BAHRAIN POLYTECHNIC بوليتكنك البحرين

Assessment Cover Sheet

Assessment Title	Industry Project -Thesis		
Assessment Type	Controlled	Individual	Must-pass
Due Date	16-May-25	**Course Code**	IT8299
Course Title	Cooperative Learning Program		
Internal Moderator's Name	Mr. Shimaz Khan		
External Examiner's Name			

Instructions:

1. The **use of generative AI tools is strictly prohibited**.
2. Submissions of this document are to be on the CLP Moodle page.
3. Late submission will be marked to a maximum of 60% or will receive a 0 grade if submitted later than 3 calendar days.
4. The document should be a maximum of 10,000 words (excluding appendices). 5% will be deducted if you go over the word limit.
5. Appendices will not be explicitly marked but may contribute to complete the information for each section.
6. This is a must pass assessment.

Learner ID	202203753	**Date Submitted**	16/5/2026
Learner Name	Ali AlRashedi		
Programme Code	IT8020		
Programme Title	Management Information Systems		
Lecturer's Name			

By submitting this assessment for marking, I affirm that this assessment is my own work.

Learner Signature	Ali AlRashedi

Do not write beyond this line. For assessor use only.

Assessor's Name			
Marking Date		**Maks Obtained**	

Comments:

LO1. Apply specialized skills and detailed knowledge gained in the Bachelor of ICT programme to an approved industry project.

LO3. Critically analyze the ethical, social, legal, and professional issues related to the project and developed ICT solution
45% of the overall mark

Branch Insight 360: A platform using OutSystems, Power BI, and Denodo to generate business insight.

by

Ali AlRashedi

A Thesis Submitted in

(Partial) Fulfillment of the

Requirements for the Degree of

Bachelor of Science

in Information & Communication Technology

at

Bahrain Polytechnic

May 2026

Title

Branch Insight 360: A platform using OutSystems, Power BI, and Denodo to generate business insight.

Copyright

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

……Ali AlRashedi…….. / …Ali AlRashedi…… / …22/2/2026…

Signature name date:

Abstract

Branch Insight 360 is an Artificial Intelligence (AI)-powered decision support solution that provides branch-level banking analysis. This tool allows users to convert natural-language questions into structured insights, dashboards, and reports without compromising governance when using stand-alone web and OutSystems applications.

Deployment uses FastAPI, Denodo Artificial Intelligence Software Development Kit (AI SDK) for data virtualization and query coordination, and an access control structure that includes Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Policy-Based Access Control (PBAC) layers. Data virtualization, query validation, and Power Business Intelligence (Power BI) inspired dashboard creation are incorporated in the model to ensure governed analytical outputs.

Development takes advantage of modular design principles with frontend-backend separation, security hardening of the application programming interface (API), and iterative validation through startup tests and scenario evaluation.

Empirical evidence suggests that the solution consistently provides accurate and ready-for-management answers and summaries while delivering dependable performance in Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) settings.

The results confirm that a governed natural language layer for analytics may facilitate more efficient reporting, faster decision-making, and enterprise data integration with executive reporting demands in branch banking.

Keywords: Artificial Intelligence (AI), FastAPI, Denodo AI SDK, Power BI, OutSystems, Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), Policy-Based Access Control (PBAC), banking analytics.

Acknowledgements

First of all, I would like to thank Al Salam Bank for providing the training and academic environment to help build this project.

Secondly, I would like to acknowledge with gratitude my line manager, Mr. Husain Alsabei, who has been guiding me, and my coworkers, Mr. Ali Al Mahdi, Mr. Sayed Husain, Mr. Yahya Thamer, and Ms. Fatima Hasan, for their support and guidance in my learning experience.

Thirdly, I would like to thank my project supervisor, Dr. Hasan Mansoor, for guiding me and providing suggestions for the project.

Lastly, I would like to thank my family and friends for their continued encouragement and support throughout my project period. Their patience and encouragement played a crucial role in my successful completion of this project.

Table of Contents

List of Figures

List of Tables

List of Symbols

No table of Symbols entries found.

This Project does not contain any Symbols.

List of Abbreviations

Abbreviation	*Full Form*
AI	**A**rtificial **I**ntelligence
SQL	**S**tructured **Q**uery **L**anguage
SDK	**S**oftware **D**evelopment **K**it
VQL	**V**irtual **Q**uery **L**anguage
Power BI	**P**ower **B**usiness **I**ntelligence
MCP	**M**odel **C**ontext **P**rotocol
LLM	**L**arge **L**anguage **M**odel
RBAC	**R**ole-Based **A**ccess **C**ontrol
ABAC	**A**ttribute-Based **A**ccess **C**ontrol
PBAC	**P**olicy-Based **A**ccess **C**ontrol
API	**A**pplication **P**rogramming **I**nterface
LESPI	**L**egal, **E**thical, **S**ocial, and **P**rofessional **I**ssues
ABW	**A**ctivity-Based **W**orking
AML	**A**nti-Money **L**aundering
APAC	**A**sia-Pacific
ASP	**A**pplication **S**ervice **P**rovider
AWS	**A**mazon **W**eb **S**ervices
BI	**B**usiness **I**ntelligence
BPAC	**B**usiness **P**rocess **A**ccess **C**ontrol
BPR	**B**usiness **P**rocess **R**eengineering
CA	**C**ertificate **A**uthority
CASA	**C**urrent **A**ccount **S**avings **A**ccount
CBB	**C**entral **B**ank **O**f **B**ahrain
CORS	**C**ross-**O**rigin **R**esource **S**haring
CRUD	**C**reate, **R**ead, **U**pdate, **D**elete
CSS	**C**ascading **S**tyle **S**heets
DAX	**D**ata **A**nalysis **E**xpressions
DFD	**D**ata **F**low **D**iagram
DSR	**D**esign **S**cience **R**esearch
EDB	**E**conomic **D**evelopment **B**oard **(**Bahrain)
ERD	**E**ntity-Relationship **D**iagram
ETL	**E**xtract, **T**ransform, **L**oad
GCC	**G**ulf **C**ooperation **C**ouncil
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
HTML	**H**ypertext **M**arkup **L**anguage
HTTP	**H**ypertext **T**ransfer **P**rotocol
HTTPS	**H**ypertext **T**ransfer **P**rotocol **S**ecure

ID	**I**dentifier / **I**dentification
IDE	**I**ntegrated **D**evelopment **E**nvironment
IS	**I**nformation **S**ystems
IT	**I**nformation **T**echnology
JDBC	**J**ava **D**atabase **C**onnectivity
JSON	**J**avaScript **O**bject **N**otation
JWT	**J**SON **W**eb **T**oken
KPI	**K**ey **P**erformance **I**ndicator
KYC	**K**now **Y**our **C**ustomer
MENA	**M**iddle **E**ast **A**nd **N**orth **A**frica
MIS	**M**anagement **I**nformation **S**ystems
NIC	**N**ational **I**dentification **C**ard
NIST	**N**ational **I**nstitute **O**f **S**tandards **A**nd **T**echnology
NL	**N**atural **L**anguage
NLIDB	**N**atural **L**anguage **I**nterface **T**o **D**atabase
NLP	**N**atural **L**anguage **P**rocessing
NPS	**N**et **P**romoter **S**core
OIDC	**O**penID **C**onnect
OS	**O**perating **S**ystem
PDPL	**P**ersonal **D**ata **P**rotection **L**aw (Bahrain)
PII	**P**ersonally **I**dentifiable **I**nformation
RDBMS	**R**elational **D**atabase **M**anagement **S**ystem
RE	**R**egular **E**xpression
REST	**R**epresentational **S**tate **T**ransfer
SSL	**S**ecure **S**ockets **L**ayer
TCP	**T**ransmission **C**ontrol **P**rotocol
TLS	**T**ransport **L**ayer **S**ecurity
UI	**U**ser **I**nterface
URL	**U**niform **R**esource **L**ocator
UX	**U**ser **E**xperience
VDP	**V**irtual **D**ataPort **(**Denodo)
VRT	**V**isual **R**egression **T**esting
XML	**E**Xtensible **M**arkup **L**anguage
JS	**J**ava **S**cript
OAuth	**O**pen **A**uthorization

Table 1: List of Abbreviations

1. Introduction

Branch Insight 360 addresses a persistent challenge in banking analytics: management teams require timely and reliable branch-level insights, yet traditional reporting workflows depend on technical query skills, fragmented data sources, and manual dashboard preparation. In many institutions, this leads to delayed decisions, inconsistent reporting across teams, and increased operational risk when performance issues are identified too late. The proposed project seeks to create a governed analytics platform where non-technical managers could ask business questions and receive policy-compliant reports based on the analyzed data.

The rationale for this work is both operational and strategic. Operationally, banking environments require fast and repeatable analysis under strict governance constraints. Strategically, institutions are modernizing through low-code and AI-enabled decision support, yet many solutions still lack a unified architecture that combines natural-language processing, secure data virtualization, and executive reporting. This project addresses that gap by integrating OutSystems, FastAPI, Denodo AI SDK, and dashboard generation services into a single modular and governed workflow designed for reliability, maintainability, and secure analytical processing.

Project Rationale

The proposed work aims to design and demonstrate a secure data architecture for automated detection of branch performance trends using OutSystems and Power BI Model Context Protocol (MCP). In many current processes, report preparation still depends on technical specialists, which introduces delays and reporting risk. Although AI has improved analytical speed, banking platforms continue to face integration challenges when combining AI-driven processing with modern low-code tools such as OutSystems. According to (Denodo, 2024), financial institutions implementing logical data layers must address data siloes and real-time visualization requirements.

The value of reliable data environments has been highlighted in studies on business AI and standardized protocols. There are already cases of Denodo-based systems being used for banking, with the Power BI MCP making it easy to turn AI responses into dashboard specs

that work in different systems. Specifically, the OutSystems platform was used to "modernize banking systems and improve customer experience through rapid application delivery" (OutSystems, 2024). The general use of AI is also examined in a research study, and it verifies the role of artificial intelligence in improving the financial sector, as trends in research are on improving capability, workflow, and efficiency (Mhlanga, 2020).

Project approach

A hybrid, modular, and iterative approach was used to build a secure banking analytics platform that translates natural-language questions into actionable management insights. The modular design allows frontend services, AI processing, dashboard generation, security controls, and backend routing to operate independently while interacting through controlled APIs. Early project stages focused on identifying functional, security, and analytical requirements through project specifications, academic research, and stakeholder discussions. Based on these findings, the system was implemented using OutSystems for the frontend, FastAPI for backend orchestration, Denodo AI SDK for natural-language-to-VQL translation, and dashboard services for report generation. Security and governance mechanisms, specifically role-based access control (RBAC), attribute-based access control (ABAC), and policy-based access control (PBAC), were integrated throughout the workflow.

Project Objectives

The aim is to offer a secure, scalable, and organized data infrastructure so as to allow fast processing of the branch performance queries.

- Architect an OutSystems and FastAPI infrastructure with secure API integration, Denodo AI SDK connections, and Power BI MCP services, validated through successful end-to-end system communication during implementation and testing.
- Apply OutSystems UI components to provide operational visibility for branch managers and senior leadership, demonstrated through functional dashboards and role-based user access.
- Support integration of the LLM workflow with front-end interfaces for branch managers through the Denodo AI SDK, with successful translation of natural language queries into valid analytical outputs.

- Ensure data security and privacy practices aligned with banking industry requirements in a hybrid cloud and low-code environment, validated through secure handling of requests and absence of unauthorized data exposure during testing.
- Implement a three-tier access control model combining RBAC, ABAC, and PBAC to enforce data governance policies, verified through successful execution of access-control test cases across all user roles.

Limitations and Concerns

- The licensing and availability of OutSystems environments may be limited for some developers, which can affect reproducibility in different development setups.
- The current deployment runs entirely on a local machine (127.0.0.1) and requires a self-signed HTTPS certificate on port 8443 to securely bridge communication between the local backend and the cloud-hosted OutSystems application.
- The complexity of Power BI MCP specifications introduces mapping and translation challenges when interpreted by AI-based systems.
- Power BI MCP services were not directly implemented due to cost constraints, with resources instead allocated toward AI processing and integration.
- A trade-off exists between rapid development using OutSystems and the financial limitations associated with AI API usage and scaling.
- Limited hardware resources on the development device may introduce latency and slower response times during high-load processing.
- Real production banking data was not used due to confidentiality constraints; instead, mock datasets generated from AI tools and Kaggle sources were used for development and testing.
- Full demonstration of ABAC and PBAC scenarios may be restricted due to infrastructure and resource limitations.
- The Denodo AI SDK occasionally generates VQL with date-function syntax that is not fully supported by the Denodo Virtual DataPort instance, resulting in failures for some time-based queries. This is a known limitation of the SDK and does not affect the core analytical workflow of the system.

Prior Work

Data systems have been used in the past to support banking and financial applications, but design and security are still significant concerns, as shown by numerous studies. According to (Ni et al., 2026), there is a need for well-designed platforms organized in well-structured components, focusing on “AI innovation, branch networks, asset quality, and privacy.” This is in line with the focus on the design of Denodo-grounded platforms and secure deployment.

AI and Natural Language Processing (NLP) has been used within actual management tasks in studies involving financial reporting, proving the capability of AI. “The use of ‘natural language interfaces …to map queries to visualizations’ showcases the benefit of automated systems in improving business reporting applications,” as mentioned in (Shen et al., 2022). The analysis of logical data fabric in the enterprise sector by (Denodo Technologies, 2024) draws attention to the significance of unified data layers, stating that the topic "provides a logical architecture for connecting disparate data sources without moving the information." This gives emphasis to the significance of organized, accurate, and safe systems.

Although these studies provide useful insights, they also point to certain weaknesses, as they are model-centric, rather than considering the overall operational environment. There are few studies on comprehensive architecture for hybrid systems, ranging from logical data virtualization to standardized protocols and reporting integration. This new industry fills the gap by developing a Denodo-grounded infrastructure tailored only for the deployment of branch performance intelligence.

Hypothesis

The project hypothesizes that a modular architecture integrating OutSystems, FastAPI, and Denodo AI SDK with explicit governance controls will improve the speed, consistency, and security of branch-performance reporting compared to manual, technically intensive workflows. If the architecture is correctly implemented, the system will provide timely analytical outputs, preserve access-policy boundaries across user tiers, and maintain stable operation for practical management reporting scenarios.

Proposed Solution

The suggested solution utilizes the integration of four services. The front-end will be provided by a cloud-based OutSystems application. Upon loading the page, it will embed the user's identity and API keys into the browser environment. The backend service will provide security access control and route requests based on whether they are handled by Denodo AI SDK (natural language processing) or the Power BI MCP server (dashboards).

The Denodo AI SDK will translate natural language to VQL and execute it against Denodo Virtual DataPort, connecting to a Supabase PostgreSQL database using JDBC. Once filtered for governance purposes through the backend interface, the entire result of the query is passed back to the front end to be visualized.

Description of the Report

Chapter 2: Background

This chapter will cover the concepts and technology being used in the proposed project, including data virtualization, Denodo AI SDK, logical data fabric, and natural language processing concepts in banking reports. The chapter will also compare other platforms, studies, and the gaps this proposed endeavor will fill, as well as conduct research on the current state of the market for banking intelligence systems and how the proposed system will improve those systems.

Chapter 3: Design

The topic of this chapter is system requirements, architecture, and designs. The proposed deployment of OutSystems components, FastAPI backend, and Denodo AI integration is examined and analyzed.

Chapter 4: Implementation

This chapter will detail how this system is developed, a step-by-step guide based on the Project Implementation phase, highlighting key aspects of implementation along with guidance.

Chapter 5: Testing

The chapter describes the test plan cases, processes, and environments used in validating system behavior. Performance, security, stress, and functional tests are used, and their outcomes and findings are shown.

Chapter 6: Discussion

In this chapter, the outcome of the tests and their relation to the objectives will be assessed. This chapter will discuss achievements, limitations, challenges, Bahraini perspectives, lessons learned during development, future work, and LESPI reflection.

Chapter 7: Conclusion

In this chapter, the findings obtained from this project will be summarized, and the hypothesis will be addressed.

Chapter 8: References

All the sources used are referenced in APA format.

Chapter 9: Appendices

The following chapter contains more details, including diagrams, configurations, and screenshots.

2. Background

Introduction

This section supports the development of a secure data environment for branch performance analysis and reporting in banking. The project applies data virtualization to enable controlled data processing, AI-based computation, and secure connectivity across distributed branch networks. To establish a clear foundation, this chapter introduces the concepts, technologies, and prior research required to design infrastructure for sensitive banking operations within a controlled and governed environment. It also identifies key factors in API security, data transfer, and challenges associated with moving financial data from internal banking systems to logical cloud-enabled architectures. The remaining subsections present data virtualization concepts, natural-language reporting environments, and relevant Denodo services, together with prior studies and industry context that inform the proposed solution.

Related Theory

The Branch Insight 360 infrastructure is grounded in data virtualization, logical data fabric, secure API management, and hybrid communication principles. Banking reporting platforms process large volumes of sensitive financial information and therefore require data segmentation, encrypted communication, and strict information-security controls. Theoretical foundations such as zero-trust architecture, logical network isolation, and encrypted routing are essential for securing interactions across multi-layer environments. High availability and redundancy are also critical in financial systems due to the direct impact of service disruption on operational decision-making.

Many banking environments operate with mixed on-premise and cloud infrastructures, requiring secure API gateways for controlled communication, which reinforces the relevance of hybrid architectural models. Regulatory frameworks such as PDPL and GDPR impose additional constraints through requirements for encryption, controlled routing, auditability, data retention limits, and restricted administrative access (Wu et al., 2023). These principles directly influence the system topology, including FastAPI service boundaries, restricted traffic routing, secure data transport between branches, and separated administrative channels.

Natural Language Interfaces to Databases

Natural Language Interfaces to Databases (NLIDBs) enable users to query structured data using natural language instead of formal query languages such as SQL (Androutsopoulos et al., 1995). Early NLIDBs relied on rule-based approaches that required manual grammar definitions, limiting scalability and adaptability. Contemporary approaches use neural sequence-to-sequence models trained on datasets such as Spider (Yu et al., 2018) and WikiSQL (Zhong et al., 2017). Recent enterprise implementations, including the Denodo AI SDK, use retrieval augmented generation (RAG), where schema-relevant context is retrieved before query generation. This improves accuracy in enterprise environments where users lack awareness of underlying data structures.

Data Virtualization

Data virtualization provides a logical abstraction layer that integrates multiple data sources without physically moving or duplicating data (Chaudhuri et al., 2011). In Denodo Virtual DataPort, virtual views function as relational interfaces that behave like physical tables from the client perspective. For Branch Insight 360, this allows the AI SDK to generate and execute queries against business-defined views such as branch_performance_monthly, while preserving separation between data sources and enforcing governance rules.

Access Control Models

In Role-Based Access Control (RBAC), permissions are assigned to roles rather than individual users, and users inherit permissions through assigned roles (Barkley, 1997). In Branch Insight 360, roles include Administrators, Normal Allowed, Partial, and Blacklist. RBAC improves governance and auditability but may become rigid in dynamic environments due to role dependency.

Attribute-Based Access Control (ABAC) evaluates access using contextual attributes such as user identity, request environment, and resource metadata (Hu et al., 2014). In this system, attributes such as X-User-Group, X-User-Name, and X-API-Key are used to support dynamic access decisions.

Policy-Based Access Control (PBAC) introduces centralized policy enforcement that combines role and attribute evaluation with runtime rules. In Branch Insight 360, FastAPI middleware applies authentication, blacklist enforcement, and request validation to ensure consistent governance during execution.

Used and Considered Technology

Category	Selected	Considered Alternatives	Reason for Selection
Frontend Framework	OutSystems	React, Angular, Vue.js	Supports rapid development, cloud deployment, and integrated identity handling.
Backend API	FastAPI (Python)	Express (Node.js), Django, Flask	Provides high-performance asynchronous execution and strong AI integration support.
Virtualization Layer	Denodo Platform	Physical ETL, Manual API Aggregation, IBM Cloud Pak	Enables logical data access without duplication and supports governed query execution.
Query Engine	Denodo AI SDK	Direct LLM prompting, Vanna.ai, LangChain SQL	Provides schema-aware VQL generation and metadata-driven retrieval.
Database System (DBS)	Supabase (Postgres), MongoDB	MySQL on AWS RDS, Azure DB, AWS Dynamo DB	Offers fast deployment, cost efficiency, and strong PostgreSQL compatibility.
Dashboard Service	Power BI MCP	Chart.js direct rendering, Tableau Embedded	Provides enterprise-grade reporting structure and standardized dashboard modeling.

Table 2: Used and Considered Technology

Related Work & Literature Review

Most studies on banking branch performance systems focus on algorithmic performance and model accuracy, with limited emphasis on full infrastructure deployment in regulated environments. Prior research demonstrates the value of natural-language processing in improving reporting accessibility, but often lacks detailed architectural implementation for governed enterprise systems (Aman, 2024). Security-focused studies highlight encrypted

data transfer in financial systems but provide limited coverage of integrated architectures involving API gateways, logical isolation, and runtime governance (Shen et al., 2022). Additional research identifies operational challenges in AI-driven environments, including reliability, fault tolerance, and metadata management (Mhlanga et al., 2024).

These limitations indicate a need for infrastructure-oriented architectures that integrate security, scalability, and operational governance. The Branch Insight 360 system addresses this gap through a Denodo-based virtualized backend, secure API routing, private data connectivity, and policy-driven execution workflows. This design aligns with infrastructure-centric recommendations for financial systems in regulated environments (Ni et al., 2026).

Market Research

The global business intelligence market was valued at USD 29 billion in 2023 and is projected to grow at approximately 9% CAGR through 2030, driven by demand for real-time analytics, AI integration, and regulatory compliance requirements such as PDPL and GDPR. In the banking sector, GCC institutions are accelerating digital transformation through AI analytics and natural-language interfaces. Branch Insight 360 aligns with this trend by enabling governed natural-language analytics for branch-level decision-making.

Bahrain is a leading GCC fintech hub, supported by Economic Vision 2030, which promotes a transition toward a technology-driven and sustainable economy. Initiatives such as Bahrain FinTech Bay have expanded AI, cloud, and financial data experimentation through regulatory sandboxes. Open banking adoption further supports interoperable financial data systems and analytics-driven services.

Denodo also provides infrastructure efficiency benefits, including reduced hardware dependency and lower operational costs, supporting digital transformation strategies in Gulf banking systems (PwC, 2025). Within this context, Denodo represents a suitable foundation for scalable and governed analytics infrastructure.

The design of Branch Insight 360 is also guided by the Central Bank of Bahrain (CBB) Rulebook Volume 1 for Conventional Banks OM Module, which defines requirements for cybersecurity governance, authentication, outsourcing controls, audit logging, and data retention. These requirements align with PDPL regulations and directly inform the system's RBAC, ABAC, and PBAC security architecture. Additionally, Bahrain's bilingual environment introduces future requirements for Arabic natural-language processing support.

3. Methodology

Introduction

This chapter describes the methodology used to design, develop, and assess the Branch Insight 360 Analytics System. It explains the procedures used to gather functional and non-functional requirements, analyze the system, design the architecture, implement the solution, and validate the final system. The objective is to ensure that the proposed solution addresses the banking analytics, governance, and security requirements identified for this project. The requirements were collected from the project brief, stakeholder interaction, interviews, and academic literature, which helped define the functionality of the system. The chapter also presents the system constraints, architectural decisions, and the design models applied in the development process, including workflow, data flow, and access control models.

Requirements and Design

Branch Insight 360's requirement gathering was informed by three main sources, ensuring alignment between business needs, technological feasibility, and compliance.

First, the project's brief provided the scope of the system, which included natural language questioning and the integration of OutSystems with Denodo. Moreover, it contributed to the definition of the primary objectives, which included converting user inquiries into VQL queries.

Second, consultations with the project supervisor and fictional banking system users assisted in defining more requirements. During these consultations, user classes such as Administrator, Normal, Partial, and Blacklisted were recognized. In addition, these consultations contributed to the development of the Role-Based Access Control, Attribute-Based Access Control, and Policy-Based Access Control models.

Third, reviews of academic articles, real-life banking systems, and technical documentation assisted in identifying additional requirements, which were incorporated into the design.

Functional Requirements

Functional requirements define the specific functions and capabilities that the system must provide to satisfy both user and organizational demands. They describe what the system should be capable of doing, such as allowing branch managers and designated users to access and analyze banking information through natural language queries. Furthermore, the sources used to identify these requirements are listed in Appendix 1.

Req ID	Source	Requirement Description	Data	Process	Communication	Scalability	Security	Priority	Status
Req-FR-01	Project Brief	Accept natural language queries from OutSystems		✓	✓			High	-
Req-FR-02	Research	Translate English questions into valid VQL via AI SDK	✓	✓	✓		✓	High	-
Req-FR-03	Interview	Enforce role-based authorization (RBAC) levels	✓	✓	✓	✓	✓	High	-
Req-FR-04	Project Brief	Generate interactive management-ready dashboards	✓	✓	✓			High	-
Req-FR-05	Client	Deny analytical access requests from blacklisted users		✓	✓		✓	High	-

Req-FR-06	Research	Redact sensitive info for partially privileged users	✓	✓			✓	Medium	-
Req-FR-07	Client	Differentiate aggregate intent from record-level requests	✓	✓			✓	Medium	-
Req-FR-08	Project Brief	Provide branch-specific 360-degree analytics	✓	✓				Medium	-
Req-FR-09	Research	Invoke fallback execution path when primary flow fails	✓	✓	✓	✓	✓	High	-
Req-FR-10	Client	Export insights via print/PDF executive workflow	✓	✓				Medium	-
Req-FR-11	Project Brief	Support OutSystems embedding via runtime bootstrap		✓	✓	✓	✓	High	-
Req-FR-12	Research	Classify analytical intent (trend, ranking, summary)	✓	✓				Medium	-
Req-FR-13	Client	Provide clarification prompts for ambiguous requests		✓	✓			Medium	-

Table 3: Functional Requirements

The most critical functional requirements are Req-FR-02 and Req-FR-03, which represent the intelligence and governance components of the system. Natural-language-to-VQL translation and multi-level policy enforcement application during the execution process should contribute to equal access to data and maintain banking security.

Non-Functional Requirements

Non-functional requirements define the quality attributes of the system and how it should operate. These requirements establish the required performance, security, and service attributes that are needed in a professional banking context (Hevner et al., 2004).

Req ID	Source	Requirement Description	Performance	information	Economics	Control/ Security	Efficiency	Service	Priority	Status
Req-NFR-01	Research	Sub-500ms health check response time	✓				✓		High	-
Req-NFR-02	Interview	Enforce HTTPS for all OutSystems traffic				✓		✓	High	-
Req-NFR-03	Research	Environment-based credential isolation		✓		✓			High	-
Req-NFR-04	Interview	Policy checks executed in cheapest-first order	✓				✓		Medium	-
Req-NFR-05	Project Brief	Deployable on standard developer hardware			✓		✓	✓	Low	-
Req-NFR-06	Project Brief	PDF Generation Report		✓				✓	Low	-

Table 4: Non-Functional Requirements

The most critical non-functional requirements are Req-NFR-02 and Req-NFR-03, since these non-functional requirements have a direct impact on CBB technology risk

management criteria. These requirements help ensure that sensitive banking metadata and API keys are not exposed in transport or in frontend code.

Constraints

- Technical Constraint: The application uses a self-signed TLS certificate on port 8443, which causes browsers to display a security warning on initial connection.
- Security Constraint: Denodo AI SDK can generate simple VQL queries without session-token checks, which could increase the risk of VQL injection if security controls are bypassed. To reduce this risk, PBAC middleware with multiple verification layers was introduced.
- Compliance Constraint: The CBB requires full audit logging and monitoring due to the nature of the application. This influenced the volume of logging and stored data in the project. The issue was managed through FastAPI request logs and tracking.
- Data Constraint: Due to confidentiality issues, live banking data could not be used. Therefore, mock banking data was used in the application, which does not represent every real-world case. To improve test diversity, eleven distinct banking datasets were generated.
- Time Constraint: The application had to be completed within a fixed development period, which limited the implementation of advanced features such as Arabic natural-language processing and additional optimization work.

System Design

This subsection presents the design methodology used to translate the requirements of Branch Insight 360 into a structured system architecture. It explains how the system requirements shaped the design decisions and how the selected models support the development of a secure and governed banking analytics platform. The section focuses on the main design elements, including the system architecture, data flow, use-case structure, and business workflow, while additional supporting diagrams are provided in Appendix 2. The purpose of this subsection is to show how the proposed solution was designed to satisfy the functional, security, and governance requirements identified in the previous chapter.

Branch Insight 360 follows a layered design approach in which each decision is derived directly from the requirements in Section 3.1. The process began with context modelling to define system boundaries, followed by data flow analysis to establish governance checkpoints, and concluded with component and access control specification. Design elements are modelled using UML and flowchart notation; the two most critical are detailed below and the full catalogue is in Appendix 2.

