IT Project Portfolio

Juan Miguel Rodriguez Ramirez

Project Overview

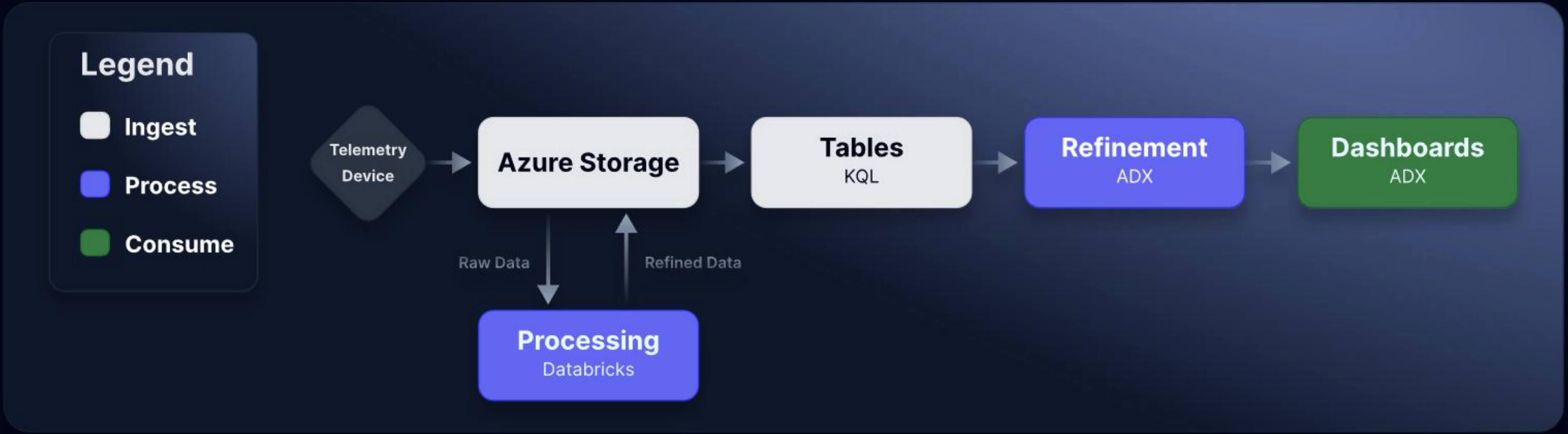
Title	Focus Area	Main Technologies
Cloud-Native Automotive Telemetry Data Pipeline	Data Engineering Pipeline	Azure, ADX, KQL, Python
User Provisioning Integration API	Enterprise Backend API	Java, SCIM, Azure AD, PostgreSQL
Full-Stack E-commerce Platform	Full-Stack Web Application	Java, Spring Boot, Angular, MySQL
Real-Time Multilingual Speech Translation System	Full-Stack Web Application	Azure, Python, Streamlit
Probabilistic Reliability Modeling for Water Treatment	Machine Learning Modeling	MATLAB, GeNIe
Training Log Management System	Backend Data Application	Python, PostgreSQL, NumPy
Game Inventory & Progress Tracker	Backend Data Application	C++, SQLite

Cloud-Native Automotive Telemetry Data Pipeline

Description & Architecture

Cloud-Native Automotive Telemetry Data Pipeline

Flowchart



Cloud-Native Automotive Telemetry Data Pipeline

Introduction

Context & Challenge

High-frequency telemetry data from distributed sensing devices generated large volumes of raw, inconsistent time-series logs. A cloud-native pipeline was required to ingest, preprocess, and structure this data for efficient analysis and visualization.

Objective & Role

Build a scalable telemetry pipeline to transform raw device logs into analysis-ready datasets and dashboards.
Role: Data Engineer responsible for data modeling, KQL preprocessing, ADX dashboards, and analytics support.

