Project-X: Development Roadmap

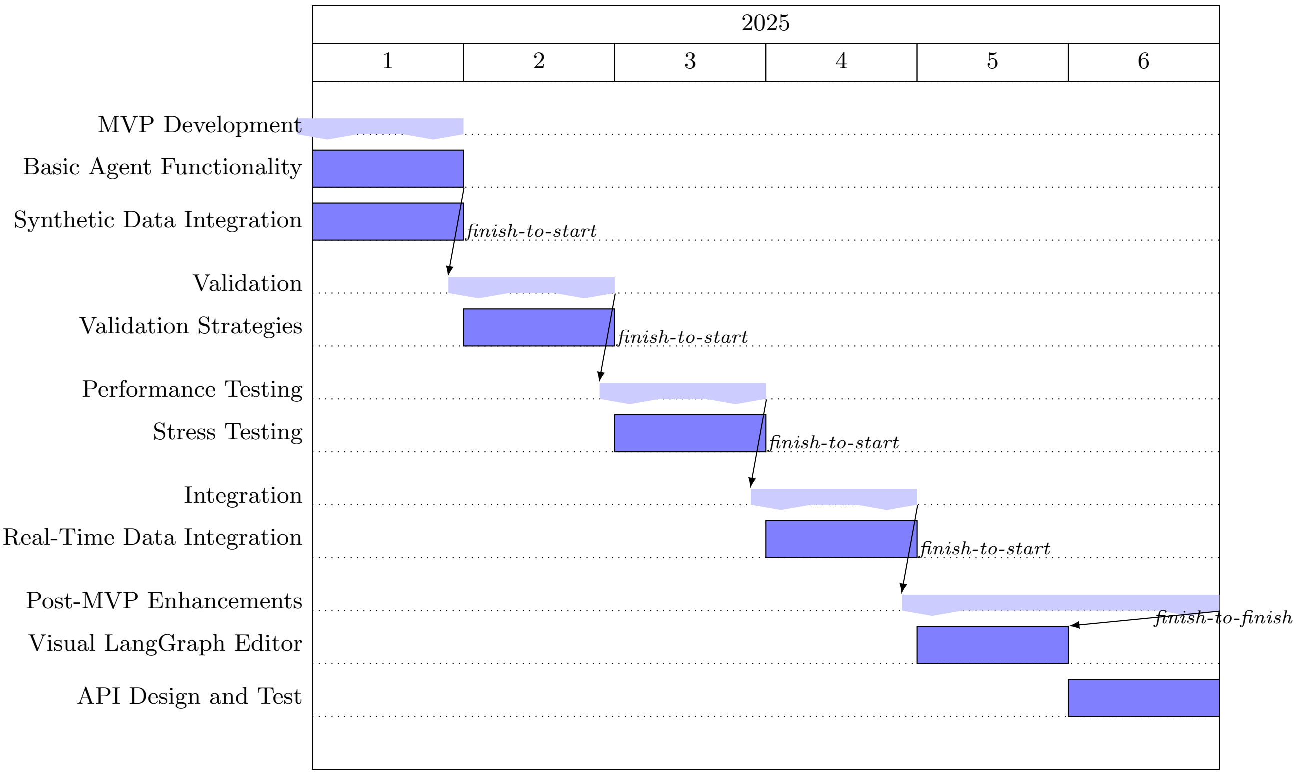
Alberto Espinosa
KSquare Group

July 21, 2025

This document outlines the roadmap for the Project-X agent-based architecture, detailing milestones, timelines, and dependencies to guide development from MVP to full real-data integration, supporting use cases such as fraud detection, supply chain optimisation, and customer analytics.

1 Roadmap

The following roadmap is defined in blocks of one week, starting Monday, 21 July 2025, and concluding on Friday, 29 August 2025.



2 Roadmap Stage Descriptions

This section provides an overview of each stage in the Project-X development roadmap, elucidating the objectives and expected outputs. The descriptions are structured to guide developers through the sequential progression of the agent-based architecture, ensuring clarity and alignment with the project's goal of delivering a functional minimum viable product (MVP) within six weeks. Each stage builds upon the previous, culminating in a cohesive system capable of processing datasets, generating synthetic data, training machine learning models, applying business rules, and delivering comprehensive reports.

2.1 MVP Development (Week 1)

Objective: Implement the core functionality of all eight agents (User Prompt Parser, Code Generation for Synthetic Data, EDA, PyCaret Model Running, Model and Results Interpreter, Business Rule, Configuration, Report Formatter) independently, ensuring each operates correctly in isolation outside the LangGraph environment. This includes developing synthetic data generation capabilities to support subsequent pipeline stages.