System Architecture Diagram

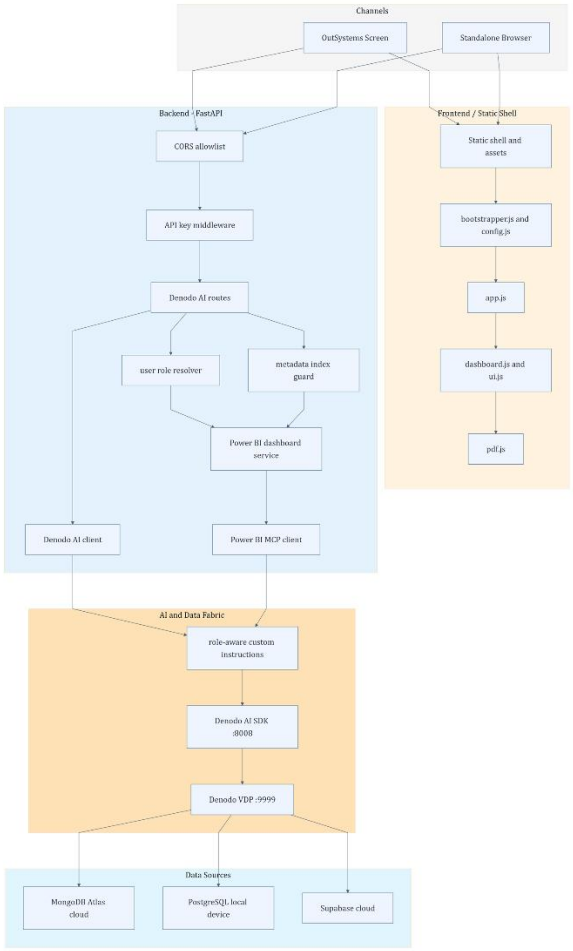


Figure 1: Component Architecture Diagram

The component diagram shows the overall design of the Branch Insight 360 platform and the way the cloud-based OutSystems frontend interacts with the modular Python backend. This diagram illustrates the primary functions of the platform, namely natural-language processing and multilayered access control, through modules such as FastAPI and Denodo AI SDK. This architecture helps keep banking credentials within the local runtime environment.

The architecture establishes OutSystems as the cloud frontend, FastAPI as the orchestration layer, and Denodo Virtual DataPort as the governed data access layer. This three-tier separation satisfies REQ-FR-01 and REQ-NFR-03. The HTTPS bridge on port 8443 was

introduced as a design constraint to allow the cloud frontend to reach the local backend without exposing the data layer directly.

The architecture also supports compliance with REQ-FR-03 (RBAC Enforcement) and REQ-NFR-03 (Environment Isolation) through limited service boundaries and controlled routing. In this design, FastAPI coordinates communication between the Denodo intelligence service on port 8008 and the dashboard generation service on port 8011.

Data Flow Diagram

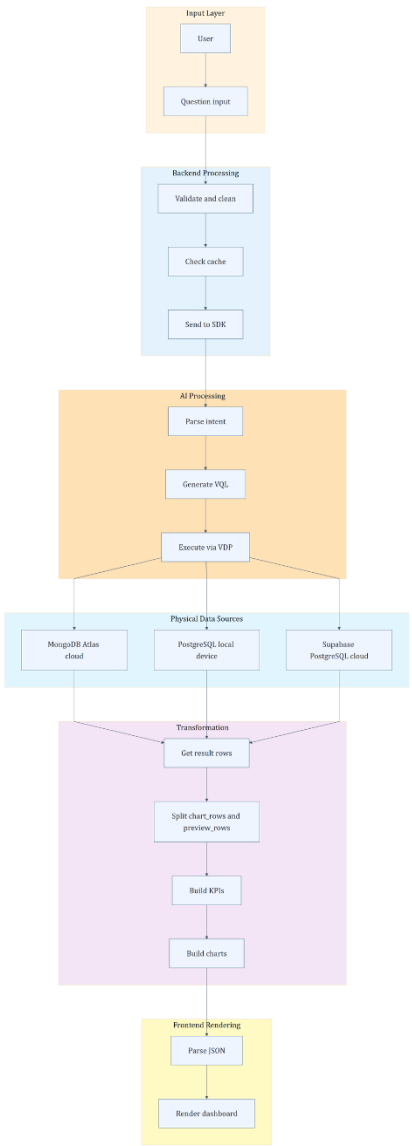


Figure 2: Data Flow Diagram

The Data Flow Diagram (DFD) demonstrates how data moves from the initial natural-language statement in the OutSystems interface to the final dashboard seen by the user. It explains how sensitive banking data is processed, verified, altered, and filtered within the overall workflow process.

The data flow traces a query from ingress through policy validation, VQL generation, execution, and response delivery. Each transition represents a governance checkpoint: requests are authenticated before routing, VQL is executed only after policy approval, and results are filtered before return. This satisfies REQ-FR-05, REQ-FR-06, and REQ-NFR-02.

This flow facilitates the implementation of REQ-FR-02 (Natural Language to VQL Translation), where the user input is translated into structured VQL queries. It also assists in the implementation of REQ-FR-07 (Diagnostic Scrubbing) since backend filtering takes place and sensitive metadata are scrubbed before being passed to the frontend.

Use-Case Diagram

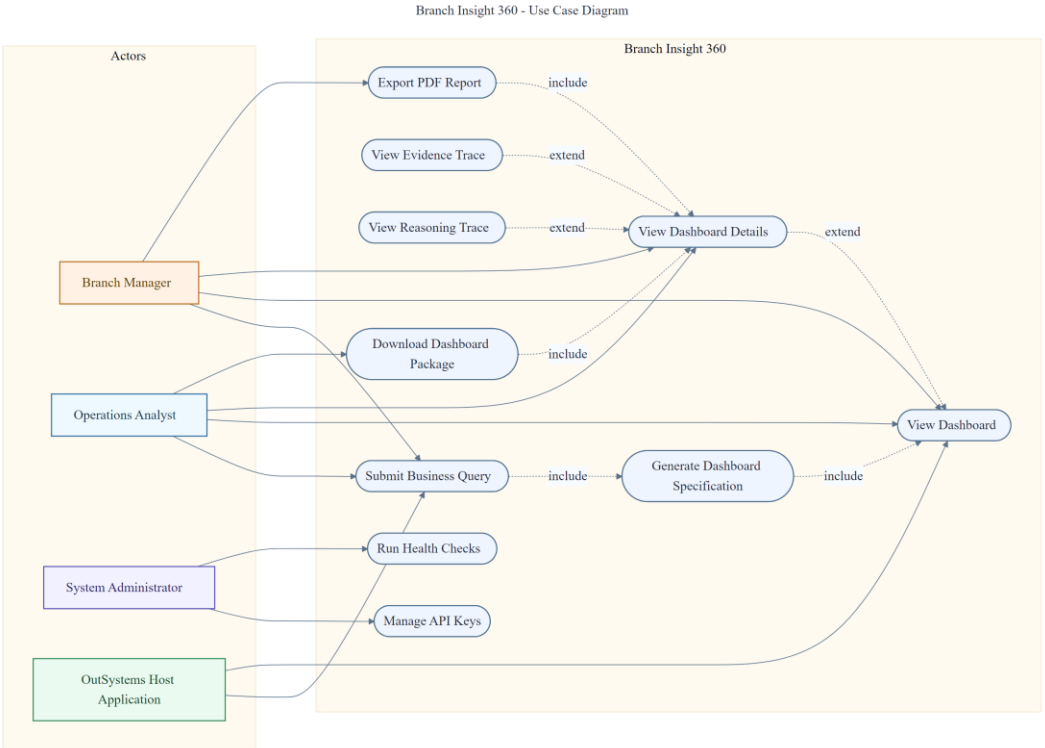


Figure 3: Use Case Diagram

The use-case diagram was selected because it clearly describes interactions between actors and systems within Branch Insight 360. It presents the actors and the use cases available to them according to their responsibilities. This diagram provides useful insight into the system from different perspectives.

Use-case diagrams are valuable in requirements analysis and access-control design because they show the relationship between actors and system functionalities. The include and extend relationships reveal mandatory processes and supplementary functions related to the primary use cases.

Overall, the use-case diagram provides a concise and comprehensive representation of the system and the roles of its actors, which benefits both stakeholders and software designers.

Business Workflow diagram

The business workflow diagram illustrates the end-to-end operational sequence from a branch manager submitting a natural-language query to receiving a governed dashboard response. It maps human decision points and system handoffs, highlighting where access-control gates and fallback decisions are applied. This supports REQ-FR-08 and REQ-FR-09 by showing that escalation and retry behavior are designed into the workflow rather than handled ad hoc.

Additional diagrams can be found in Appendix 2.2.

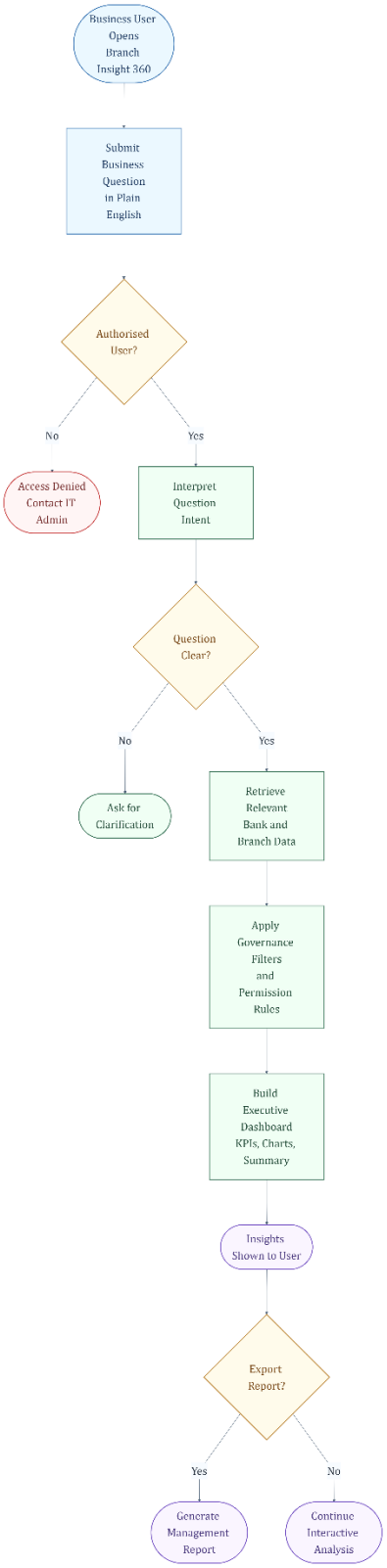


Figure 4: High-Level Business Workflow

Algorithm Design

The access control pipeline, shown in full in Figure 44 (Appendix 2.2), operates as follows: (1) authenticate the API key, (2) retrieve and match the user role via RBAC, (3) evaluate contextual attributes via ABAC, (4) check the PBAC blacklist for sensitive fields, (5) issue allow or deny, (6) if allowed, forward to the Denodo AI SDK for VQL generation, and (7) redact any sensitive fields before returning the result. This cheapest-first order minimizes processing cost on rejected requests, satisfying REQ-NFR-04.

The decision model is rule-based with deterministic policy evaluation, ensuring predictable access control outcomes. Runtime execution follows a sequential pipeline where each stage must pass validation before proceeding, preventing unauthorized propagation of queries.

Data Structures

JSON is used for all inter-service communication due to native compatibility with OutSystems REST connectors and FastAPI. Python dictionaries store RBAC and ABAC policies in memory, providing constant-time lookups that support the cheapest-first evaluation order above. Relational virtual views, modelled in Figure 41 (Appendix 2.2), were chosen over direct database tables to enforce governance at the schema level and ensure the Supabase PostgreSQL database is never accessed directly, satisfying REQ-FR-03 and REQ-NFR-01.

Alternative structures such as direct SQL API access or document-based NoSQL storage were rejected due to lack of centralized governance control and reduced query traceability. JSON and relational virtual views were selected because they provide schema consistency, auditability, and compatibility with both AI-generated queries and policy enforcement layers.

4. Implementation

Database Configuration and Connection String.

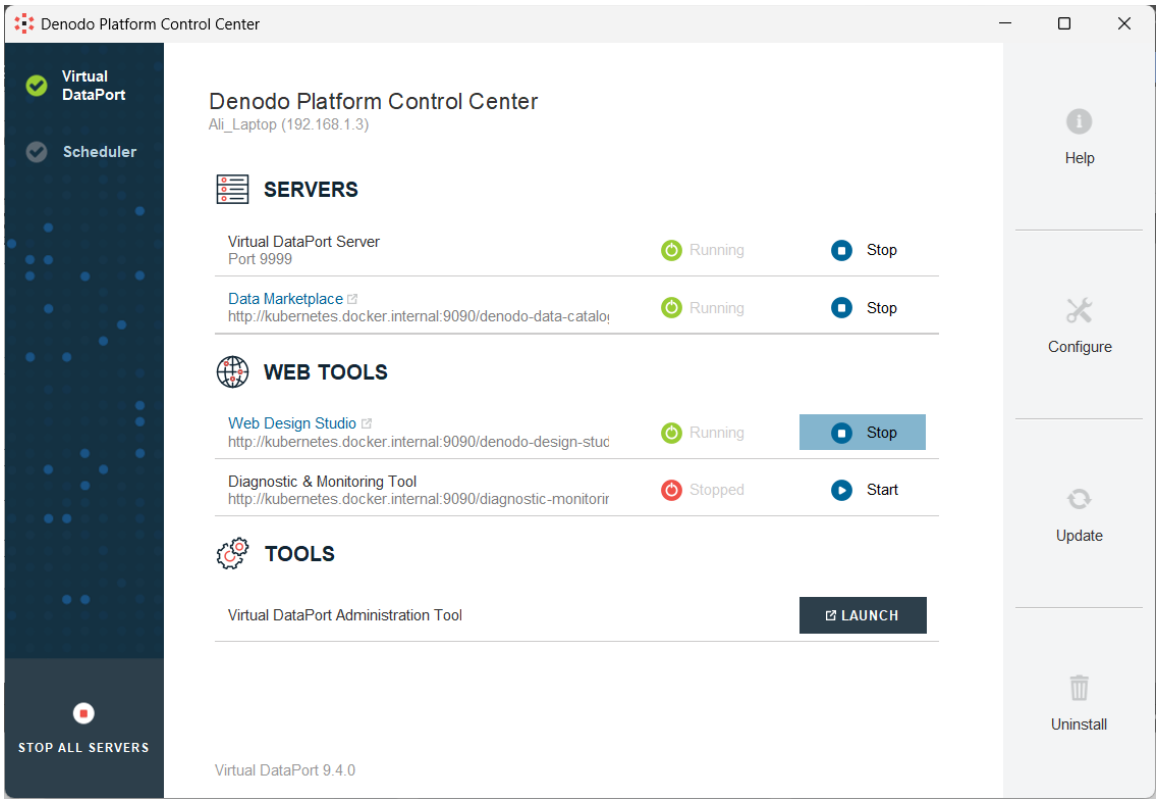


Figure 5: Denodo Platform Control Center

Having these services running is important as they are the heart of the System. Opening the web design studio to access the Denodo platform.

By default, the username and password for the Denodo server are admin, admin.

The data Marketplace is mainly used to sync the data in the web studio, and acts as a layer for the AI or any other software that requires access to data by Denodo.

Afterwards, by Pointing towards a Database, click on the "+" button and then select Create New Data Source based on the figure below.

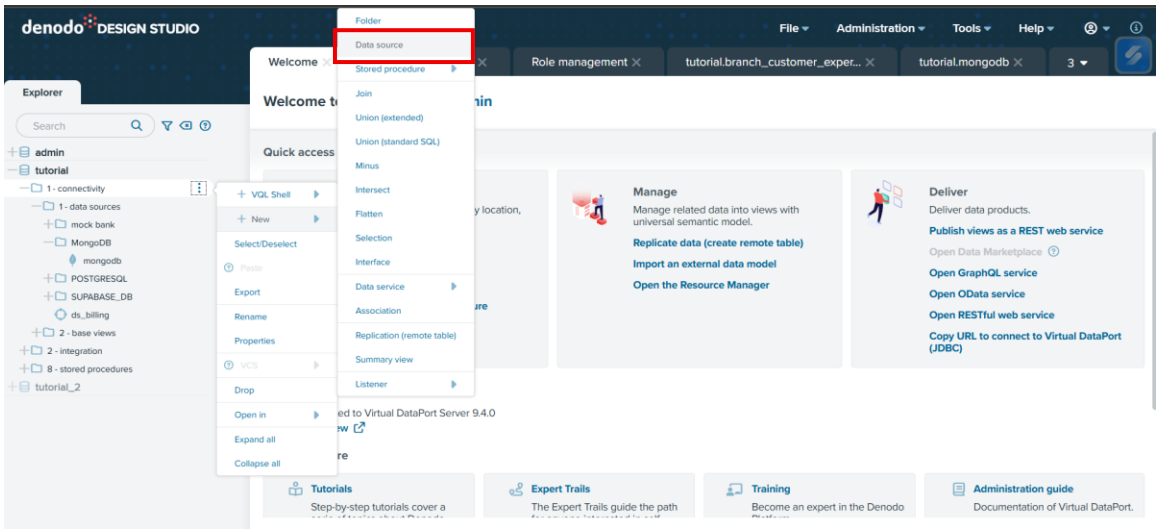


Figure 6: Data Source Adding

By default, the databases are empty, so creating folders and following Denodo's best practices will result in an organized folder structure.

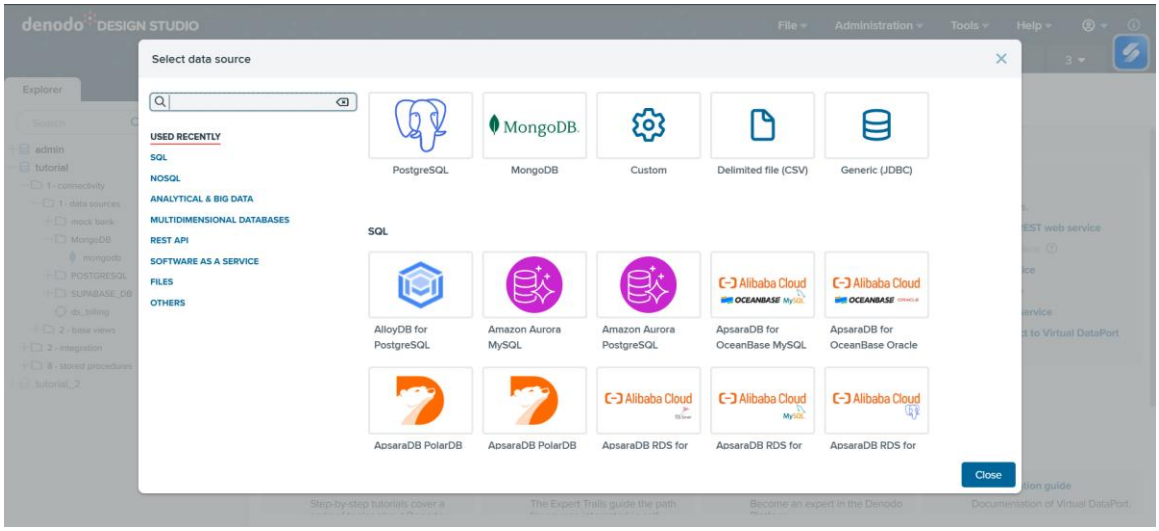


Figure 7: Data Menu

Denodo supports many data sources, from SQL to NoSQL, such as MongoDB. For this example, we will use MongoDB and show the connection string and its connectivity.

In Denodo, after selecting MongoDB, the figure below will show the connection to the figure 6.

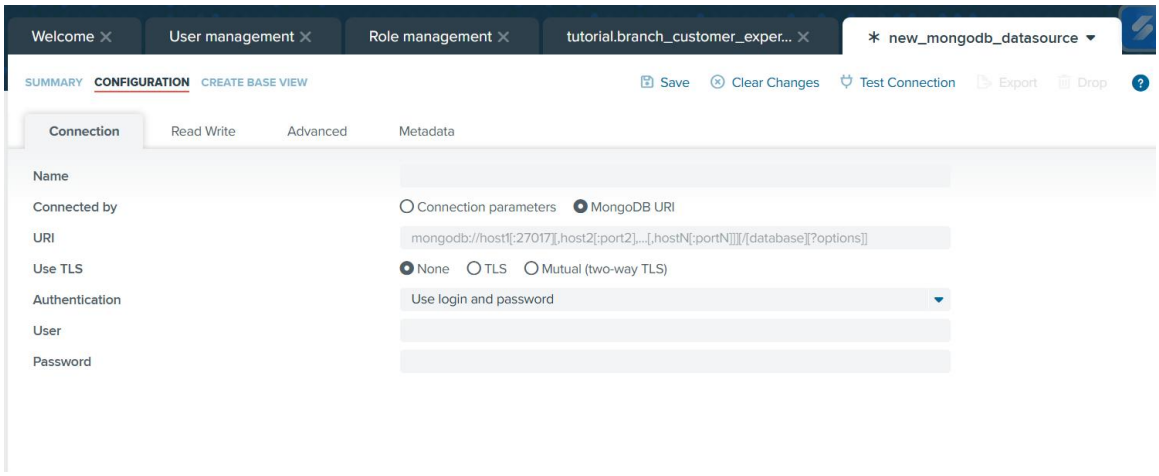


Figure 8: Denodo MongoDB Configuration tab.

By clicking on the MongoDB URI option, we will need to get the DB string from MongoDB.

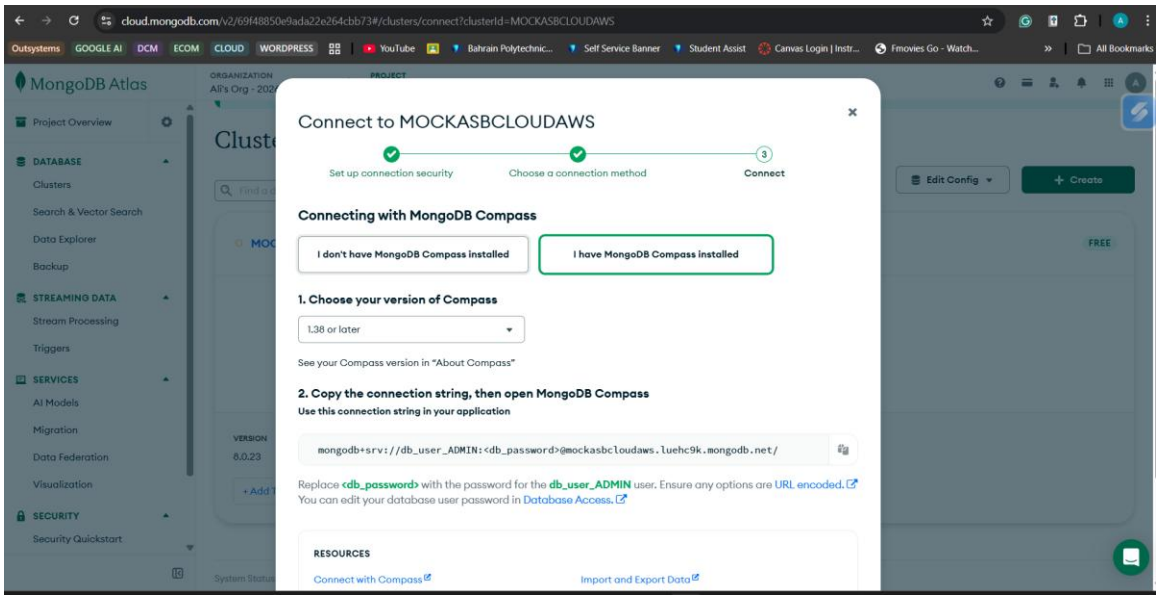


Figure 9: MongoDB Connection String

By clicking on Connect to DB, several options appear. Select Connect using MongoDB Compass.

We will find the connection string and modify it by adding the password to it.

Furthermore, we will need to add the username and password to authenticate the connection.

If everything was done correctly, click on “Test Connection” as it will the connection from Denodo to the MongoDB database.

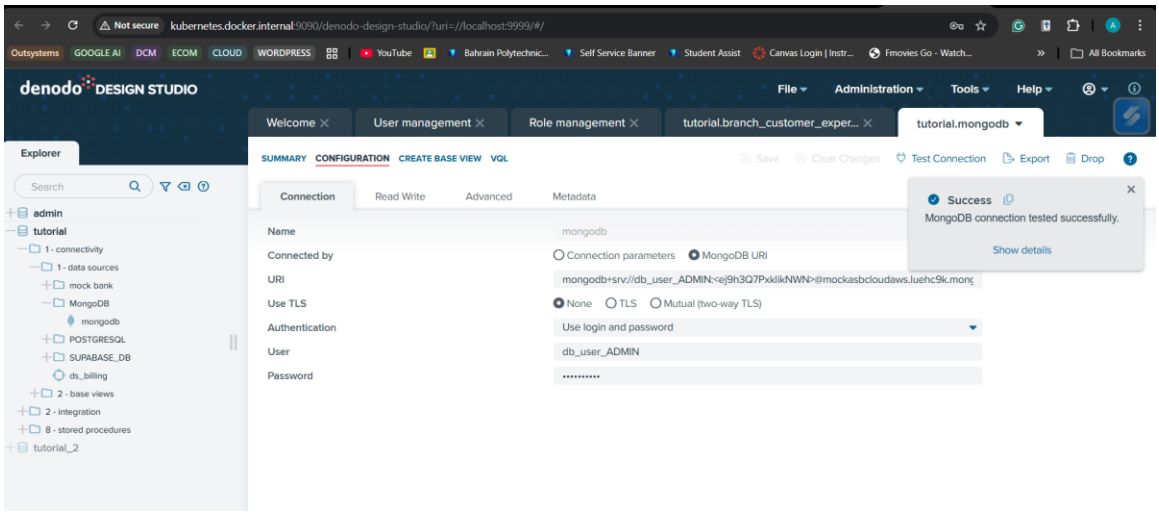


Figure 10: Connection Tested Successfully

Creating Views and connections

Creating the views from the database onto Denodo servers

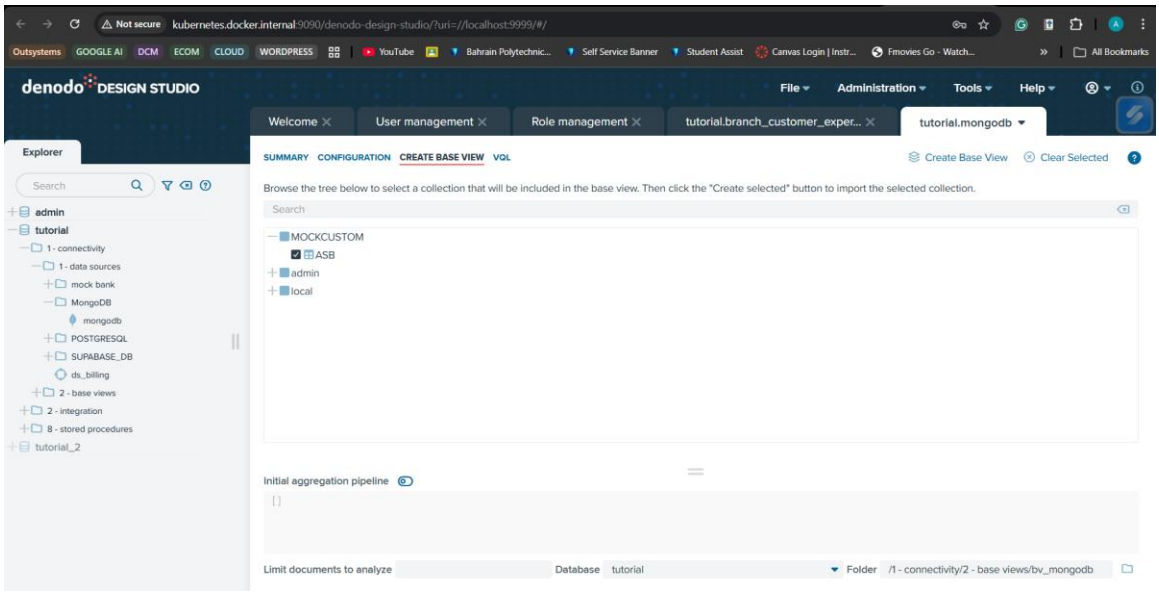


Figure 11: Creating Base Views

After creating the base views and forming the associations, we will see the views created, and we can display what kind of data they contain.

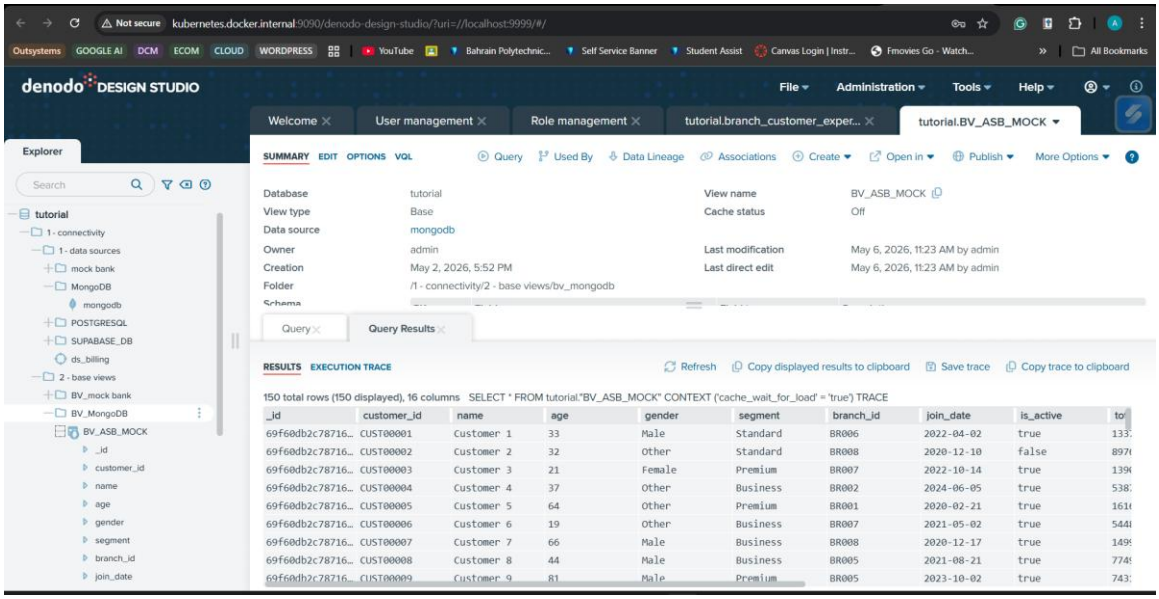


Figure 12: MongoDB Base View Query result.

Data Market Place Synchronization

Heading to the Market Place, we will be able to synchronize the data from the web studio to it.