Cloud-Native Automotive Telemetry Data Pipeline

Methodology

Solution Approach

A cloud-first pipeline processed raw telemetry in Azure Storage and ADX using KQL for preprocessing, enrichment, and analysis. Additional transformations were handled with Python and Databricks.

System Architecture & Implementation

- Centralized raw telemetry in Azure Storage and ADX.
- Implemented preprocessing, feature extraction, and normalization directly in KQL.
- Built ADX dashboards for near real-time visualization and analysis.

Outcome & Impact

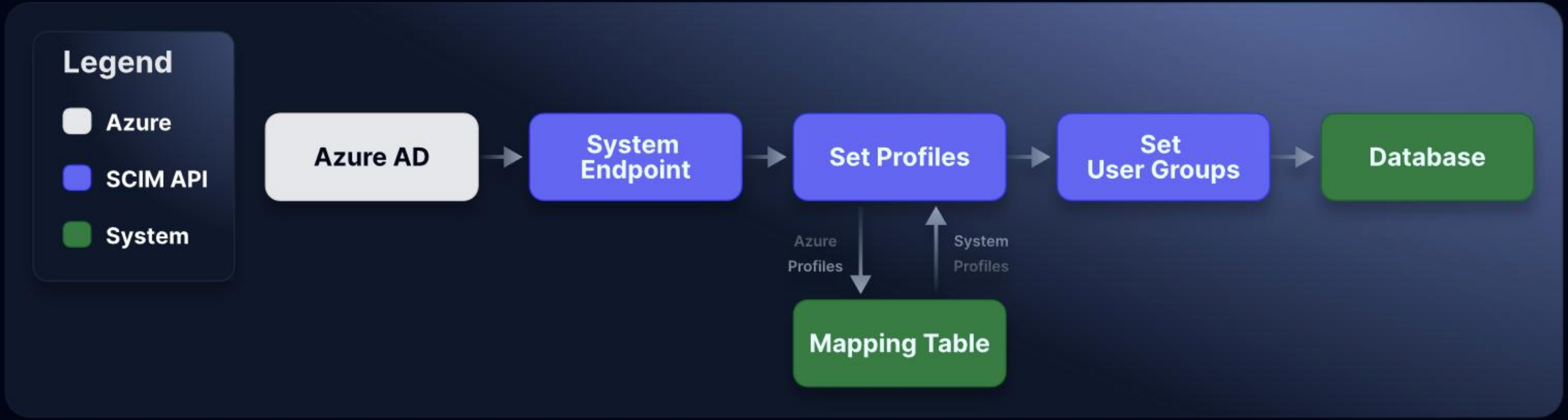
- Enabled near real-time analysis of high-frequency sensor behavior.
- Reduced manual preprocessing through automated KQL transformations.
- Improved debugging and traceability via structured datasets and dashboards.

User Provisioning Integration API

Description & Architecture

User Provisioning Integration API

Flowchart



User Provisioning Integration API

Introduction

Context & Challenge

Enterprise environments rely on Azure Active Directory (Azure AD) to manage user identities and lifecycle events. A backend integration was required to reliably synchronize users, teams, and permissions based on provisioning messages while preserving organizational hierarchy and access consistency.

Objective & Role

Design and implement a backend service to process Azure AD SCIM provisioning events, map users to internal roles and teams, and keep user states synchronized.
Role: Backend Engineer responsible for API design, provisioning logic, data mapping, and database updates.

User Provisioning Integration API

Methodology

Solution Approach

A SCIM-compatible REST API was implemented using Java and Spring Boot to process Azure AD provisioning messages and deterministically map identities to internal roles and teams.

System Architecture & Implementation

- Implemented SCIM-compatible REST endpoints to process user lifecycle events from Azure AD.
- Designed mapping logic to translate groups and attributes into internal roles, teams, and hierarchies.
- Persisted user state and permissions in PostgreSQL using Hibernate/JPA with transactional consistency.

