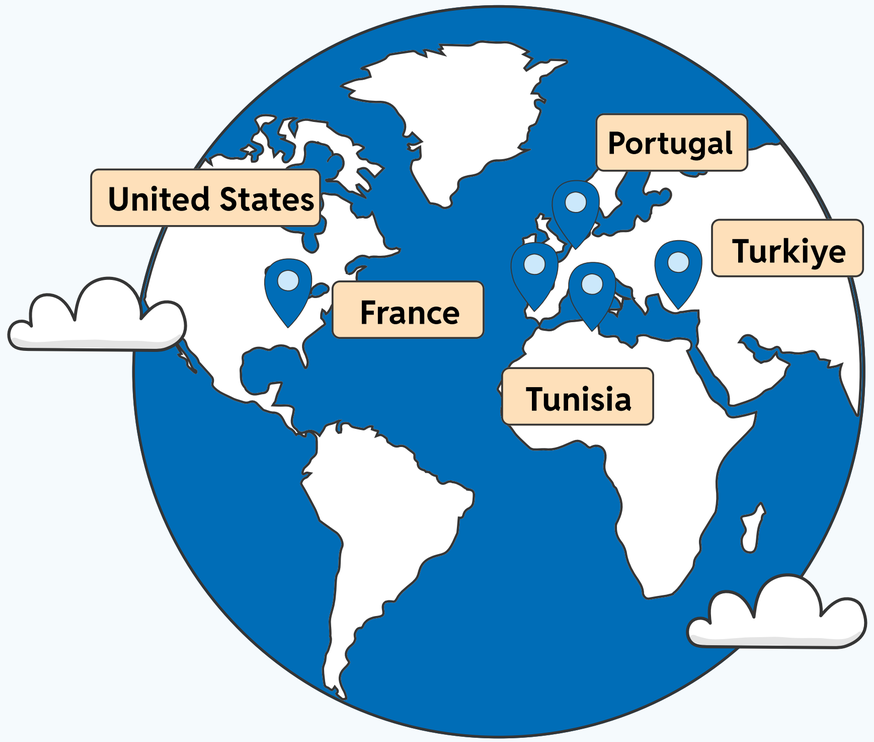
End of studies projects 2027

Tunisia

COGNIRA

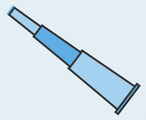


Cognira is the leading promotion management solution provider for retailers and wholesalers, making it easy to collaborate across departments, analyze past performance and run smarter, more effective promotions.
Our single promotion solution – PromoAI – leverages data science and AI to effectively manage the entire end-to-end promotion lifecycle.

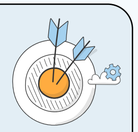


Vision

Cognira's vision draws on the company's desire to bridge the gap between retail and science in an easy, accessible way.



Mission



Our **mission** is to make data science effortless and approachable for our customers and provide business insights that maximize efficiency and ROI.

internships Topics



- Data Science Consulting
- AI-Powered Release Documentation
- AI-Assisted Log Analysis and Incident Reproduction
- AI-powered test management tool, from requirements to risk-based regression
- AI-based framework for Hybrid API & UI Performance Testing (k6 / xk6)
- Test-Impact Selection Service
- Human In The Loop
- Semantic Similarity
- Data Pipelines with DuckDB
- Interactive MCP Apps for PromoAI's Agent Cora
- Blueprint: A Visual Screen Builder with AI-Ready Design Handoff
- Cognira AI Agent for Forecast Debugging
- Deep Learning for Promotional Demand Forecasting
- Agentic DAST and LLM Red Teaming

Data Science Consulting

Project Description

Assist senior Data Science Consultants with delivery of new client implementations

Expected Deliverables

1) Check client data feeds against the **CDF** (completeness, mappings, promo detection)
2) Build and test Databricks notebooks and jobs for ingestion and feature tables
3) Review forecast accuracy and help tune models for seasonal and category exceptions
4) Build Merchant Insights dashboards and weekly reports
5) Write release test cases and check results after each release
6) Prepare data exploration summaries and client-facing slides

Required Skills

1) Data analytics and reporting experience
2) Python (pandas) and SQL
3) Statistics and forecasting basics (regression, time series, error metrics such as MAPE and bias)
4) Data validation and quality checking
5) Clear written and spoken English for working with client teams

Nice-to-have Skills

1) Databricks / PySpark and Delta tables
2) Retail or CPG knowledge (promotions, pricing, merchandising)
3) Dashboards and data visualization
4) Git / GitHub
5) Using AI coding assistants (e.g., Claude Code) for analysis and testing
6) Causal inference methods (e.g., difference-in-differences)

AI-Powered Release Documentation

Project Description

Problem: Product teams spend a large share of every release turning scattered knowledge (tickets, specs, design files, meeting decisions) into documentation such as release notes, feature guides and knowledge bases. The work is manual, repetitive and inconsistent, and it slows the team down as the product grows.
Intent: Design and build AI-assisted workflows that generate and check documentation from the team's existing tools, so PMs review and approve instead of writing from scratch. The intern will apply this to Cognira's product team and leave behind a repeatable process.