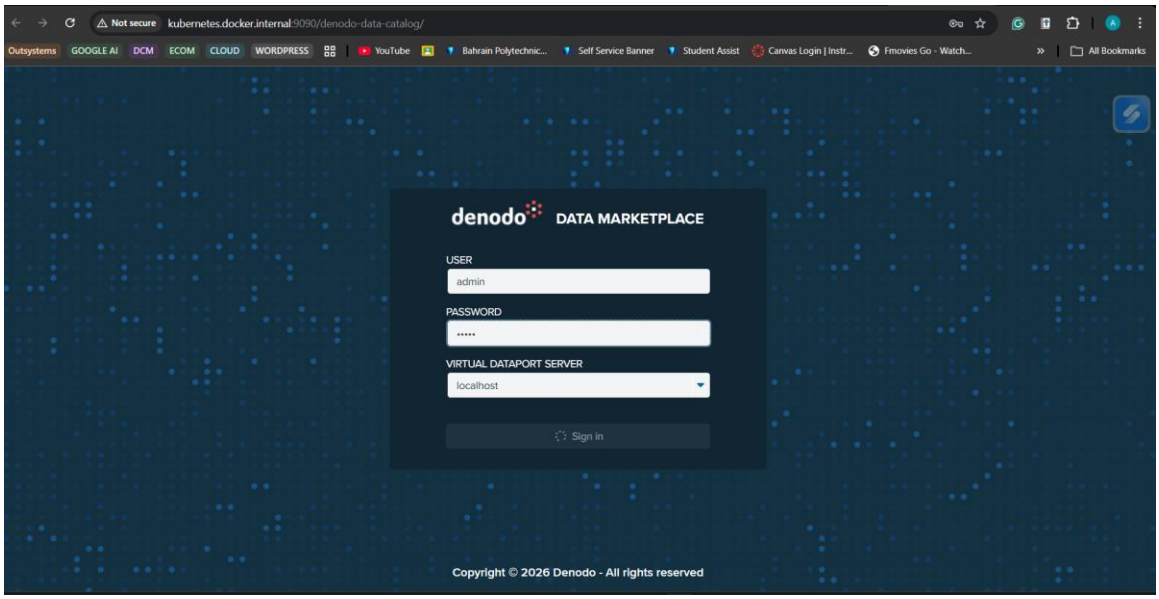


Figure 13: Login Screen to Data MarketPlace

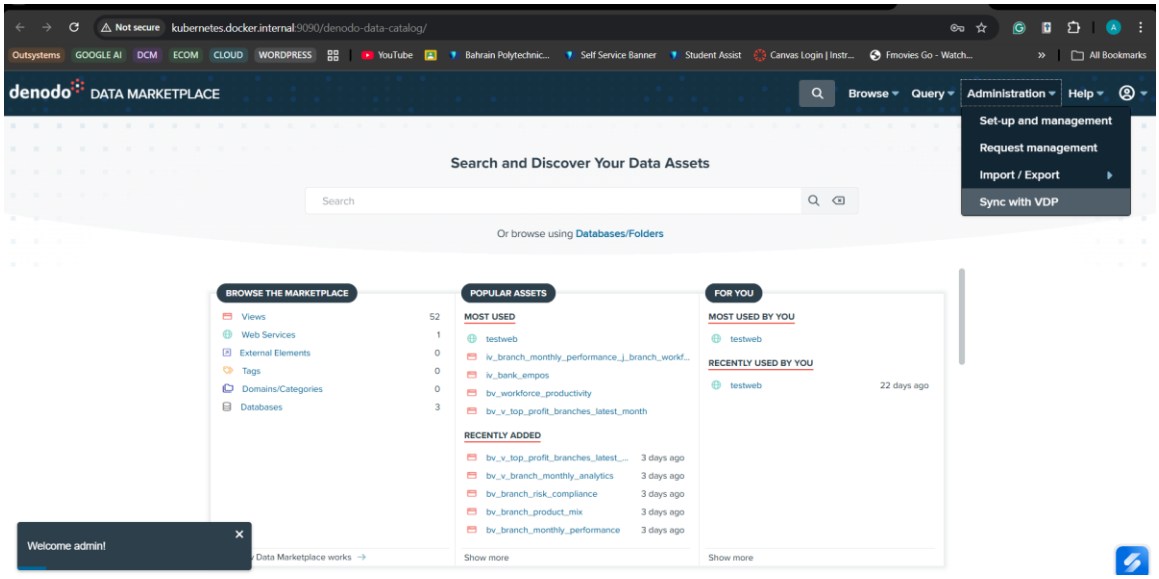


Figure 14: VDP Sync

By clicking on it and clicking on the continue button, the data will be synced to the marketplace.

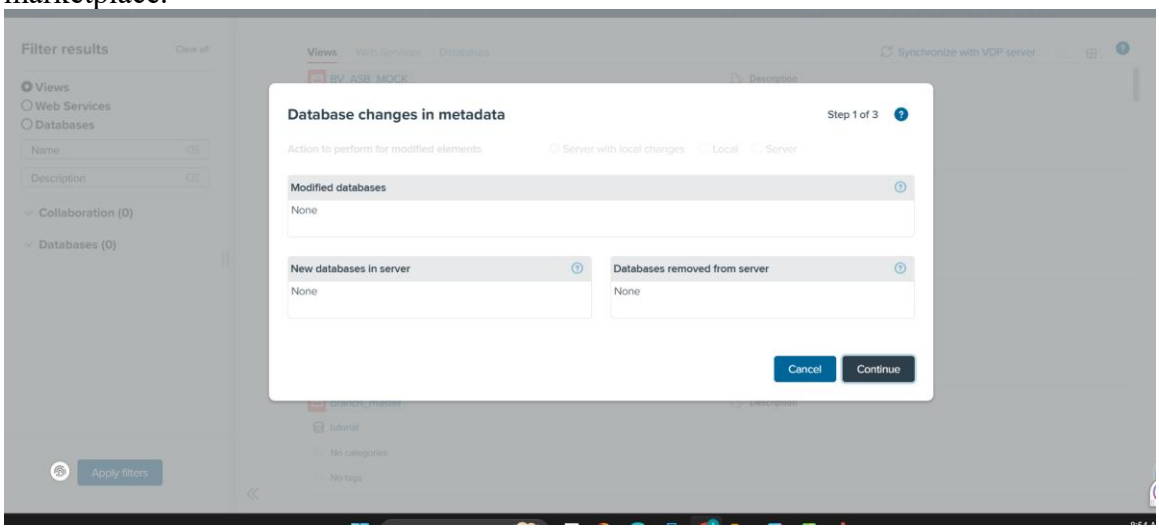


Figure 15: Pop-up menu.

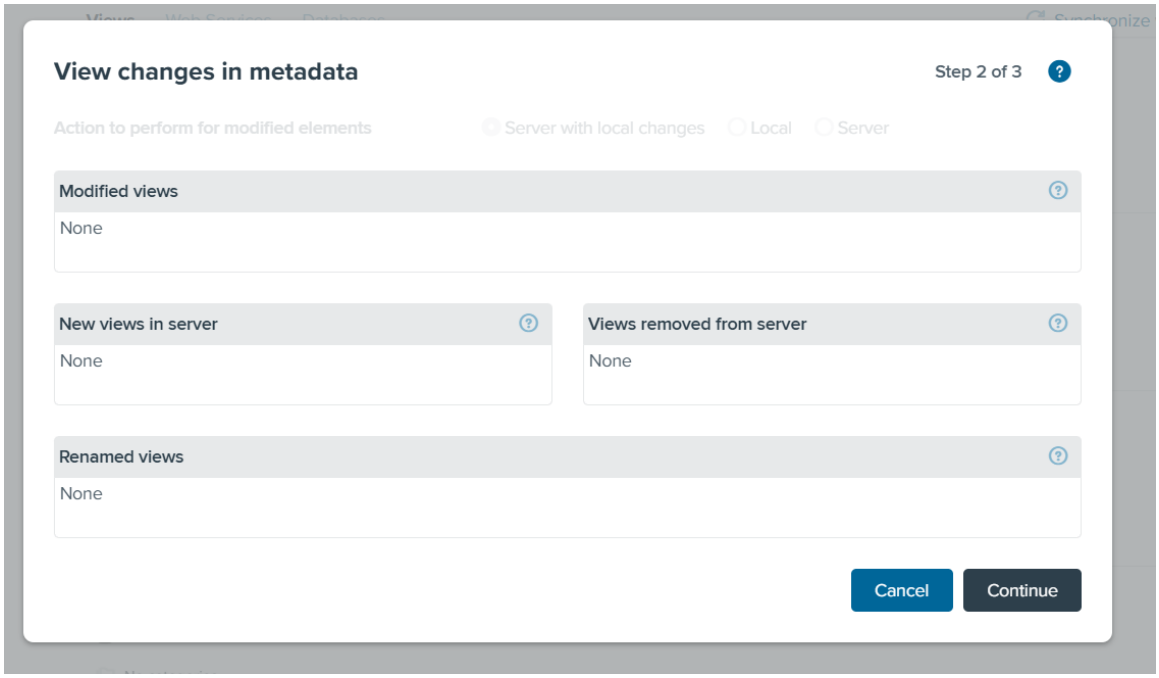


Figure 16: View changes in metadata.

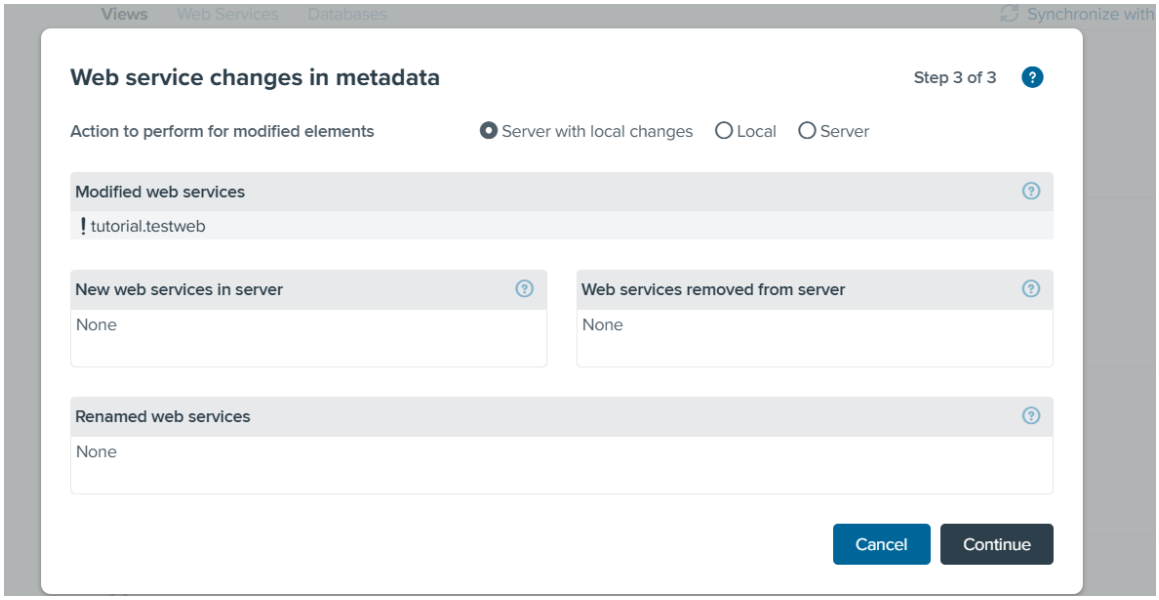


Figure 17: Web service changes in metadata.

The figures above may differ from user to user, in any view modified or created from the web studio can be synced to the marketplace.

OutSystems Configuration

Either import the following file from GitHub, branch360.oml. Or start from scratch by creating new applications.

Using OutSystems Service Studio to host our front end, import the following file from GitHub, branch360.oml.

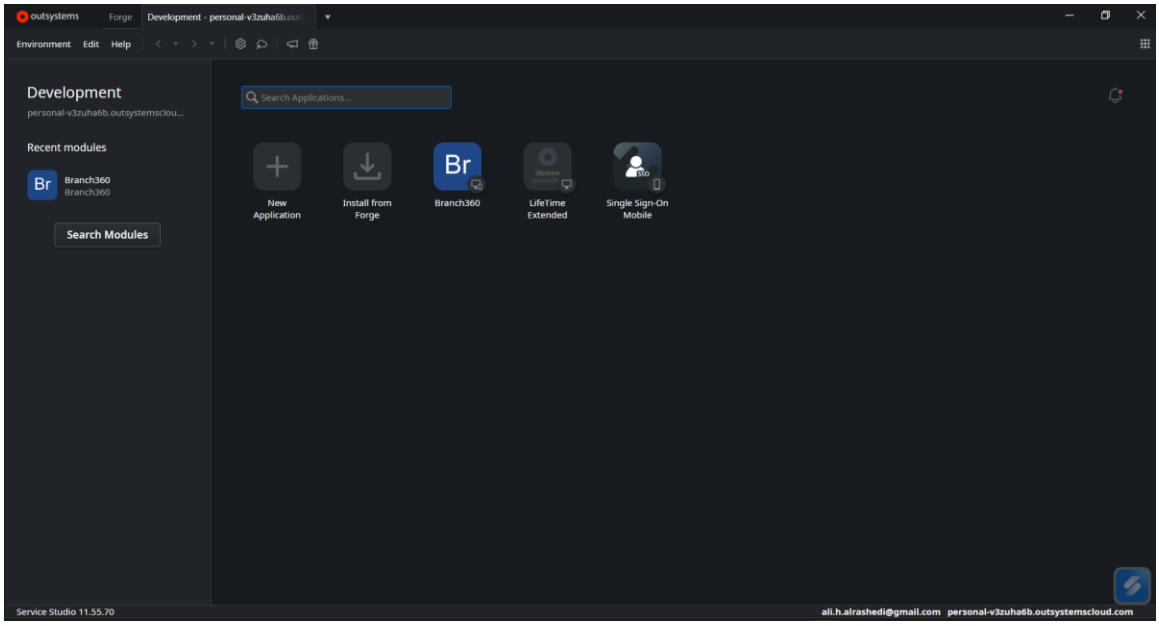


Figure 18: OutSystems Service Desk Home Page

Click on Branch360 and then click on the module.

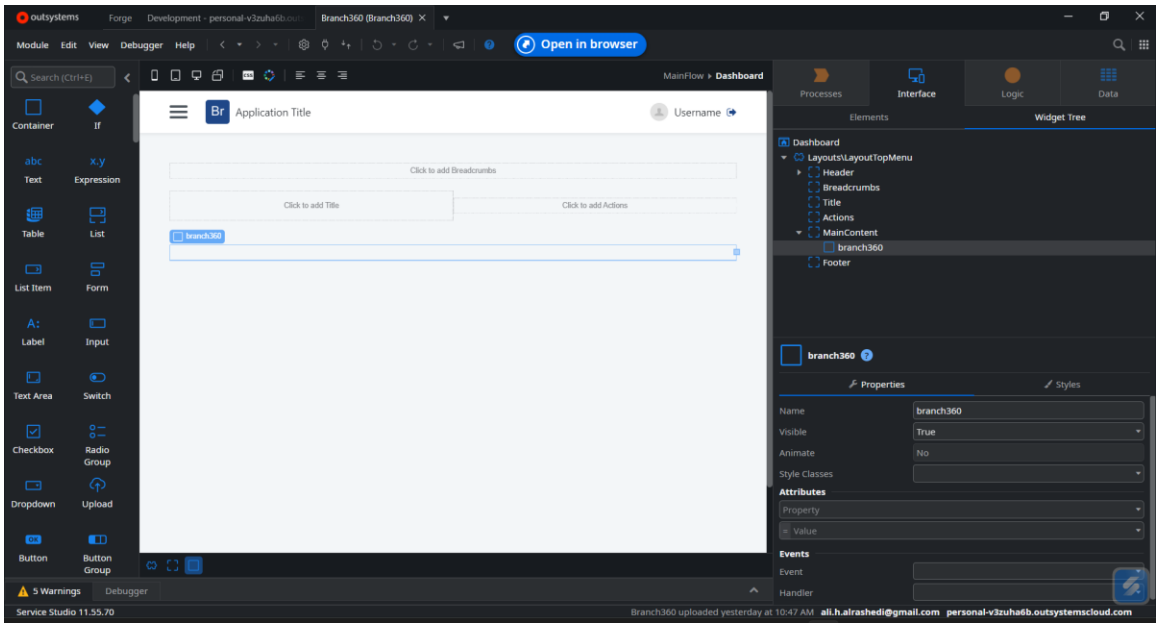


Figure 19: OutSystems Main Screen

If we just imported everything from the branch360.oml then, the following steps will be unnecessary.

Setting up OutSystems

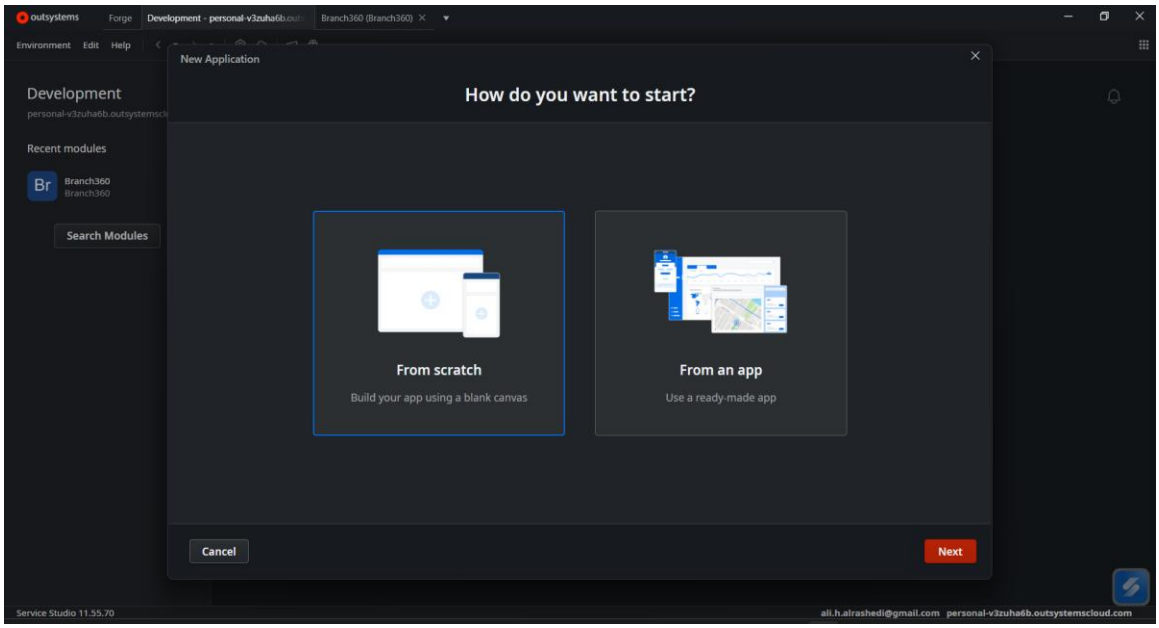


Figure 20: New Application pop-up.

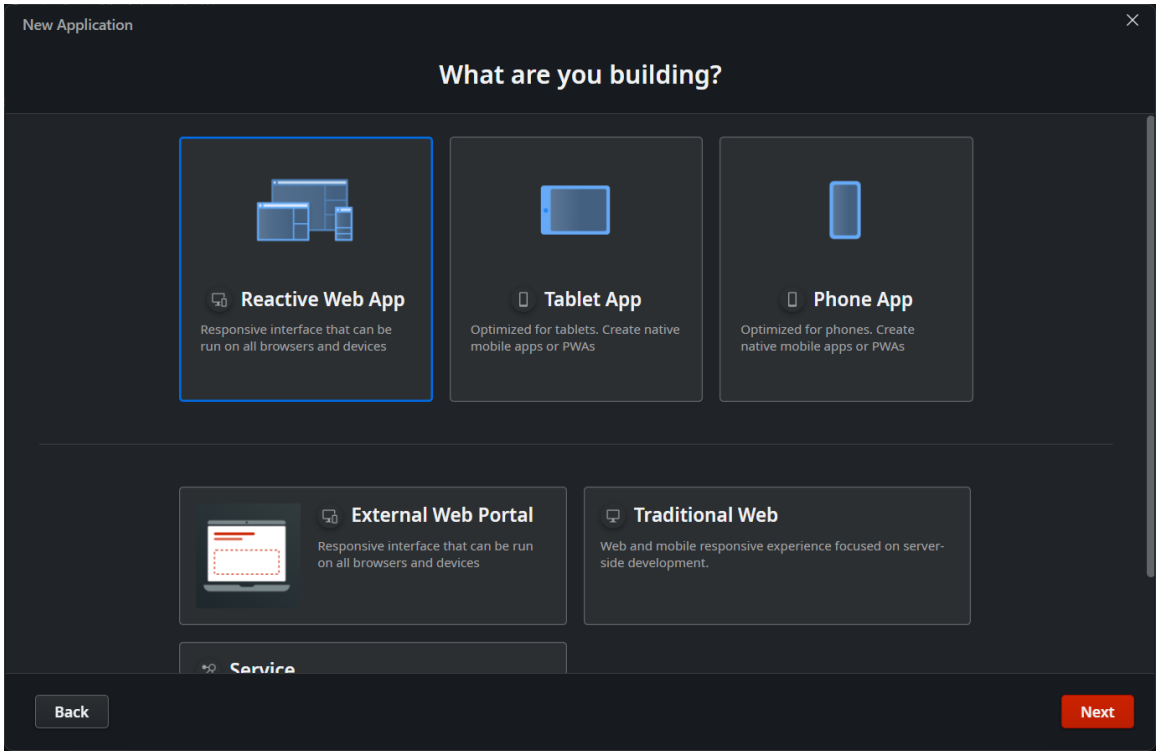


Figure 21: New Application pop-up Reactive Web App

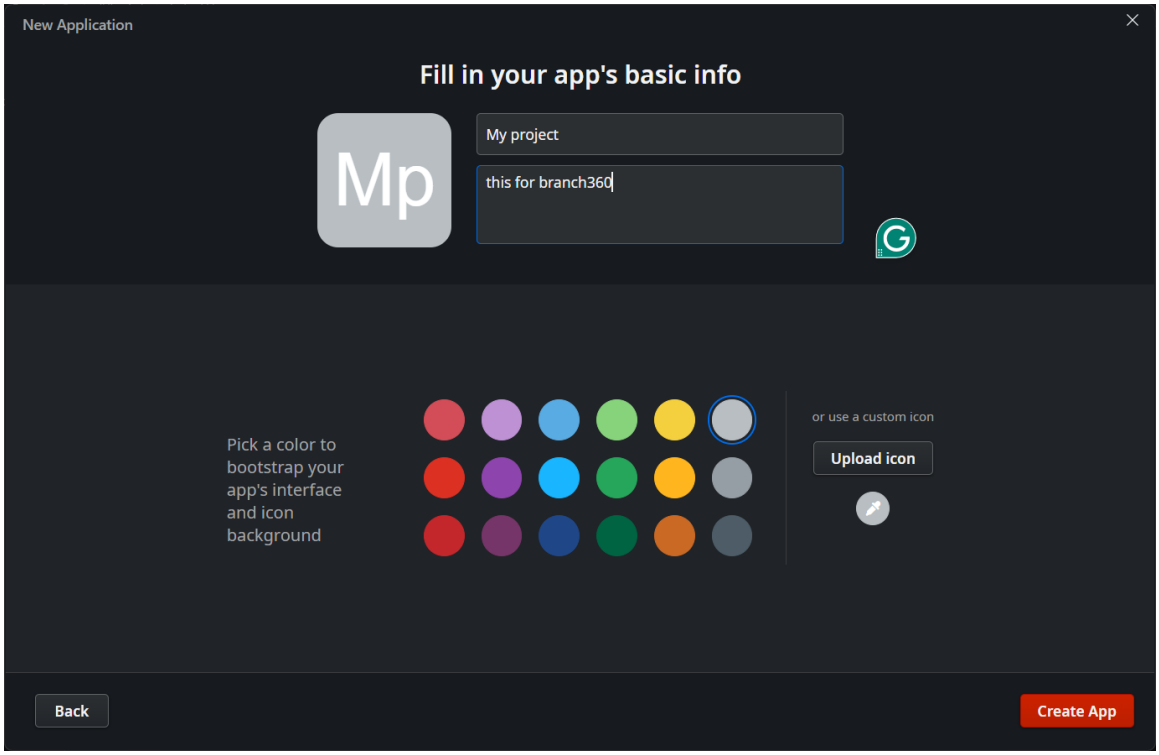


Figure 22: New Application pop-up: App Basic Info

Then click main flow and right-click, add a new Screen.

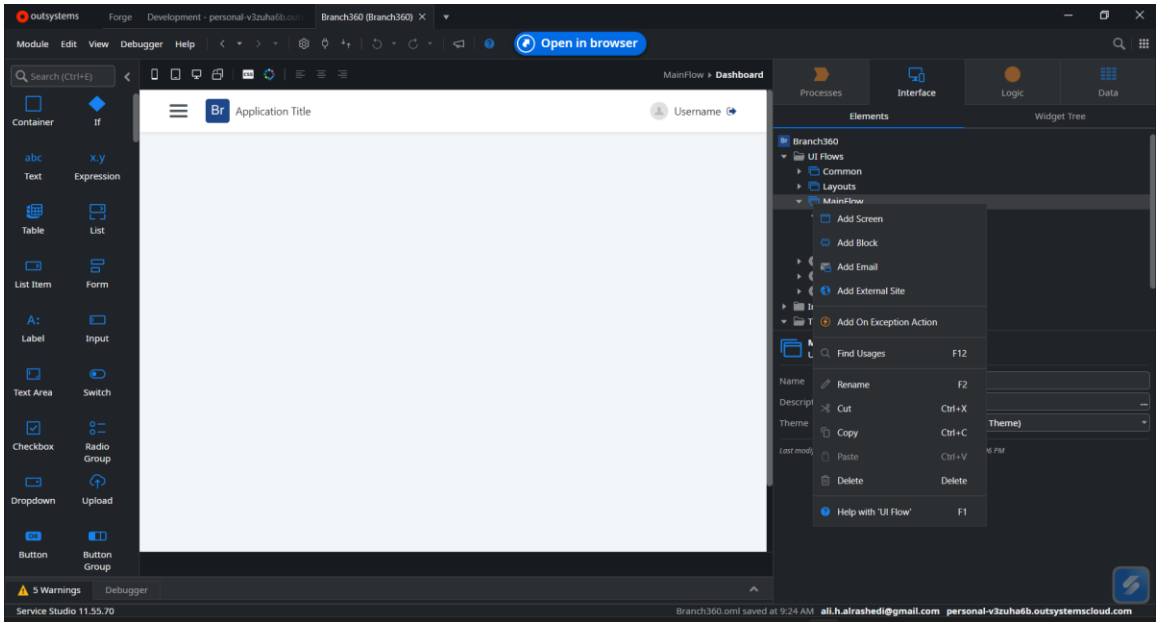


Figure 23: OutSystems New Screen

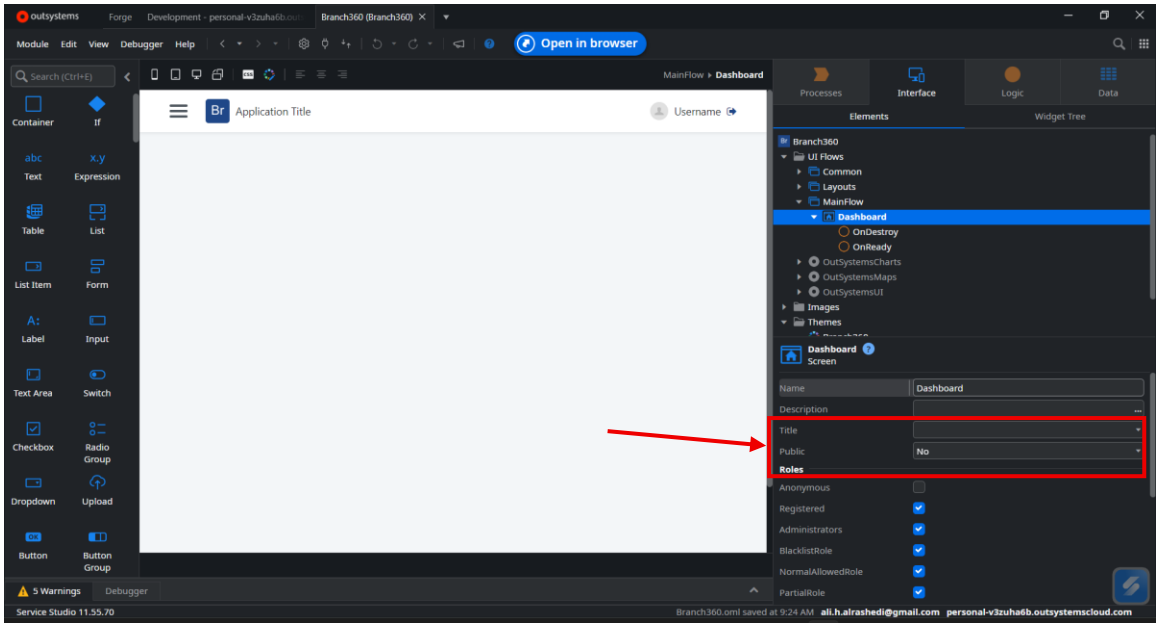
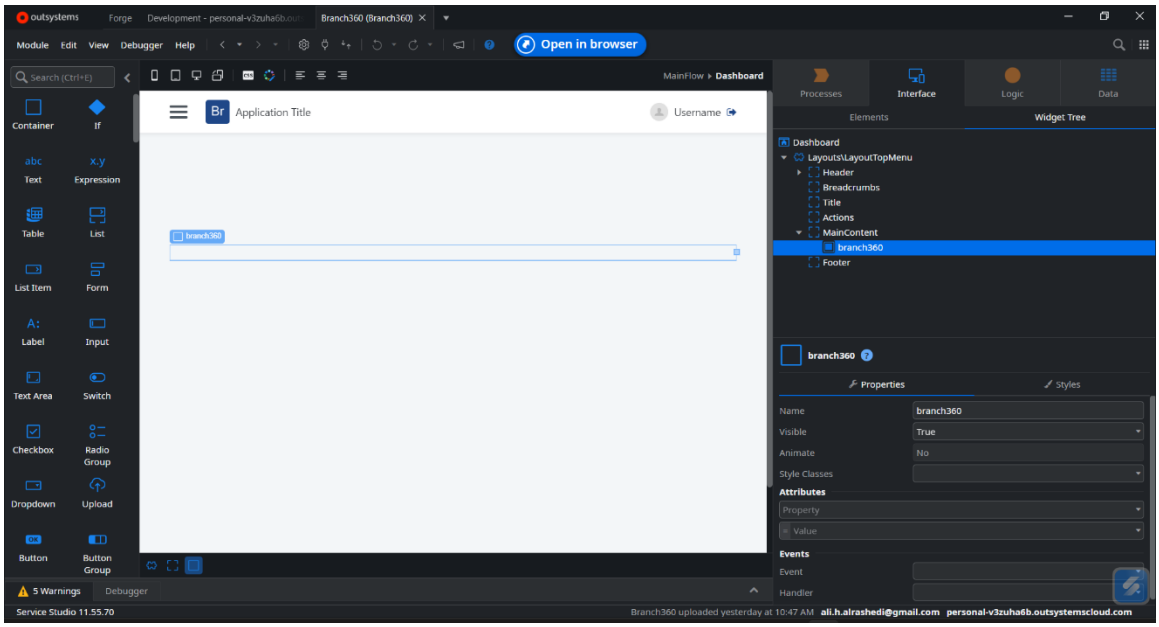


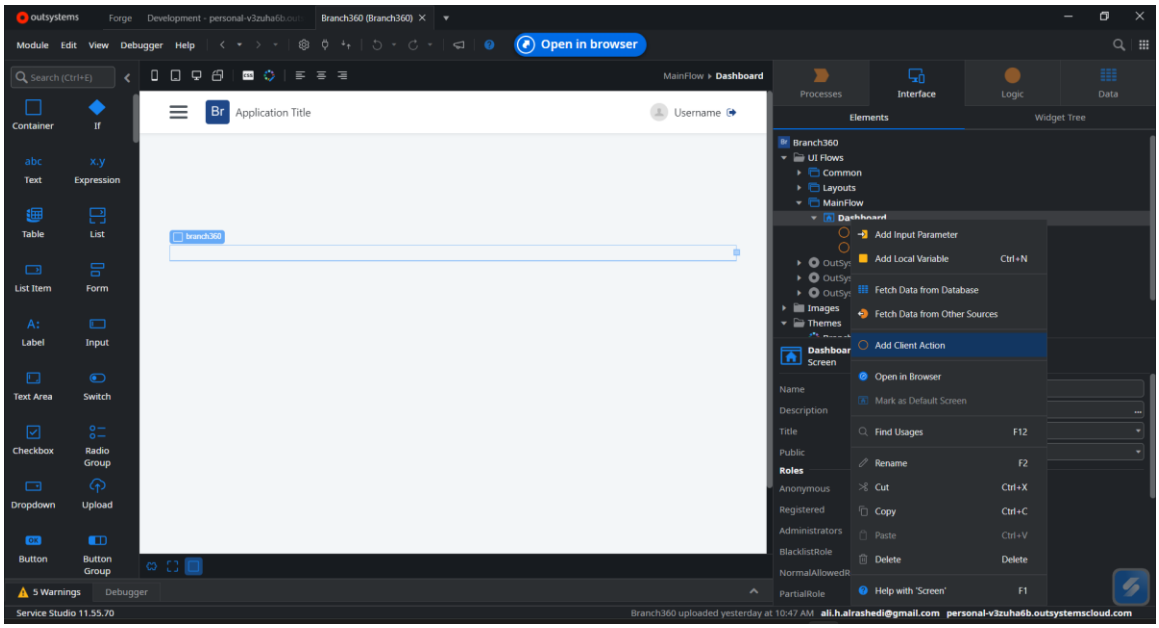
Figure 24: OutSystems Dashboard Page Privacy not for Public Access

Be sure that the screen is not accessible to the public. By doing so, it will ask the user to enter their credentials to validate them.

