and make-up of individual units. The LLD is platform divided, units generated or developed in the same platform are part of the same LLD design.
TO BE UPDATED DURING DEVELOPMENT!!!!!!

7 High level technical requirements

This section is a high level explaination of the main tools or services that are required to achieve the project objectives. The methods sections does not contain pseudo code or any technical indicators. The section is intended to be used by individuals who have knowledge of the systems, technologies or concepts used for the project. If any clarification is required, the author should be contacted.

7.1 Fast API

7.1.1 Supabase Data integration

To imitate real world third party tools, Supabase is used as a third party data storage. The Supabase database will contain synthetic data that resembles customer transactions that have anomalies that indicate fraudulent/suspicious transactions.

- Create an AI agent that generates synthetic data. The data generated should contain anomalies that indicate fraudulent/suspicious activities.
  - The synthetic data should customer reference information and transactions

  - The Agent must provide definitions about what is an anomaly or fraud based on the generated data.
  - The Agent must mention the records that considered an anomaly or fraud.
  - **IMPORTANT**: The agent must respond in JSON.
- Create a supabase project and retrieve the database API.
- Create four SQL table in supabase.
  - The first table must contain information about what is anomaly, suspicious or Fraud.
  - The second table must contain the synthetic data and an additional boolean field called 'user_feedback_received' cdefault value false. Ensure that the API can be used to change the feedback field status.
  - The third table must contain records that show anomaly.
  - The fourth table must contain that explaination of why the records were considered an anomaly.
- Create a FAST API that transfers A.I generated data to the supabase table.
- **IMPORTANT**: The schema of the tables will be inherited from the inherited synthetic data.

7.1.2 Azure Data Factory transfer

Azure data factory (ADF) is a tool used to transfer and transform data from one source to another. In this project it is used to call the Fast API linked to the Supabase database. ADF adheres to the medallion architecture, which consists of three layers Bronze, Silve and Gold. Bronze is associated with raw unchanged data, Silver is enhanced or enriched data (e.g., changing date formats and field naming convention) and Gold is a ready to consume layer that has business logic imposed.

- Create a pipeline that copies the information from Supabase to a Azure blob storage that has hierarchical features (datalake).
- Create a pipeline that run databricks notebooks when a new file inserted to the Azure blob storage (Datalake) - **The step is dependent on databricks notebooks, complete the step when the books have been created**.

7.1.3 User interface interaction

The user interface has two integrated interfaces. The first will be a Gradio chatbot interface that sends queries to a AI agent. The AI agent will query all available resources to provide feedback to a user or admit if it does not have the information requested. The second interface is react programmed interface. The interface processes commands from the user based on available functionalities.

- Create an A.I agent that queries various tools (e.g., data sources) and provides feedback in natural language.
- Create an interface for user query the A.I agent.
- Create a frontend that contains a chatbot interface and a page to manage the information on the postgres table (working paper).

7.2 Databricks

Databricks is a apache spark platform used to perform analytics and data science tasks using big data. In this project it used to create notebooks that will run predefined logic written in SQL. Thereafter, store the results in two databases, Postgres SQL table and an account storage container (Set up as an externally managed table).

- Create a databricks container.
- Create a databricks connector and assign it contributor privileges.
- Create three container called bronze, silver and gold.
- Extract the data from the landing storage to the raw container (adhere to ingestion audit best practices)
- Ingest the data into the silver container making neccessary changes in preparation for consumption.
- Run predefined logic and store the results in a silver container table.
- Perform standard aggregation or summary on the predefined logic output information.
- Push the standard aggregation or summary to a gold container and postgres table.

7.3 AI agent and machine learning model

AI agent will query information based on the users request using a chatbot interface. Machine learning inference models are going to be functionalised for the agent to call if needed. The AI design pattern (workflow) used is Routing. Routing is a design pattern where an LLM decides which child LLM or process is executed. There is no aggregation or synthesis used for the final outcome.

- Create an N number of agents and one parent Agent where $N = 1, 2, 3....$
- Functionalise child agents to be used as tools for the Parent agent.
- One of the child agents must call the postgres database using API (dveloped using Fast API).
- Functionalise an API call to the ML model inference.
- Each agent should have guardrails.

7.4 Special considerations

7.4.1 Deployment approach

The project has locally developed features that are packaged using Docker images. These docker images are deployed on docker hub and Azure container registry. Docker hub will be used to store **development** images and Azure container registy stores production images. There are 3 environments in this project, Production, Development and Test environment. Each environment will have its own database instances, their structure is replicated for the other environment. Semantic versioning will be used to track software versioning. Docker ensures the behavoiur and packages used during development remain consistent during production deployment.

7.4.2 Security approach

Local development uses the .env file to store confidential or secretive information such passwords and API keys. The **.env** file is excluded from git tracking and docker hub deployment using gitignore and dockerignore.

7.4.3 Collaborative development package management

To minimize module and library conflicting versions during development a virtual environment is created to ensure consisted version compatibility across different machines.

7.4.4 code consideration

Install packages as modules if they are giving you issues. Use the following notation:

```
python -m pip install <package name>
```

8 Test results

9 Discussion

10 Conclusion

11 Appendix

11.1 Code