Tasks:

- **Basic Agent Functionality**: Develop the standalone logic for each agent using Python and pre-trained Ollama models, ensuring individual functionality for tasks such as prompt parsing, model training, and report generation.
- **Synthetic Data Integration**: Implement the Code Generation for Synthetic Data Agent to produce well-formed synthetic datasets based on user-specified descriptions, ensuring compatibility with downstream analysis.

Expected Outputs:

- Functional agents, each producing standalone outputs, including:
  - JSON configurations from the User Prompt Parser Agent (e.g., {`"columns": [{"name": "transaction_id", "type": "int"}, {"name": "is_fraud", "type": "bool"}], "target": "is_fraud", "rows": 1000`}).
  - Synthetic datasets in CSV format from the Code Generation Agent (e.g., `data.csv` with columns like `transaction_id, amount, country, is_fraud`).
  - JSON reports from the EDA Agent (e.g., {`"columns": [...], "target_suitable": true`}).
  - Trained models (e.g., `model.pkl`), metrics (e.g., {`"accuracy": 0.85`}), and plots (e.g., `confusion_matrix.png`) from the PyCaret Model Running Agent.
  - JSON performance reports from the Model and Results Interpreter Agent.
  - Modified datasets (e.g., `data_ruled.csv`) and rule summaries from the Business Rule Agent.
  - JSON configurations from the Configuration Agent (e.g., {`"pycaret_model": "random_forest"`}).
  - Unified JSON reports from the Report Formatter Agent.

2.2 Validation (Week 2)

Objective: Ensure seamless integration of all agents within the LangGraph pipeline by synchronising their inputs and outputs, validating that each agent correctly processes the previous agent's output to produce usable results for the next.

Tasks:

- **Validation Strategies**: Integrate all agents into the LangGraph environment, testing the flow of data through the pipeline (e.g., JSON prompts to synthetic datasets to model training) to confirm compatibility and correctness of input-output handoffs.

Expected Outputs:

- A fully integrated LangGraph pipeline with all eight agents, producing a cohesive sequence of outputs, including JSON configurations, synthetic datasets, EDA reports, trained models, performance metrics, rule-applied datasets, configuration files, and unified reports, validated for the five predefined use cases.

2.3 Performance Testing (Week 3)

Objective: Evaluate the performance and robustness of the integrated pipeline by conducting stress tests across the five predefined use cases, ensuring reliability under varying conditions.

Tasks:

- **Stress Testing**: Execute the integrated pipeline, including the PyCaret Model Running Agent, on the five use cases (e.g., fraud detection, customer analytics), assessing performance metrics and system stability under different dataset sizes and complexities.

Expected Outputs:

– Validated pipeline outputs, including saved machine learning models (e.g., `model.pkl`), performance metrics in JSON format (e.g., {`"accuracy": 0.85, "auc": 0.90`}), and visualisation plots (e.g., `confusion_matrix.png`), confirming robustness across use cases.

2.4 Integration (Week 4)

Objective: Incorporate domain-specific business rules into the pipeline to enhance dataset utility and align with use case requirements.

Tasks:

– **Real-Time Data Integration**: Deploy the Business Rule Agent to apply predefined business rules (e.g., flagging transactions based on country or amount thresholds) to datasets, generating modified datasets with rule-based annotations.

Expected Outputs:

– Modified datasets in CSV format (e.g., `data_ruled.csv` with additional columns like `suspicious`) and JSON rule summaries (e.g., {`"rule": "Flag transactions from countries X, Y", "impact": "10% transactions flagged"`}).

2.5 Post-MVP Enhancements (Weeks 5–6)

Objective: Augment the system with advanced functionalities and finalise the integration of an API to deliver comprehensive, user-accessible outputs.

Tasks:

– **Visual LangGraph Editor**: Develop the Configuration Agent to provide a text-based interface for adjusting parameters (e.g., model types, rule thresholds), enhancing system flexibility and user interaction.

– **API Design and Test**: Implement the Model and Results Interpreter and Report Formatter Agents to produce unified reports and validate the API for seamless delivery of pipeline outputs.

Expected Outputs:

– Configuration JSON files (e.g., {`"pycaret_model": "random_forest", "fraud_threshold": 500`}) generated by the Configuration Agent.

– Unified JSON reports (e.g., {`"eda": {...}, "model": {"accuracy": 0.85}, "rules": {...}, "summary": "Robust system for fraud detection"`}) and a validated API for report delivery.

3 Team Composition

Name	Role	Agent	Comments
Rafael	ML and APIs for Models, AI-Dev	***	
Andrés	Frontend, AI-Dev	***	
Carlos	Frontend, AI-Dev	***	Integrator
Mauricio	Backend, AI-Dev	***	
Johnattan	DevOps, AI-Dev	***	
Gastón	Tech Lead, AI-Dev	***	
Jan	AI–developer	***	
Enrique	AI–developer	***	

Table 1: Team Members and Assigned Agents