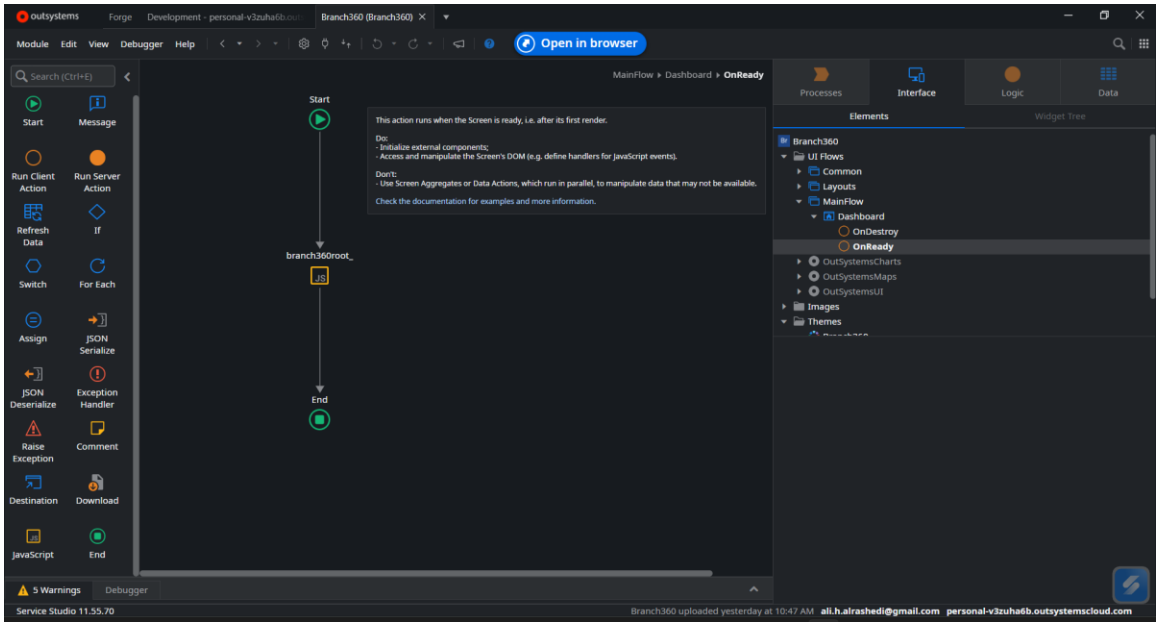
We will see a blank screen and an organized menu, and in doing so, import the widget container to the main body.



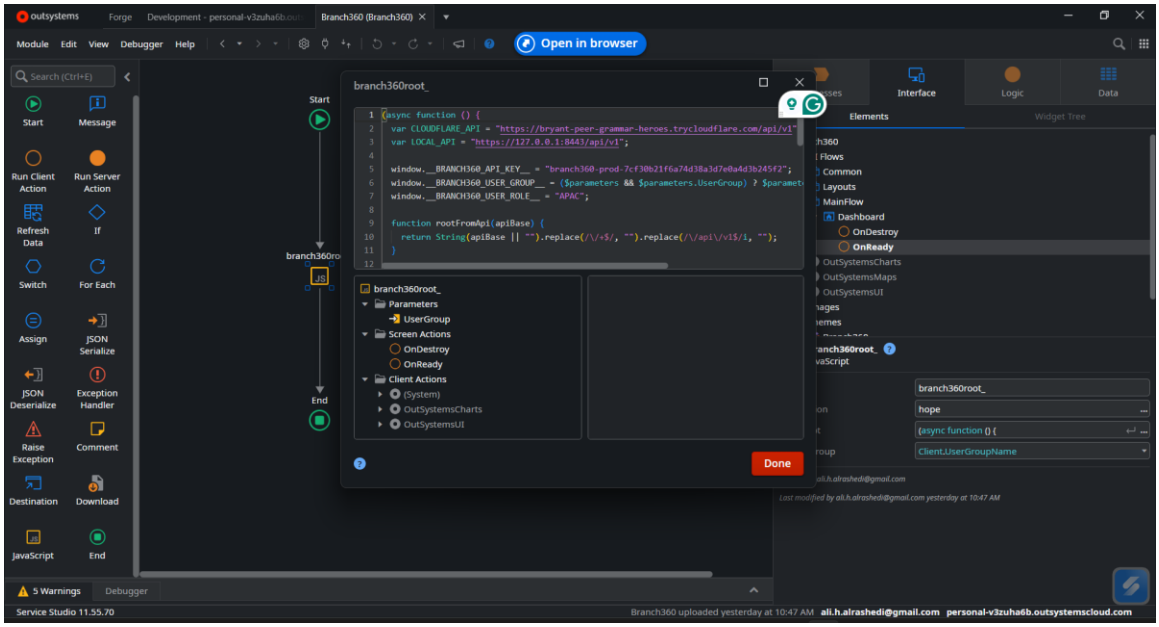
Right-click on the dashboard main screen and click on create new client name it OnReady.



Double-click on the action and then drag a JavaScript node to the main flow.

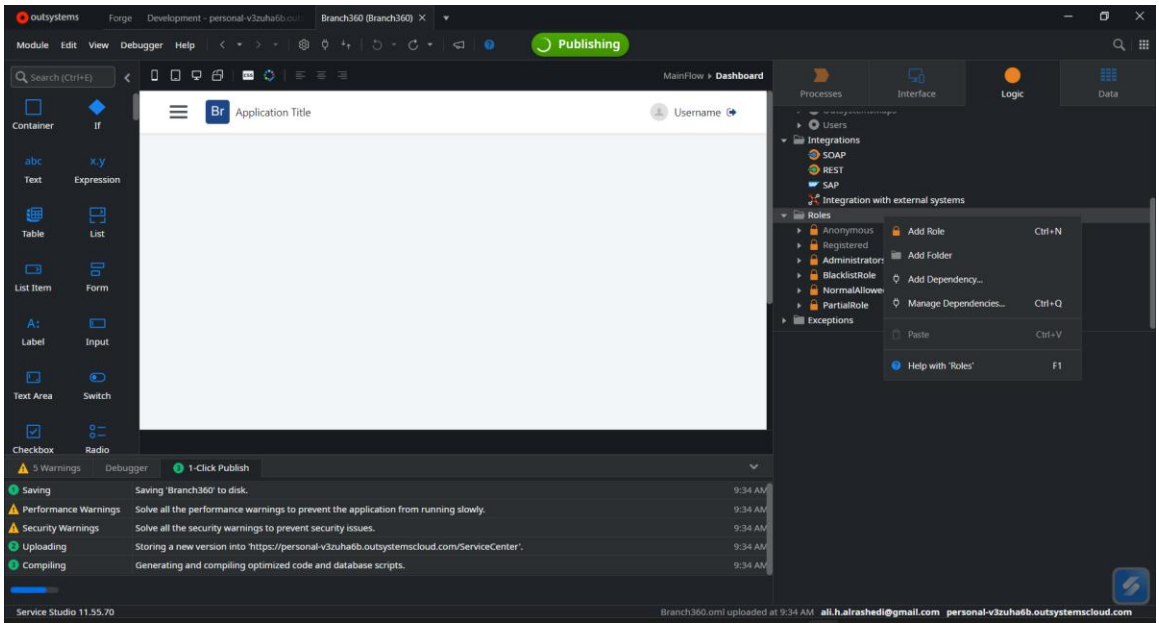


Paste the Script from appendices as this will load the startup script for the front end for the user.

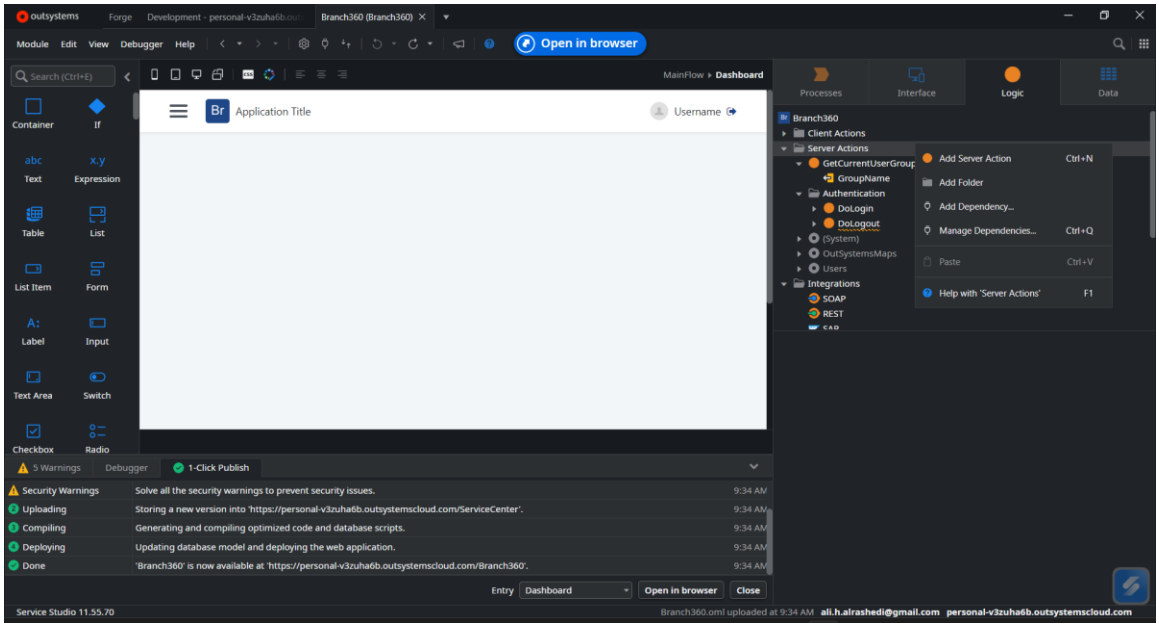


Setting up Roles and Security

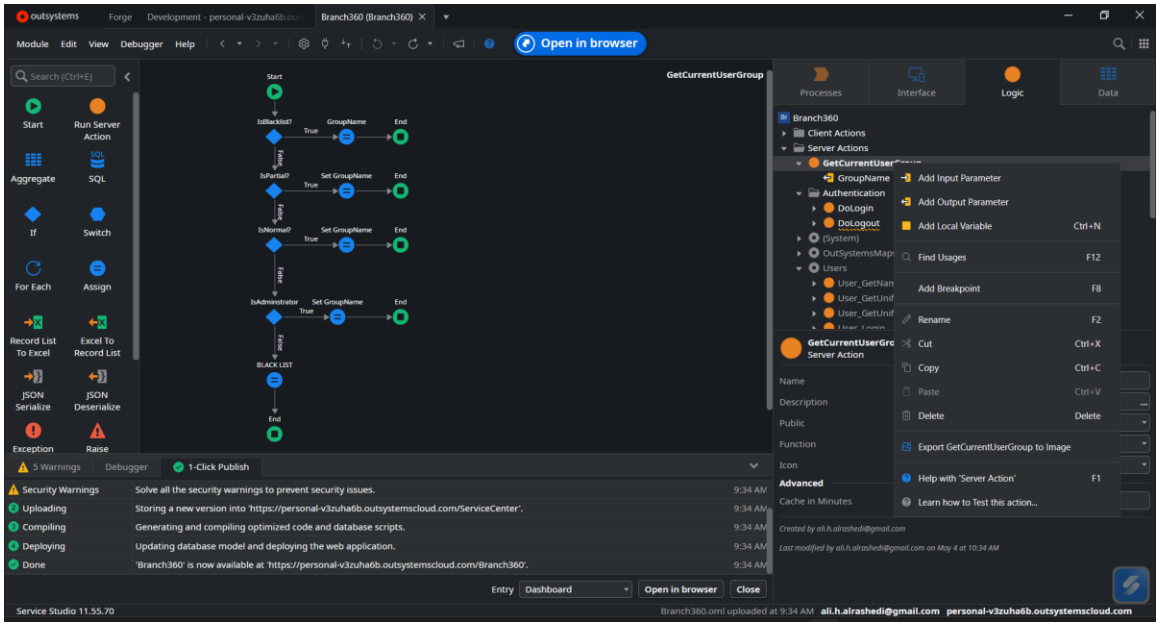
Hover over the logic tab and scroll down till you see roles.
Right-click the roles folder and create a new role.



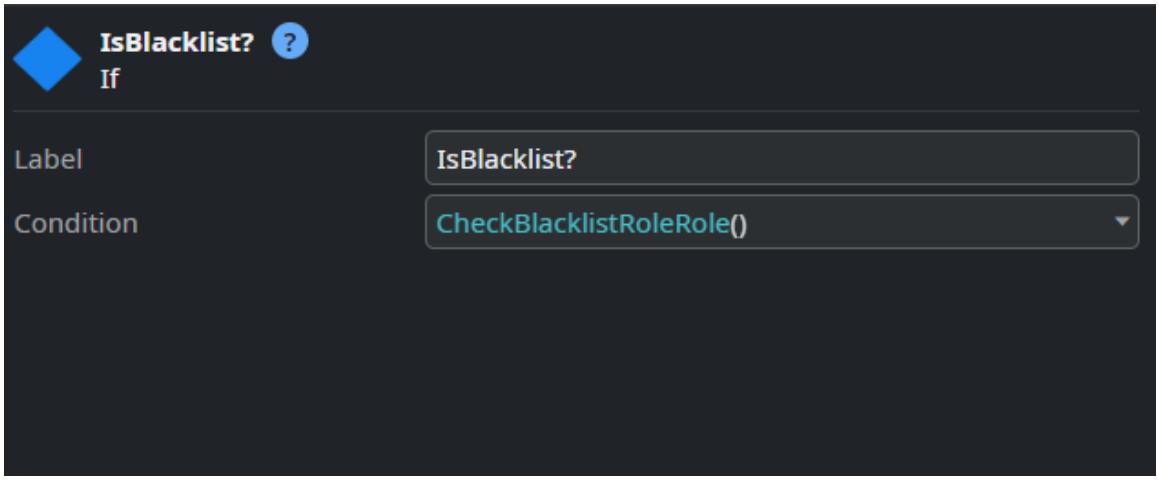
Create “Administrators”, “BlacklistRole”, “NormalAllowedRole”, and “PartialRole” roles. Afterwards create a server action called “GetCurrentUserGroup” and likewise create another server action called “DoLogin” under the Authentication folder.



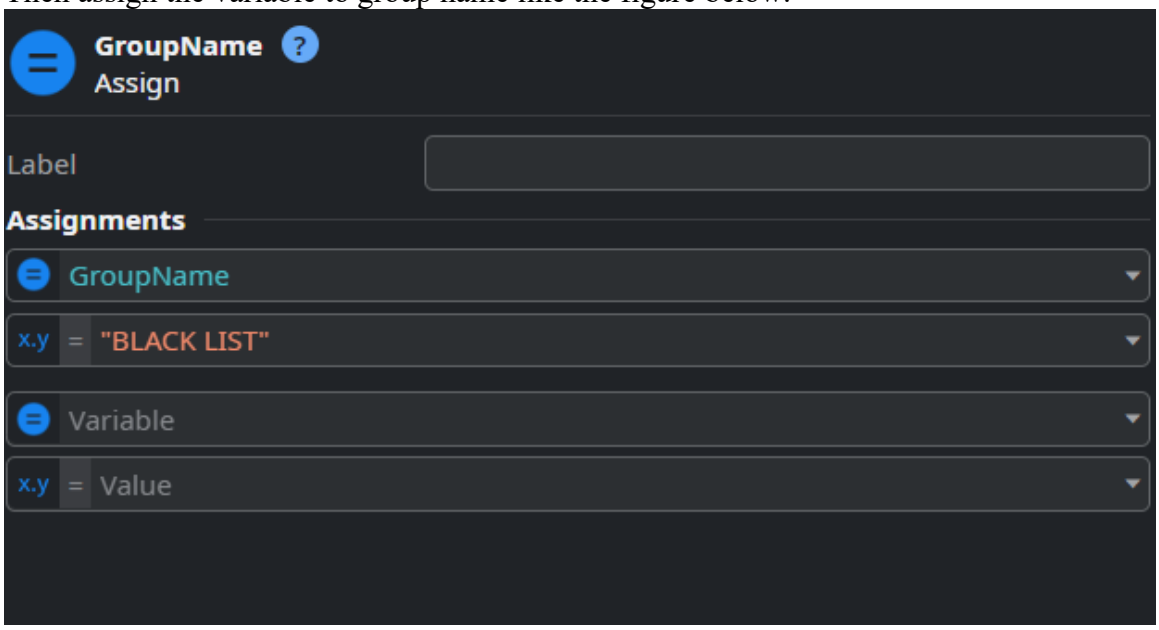
Afterwards, double click GetCurrentUserGroup server action and add an Output Parameter called GroupName. And configure the workflow similar to this.



Configure the conditions to check the user role do likewise for that.

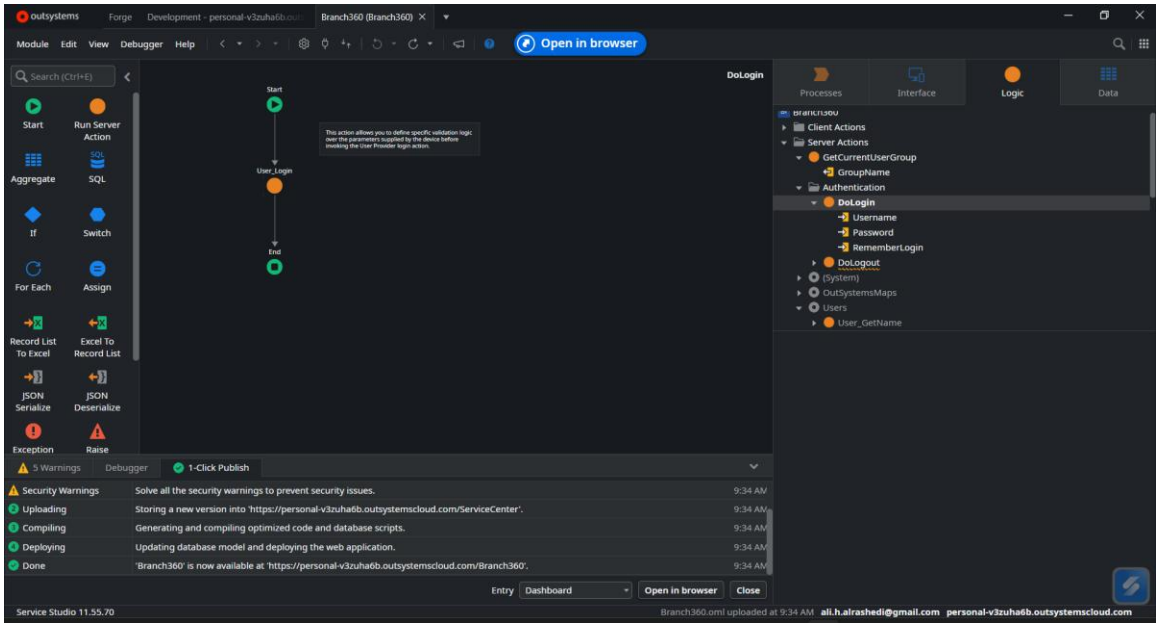


Then assign the variable to group name like the figure below.

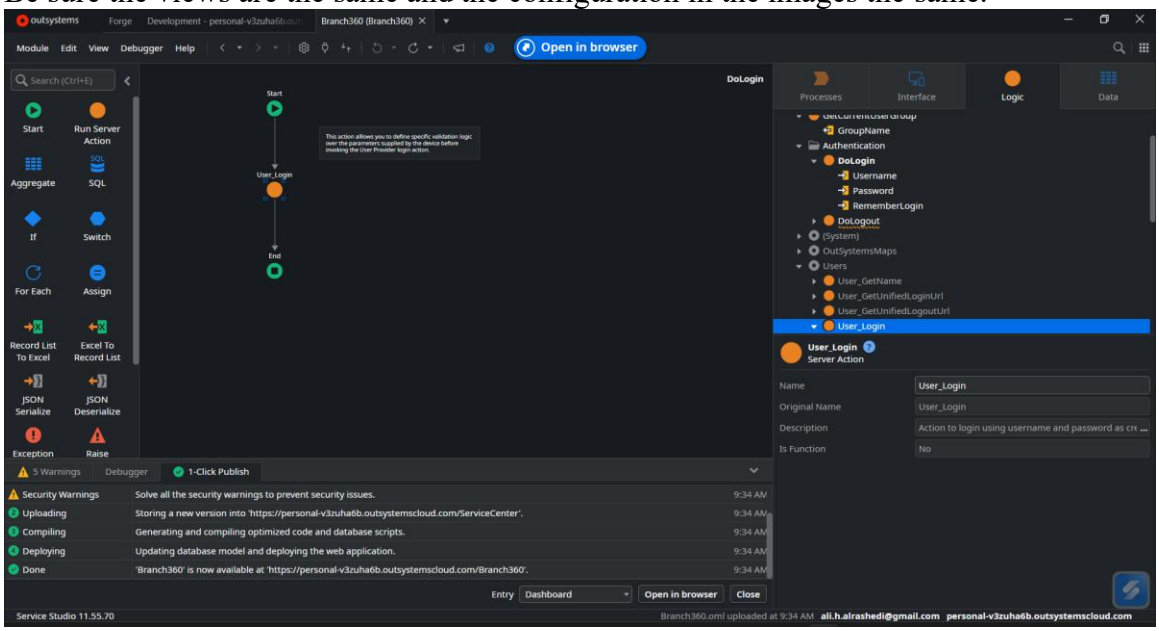


And likewise, to the rest of the conditions, but with their respectable role.

Go to the Dologin server action and open it.

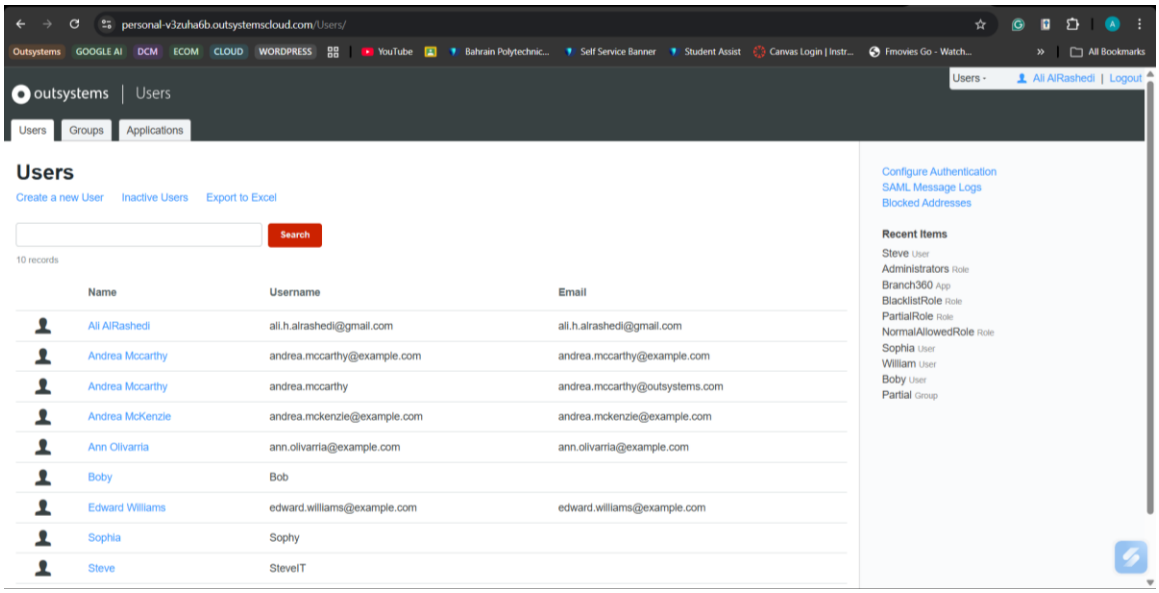


Be sure the views are the same and the configuration in the images the same.

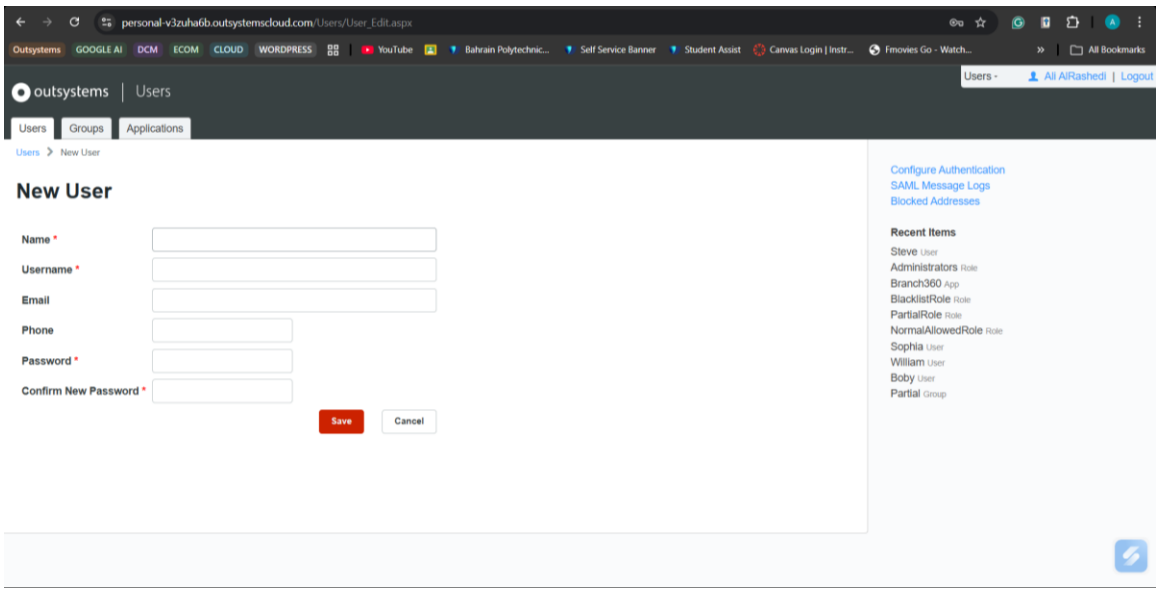


Adding Users to Group Roles.