Outcome & Impact

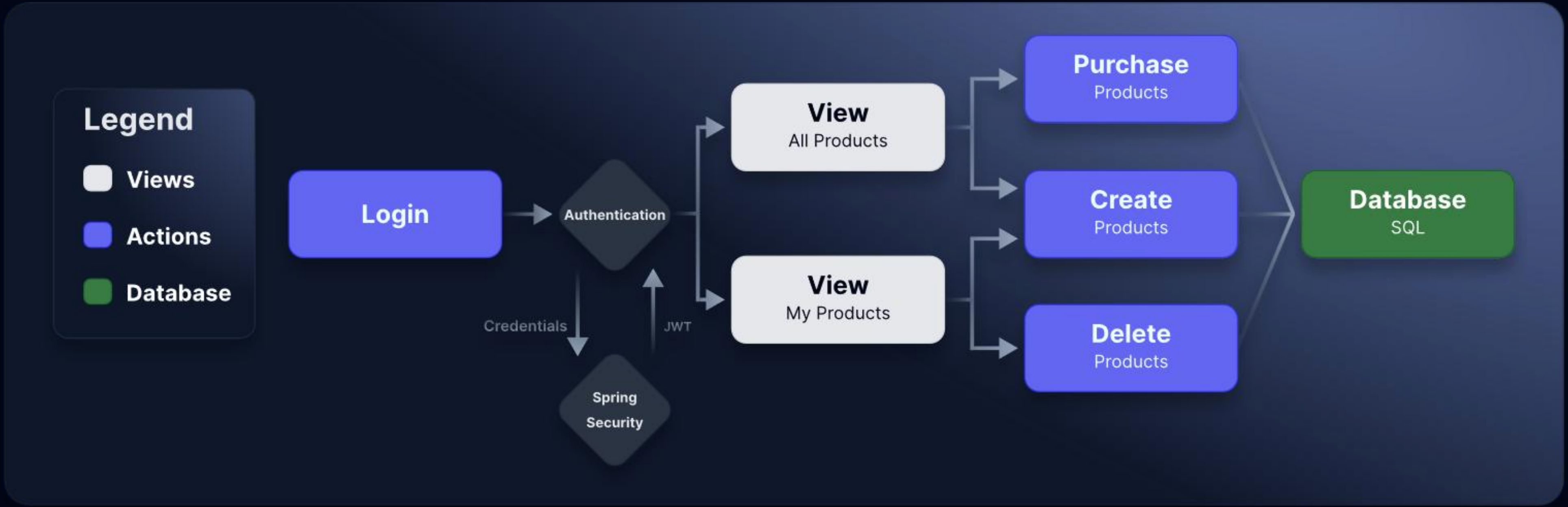
- Delivered a reliable provisioning integration synchronizing Azure AD identities with internal systems.
- Automated user onboarding, updates, and deactivation, reducing manual administration.
- Ensured consistent and auditable role and permission assignment across teams.

Full-Stack E-commerce Platform

Description & Architecture

Full-Stack E-commerce Platform

Flowchart



Full-Stack E-commerce Platform

Introduction

Context & Challenge

Small retailers often lack a unified system to manage products, inventory and sales transactions. A full-stack solution was needed to support authentication, inventory management, purchases, and persistent transaction tracking in a relational database.

Objective & Role

Design and implement a full-stack e-commerce application with a secure backend API, a responsive frontend, and consistent persistence of inventory and purchase data.
Role: Full-Stack Developer responsible for backend, frontend, database integration, and deployment.

Full-Stack E-commerce Platform

Methodology

Solution Approach

A REST-based backend was built in Java with authentication process, paired with an Angular frontend for product workflows. The system was containerized to enable reproducible local deployment.

System Architecture & Implementation

- Designed a relational model for users, products, and purchase transactions.
- Implemented REST APIs with Spring Boot, Hibernate/JPA, and JWT-secured endpoints.
- Built a responsive Angular frontend and containerized the full stack with Docker.