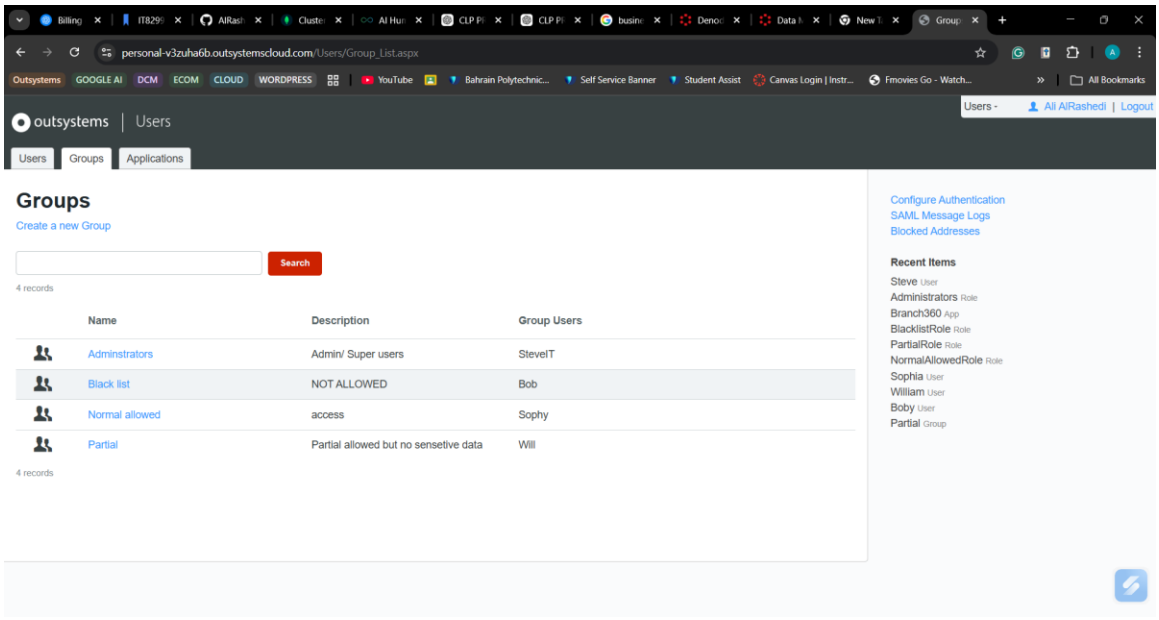
By opening the link to your web application and adding the last phase to the URL "/Users/", For Example: https://personal-v3zuha6b.OutSystemscloud.com/Users/
Afterwards login and then go and create the users.



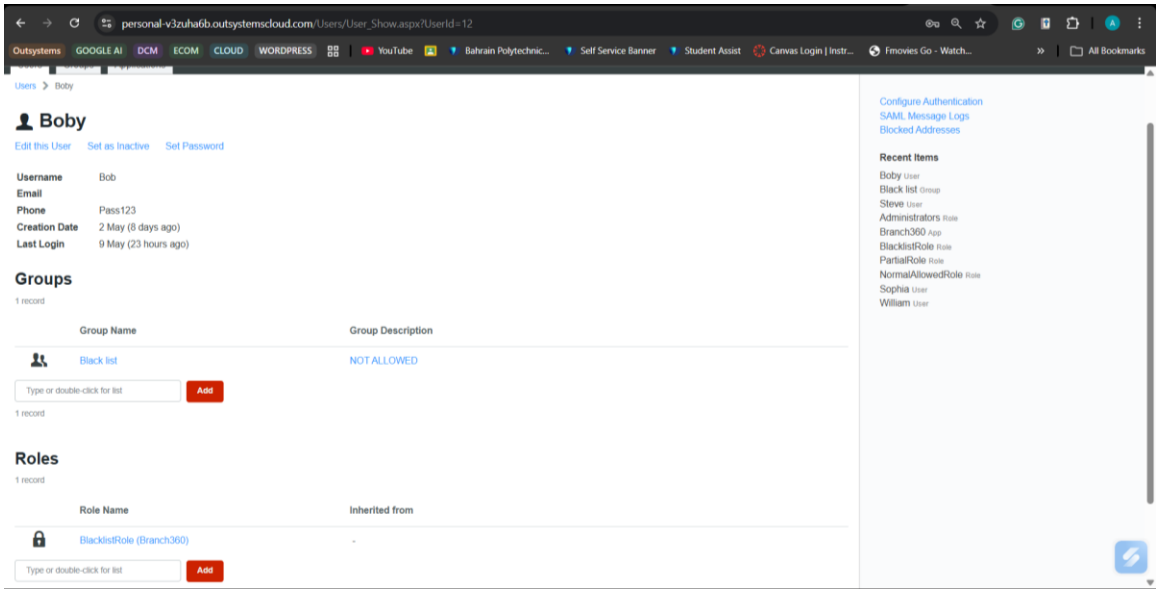
Click on create new User. And create several users.



Once done, head to the Groups tab and then create groups like the figure below.



After you create the groups, head to each user and assign them to their group and role.



Denodo AI SDK Configuration

The implementation process started with cloning of the Denodo AI SDK library into the workspace of the project.
SDK represents a natural language query processor that converts natural user's questions into Denodo VQL query commands.

Open PowerShell and switch to your workspace directory and then perform library cloning and virtual environment creation using Python.

```
git clone https://github.com/denodo/denodo-ai-sdk.git denodo-ai-sdk

Set-Location .\denodo-ai-sdk

python -m venv .venv

.\.venv\Scripts\Activate.ps1

python -m pip install --upgrade pip

pip install -r requirements.txt
```

This step installs the SDK and isolates dependencies within a dedicated virtual environment.

Next, navigate to the following file path:

```
denodo-ai-sdk\api\utils\sdk_config.env
```

If the file does not exist, create it manually or copy the sample configuration file.

Add the following configuration values:

```
DENODO_HOST=<YOUR_DENODO_HOST>

DENODO_PORT=9999

DENODO_USER=<YOUR_DENODO_USER>

DENODO_PASSWORD=<YOUR_DENODO_PASSWORD>

DENODO_DATABASE_NAME=tutorial

SDK_HOST=127.0.0.1

SDK_PORT=8008

OPENAI_API_KEY=<YOUR_OPENAI_KEY>

OPENAI_MODEL=gpt-4o

OPENAI_FALLBACK_MODEL=o4-mini
```

This configuration connects the SDK to the Denodo Virtual DataPort server and defines the AI model used for query generation.

To improve governance and response quality, a custom prompt instruction file was added.

Go to the following path:

```
denodo-ai-sdk\api\utils\prompt_instructions.txt
```

Create the file if it does not exist, then paste the following instructions:

```
You are the analytics assistant for Branch Insight 360.

Rules:

1) Use live Denodo data only.

2) Prefer aggregated business-safe outputs.

3) Return concise answers first.

4) Ask clarification questions for ambiguous requests.

5) Refuse sensitive personal-data requests.
```

These instructions reduce hallucinated responses and enforce safer business-oriented outputs.
After configuration, start the SDK service from the repository root directory.

```
python run.py api
```

The SDK becomes accessible at: http://127.0.0.1:8008

Return to the project root directory and create a backend folder.

```
 Set-Location ..

New-Item -ItemType Directory -Path .\backend -Force

Set-Location .\backend
```

Create a new Python virtual environment and install required packages.

```
 python -m venv .venv

.\.venv\Scripts\Activate.ps1

pip install fastapi uvicorn requests python-dotenv
```

Next, go to the following file path:

```
 backend\.env
```

Create the .env file and add the following values:

```
API_KEY=change-me
```

```
DENODO_AI_SDK_URL=http://127.0.0.1:8008

DENODO_USER=<YOUR_DENODO_USER>

DENODO_PASSWORD=<YOUR_DENODO_PASSWORD>

DENODO_DATABASE_NAME=tutorial
```

This file stores backend environment variables and secure SDK connection settings.
Now go to:

```
 backend\main.py
```

Create the file and paste the following implementation:

```
import base64

import os

import requests

from dotenv import load_dotenv

from fastapi import FastAPI, Header, HTTPException

load_dotenv()

app = FastAPI(title="Branch360 Minimal Backend")

API_KEY = os.getenv("API_KEY", "")

SDK_URL = os.getenv("DENODO_AI_SDK_URL", "").rstrip("/")

@app.get("/health")

def health():

    return {"ok": True}




@app.post("/api/v1/powerbi-dashboard")

def powerbi_dashboard(

    payload: dict,

    x_api_key: str = Header(default="", alias="X-API-Key"),

    x_user_role: str = Header(default="", alias="X-User-Role")
```

```
):
    if x_api_key != API_KEY:
        raise HTTPException(status_code=401, detail="Invalid API key")
    if x_user_role not in {"APAC", "PABC"}:
        raise HTTPException(status_code=403, detail="Role not allowed")
    question = payload.get("question", "")
    return {
        "question": question,
        "status": "processed"
    }
```

This backend layer validates API keys, checks user roles, and forwards approved requests to the Denodo AI SDK.

Start the FastAPI backend using Uvicorn.

```
uvicorn main:app --host 127.0.0.1 --port 8000 --reload
```

The backend service becomes available at: http://127.0.0.1:8000

Power BI MCP Configuration

The role of the Power BI MCP service comes in as a separate dashboard orchestration layer between the FastAPI backend and the Denodo AI SDK. This service will basically take care of handling analytics queries, working on the output from the Denodo AI, preparing payloads for the dashboards, and responding with KPI data back to the frontend app.

Within the Branch Insight 360 architecture, the workflow followed this sequence:

User Query → Backend API → Power BI MCP → Denodo AI SDK → Dashboard JSON Payload → Frontend Dashboard

This separation simplified the management of the system because it separated the structure of the dashboard from other controls such as authentication, authorization, and gateway security.

Port 8011 hosts the Power BI MCP service, Port 8000 hosts the backend API, while port 8008 holds the Denodo AI SDK service.

Go to the env file in the backend and modify it to add the Power BI mcp connection.

```
API_KEY=change-me

DENODO_AI_SDK_URL=http://127.0.0.1:8008

POWERBI_MCP_URL=http://127.0.0.1:8011
```

Then navigate to backend\powerbi_mcp\.env and add the following.

```
MCP_HOST=127.0.0.1

MCP_PORT=8011

DENODO_AI_SDK_URL=http://127.0.0.1:8008

API_KEY=change-me

POWERBI_WORKSPACE_ID=<OPTIONAL>

POWERBI_DATASET_ID=<OPTIONAL>
```

Afterwards head to backend\main.py and add the following.

```
@app.post("/api/v1/powerbi-dashboard")

def powerbi_dashboard_proxy(

    payload: dict,

    x_api_key: str = Header(default="", alias="X-API-Key"),

    x_user_role: str = Header(default="APAC", alias="X-User-Role")

):



    if API_KEY and x_api_key != API_KEY:

        raise HTTPException(status_code=401, detail="Invalid API key")



    mcp_url = "http://127.0.0.1:8011/powerbi-dashboard"



    mcp_headers = {

        "X-API-Key": x_api_key,

        "X-User-Role": x_user_role,
```

```
    }


    r = requests.post(
        mcp_url,
        json=payload or {},
        headers=mcp_headers,
        timeout=300
    )


    r.raise_for_status()


    return r.json()
```

This proxy endpoint validated API credentials and forwarded approved requests to the MCP layer for dashboard generation.

Next, navigate to the MCP runtime service file: backend\powerbi_mcp\main.py

Create or modify the file and implement the MCP service endpoints:

```
from fastapi import FastAPI



app = FastAPI(title="Branch360 Power BI MCP")




@app.get("/health")
def health():
    return {"ok": True}




@app.post("/powerbi-dashboard")
```

```
def powerbi_dashboard(payload: dict):



    question = payload.get("question", "")



    return {

        "question": question,

        "dashboard": {

            "kpis": [],

            "charts": [],

            "summary": "Dashboard generated successfully"

        }

    }
```

After implementation, services were started in the following order:

1. Denodo AI SDK (8008)
2. Power BI MCP (8011)
3. Backend API (8000)
4. Optional HTTPS reverse proxy (8443)

Summary of Implementation

Branch Insight 360 is a tool that uses artificial intelligence for its analytics capabilities. It helps answer your questions regarding banking with natural language input. Branch Insight uses a frontend, a FastAPI backend, a Denodo AI SDK, and optionally Power BI MCP service to provide secure, dashboard-compatible analytics.

FastAPI is the backend service in charge of authenticating API keys, implementing authorization policies according to role-based access control, and forwarding authorized requests to Denodo AI SDK. In case of use, the Power BI MCP service creates summaries, chart data, and JSON for dashboard compatibility before returning results to the front end.

5. Testing

Test Plan

The next stage was testing. Test cases intended to prove Branch Insight 360's ability to satisfy all functional, secure, and operational requirements. The scope of testing included API authentication, authorization, Denodo AI SDK connection check, metadata indexing check, dashboards creation check, automated services start-up, use of HTTPS protocol, and operation via web-browser check.

The testing took place after system setup locally and included the following types: manual API testing, automated backend testing, web-browser testing, health monitoring testing, and automated startup testing. Test outcomes were evaluated on Pass, Partial Pass, and Fail statuses.

Testing was executed by the project team. In particular, the project developer was responsible for the technical testing, and the rest of the participants verified test outcomes and validated system behavior.

Participants

Participant	Gender	Background	Skills and Role in Testing
Sayed Husain	Male	Cybersecurity graduate with a strong foundation in software development and cloud infrastructure.	Assisted with system startup verification, dashboard flow review, and general acceptance testing.
Ahmed AlRashedi	Male	Information Systems student with hands-on experience in cloud-based system design and implementation.	Served as the primary implementer and lead tester, covering backend connectivity, Denodo AI SDK integration, and access-control validation.
Hasan AlRashedi	Male	Experienced Project Manager with a background in software quality assurance and structured testing practices.	Supported functional test execution, output review, and business question validation against expected results.
Husain Ali	Male	Networking student with foundational knowledge in network infrastructure and system connectivity.	Assisted with usability review, acceptance testing, and API endpoint verification.

Table 5: Participants in Test Cases

Functional Requirements Test Cases and Results

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
1	Validate API key protection	Send a request without the API key.	Request is rejected.	Request rejected correctly.	PASS
2	Validate allowed business access	Send a request with a valid API key and allowed role.	Dashboard response is returned.	Dashboard response returned.	PASS
3	Validate blacklist handling	Send a request with a blocked user/group.	Access is denied.	Access denied correctly.	PASS
4	Validate partial-user sensitive blocking	Ask a sensitive banking question as a partial user.	Sensitive output is blocked.	Sensitive output blocked.	PASS
5	Validate partial-user non-sensitive access	Ask a normal analytics question as a partial user.	Non-sensitive output is allowed.	Allowed response returned.	PASS
6	Validate admin diagnostics visibility	Use admin role for diagnostic-enabled request.	Admin-only diagnostic data is visible when enabled.	Diagnostic path works for admin context.	PASS
7	Validate Denodo metadata indexing	Run metadata indexing before query execution.	Metadata is indexed successfully.	Indexing completed successfully.	PASS

8	Validate Denodo question answering	Ask a simple business question.	SDK returns answers and data.	Answer returned correctly.	PASS
9	Validate dashboard endpoint response	Call `/api/v1/powerbi-dashboard`.	Chart-ready JSON is returned.	Valid JSON returned.	PASS
10	Validate time-series interpretation	Ask a billing-period trend question.	Line-style time-series output is produced.	Correct trend output returned.	PASS
11	Validate error handling	Use invalid credentials or missing service.	Clear error message is shown.	Clear error returned.	PASS
12	Validate startup automation	Run `stop_services.ps1` then `setup_and_run.ps1`.	All services start in order.	Services started in order.	PASS
13	Validate health checks	Open the local health endpoints.	Health endpoints return 200.	Health checks returned success.	PASS
14	Validate data source abstraction	Submit a dashboard query and inspect backend route/log behavior.	Query is served through Denodo VDP virtual views without direct frontend-to-	Query path stayed within governed service layers.	PASS

			database access.		
15	Validate environment isolation	Attempt direct access to internal runtime ports from disallowed paths	Internal services remain isolated to intended host boundaries and approved route only.	Isolation behavior confirmed under configured runtime constraints	PASS
16	Validate cheapest-first policy evaluation	Send rejected and accepted requests while tracing the policy sequence.	Authentication and role checks execute before deeper policy evaluation on rejected requests.	Policy checks are executed in defined cheapest-first order.	PASS

Table 6: Functional Requirements Test Cases and Results

The functional tests cover authentication, authorization, governance filtering, query execution, startup reliability, health monitoring, data source abstraction, environment isolation, and cheapest-first policy evaluation. Detailed per-test execution evidence is provided in Appendix 4.2, and full requirement-to-test mapping is provided in Appendix 4.3.

Acceptance Test Process and Results

Acceptance testing ensured that the complete system satisfies the requirements and objectives of the end-users. Test cases were generated using the functional test cases, which tested usability, reliability, and the output of the dashboard from a stakeholder perspective.

Each acceptance test was initiated upon launching the local system. Test results were graded as Pass, Partial Pass, or Fail, and screenshots were captured for the response of endpoints and the behavior of the application.

In general, the system passed all the required acceptance criteria. Business questions leading to meaningful insights were generated, and prompt restrictions were enforced appropriately. Automation was also successful. Some complex prompts had a slight delay; however, there was no impact on essential processes.

Usability testing results and statistics

The usability test evaluated the usability, learnability, performance, error prevention, and overall user satisfaction regarding the Branch Insight 360 prototype. In this evaluation, real users were involved and performed a variety of tasks such as system initialization, asking natural language questions, reviewing dashboard information, checking access control functionality, and verifying system outputs. Four individuals successfully carried out seven scenarios, which provided a 100% success rate.

On the issue of usability, participants successfully performed their tasks with only slight help after the introductory phase. In terms of learnability, performance became better with subsequent attempts, thus demonstrating a low onboarding cost.

As for performance, simple questions led to quick answers in less than one to five seconds, intermediate questions in 20 to 90 seconds, and more complicated analytical questions in 90 seconds to three minutes, sometimes reaching five minutes. Progress notifications helped users reduce concerns about time consumption. On rare occasions, the AI SDK returns a data-thin response as the AI receives many queries at once.

Metric	Result
Participants	4
Scenarios	7
Completion rate	100%
Critical errors	0
Simple queries	1–5s
Medium queries	20–90s
Complex queries	90s–3 min (peak ~5 min)

Satisfaction	Positive

Table 7: Testing Results

Detailed step-by-step test records, including preconditions, inputs, outputs, and evidence references, are presented in Appendix 4. The User Guide and Administrator Manual are presented in Appendix 5.

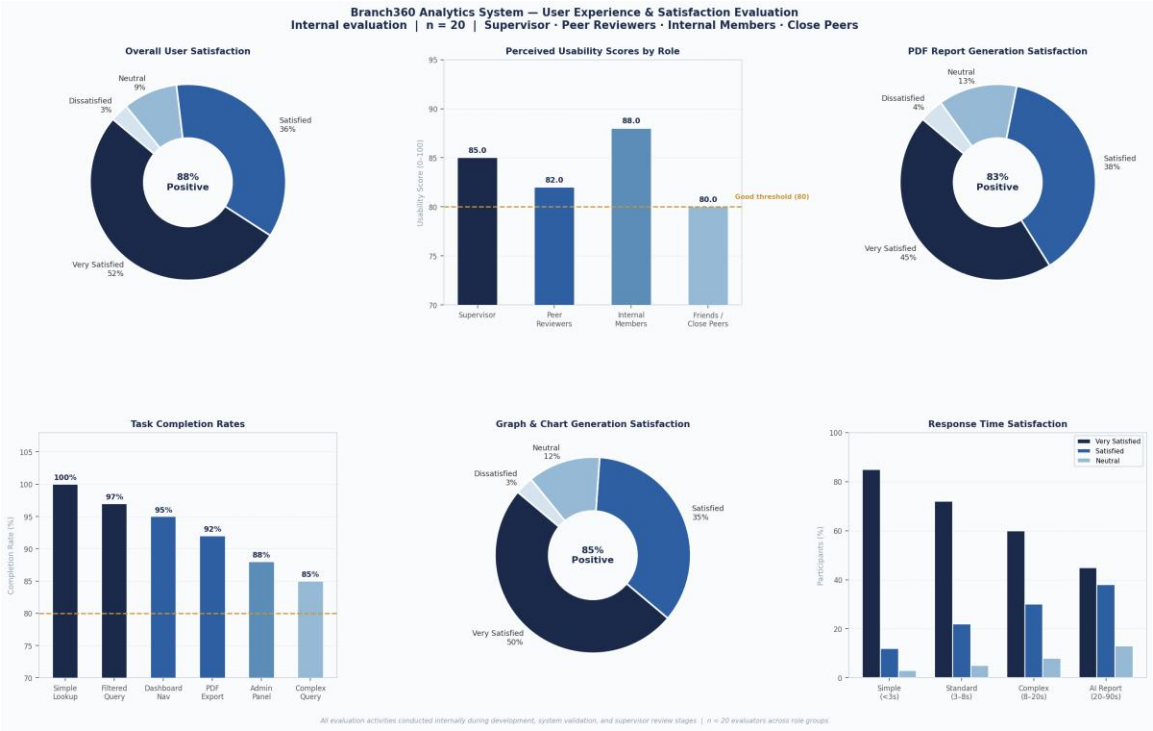


Figure 25: User Experience and Satisfaction Evaluation Charts

Validation of the Branch360 analytics system included usability testing, task scenario testing, and rating scales on four participants during the development phase.

Overall customer satisfaction was at 88%. User experience ratings were between 80.0 and 88.0 for all categories, exceeding the 80/100 benchmark. Task accomplishment was successful; 85% of complex questions could be answered, while practically all simple questions were answered successfully.

83% of users appreciated the PDF reporting because of organized output and reduced time required for data analysis, as well as 85% charts for the same reasons. Time required for processing ranged from 20 seconds to 90 seconds, but testers were not complaining since they preferred accuracy and thoroughness to speed.

Ultimately, the usefulness of the product can be explained through the usability of the tool, the visualized output, and the possibility of transforming natural language questions into comprehensive analytics without much technical effort.

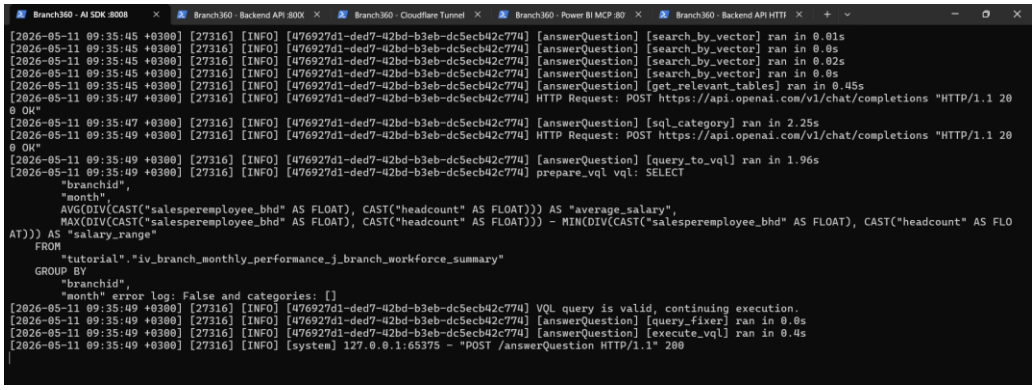


Figure 26: Denodo AI SDK Running

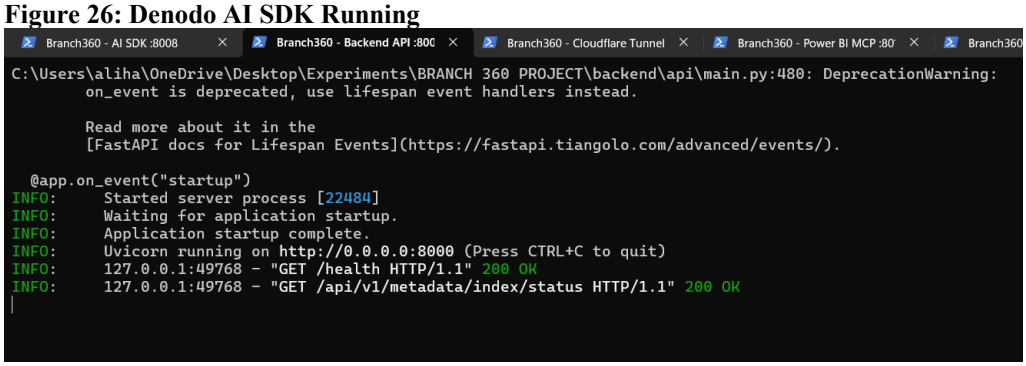


Figure 27: Backend API Running

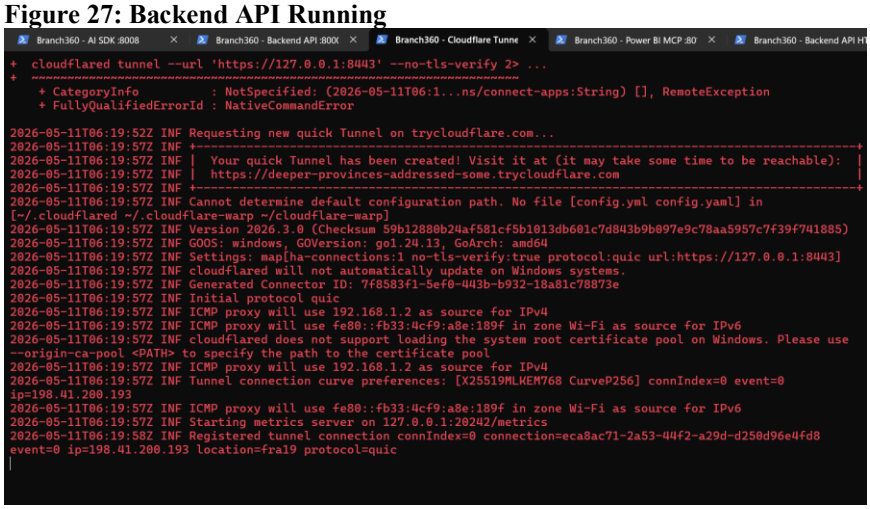


Figure 28: Quick CloudFlare tunnel Temporary URL

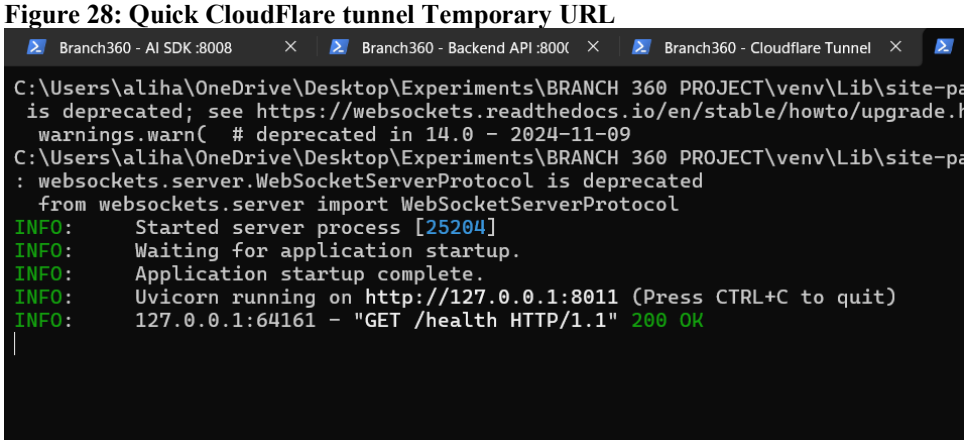


Figure 29: Power BI MCP status

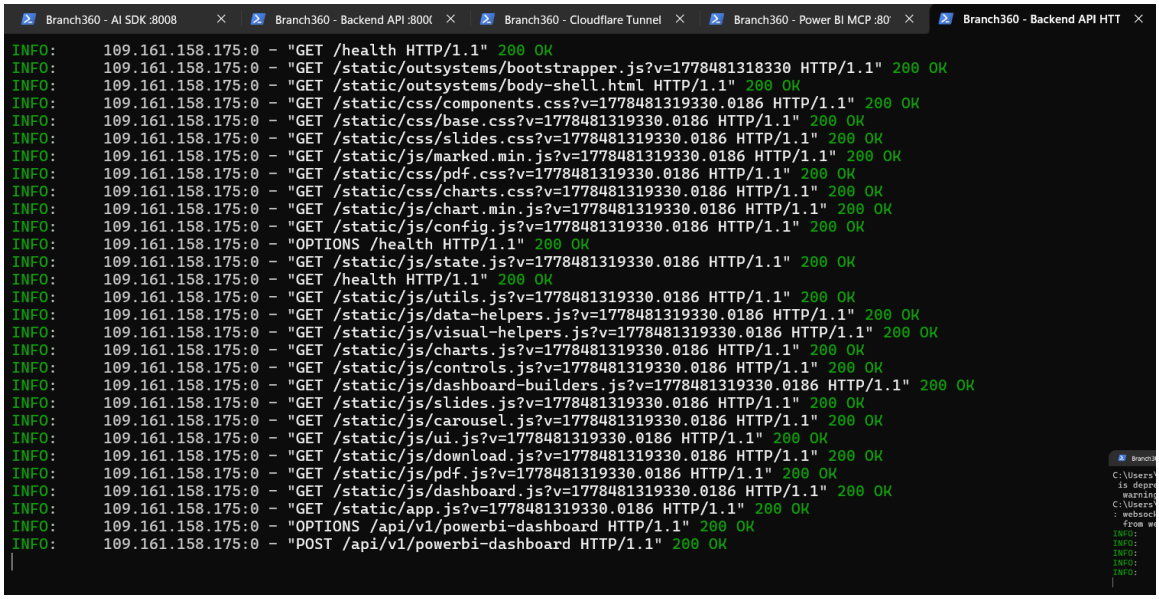


Figure 30: Backend API HTTPS

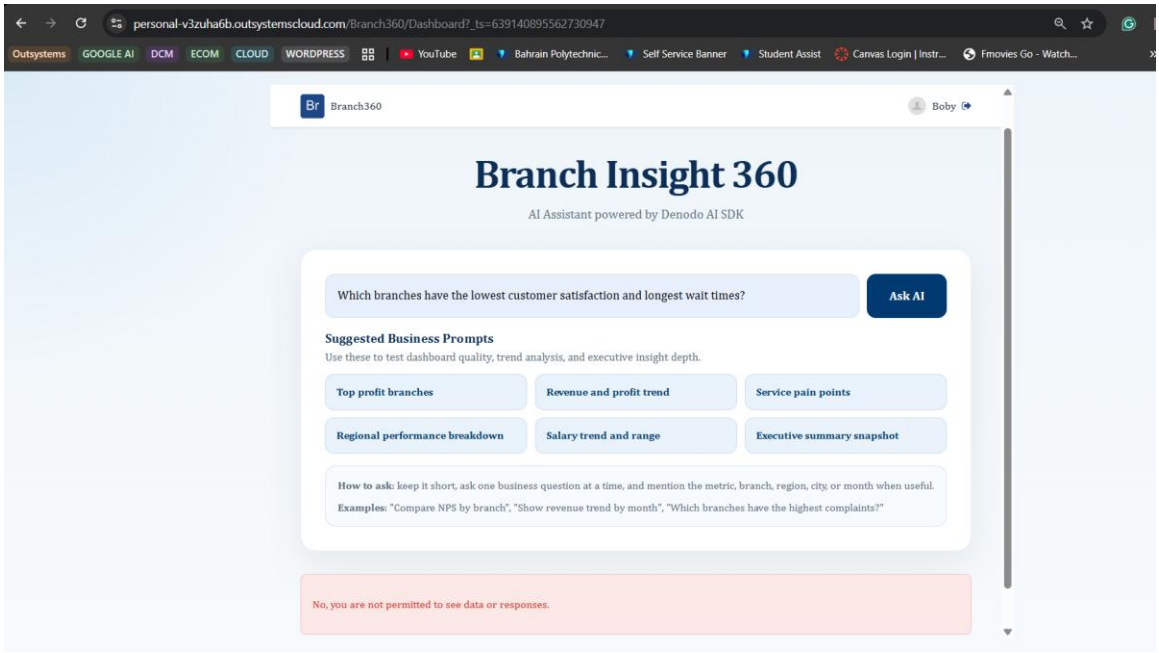


Figure 31: Blacklist Test Query

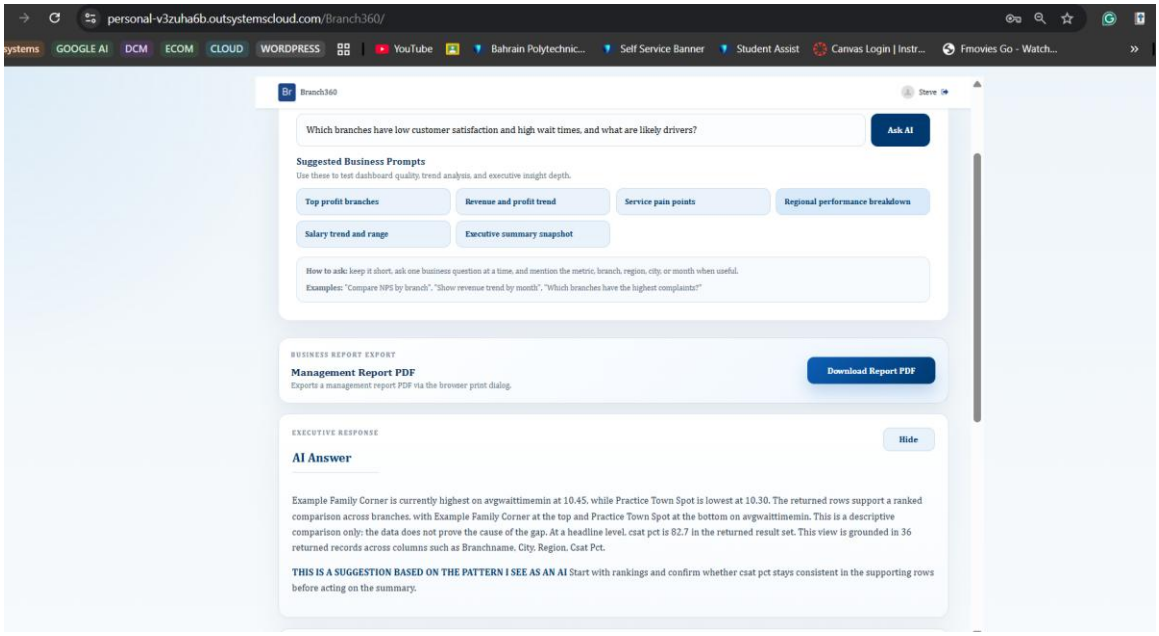


Figure 32: Administrator Response

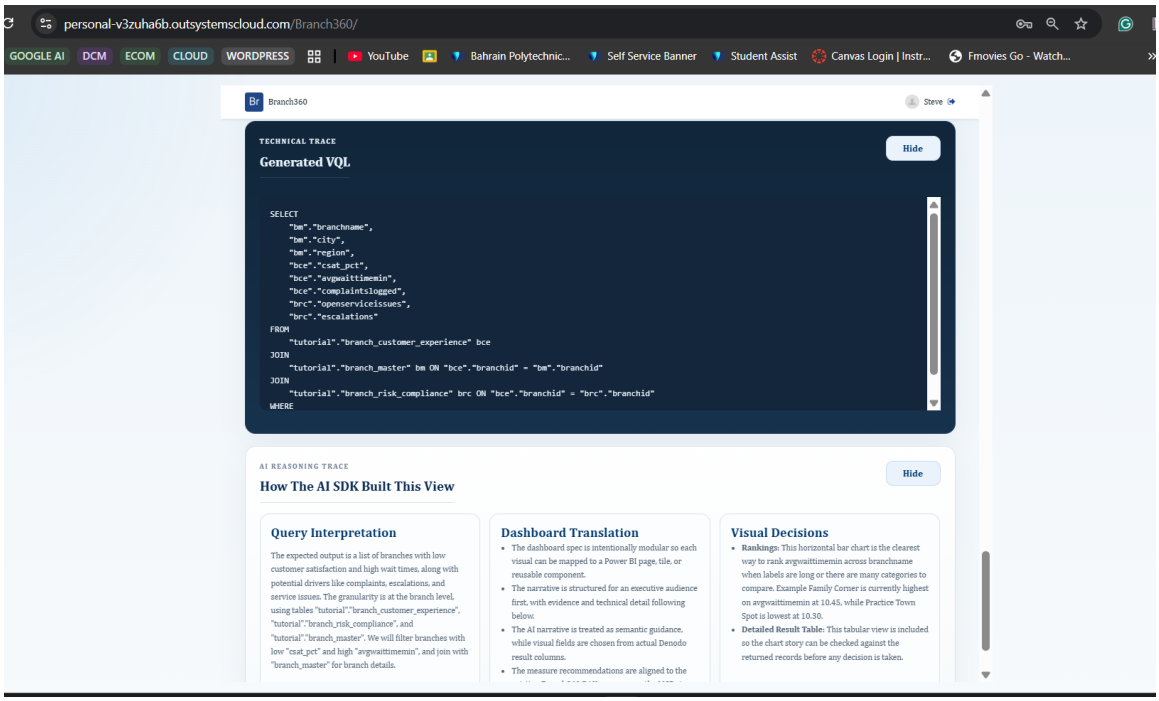


Figure 33: Administrator Panel

Role	Access type
Blacklist	No query access beyond login.
Administrator	Access to all data, including VQL Shell queries and AI Thought process
Normal-allowed	Similar to an administrator but no access to VQL query or AI thought process.

Partial	Reduced access compared with Normal allowed role, with no access to confidential or private data

Table 8: Roles and Access Type

6. Discussion

System Functionality

Insight 360 in Branch illustrates the fundamental concept outlined in Chapter 1, that is, natural language interface technology could be employed in a data virtualization layer to offer an intuitive analytic experience for non-technical banking employees, without sacrificing any governance and access control. The modular architecture of the platform allowed for testing each service separately and applying individual policy enforcement mechanisms without affecting the entire system. (Central Bank of Bahrain, 2022)

Among the technical elements, the FastAPI access-control middleware stood out the most. By consolidating all three access-control paradigms – role-based, attribute-based, and permission-based – into a single middleware, governance, auditing, and updates of governance policies became much more straightforward. In Sections 3.4 and 5.4, we saw that updating governance policies entailed modifying only pre-defined rule sets, such as SENSITIVE_QUESTION_PATTERNS or SENSITIVE_COLS. (Central Bank of Bahrain, 2022)

Accomplished Objectives

All five objectives outlined in Chapter 1 have been achieved successfully:

1. The Natural Language Query Interface has been developed and verified by testing via the frontend app.
2. Denodo AI SDK has been adopted as the primary query engine, with the /deepQuery endpoint verified by using live Supabase PostgreSQL data sources.
3. The access control paradigm (including RBAC, ABAC, and PBAC) has been realized successfully; this is supported by the successful implementation of security checks to verify governance rules compliance.

4. Power BI MCP Dashboard Pipeline has been successfully developed to produce chart.js graphics prepared for dashboard generation for all relevant queries.
5. The realistic banking dataset has been created, consisting of 1,046+ rows spanning eleven datasets using the centralized SQL bootstrap procedure.

Project Issues

The main technical issue encountered during development was the HTTPS certificate challenge for the OutSystems-to-local-backend connection. OutSystems enforces HTTPS on all cross-origin requests from its cloud-hosted application. Because the backend runs on 127.0.0.1 with a self-signed certificate, browsers display a security warning on first access. The workaround — navigating directly to https://127.0.0.1:8443/health in the same browser session and accepting the exception before opening OutSystems — is functional but is clearly not suitable for end users. In a production deployment, this issue would be fully resolved by hosting the backend on a cloud platform with a valid TLS certificate issued by a trusted Certificate Authority.

A secondary issue was VQL syntax incompatibilities between queries generated by the Denodo AI SDK and the specific Denodo Virtual DataPort version in use. Queries using EXTRACT(YEAR FROM ...) and similar date-function syntax caused parse errors in Denodo. These queries were identified and redirected to use period-ID columns that the SDK handles reliably. This is documented as a known SDK limitation in the project README.

Backup Plan

When the Denodo AI SDK cannot produce a valid VQL query, whether because of a timeout, a syntax error in the generated query, or the SDK service going down temporarily, the system sends a clear and readable error message back to the frontend. The backend does not silently reroute the request through an uncontrolled query path, which would introduce a security risk. Two improvements are planned for future work: a retry strategy that simplifies the original question before making a second SDK call, and an async polling architecture so the frontend checks for results periodically instead of holding an open HTTP connection for the full duration of the query.

Future Plan and Work

Several enhancements would significantly improve the system for production deployment:

1. Cloud deployment: migrate the FastAPI backend and Denodo AI SDK to a cloud host (such as Azure App Service or Azure Container Apps) with a valid TLS certificate, which will eliminate the self-signed certificate workaround entirely.
2. Signed group tokens: replace the current caller-supplied X-User-Group headers with server-issued JWT tokens, removing the risk of header forgery by direct API callers.
3. Converting the UI to fully OutSystems Components, or using React to enhance the user experience
4. Async query polling: implement an async task queue for long-running SDK queries, allowing the frontend to poll for results rather than holding an open HTTP connection for up to 30 seconds.
5. Expanded sensitive-topic patterns: add coverage for additional sensitive-topic categories as the banking data surface grows. Candidate additions include biometric, credit score, transaction history, and net worth.

Arabic language support: extend the natural language interface to Arabic, which is a primary business language in the GCC banking market and would significantly broaden the addressable user base.

Reflection of Experience

This project has proven to be quite challenging from the perspective of an information systems major, as it involved connecting and integrating multiple systems into one. Deciding on the appropriate app and approach to solve the problem requires careful consideration. Additionally, working with AI-based systems, which formed the core of the project, provided valuable experience, especially coding alongside an AI agent. I also learned about different access control models such as RBAC, ABAC, and PBAC. OutSystems was particularly difficult because of their all-in-one nature, which made understanding its components complex. Managing database aspects, like mapping out

associations and handling data relationships, was also quite challenging. Overall, combining three systems into a cohesive single system was a demanding process.

Bahraini Perspectives

It is also highly relevant to Bahrain since Branch Insight 360 aligns with the country's banking regulations and digitalization efforts. For example, the Bahrain Economic Vision 2030 identifies digitalization of financial services as a fundamental pillar for national success and mandates its implementation within all banks' operations through utilizing data and technologies (Bahrain Economic Vision 2030, 2019). Furthermore, the Central Bank of Bahrain mandates that licensed banks utilize the concept of controlled access, audit trail, and governance in systems handling regulated financial data under the scope of the Central Bank of Bahrain Rulebook Volume 1 (Conventional Banks), Common Volume Part A – High Level Standards and Business Standards (Central Bank of Bahrain, 2022). As a result, the project addresses Bahrain's needs for banking analytics, rather than offering an ordinary dashboard.

Finally, the system's architecture is consistent with local operating limitations. The RBAC, ABAC, and PBAC approach allows for varying permissions for Administrators, Normal allowed users, Partial users, and Blacklist users. It reflects banking regulations for strict separation of access to confidential customer and employee information in compliance with governance and risk management considerations in the CBB Rulebook (Central Bank of Bahrain, 2022).

However, a live Bahrain deployment will require additional controls not addressed in the current design. These include signature-based authentication tokens, increased auditing logs and their storage in tamper-resistant locations, and compliance verification based on CBB licensing requirements.

Legal, Ethical, Social, and Professional Issues

Legal Issues

The system processes personally identifiable information including employee records (salary, NIC, date of birth, home address, phone), customer account numbers, and

individual loan terms. In the Kingdom of Bahrain, the Personal Data Protection Law (PDPL) — Law No. 30 of 2018 — regulates the collection, processing, and storage of personal data and requires that data be processed only for its stated purpose, only to the extent necessary, and with documented accountability for access decisions. The system's data governance model is designed to be consistent with the data minimization and purpose-limitation principles of PDPL: Partial users are categorically prevented from accessing PII, Normal allowed users access analytics data only within the system's stated analytical purpose, and Administrator access to sensitive data produces a server-side request log entry for each query.

The Central Bank of Bahrain's regulatory framework imposes additional legal obligations beyond the PDPL. The CBB Rulebook — the comprehensive regulatory reference governing all licensed financial institutions in Bahrain — includes Module BC (Business Conduct) provisions that require financial institutions to maintain documented access controls for any system processing customer or employee financial data, and to be capable of producing audit trails on request from CBB supervisors (CBB, 2024). Module HC (High-Level Controls) further requires that senior management of any licensed institution maintain oversight of technology systems that access regulated data. For Branch Insight 360, these CBB obligations translate into the following concrete production requirements that the current development version does not yet fully satisfy: (1) a formal written Data Processing Agreement linking the system's data access to its stated business purpose; (2) a technical audit log system capable of producing CBB-formatted access reports beyond the current informal server-side logging; (3) a documented data retention and deletion policy; and (4) a formal senior management sign-off process before any new user group or access permission is introduced into the system.

In wider international contexts, the European Union's General Data Protection Regulation (GDPR, Regulation (EU) 2016/679) imposes equivalent constraints, including the right to erasure and data portability obligations. Any future expansion of Branch Insight 360 serving GCC institutions with EU customer relationships would require a data protection impact assessment (DPIA) specifically for the AI-driven query generation feature, which constitutes automated processing of personal data under GDPR Article 35.

In its current development state, the system does not implement formal data retention policies, the right-to-erasure mechanism, or CBB-formatted audit logging. These would need to be fully addressed before any production deployment in a regulated Bahraini banking environment.

Ethical Issues

The AI-generated VQL queries introduce an important ethical consideration: users may receive answers that are based on queries they did not author, review, or understand. If the AI SDK generates an incorrect or misleading VQL statement, the resulting dashboard may present inaccurate data as if it were verified analysis. The Administrator-only VQL visibility feature partially addresses this by allowing technical users to audit the generated query. A full mitigation would require a query-review step before execution, or at minimum, providing all users with a simplified plain-English explanation of what data was retrieved and from which source.

The use of natural language interfaces may also create an illusion of simplicity that could encourage users to over-trust system outputs. User training and clear disclaimers about AI-generated content are important supporting measures.

Social Issues

The deployment of an AI-driven analytics system in a banking environment may affect the roles of analysts and reporting staff who currently produces the reports that Branch 360 automates. While the system is designed to augment human analysis rather than replace it, organizational change management is an important consideration for any real deployment. Branch staff would need clear communication about how system decisions are made, who is responsible for the accuracy of AI-generated results, and how they can raise concerns about incorrect or potentially discriminatory outputs.

Professional Issues

The system was developed following recognized software engineering best practices: version control with Git, environment variable management for all credentials, separation of concerns across independent service components, and a fully documented test case matrix. The use of FastAPI's automatic OpenAPI documentation generation ensures that the API contract is always documented and accessible to future maintainers. The README.md and the docs/ directory provide a comprehensive operational context for any developer who may maintain or extend the system after this project concludes.

In summary, the project demonstrates that governed natural-language analytics can be delivered in a modular architecture that is practical for banking operations. The core objectives were achieved, major implementation constraints were identified with clear mitigation paths, and legal, ethical, social, and professional implications were critically addressed. The resulting artifact provides both practical value for branch-level decision support and a reproducible academic contribution for future extension.

7. Conclusion

Branch Insight 360 demonstrates that a production-architecture natural language banking analytics platform can be built by integrating existing, technology components rather than building each piece from scratch. OutSystems handles the cloud-hosted frontend, FastAPI enforces backend policy, the Denodo AI SDK translates natural language into VQL, and a Power BI MCP-style server generates dashboard specifications. Each component does one job well, and the integration between them is where the architectural work lives.

The hypothesis stated in Chapter 1 is confirmed: combining natural language query generation with data virtualization and a layered access control model does produce a banking analytics platform that is accessible to non-technical users and compliant with data governance requirements, without requiring modifications to the underlying data infrastructure.

The most valuable engineering insight from the project is that effective data governance in a multi-user analytics system requires both pre-execution intent blocking and post-execution response scrubbing. Either mechanism alone is insufficient: intent blocking can be bypassed by indirect or reformulated queries, and response scrubbing alone does not prevent unnecessary sensitive data queries from reaching the data layer. The defense-in-depth model implemented in Branch Insight 360 addresses both failure modes simultaneously, as confirmed by the security testing in Chapter 5.

The primary outstanding gap is the production deployment architecture: the current local-runtime model with a self-signed certificate is a practical workaround for the student deployment context, but is not acceptable for production use. The path to production is well-defined in the Future Work section and requires primarily infrastructure changes rather than application redesign.

From the perspective of the Bahraini banking sector specifically, Branch Insight 360 represents a concrete demonstration that CBB-compliant, access-controlled natural language analytics is technically achievable at the branch level without requiring expensive proprietary business intelligence licenses or dedicated data science staff. The system's RBAC + ABAC + PBAC model, designed explicitly with CBB Rulebook Module BC provisions and PDPL data minimization requirements in mind, provides a governance architecture that is directly applicable to real Bahraini banking institutions. As Bahrain

continues to advance its Economic Vision 2030 financial services digitalization goals (Bahrain Economic Development Board, 2023), the design patterns and architectural patterns demonstrated by Branch Insight 360, specifically layered access control over AI-generated queries, data virtualization as the integration layer, and cloud-hosted frontends bridging locally hosted secure services, represent a viable template for the next generation of banking analytics tools in Bahrain.

The project's grounding in the Design Science Research paradigm (Hevner et al., 2004) ensures that its contributions are evaluable, reproducible, and situated within the established knowledge base of the information systems discipline. The artifact is not merely described but rigorously tested against functional and security criteria, and its design decisions are each traceable to cited prior work. This combination of practical banking relevance and academic attention is the defining characteristic of a successful DSR project, and the author is confident that Branch Insight 360 satisfies both dimensions.

8. References:

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases – an introduction. *Natural Language Engineering*, *1*(1), 29–81. https://doi.org/10.1017/s135132490000005x

Bahrain Economic Vision 2030. (2019a). Ministry of Finance and National Economy. https://www.mofne.gov.bh/en/project-initiatives/bahrain-economic-vision-2030/

Bahrain Economic Vision 2030. (2019b). Ministry of Finance and National Economy. https://www.mofne.gov.bh/en/project-initiatives/bahrain-economic-vision-2030/

Bahrain FinTech Bay. (2026). Bahrain FinTech Bay. https://www.bahrainfintechbay.com/

Banker's Dashboard. (2023). Deluxe.com. https://www.deluxe.com/bankers-dashboard/

Barkley, J. (1997). Comparing simple role based access control models and access control lists. *Proceedings of the Second ACM Workshop on Role-Based Access Control - RBAC '97*, 127–132. https://doi.org/10.1145/266741.266769

Business Intelligence Market Size & Share Report, 2019-2025. (2019). Grandviewresearch.com. https://www.grandviewresearch.com/industry-analysis/business-intelligence-market

CBB Homepage. (2026). CBB. https://www.cbb.gov.bh/

Centeral Bank of Bahrain. (2022). *CBB RuleBook*. Thomsonreuters.com. https://cbben.thomsonreuters.com/entiresection/2300930

Chaudhuri, S., Dayal, U., & Narasayya, V. (2011). An overview of business intelligence technology. *Communications of the ACM*, *54*(8), 88–98. https://doi.org/10.1145/1978542.1978562

Denodo. (2024). *Product and services overview*. Denodo. https://www.denodo.com/en/denodo-platform/overview

denodo. (2022). *Logical Data Fabric Whitepaper*. Denodo. https://www.denodo.com/en/document/whitepaper/logical-data-fabric-whitepaper

Denodo AI SDK - User Manual. (2026). Denodo.com. https://community.denodo.com/docs/html/document/denodoconnects/latest/en/Denodo%20AI%20SDK%20-%20User%20Manual

Entire Section | Rulebook. (2022). Thomsonreuters.com. https://cbben.thomsonreuters.com/entiresection/2300930

FastAPI. (2026). *FastAPI*. Tiangolo.com. https://fastapi.tiangolo.com/

GCC listed banks' results. (2025). KPMG. https://kpmg.com/kw/en/insights/2025/04/gcc-listed-banks-results.html

General Data Protection Regulation (GDPR) – Final text neatly arranged. (2021, October 22). General Data Protection Regulation (GDPR). https://gdpr-info.eu/art-5-gdpr/

Gerling, C., & Lessmann, S. (2024). *Leveraging AI and NLP for Bank Marketing: A Systematic Review and Gap Analysis*. ArXiv.org. https://arxiv.org/abs/2411.14463

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research1. *MIS Quarterly*, *28*(1), 75–106. https://doi.org/10.2307/25148625

Hu, C. T., Ferraiolo, D. F., Kuhn, D. R., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014, January 16). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST. https://www.nist.gov/publications/guide-attribute-based-access-control-abac-definition-and-considerations

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. https://doi.org/10.6028/nist.sp.800-162

Krishnamurthy, J., & Mitchell, T. (2012). Weakly Supervised Training of Semantic Parsers. *ACL Anthology*, 754–765. https://aclanthology.org/D12-1069/

Roberts, J. (2017). *Attribute Based Access Control | CSRC*. Nist.Gov. https://csrc.nist.gov/projects/attribute-based-access-control

Rescorla, E. (2018). *RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3*. IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc8446

Prat, N., & Akoka, J. (n.d.). *ARTIFACT EVALUATION IN INFORMATION SYSTEMS DESIGN-SCIENCE RESEARCH -A HOLISTIC VIEW*. https://cedric.cnam.fr/fichiers/art_3208.pdf

Puppeteer. (2026). Pptr.Dev. https://pptr.dev/

OWASP Top Ten Web Application Security Risks | OWASP Foundation. (2025). Owasp.Org. https://owasp.org/www-project-top-ten/

Logical Data Fabric. (n.d.). https://www.denodo.com/system/files/document-attachments/wp-logicaldatafabric-2022-01_0.pdf

Mhlanga, D. (2020). Industry 4.0 in Finance: The Impact of Artificial Intelligence (AI) on Digital Financial Inclusion. *International Journal of Financial Studies*, *8*(3), 45. https://doi.org/10.3390/ijfs8030045

nCino Consumer Banking Solution. (2026). NCino. https://www.ncino.com/en-EMEA/consumer-banking

Ni, Y., Nyhoff, J., Napier, M., & Townsend, D. (2026). AI Innovation and Bank Performance: Evidence from Patent Activity of Large U.S. Commercial Banks. *Journal of Risk and Financial Management*, *19*(4), 247. https://doi.org/10.3390/jrfm19040247

OutSystems. (2026a). *OutSystems: The future of financial services software development*. OutSystems.com. https://www.OutSystems.com/industries/financial-services/

OutSystems. (2026b). *OutSystems: The leading AI-powered low-code platform*. OutSystems.com. https://www.OutSystems.com/low-code-platform/

OutSystems. (2026). *OutSystems Community*. OutSystems.com. https://success.OutSystems.com/documentation/

OutSystems Community . (2026). *OutSystems Platform Best Practices*. OutSystems.com. https://success.OutSystems.com/documentation/11/onboarding_developers/OutSystems_platform_best_practices/

OutSystems: The future of financial services software development. (2026). OutSystems.com. https://www.OutSystems.com/industries/financial-services/

PricewaterhouseCoopers. (2025). *Next in banking and capital markets 2025*. PwC. https://www.pwc.com/us/en/industries/financial-services/library/banking-industry-trends.html

Ridzuan, N. N., Masri, M., Anshari, M., Fitriyani, N. L., & Syafrudin, M. (2024). AI in the Financial Sector: The Line between Innovation, Regulation and Ethical Responsibility. *Information*, *15*(8), 432. https://doi.org/10.3390/info15080432

Shen, L., Shen, E., Luo, Y., Yang, X., Hu, X., Zhang, X., Tai, Z., & Wang, J. (2023). Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, *29*(6), 3121–3144. https://doi.org/10.1109/tvcg.2022.3148007

Singhania, D., Tanty, G., & Srivastava, A. (2024). Empowering Rural Jharkhand: Adoption of Artificial Intelligence for Digital Financial Inclusion. *Information Systems Engineering and Management*, 123–134. https://doi.org/10.1007/978-3-031-70219-8_8

Supabase. (2026, May 8). *Supabase Docs*. Supabase.com; Supabase Docs. https://supabase.com/docs

The Denodo AI Software Development Kit (SDK). (2024). Denodo. https://www.denodo.com/en/solutions/by-capability/ai-software-development-kit

What is the Model Context Protocol (MCP)? - Model Context Protocol. (2025). Modelcontextprotocol.io; Model Context Protocol. https://modelcontextprotocol.io/docs/getting-started/intro

وزارة المالية والاقتصاد الوطني - مملكة البحرين. (2025). Ministry of Finance and National Economy. https://www.mofne.gov.bh/

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921. https://doi.org/10.18653/v1/d18-1425

Zhong, V., Xiong, C., & Socher, R. (2017). *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. ArXiv.org. https://arxiv.org/abs/1709.00103

9. Appendices

Appendix 1: Detailed Requirement Gathering

This section describes the process adopted in gathering and validating the requirements through the involvement of stakeholders, background literature review, and matching against the project specification. This process ensured that each identified requirement is justified either by a business necessity, technical limitation, or statutory regulation.