Outcome & Impact

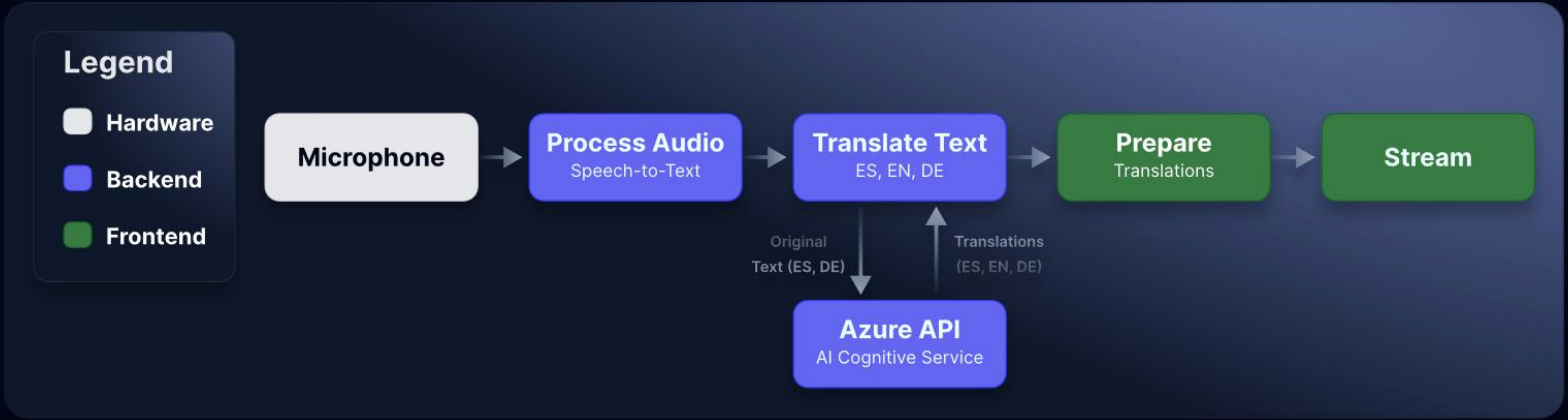
- Delivered a functional end-to-end e-commerce app with persistent transactions.
- Demonstrated full-stack development across backend, frontend, and database layers.
- Established a deployable architecture suitable for local and iterative development.

Real-Time Multilingual Speech Translation System

Description & Architecture

Real-Time Multilingual Speech Translation System

Flowchart



Legend

⬜ Hardware

🟦 Backend

🟩 Frontend

Microphone → Process Audio (Speech-to-Text) → Translate Text (ES, EN, DE) → Prepare Translations → Stream

Translate Text → Original Text (ES, DE) → Azure API (AI Cognitive Service)

Azure API → Translations (ES, EN, DE) → Translate Text

Real-Time Multilingual Speech Translation System

Introduction

Context & Challenge

Live events are often inaccessible to attendees who do not speak the primary language. A real-time system was needed to capture live speech, transcribe it, translate it into multiple languages, and display readable captions on external screens during the event.

Objective & Role

Build an application capable of real-time speech transcription, multilingual translation, and browser-based caption display.
Role: Full-Stack Developer responsible for audio capture, transcription, translation, and visualization.

Real-Time Multilingual Speech Translation System

Methodology

Solution Approach

A Python pipeline was designed to capture live audio, transcribe speech, translate text, and render captions in real time. The system was built to support interchangeable input and output languages.

System Architecture & Implementation

- Implemented continuous audio capture and streaming into a speech-to-text pipeline using Azure services.
- Integrated speech-to-text and translation services to generate multilingual near-real-time captions.
- Built a Streamlit-based frontend to display live captions in the browser per target language.