Appendix 1.1: Interview and Stakeholder Summary

Interviewee	Role	Key Question	Response Summary	Requirement Identified	Type
Line Manager	Line Manager	What is the highest operational risk if branch users can query data directly?	Sensitive data exposure through reformulated questions is the main risk.	Sensitive topic blocking and response scrubbing	Functional
		What response behavior is needed for branch teams?	Managers need fast, readable output without waiting for technical report cycles.	Dashboard-ready response generation	Functional
		What controls are mandatory before production use?	Access must be role-based, auditable, and centrally governed.	RBAC + ABAC + PBAC access model	Functional
		How should platform trust and traceability be handled?	Administrators must inspect diagnostic metadata for verification.	Admin-only diagnostic visibility	Functional
		How should ambiguous business questions be handled?	The system should ask clarification prompts before execution.	Clarification prompt workflow	Functional
		What should happen when unauthorized data is requested?	Deny safely and log the event without exposing schema details.	Secure denial and audit logging	Functional
		What should users see first in output?	KPI summary first, then chart, then optional table details.	Business-first response layout	Functional
		What should happen if AI translation fails?	Fallback execution path should return usable output with warning.	Fallback query execution path	Functional

		How should branch scope be enforced?	Users should only see own-branch scope unless elevated role allows wider scope.	Branch-scoped analytics enforcement	Functional
		What export behavior is required for meetings?	Exports should include timestamp, filters, and clean formatting.	Management-ready PDF export	Functional
		How should large record requests be handled?	Cap row count and ask user to refine filters.	Record-level limits and guidance	Functional
		What should happen to blacklisted users?	Requests should be blocked before query processing starts.	Blacklist-first gating	Functional
		How should authorization checks be evaluated for scale?	Evaluate cheapest deny conditions first to reduce latency.	Policy check ordering	Non-Functional
		What local setup constraints apply for onboarding?	System should run on standard developer machines.	Standard developer hardware deployment	Non-Functional
		What reporting output quality is required?	PDF generation must be reliable and fast for operations.	Production-grade PDF generation capability	Non-Functional
		What uptime behavior is expected during partial dependency failures?	Return degraded state clearly and continue safe service.	Graceful degradation behavior	Non-Functional
		What latency target should authorization decisions meet?	Access decisions should complete quickly and consistently.	Authorization overhead under 50ms	Non-Functional
		What security metadata must be retained?	Retain operational logs for post-incident investigation.	Audit-log retention and traceability	Non-Functional
		What should deployment consistency look like?	Behavior should be reproducible from dev to production.	Environment parity and reproducible deployment	Non-Functional
		What availability objective should be set?	Define explicit monthly SLO/SLA targets with monitoring.	Service availability objective definition	Non-Functional

		How should the platform behave under peak concurrency?	Response should degrade gracefully with no security bypass.	Load resilience and safe degradation	Non-Functional

Table 9: Interview Summary

Appendix 1.2: Research-Based Requirement Gathering

Research Source	Citation	Key Finding	Requirement Derived	Type
Central Bank of Bahrain	(CBB Homepage, 2026)	Regulated systems require strict access control and governance.	RBAC + ABAC + PBAC enforcement	Functional
NIST ABAC Guidance	(Roberts, 2017)	Runtime attributes improve fine-grained authorization decisions.	Header and context-driven access checks	Functional
ARTIFACT EVALUATION IN INFORMATION SYSTEMS DESIGN-SCIENCE RESEARCH – A HOLISTIC VIEW	(Prat & Akoka, n.d.)	Artifacts should be evaluated against real environment and goals.	Structured validation and traceability	Non-Functional
NLIDB literature	(Androutsopoulos et al., 1995)	Query ambiguity and schema mapping are core risks in NL interfaces.	Clarification handling and governed output	Functional
NIST Cybersecurity Framework	(NIST CSF 2.0, 2024)	Layered controls and governance reduce operational cyber risk.	Defense-in-depth authorization chain	Non-Functional
OWASP Top 10	(*OWASP Top Ten Web Application Security Risks* |	Common web risks must be handled by	Input safety and secure error behavior	Non-Functional

	OWASP Foundation, 2025)	default secure coding controls.		
TLS Specification	(Rescorla, 2018)	Modern TLS is baseline for secure in-transit communication.	HTTPS/TLS enforcement standard	Non-Functional
GDPR Principles	(*General Data Protection Regulation (GDPR) – Final Text Neatly Arranged*, 2021)	Data minimization supports role-limited visibility and redaction.	Sensitive-data redaction policy	Functional
ACL Anthology	(Krishnamurthy & Mitchell, 2012)	Semantic parsing research supports intent-aware query understanding.	Intent classification and translation quality	Functional
Puppeteer Docs	(*Puppeteer*, 2026)	Server-side rendering provides stable report export behavior.	Reliable PDF generation capability	Non-Functional

Table 10: Research-Based Requirement Gathering

Appendix 1.3: Project Brief to Requirement Mapping

Project Brief Statement	Requirement Extracted	Category
Users should ask questions in plain English.	Natural language query intake	Functional
The system should return to usable analytics, not raw technical output.	Dashboard-ready structured responses	Functional
Access must differ by user role/group.	Multi-tier access control model	Functional
Solution must be secure and manageable.	API protection, CORS, and logging	Non-Functional
Branch managers need holistic branch insights.	360-degree branch analysis support	Functional

Table 11: Requirement Mapping

Appendix 2: Detailed Design Document

This appendix provides the full design visual repository, including architecture, process, data, security, and governance diagrams.

Appendix 2.1: Diagram Catalog

Num	Diagram	Purpose
01	System Context	End-to-end service boundaries
02	API Request Sequence	Request/response lifecycle
03	Frontend Module Graph	UI module organization
04	Dashboard Build Pipeline	Dashboard composition flow
05	Startup Operations Flow	Service bootstrap sequence
06	Retry and Fallback Flow	Failure and retry behavior
07	Comprehensive Dataflow	Cross-service data movement
08	ER Diagram	Data structure view
09	Threat Model	Security threat mapping
10	Security Controls Map	Control-to-risk mapping
11	Algorithm Policy Flowchart	Access decision logic
12	State Diagram	Runtime state transitions
13	Class Domain Model	Domain class relationships
14	Component Architecture	Major component boundaries
15	Deployment Architecture	Runtime deployment topology
16	Main Sequence Flow	Main user flow sequence
17	Activity Dashboard Pipeline	Visual activity workflow
18	Package Module Organization	Codebase package layout

Table 12: Diagram Catalog

Appendix 2.2: Design Diagrams

System Context

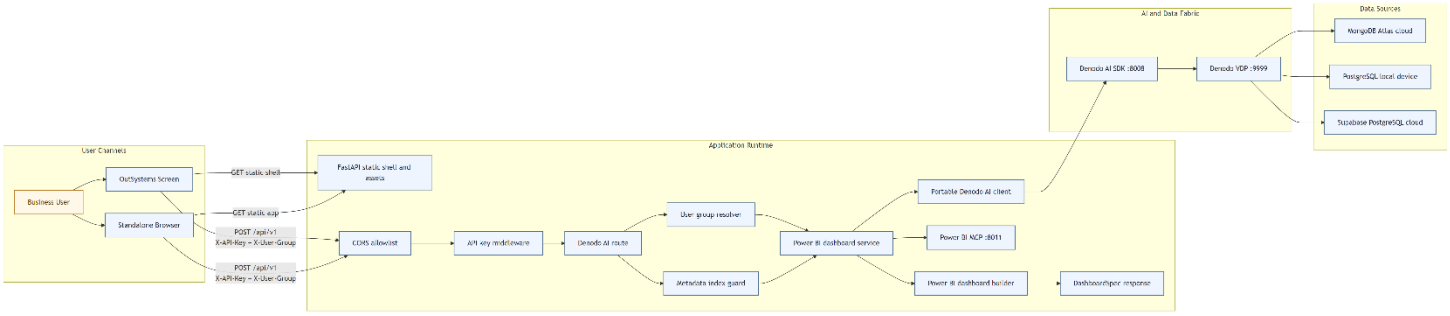


Figure 34: System Context

The figure describes the platform's boundary in its entirety, the entities involved outside of it, trust boundaries, and who is accountable for services ranging from the user's trigger actions through the governed process. One should imagine the figure as the entire system or process, as all blocks in this diagram contribute to the complete end-to-end workflow

control, rather than representing just one aspect. The diagram illustrates input data, data changes within the process, and the outputs in the form of the governed result.

It is important as it validates the scope of the architecture and provides accountability across channels, orchestration services, and downstream analytics integration. As described in the thesis, it demonstrates deliberate design decisions aligned with technical qualities and governance needs. Furthermore, it is helpful in establishing traceability between architectural or process-related decisions and measurable criteria such as reliability, security, maintainability, and explainability.

On the side of implementation and validation, the diagram is being used to verify boundary assumptions, validate proper assignment of authoritative systems for each of the decisions and the lack of ungoverned data routes around the gates. Such use makes the diagram applicable to the implementation review, test design, and even troubleshooting, as each element of the diagram (a block, branch, or transition) corresponds to a behavioral characteristic. Maintaining the diagram in its completeness in the appendix allows for cross-referencing purposes.

API Request Sequence

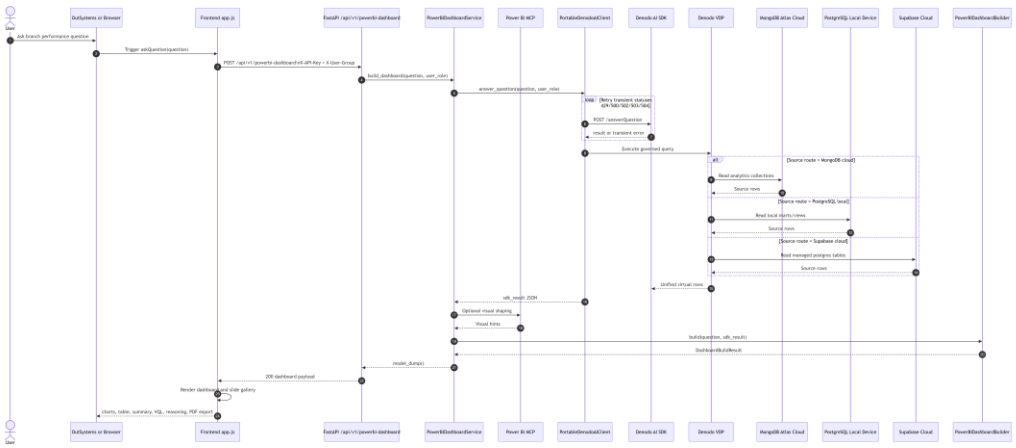


Figure 35: API Request Sequence Diagram

This figure shows the complete flow of API, from the request being sent to its validation, coordination, execution, and response formation. One should consider it an architectural view of a whole system, where each of its parts operates in an organized and controlled

end-to-end workflow, rather than a standalone entity. It helps demonstrate what is processed during the process, what changes occur within it, and what becomes the governed output.

This figure plays an important role since it helps understand dependencies between the stages and their points of potential failures. Thus, reliability and governance control can be assessed before implementing the system into the production phase. In the thesis, the figure proves the fact that all design decisions were done intentionally and with regard to technical quality, governance requirements, and operational aspects. At the same time, it increases traceability since links between architecture/process decisions are made with respect to concerns such as reliability, security, maintainability, and explainability.

To be used and evaluated further in practice, it allows designing testing cases for the specified process by correlating each of its stages with inputs, outputs, and potential failures in particular situations. It can be used when assessing and debugging implementation since each block, branch, or transition can be correlated with concrete behavior of the software under consideration. The complete diagram should remain in the appendix for the sake of completeness of information.

Frontend Module Graph

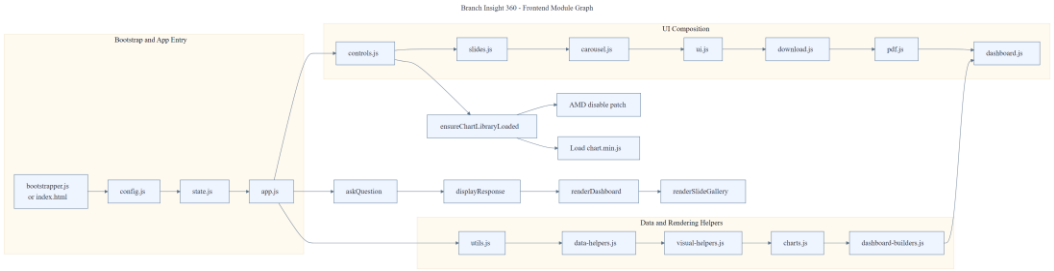


Figure 36: Frontend Module Graph

This figure illustrates the ownership of input handling, dashboard rendering, shared UI utilities, and the management of state. It is intended to illustrate the full system view where different pieces collaborate to execute the entire end-to-end process. This system view describes inputs required for the process, activities done, and the outputs that result from it as governed output.

This diagram reduces the risk of unwanted coupling by clarifying dependency directions and keeping presentation logic and data manipulation logic decoupled. In my thesis, it proves that I have made well-considered design decisions based on technical quality, requirements for governance, and operational needs of the process. Furthermore, the diagram supports traceability by tying architecture/process-related decision to concerns such as reliability, security, maintainability, and explainability.

The diagram is referenced while performing refactoring and onboarding tasks to understand safe limits for changes, the impact of such changes, and manageability of contracts between components involved in the process. As a result, the diagram proves to be useful for conducting implementation reviews, designing tests, and troubleshooting issues related to different blocks, branches, and transitions that map onto observable behavior.

Dashboard Build Pipeline

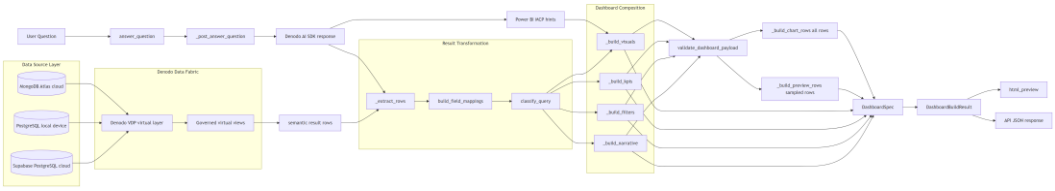


Figure 37: Dashboard Build Pipeline

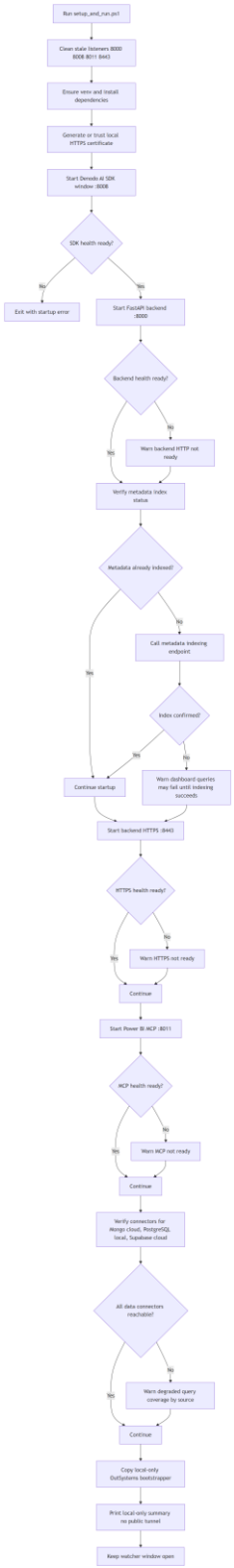
The pipeline demonstrates the transformation of normalized analytical payloads into KPI cards, and the related narrative blocks can be seen. Consider the figure as a holistic system design where all parts act in concert throughout an end-to-end and tightly governed workflow, not as loosely connected components. It visualizes what input is taken, what processing takes place, and how the output is achieved.

The significance of this figure lies in its representation of presentation consistency via a process of structuring data in stages rather than rendering it according to ad-hoc frontend design decisions. It provides evidence that all the design decisions were made purposefully and aligned with technical excellence, governance principles, and operational aspects of the system. Furthermore, such a figure assists in tracking the alignment of architectural or

process choices against concerns such as reliability, security, maintainability, and explainability.

For validation purposes, one should trace sample payloads from start to finish through every pipeline stage, verifying that the output structures are deterministic for certain query classes. Such a diagram can be very helpful during the implementation review, test case design, and debugging, as any block, branch, or transition may have observable effects that can be traced back to this figure.

Startup Operations Flow



The Startup Operations flow diagram depicts the bootstrapping order, readiness checks, health gates, and failure handling during service initialization. It should be interpreted as a full system or process view, where each element contributes to an end to end controlled workflow rather than standing alone. This view helps explain what enters the process, what is transformed internally, and what exits as governed output.

This diagram matters because it establishes reproducible startup behavior and prevents partial system activation that could expose users to unstable or ungoverned states. In the thesis context, it demonstrates that design decisions were intentional and aligned with technical quality, governance, and operational needs. It also supports traceability by connecting architecture or process choices to measurable concerns such as reliability, security, maintainability, and explainability.

In practical operation and validation, it is applied operationally as a runbook reference to verify healthy component progression and to diagnose startup stalls at specific readiness checkpoints. This makes the diagram useful for implementation review, test design, and troubleshooting because each block, branch, or transition can be tied to observable behavior. Keeping the complete diagram in the appendix preserves cross reference context required for reproducibility and academic assessment.

Figure 38: Startup Operations Flow

Retry and Fallback Flow

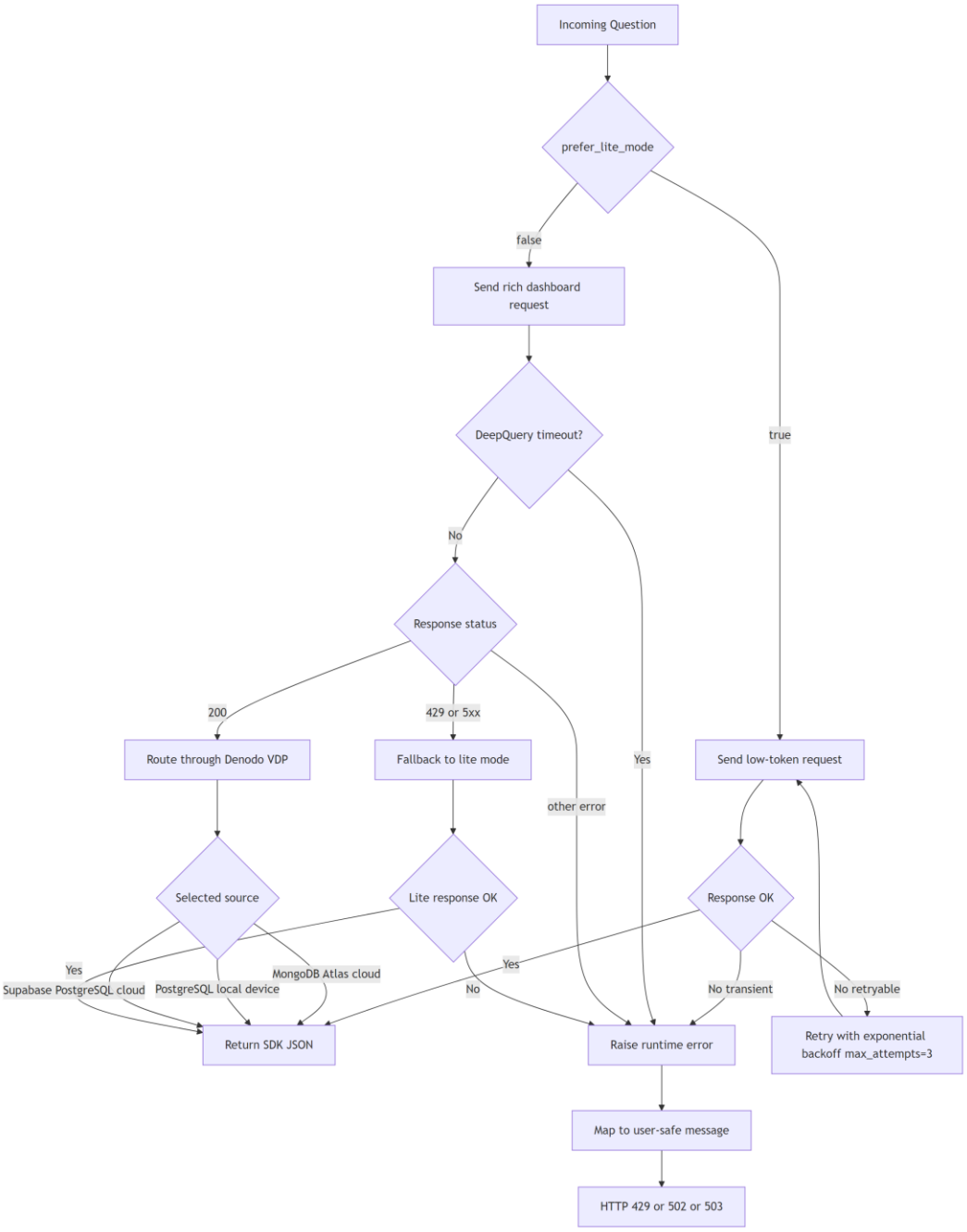


Figure 39: Retry and Fallback Flow

The figure above presents retry decisions, timeout thresholds, escalation paths, and fallback response strategies under degraded service conditions. It should be interpreted as a full system or process view, where each element contributes to an end to end controlled workflow rather than standing alone. This view helps explain what enters the process, what is transformed internally, and what exits as governed output.

This diagram matters because it documents resilience policy so reliability behavior is explainable and consistent with user facing clarity and governance obligations. In the thesis context, it demonstrates that design decisions were intentional and aligned with technical quality, governance, and operational needs. It also supports traceability by connecting architecture or process choices to measurable concerns such as reliability, security, maintainability, and explainability.

In practical operation and validation, it is tested using controlled fault injection to verify retry limits, fallback triggers, and graceful degradation outcomes across critical service calls. This makes the diagram useful for implementation review, test design, and troubleshooting because each block, branch, or transition can be tied to observable behavior.

Comprehensive Dataflow

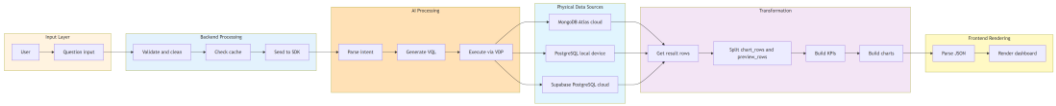


Figure 40: Comprehensive Dataflow

This data flow diagram represents cross-layer data movement from ingress through governance, transformation, execution, and egress across the full architecture. It should be interpreted as a full system or process view, where each element contributes to an end-to-end controlled workflow rather than standing alone. This view helps explain what enters the process, what is transformed internally, and what exits as governed output.

This diagram matters because it makes trust transitions and transformation boundaries explicit, supporting explainability, maintainability, and audit-ready reasoning. In the thesis context, it demonstrates that design decisions were intentional and aligned with technical quality, governance, and operational needs. It also supports traceability by connecting architecture or process choices to measurable concerns such as reliability, security, maintainability, and explainability.

In practical operation and validation, it is applied to trace data lineage, identify control ownership at each handoff, and verify that no stage bypasses policy enforcement. This

makes the diagram useful for implementation review, test design, and troubleshooting because each block, branch, or transition can be tied to observable behavior.

ER Diagram

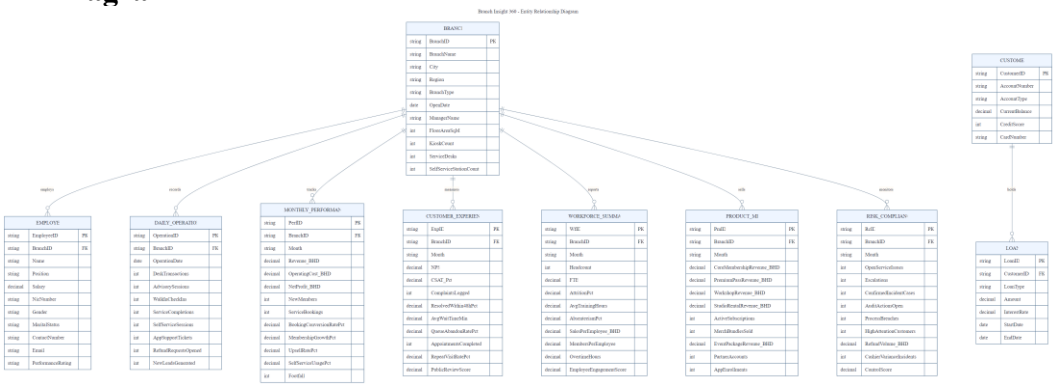


Figure 41: ER Diagram

The following ERD describes the key entities and associations that are fundamental to the branch analysis field. The ERD merges the conceptual data model with operational reporting and querying actions, hence providing a basis for schema-based reasoning. Keeping the ERD intact ensures that the relationships remain intact throughout the entire model.

Threat Model

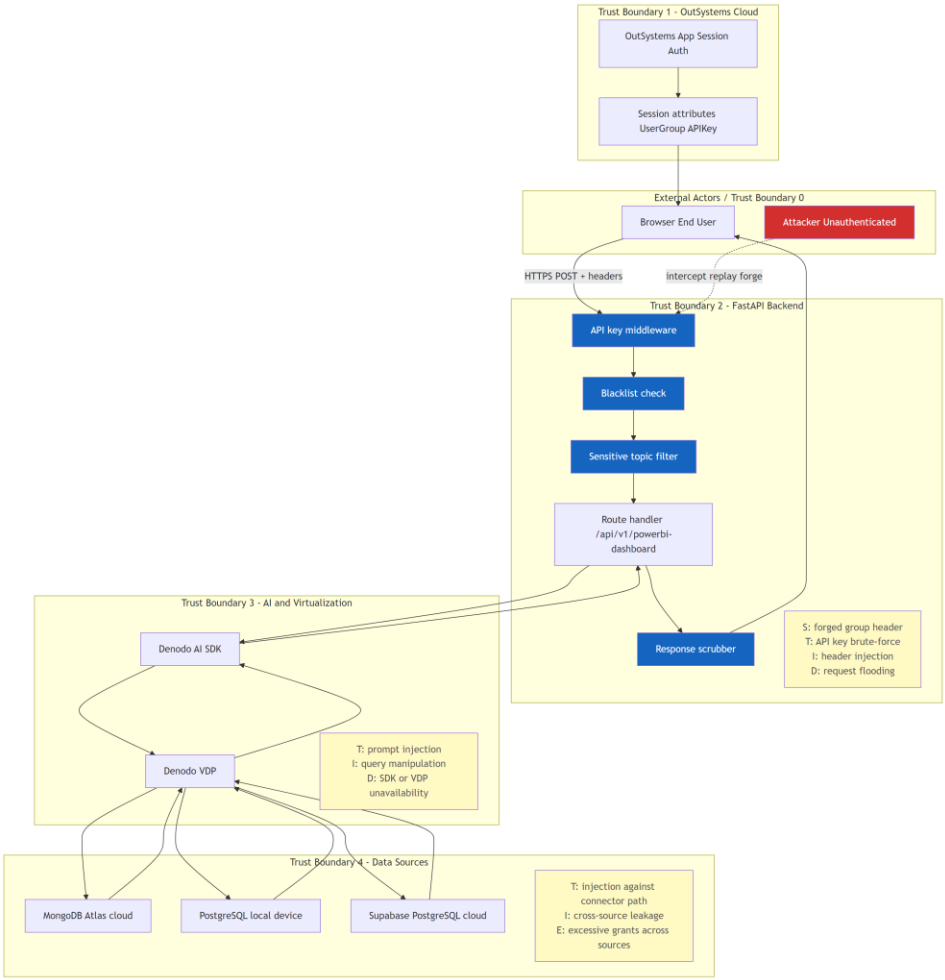


Figure 42: Threat Model

The threat model identifies the main vulnerabilities, methods of abuse, and relevant architectural areas. It supports threat-based design verification through associations between threats, components of the system, and associated controls. The entire diagram has been retained for consistency of mitigations within one security perspective.

Security Controls Map

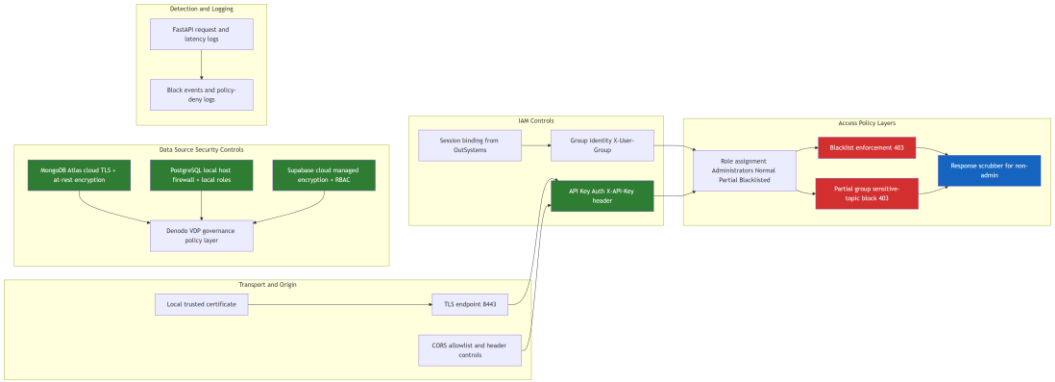


Figure 43: Security Controls Map

The security map of control shows how the multilayered approach aligns with different categories of risks. This involves enforcement rules for authentication, role/policy-based controls, and secure responses. Keeping the comprehensive map as it helps in interpreting the concept of defense-in-depth for audit purposes.

Algorithm Policy Flowchart

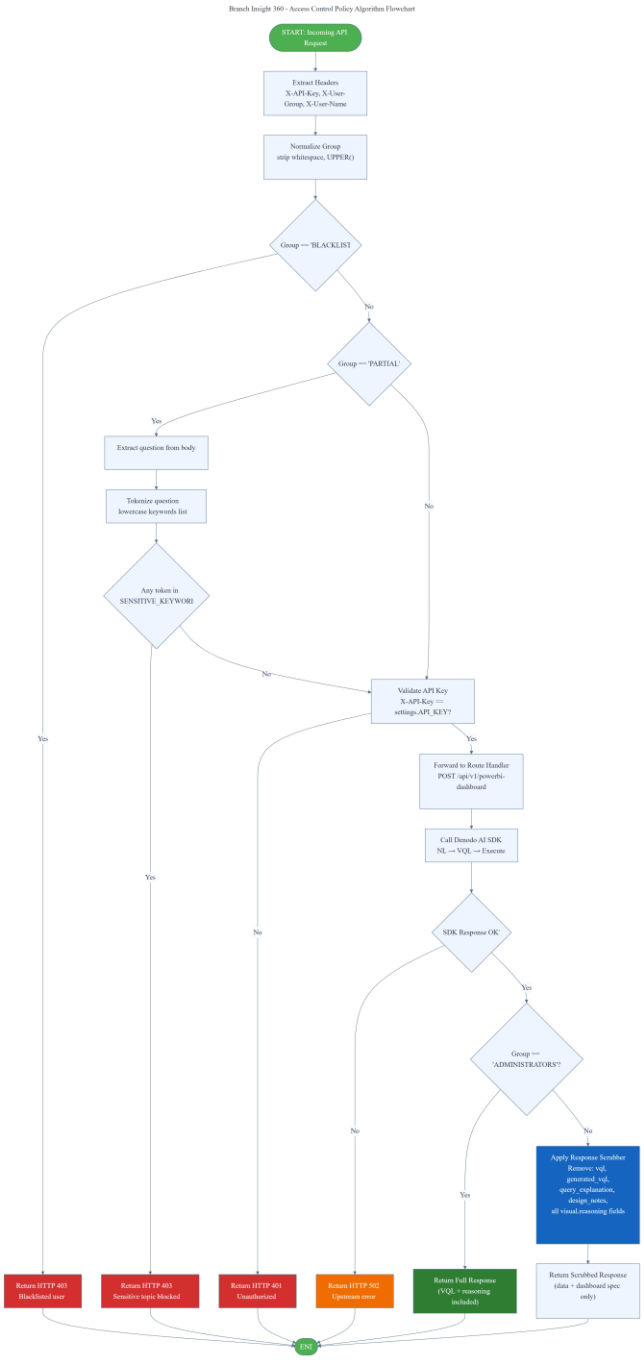


Figure 44: Algorithm Policy Flowchart

This diagram explains the entire policy decision algorithm process, starting from request submission until the final response determination stage. This includes authentication, role recognition, handling sensitivity, and allowed or not allowed decision paths. The full version includes all of these elements in their entirety for evaluation purposes.

State Diagram

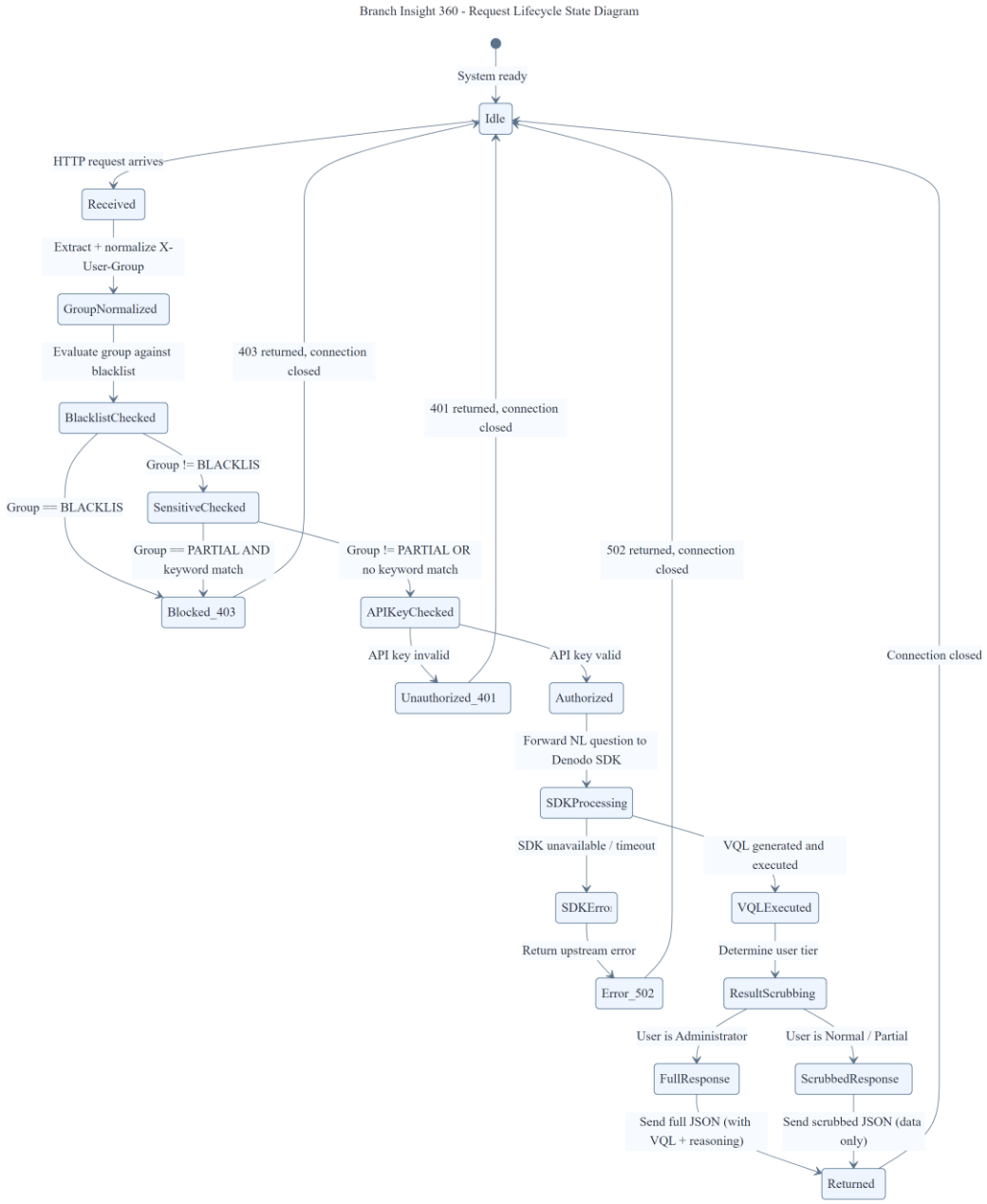


Figure 45: State Diagram

The figure above shows the different run-time paths that exist from the normal, exception, to recovery states involved in request processing. It explains how the system goes through the state life cycle as well as where the feedback controls are placed.

Class Domain Model

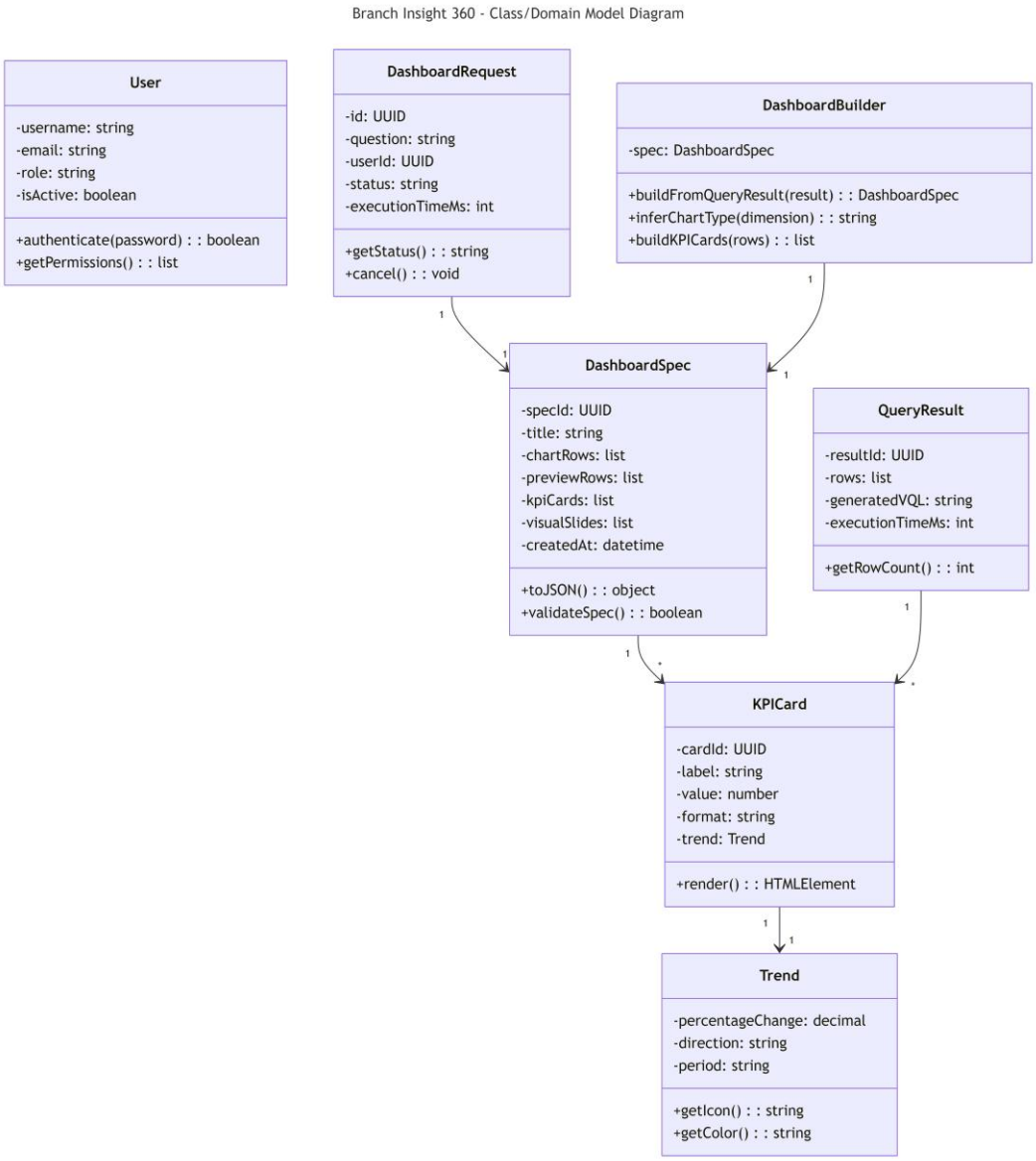


Figure 46: Class Domain Model

The figure above presents class responsibilities, associations, and abstraction boundaries for orchestration, policy, and response shaping objects. It should be interpreted as a full system or process view, where each element contributes to an end to end controlled workflow rather than standing alone. This view helps explain what enters the process, what is transformed internally, and what exits as governed output.

This diagram matters because it supports maintainable implementation by making ownership explicit and reducing accidental coupling between domain responsibilities. In the thesis context, it demonstrates that design decisions were intentional and aligned with

technical quality, governance, and operational needs. It also supports traceability by connecting architecture or process choices to measurable concerns such as reliability, security, maintainability, and explainability.

In practical operation and validation, it guides code-level design reviews by checking whether implementation classes still align with the modeled responsibilities and relations. This makes the diagram useful for implementation review, test design, and troubleshooting because each block, branch, or transition can be tied to observable behavior.

Component Architecture

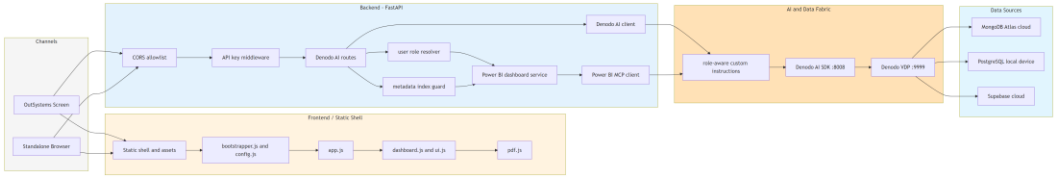


Figure 47: Component Architecture Diagram

The component architecture represents the major logical components and their interface boundaries within the entire solution stack. This is useful for modularity of thinking and future extension planning through clear definition of responsibilities and interaction. The initial image is kept intact to preserve the component topology.

Deployment Architecture

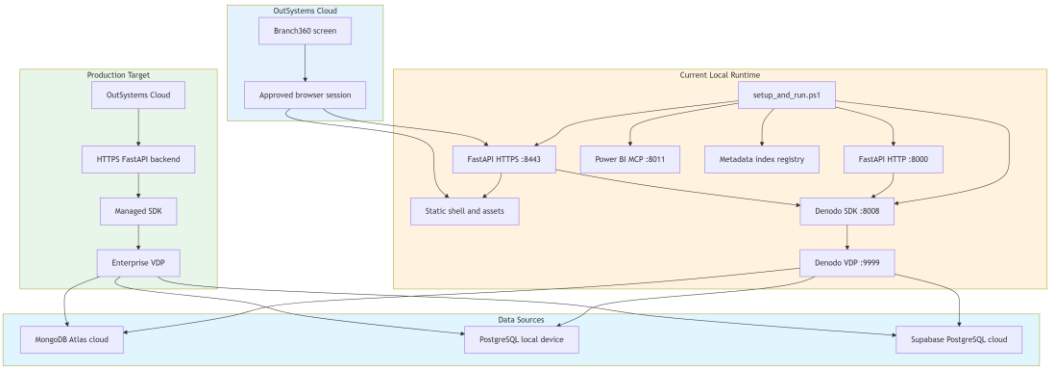


Figure 48: Deployment Architecture

Deployment architecture depicts the operation, the placement and connections to other networks of each component, environmental constraints, and integration of external systems. The diagram may be considered as a representation of an entire process where each constituent contributes to the overall process workflow. Thus, the diagram

demonstrates all the necessary components to consider when analyzing the inputs, processes, and outputs within the system under consideration.

The relevance of such a representation lies in its ability to connect the software design with practical applications, which provides us with the ability to examine such aspects as performance, security, and availability. Within the scope of this thesis, this particular diagram is used to confirm that certain design decisions are purposeful and aligned with technical, architectural, or operational considerations, including the quality of the system.

In terms of implementation, the operations department uses this particular scheme while considering such problems as environmental improvement, potential points of failure, and hardening the production environment. In this case, this diagram serves as useful support for designing tests and tracing potential issues, since any element can be connected to observable properties of reliability, security, maintainability, and explainability.

Main Sequence Flow

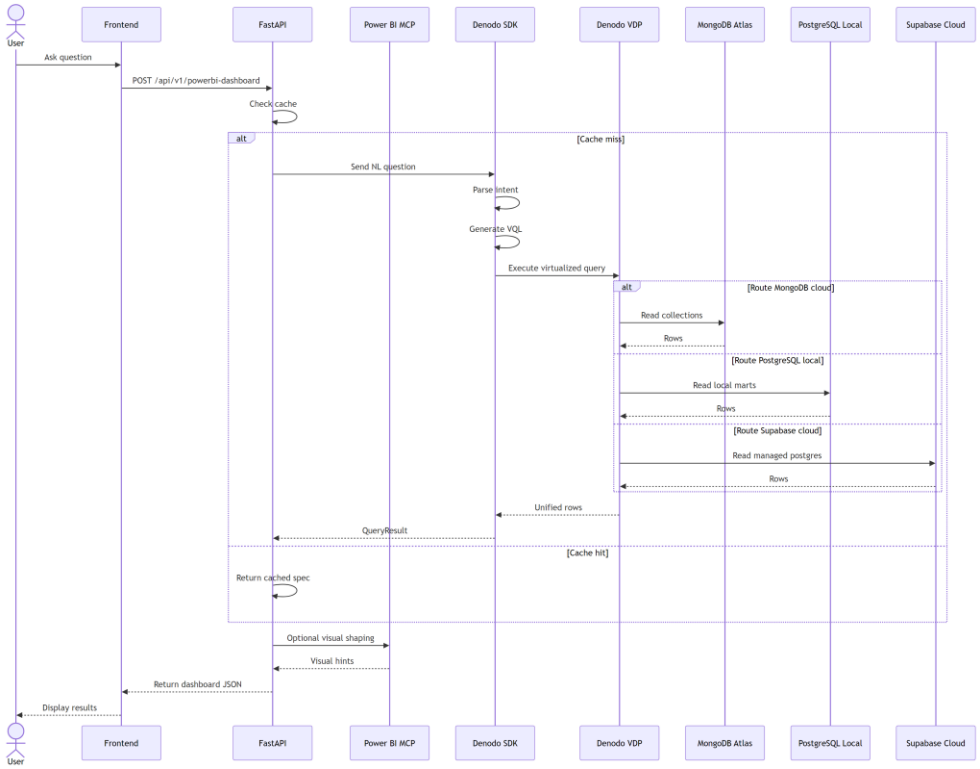


Figure 49: Main Sequence Flow Diagram

The figure demonstrates the major path for the request capture and processing, including rules enforcement, and its normal delivery. It is important to view this diagram not as an aggregation of parts but as a comprehensive representation of how the entire process works from end to end, with all parts working in concert.

It is important in our context, as it focuses the view on the major value chain, providing the means for improving and validating the basic behavior independently of the complications introduced by exceptions. The illustration shows in the thesis that design decisions made were deliberate and consistent with the criteria for the technical quality, governance, and operational considerations of the design.

As a tool for validation, it is used in acceptance testing through the verification of milestone events and expected results. It is helpful as such due to the ability to trace architectural and operational decisions to requirements and concerns for reliability, security, maintainability, and explainability.

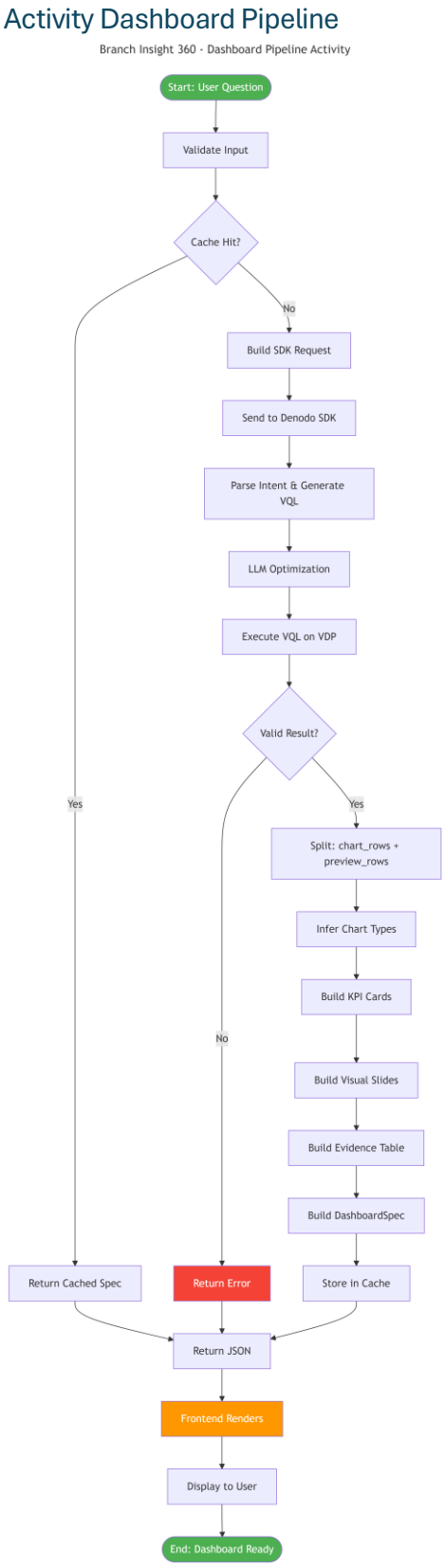


Figure 50: Activity Diagram Pipeline

The figure above presents process level activity flow and branching used to transform validated results into dashboard ready visual outputs. It should be interpreted as a full system or process view, where each element contributes to an end to end controlled workflow rather than standing alone. This view helps explain what enters the process, what is transformed internally, and what exits as governed output.

This diagram matters because it explains operational logic behind output assembly and highlights where conditional processing controls quality and user clarity. In the thesis context, it demonstrates that design decisions were intentional and aligned with technical quality, governance, and operational needs. It also supports traceability by connecting architecture or process choices to measurable concerns such as reliability, security, maintainability, and explainability. In practical operation and validation, it is applied during troubleshooting to localize failures to specific activity blocks and verify branch convergence toward stable output. This makes the diagram useful for implementation review, test design, and troubleshooting because each block, branch, or transition can be tied to observable behavior.

Package Module Organization

Figure 51: Package Module Organization

Package diagrams describe the boundaries within which the codebase should be written and how dependencies should flow between them. They are particularly useful during onboarding, establishing ownership of different parts of the code, and future maintenance work. The original uncropped diagram is included below to provide the complete picture.

Appendix 3: Detailed Implementation

Appendix 3.1: Component Implementation Summary

Component	Implementation Focus	Key Output
FastAPI Backend	Access control middleware, routing, response shaping	Governed API responses
Denodo AI SDK Integration	NL-to-VQL query orchestration	Data query execution path
Power BI MCP Module	Dashboard specification generation	Visual-ready JSON structures
Frontend/OutSystems	Query intake, dashboard rendering, role behavior	User-facing analytics workflow

Table 13: Component Implementation Summary

Appendix 3.2: Key Configuration Files

File Path	Purpose
`backend/.env`	API key, Denodo and runtime configuration
`backend/libs/denodo-ai-sdk/api/utils/sdk_config.env`	SDK model and Denodo connection setup
`backend/powerbi_mcp/.env`	MCP runtime and integration values
`setup_and_run.ps1`	Startup orchestration
`stop_services.ps1`	Service shutdown and cleanup
`test_connection.ps1`	Smoke and connectivity checks

Table 14: Configuration Files Table

Appendix 3.3: Runtime Ports

Service	Port
Backend HTTP	8000

Backend HTTPS	8443
Denodo AI SDK	8008
Power BI MCP	8011

Table 15: Runtime Ports

Appendix 4: Detailed Testing Results

Appendix 4.1: Usability Summary

Metric	Observation
Ease of Use	High; one startup sequence after environment setup
Learnability	High, clear run and verify flow
Response Performance	Fast for simple queries, longer for complex prompts
Error Clarity	Clear messages for auth/role/connectivity issues
User Satisfaction	Positive for readability and structured output

Table 16: Usability Summary

Appendix 4.2: Functional and Acceptance Test Execution Records

This subsection provides detailed execution evidence for each test, including preconditions, inputs, expected outputs, actual outputs, and references to supporting evidence.

Test ID	Test Type	Requirement Ref	Preconditions	Input or Action	Expected Result	Actual Result	Status	Evidence Ref
TC-01	Functional	REQ-FR-01	Backend service running; valid endpoint available	Send request without API key	Request is rejected with unauthorized response	Request rejected correctly	PASS	API response log
TC-02	Functional	REQ-FR-03	Valid API key; allowed role assigned	Send request with valid key and allowed role	Dashboard response is returned	Dashboard response returned	PASS	Figure 32
TC-03	Functional	REQ-FR-03	Blocked user or group configured	Send request as blocked user or group	Access is denied	Access denied correctly	PASS	Figure 31
TC-04	Functional	REQ-FR-06	Partial role configured; sensitive prompt available	Submit sensitive banking query as partial user	Sensitive output is blocked	Sensitive output blocked	PASS	Role policy log
TC-05	Functional	REQ-FR-05	Partial role configured; non-sensitive prompt available	Submit normal analytics query as partial user	Non-sensitive response is allowed	Allowed response returned	PASS	Dashboard response log
TC-06	Functional	REQ-FR-10	Administrator role configured	Submit diagnostic-enabled request as administrator	Diagnostic response appears for admin context	Diagnostic path works for admin context	PASS	Figure 33

TC-07	Functional	REQ-FR-04	Denodo service active; metadata endpoint reachable	Run metadata indexing before query execution	Metadata indexing completes successfully	Indexing completed successfully	PASS	SDK runtime log
TC-08	Functional	REQ-FR-02	Denodo AI SDK active and connected	Submit simple business question	SDK returns structured answer and data	Answer returned correctly	PASS	Figure 26
TC-09	Functional	REQ-FR-11	Backend and MCP services running	Call powerbi-dashboard endpoint	Chart-ready JSON payload is returned	Valid JSON returned	PASS	Figure 29
TC-10	Functional	REQ-FR-12	Time-series data available	Submit billing-period trend query	Time-series output is produced correctly	Correct trend output returned	PASS	Dashboard output capture
TC-11	Functional	REQ-FR-13	Invalid credentials or simulated missing dependency	Submit request with invalid auth or unavailable service	Clear and bounded error response is returned	Clear error returned	PASS	Backend error log
TC-12	Functional	REQ-NFR-05	Startup and stop scripts available	Run stop_services.ps1 then setup_and_run.ps1	Services start in required order	Services started in order	PASS	Script execution log
TC-13	Functional	REQ-NFR-06	Health endpoints configured	Open all local health endpoints	Endpoints return successful	Health checks returned success	PASS	Figure 27

					health responses			
TC-14	Functional	REQ-NFR-01	Frontend, backend, and Denodo VDP running	Submit dashboard query and inspect service path	Data access remains through governed Denodo virtual layer	Query path remained in governed layer	PASS	Service trace log
TC-15	Functional	REQ-NFR-03	Runtime ports configured as designed	Attempt non-approved path access and verify boundaries	Internal services remain isolated behind intended boundaries	Isolation behavior confirmed	PASS	Port and route check log
TC-16	Functional	REQ-NFR-04	Policy engine and role mappings active	Submit accepted and rejected requests while tracing policy order	Cheapest-first policy sequence is preserved	Evaluation order confirmed	PASS	Policy evaluation trace
TC-17	Functional	REQ-FR-07	Aggregate and record-level prompts prepared	Submit one aggregate prompt and one record-level prompt for the same business case	System distinguishes aggregate intent from record-level request and applies safe handling guidance	Aggregate request processed; record-level request constrained with guidance	PASS	Request-response log

TC-18	Functional	REQ-NFR-02	HTTPS endpoint configured on 8443; HTTP route available for validation	Attempt request over HTTP and then HTTPS	Insecure transport is blocked or redirected; secure transport succeeds	HTTP rejected or redirected; HTTPS request processed successfully	PASS	Transport security log

Table 17: Detailed Functional Test

Test ID	Test Type	Requirement Ref	Preconditions	Input or Action	Expected Result	Actual Result	Status	Evidence Ref
AT-01	Acceptance	REQ-FR-02, REQ-FR-11	All services running	Submit standard business analytics question from user interface	User receives understandable answer with visual-ready output	Output delivered as expected	PASS	Dashboard capture
AT-02	Acceptance	REQ-FR-03, REQ-FR-06	Users assigned across all role groups	Repeat same question across Administrator, Normal, Partial, and Blacklist users	Access behavior changes according to role policy	Role-based behavior matched expected policy	PASS	Figure 31, Figure 32, Figure 33
AT-03	Acceptance	REQ-FR-13, REQ-NFR-05	Fault simulation available	Temporarily interrupt one dependent service and re-test	System returns controlled error and recovers with	Controlled handling observed	PASS	Incident test log

					restart procedure			
AT-04	Acceptance	REQ-NFR-06	Participants briefed; test scenarios prepared	Execute end-to-end user scenarios from login to insight retrieval	Scenarios complete without critical usability blockers	Completion and satisfaction targets met	PASS	Table 8

Table 18: Acceptance Test Execution Records

Appendix 4.3: Requirement-to-Test Coverage Matrix

Requirement	Covered Test IDs	Coverage Status
REQ-FR-01	TC-01	Covered
REQ-FR-02	TC-08, AT-01	Covered
REQ-FR-03	TC-02, TC-03, AT-02	Covered
REQ-FR-04	TC-09, AT-01	Covered
REQ-FR-05	TC-05	Covered
REQ-FR-06	TC-04, AT-02	Covered
REQ-FR-07	TC-17	Covered
REQ-FR-08	AT-01, AT-04	Covered
REQ-FR-09	AT-03	Covered
REQ-FR-10	TC-06	Covered
REQ-FR-11	TC-09, AT-01	Covered
REQ-FR-12	TC-10	Covered
REQ-FR-13	TC-11, AT-03	Covered

REQ-NFR-01	TC-14	Covered
REQ-NFR-02	TC-18	Covered
REQ-NFR-03	TC-15	Covered
REQ-NFR-04	TC-16	Covered
REQ-NFR-05	TC-12, AT-03	Covered
REQ-NFR-06	TC-13, AT-04	Covered

Appendix 5: User and Administrator Guide

Appendix 5.1: User Guide

The guide describes how users should submit and understand questions from their business perspective.

A) User Workflow

- Log in to the system.
- Enter a question in natural language.
- Press the button "Submit" to obtain an answer.
- Review the answer summary, KPI cards, charts, and tables.
- Refine the question or filter more if required.

B) Asking a Question

- Stick to one objective in the question.
- Define the time interval when appropriate.
- Use understandable elements such as branch, metric or client type.
- Prefer general queries such as aggregation, comparison or ranking.

Questions Examples

- Provide total debt by billing period.
- Rank the first five cities by the number of late accounts.
- Average balance of the accounts per client type.

C) Types of Responses

- An analytical response ready for further visualization.
- A follow-up question in case of misunderstanding.
- A restriction response if a request is denied or restricted.

D) Problem Solving

- Try again after refreshing.

- Rephrase the query with a general verb.
- Make sure all services are operational.
- In case of issues, provide exact timestamp, user role/team, the query itself and an error message.

Appendix 5.2: System Administrator Manual

The following are the core actions that need to be performed to manage the system effectively.

A) Core Responsibilities

- Keeping the backend, Denodo AI SDK, and Power BI MCP services up and running.
- Role management in OutSystems and backend policy mappings.
- Managing Denodo runtime configuration and connection.
- Safeguarding the API keys, credentials, and environment files.
- Monitoring AI provider quota status and services availability.
- Observing the system status and addressing daily system faults.

B) Setup and Startup Required

GitHub repo with all the project files: https://github.com/AlRashedi-316/Branch-Insight-360.git

Note: The GitHub repo is set to private as this proof of concept will turn into an actual project. For any information regarding the repo, kindly contact the owner.

Files to maintain:

1. backend/.env
2. backend/libs/denodo-ai-sdk/api/utils/sdk_config.env
3. backend/powerbi_mcp/.env

What needs to be verified:

- Mappings of OutSystems roles/groups

- Denodo Hostname, Port, User, Password, Database
- AI Provider Keys, Quota Status

Startup commands:

```
.\stop_services.ps1
.\setup_and_run.ps1
```

Verification commands:

```
Invoke-WebRequest -UseBasicParsing https://127.0.0.1:8443/health

.\test_connection.ps1
```

C) Governance and Access Control

Access control should include the following access groups, maintained by administrators:

- Administrator
- Normal/Allowed
- Partial
- Black list

All API checks, topic-sensitive detection, and response controls should remain active post-update.

D) Incident and Maintenance Procedures

Incident Response

- Checking health APIs and active ports.
- Validating credentials and URLs of services.
- Service a restart using standardized scripts.
- Re-verifying connections and test queries.

Maintenance Actions

- Credential and certificate rotation as required.

- Performing regression testing post-update.
- Ensuring alignment between documentation and actual runtimes.

Appendix 6: Users and Passwords

6.1 Application User Accounts for Access Validation

These named users are documented for policy and role validation in the project setup guide.

Username	Role/Group	Intended Access Behavior	Password
Steve	Administrators	Full access; diagnostic visibility	Pass123
Bob	Blacklist	Access denied	Pass123
Sophie (Sophia test persona)	Normal allowed	Standard analytics access	Pass1234
Will	Partial	Partial access to analytics away from confidential data	Pass1234

Table 19: User Accounts on OutSystems

6.2 Host and Endpoint Settings (Non-secret but required)

Setting	Value/Example	Notes
DENODO_DATABASE_NAME	tutorial	Virtual DB name
DENODO_SERVER_URL	http://kubernetes.docker.internal:9999	Denodo VDP endpoint
DENODO_AI_SDK_URL	http://127.0.0.1:8008	AI SDK endpoint
POWERBI_MCP_HTTP_HOST	127.0.0.1	MCP host bind
POWERBI_MCP_HTTP_PORT	8011	MCP server port
ALLOWED_ORIGINS	https://personal-v3zuha6b.OutSystemscloud.com	CORS allow-list

Table 20: Host Endpoint Settings