Outcome & Impact

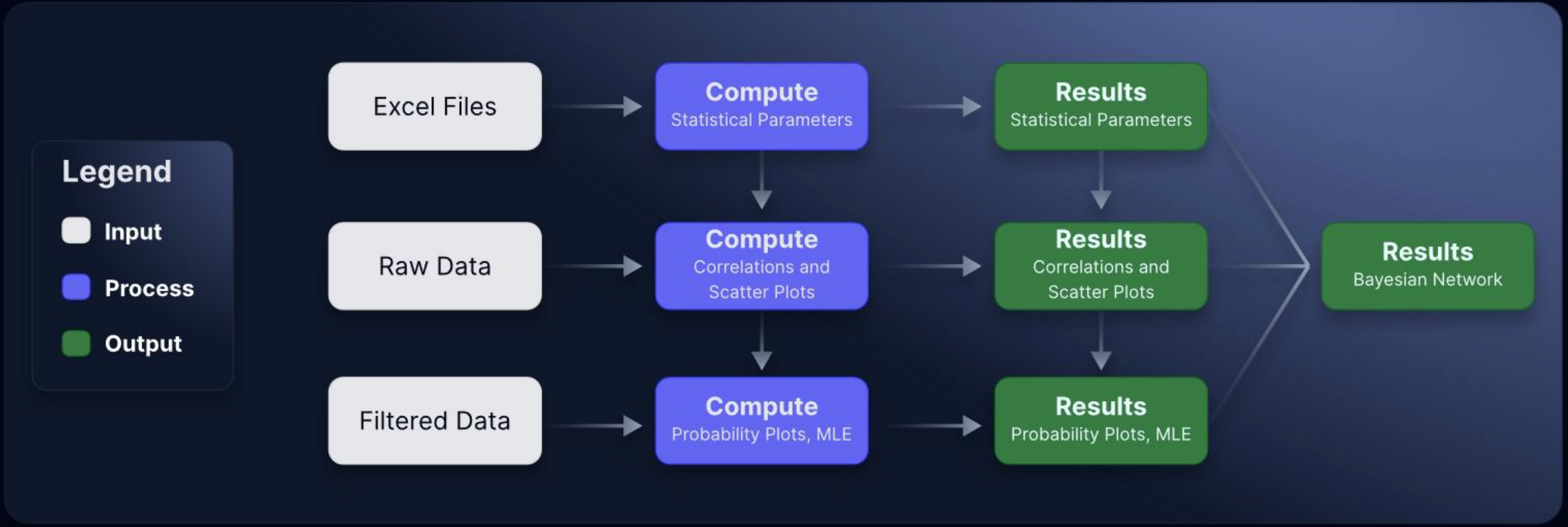
- Delivered a reusable app for real-time multilingual captioning at live events.
- Enabled non-native speakers to follow spoken content through translated captions.
- Integration of audio streaming, speech-to-text, machine translation, and visualization.

Probabilistic Reliability Modeling for Water Treatment

Description & Architecture

Probabilistic Reliability Modeling for Water Treatment

Flowchart



Probabilistic Reliability Modeling for Water Treatment

Introduction

Context & Challenge

Multi-stage wastewater treatment systems must meet strict contaminant removal requirements under significant operational variability and measurement uncertainty. Assessing system-level compliance requires propagating uncertainty across all treatment stages.

Objective & Role

Develop a probabilistic reliability model to estimate the likelihood that the full treatment train meets contaminant removal targets.
Role: responsible for data processing, stochastic modelling, Bayesian network design, and reliability analysis.

Probabilistic Reliability Modeling for Water Treatment

Methodology

Solution Approach

A data-driven probabilistic workflow combining stochastic simulation and Bayesian inference was designed to propagate uncertainty through the full treatment train and quantify compliance probabilities.

System Architecture & Implementation

- Processed and fitted statistical distributions to raw field data using MATLAB.
- Generated correlated stochastic samples via the Nataf transformation.
- Integrated stochastic inputs into a Bayesian network in GeNIe to model inter-stage dependencies.

Outcome & Impact

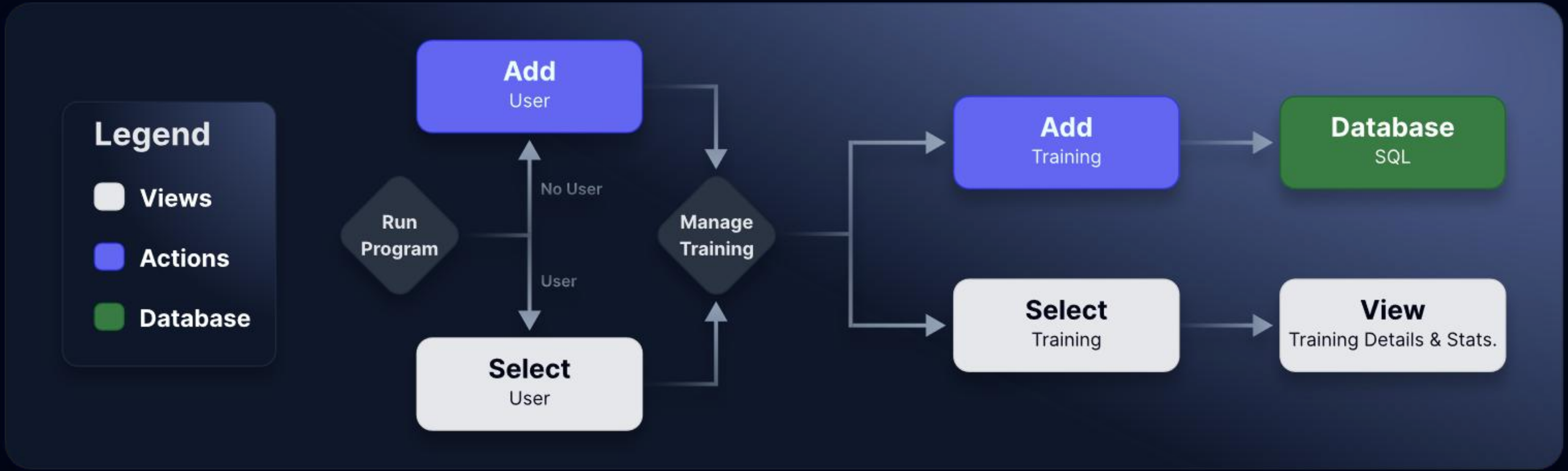
- Predicted treatment performance and compliance under uncertainty.
- Quantified reliability per treatment stage and for the full system.
- Identified dominant uncertainty drivers to support risk-based optimization.

Training Log Management System

Description & Architecture

Training Log Management System

Flowchart



Training Log Management System

Introduction

Context & Challenge

Tracking workouts in notebooks or spreadsheets leads to unstructured data, limited analysis capabilities, and poor portability across storage systems. A lightweight solution was needed to reliably store training data and support historical and volume-based analysis.

Objective & Role

Design and implement a training log system that supports multiple users, persists data in SQL databases, and enables structured inspection of training history and statistics.
Role: responsible for data modeling, application logic, and database integration.

Training Log Management System

Methodology

Solution Approach

A Python-based console application was designed with a clear separation between application logic and data access, allowing the same core logic to run on different SQL backends by swapping only the database module. A menu-driven interface guides users through account management, loggings, and data inspection.

System Architecture & Implementation

- Relational data model with users, exercises, and training logs linked via foreign keys.
- Core application logic implemented to handle user flows in Python, SQL queries and visualizations separately.

Outcome & Impact

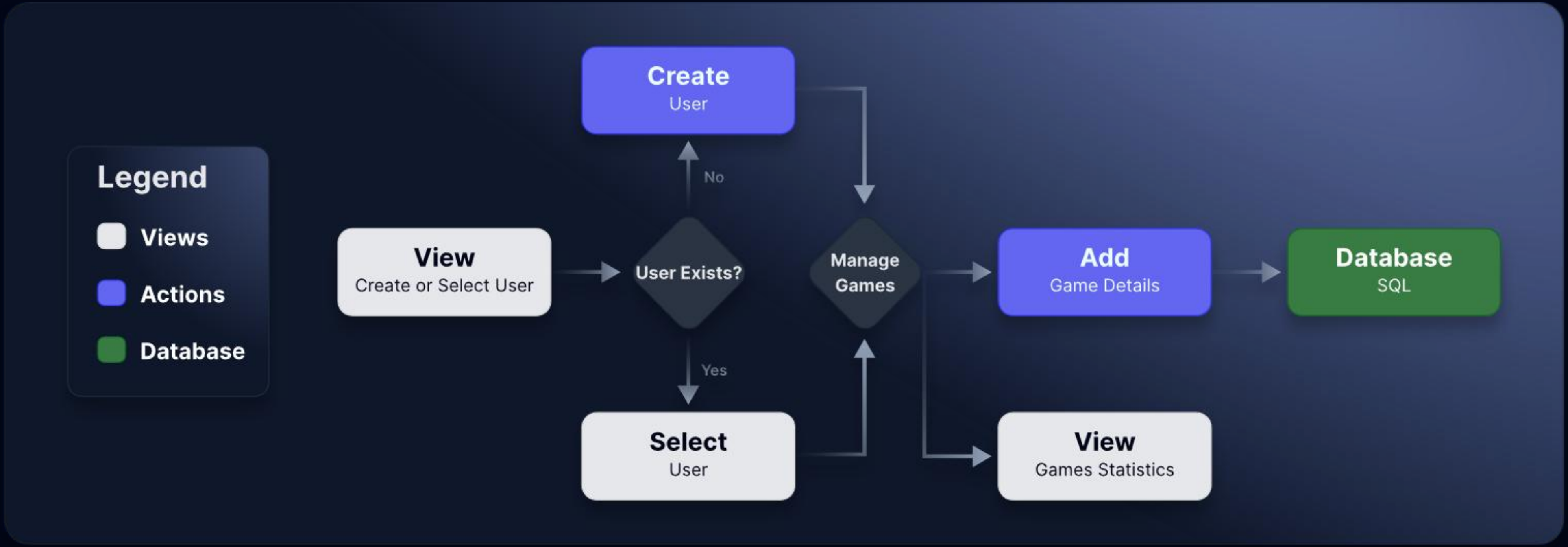
- Delivered a working, portable prototype for logging and querying structured training data.
- Enabled historical and time-window-based analysis of training volume and progression.

Game Inventory & Progress Tracker

Description & Architecture

Game Inventory & Progress Tracker

Flowchart



Game Inventory & Progress Tracker

Introduction

Context & Challenge

Tracking game collections manually in lists leads to fragmented data and limited searchability. A structured local app was needed to persist game metadata, track game status and support reliable queries.

Objective & Role

Build a desktop application to manage game metadata, user status, and progress with persistent storage in SQL.
Role: responsible for data model, application logic, UI, and SQL integration.

Game Inventory & Progress Tracker

Methodology

Solution Approach

A local desktop application designed with a clear separation between application logic and persistent storage, enabling structured tracking of game collections and user progress.

System Architecture & Implementation

- Designed a relational SQLite schema for users, games, statuses, and user–game relationships.
- Implemented a C++ data access layer handling CRUD operations via the SQLite C++ API.
- Developed application logic for game status transitions, catalog queries, and wishlist management.

Outcome & Impact

- Delivered a functional desktop application for tracking games and their respective status.
- Established a scalable foundation for future features such as ratings, tags, or analytics.

Contact Details

Juan Miguel Rodriguez Ramirez

juanm.rodriguez.dev@gmail.com

www.test.com