Expected Deliverables

1. A map of the current documentation flow (sources, owners, cadence, gaps) and a prioritized automation plan.
2. An internal release notes workflow that drafts notes from a release's Jira scope for PM review.
3. A structure and content for Cora's knowledge base, refreshed each release, plus a golden question set with verified answers to measure Cora's accuracy. Ingestion is handled by the Cora engineering team.
4. A recurring check that flags drift between stories, configuration specs and published docs.
5. PM/UX agents for recurring work, UX first (e.g., drafting design specs from user stories), with usage guides; stretch goal: a backlog management assistant for duplicate detection and grooming.
6. A runbook and handover so the team can run every workflow without the intern, plus a final demo and a short report on time saved and quality.

Required Skills

- **Structured thinking:** able to map a process into clear steps, inputs and outputs; excellent structured writing in English; hands-on experience building something with AI beyond chat (custom assistants, prompts, scripts or agents); a critical eye for AI output, autonomy and ownership in a small team; product, business analysis or business informatics background.

Nice-to-have Skills

- Experience building with LLM APIs or agent frameworks (e.g. OpenAI/ Anthropic APIs, MCP); understanding of how knowledge bases and RAG retrieve content; LLM output evaluation (test question sets, accuracy checks); basic Python for scripting and API integration; familiarity with Jira, Confluence or similar tools, UX thinking

AI-Assisted Log Analysis and Incident Reproduction

Project Description

The objective of this internship is to develop an AI-assisted tool that helps the Customer Support (CS) team investigate client-reported incidents by analyzing application logs, reconstructing the sequence of events leading to unexpected behavior, and generating evidence-based reproduction steps and investigation reports.
The tool will analyze authorized application logs to identify relevant errors, failed operations, anomalies, and correlated events. By combining log evidence with incident descriptions, it will help reconstruct what happened, identify potential root causes, and generate structured bug report drafts containing relevant evidence, reproduction steps, and missing information.

Beyond individual investigations, the project aims to transform the findings from investigated incidents into a structured and reusable knowledge base. This repository will progressively consolidate historical client issues, affected modules, reported symptoms, log patterns, reproduction steps, confirmed root causes, resolutions, and related tickets where available.
Building on this knowledge base, the project will explore an AI-assisted incident classification mechanism that can categorize new client issues and identify similar previously investigated cases. By comparing new incidents with historical findings, the tool will suggest relevant problem categories, potential root causes, and previously documented investigation paths or resolutions, helping the CS team recognize recurring issues and reduce duplicated troubleshooting efforts.
The overall objective is to establish a continuous workflow in which log analysis supports incident investigation, the results enrich historical incident knowledge, and that accumulated knowledge improves the classification and investigation of future issues.

The solution will operate in read-only mode using authorized data sources and approved historical or test incidents. All generated findings and classifications will remain subject to human review, with a clear distinction between observed evidence, inferred hypotheses, and confirmed root causes.
The internship will last four months, with the goal of delivering a functional proof of concept demonstrating the feasibility and value of this integrated approach on a selected set of incidents.

AI-Assisted Log Analysis and Incident Reproduction

Expected Deliverables

1. Process authorized application logs to identify errors, anomalies, and related events.
2. Incident Reconstruction and Reproduction
Reconstruct incident timelines and generate evidence-based reproduction steps and potential root causes.
3. AI-Assisted Investigation and Bug Reporting
Generate structured investigation summaries and bug report drafts with supporting evidence, reproduction steps, and confidence indicators.
4. Historical Incident Knowledge Base
Build a structured repository of investigated incidents, including symptoms, log patterns, root causes, reproduction steps, and resolutions.
5. Incident Classification and Historical Matching
Classify new incidents into relevant categories and identify similar historical cases, potential root causes, and documented resolutions.
6. CS-Friendly Interface
Provide a simple interface or structured output for reviewing investigation results, evidence, and classification suggestions.
7. Evaluation and Documentation
Evaluate the tool's accuracy, usefulness, and potential time savings, and document its architecture, limitations, and future improvements.

Required Skills

- Basic to intermediate programming skills in Python.
- Understanding of log files, application errors, exceptions, and debugging concepts.
- Knowledge of data processing, text analysis, and pattern recognition.
- Familiarity with APIs, databases, and structured data formats such as JSON.
- Analytical thinking and strong problem-solving skills.
- Ability to document technical findings clearly and translate them into actionable investigation steps.



AI-Assisted Log Analysis and Incident Reproduction

Nice-to-have Skills

- Experience with Large Language Models (LLMs), AI agents, or generative AI APIs.
- Familiarity with Retrieval-Augmented Generation (RAG), embeddings, or semantic search.
- Knowledge of anomaly detection, time-series analysis, or event correlation.
- Experience with observability and logging tools such as Grafana, Kibana, Elasticsearch, or the ELK stack.
- Familiarity with SQL and database querying.
- Understanding of SaaS applications, incident management, or software testing.
- Experience integrating AI tools into existing workflows.

AI-powered test management tool, from requirements to risk-based regression

Project Description

Before development: an AI assistant reviews the Jira user stories and flags vague, missing or untestable acceptance criteria and forgotten edge cases, with clarifying questions for the PO.
During testing: writes structured test cases with reusable preconditions, links them to Jira stories, groups them into test executions per release, and records results with evidence and linked bugs. Traceability with the automated Playwright scenarios is kept throughout.
Before each release: the assistant learns from past releases, bugs and test results to predict the riskiest features. It orders the regression scenarios so the riskiest are tested first and suggests new targeted scenarios, which QA reviews before adding them.

==> The result is one shared tool that helps QA catch gaps early, test smarter, and let fewer regressions reach clients

Expected Deliverables

- Requirements and design document, validated with the QA team
- **Test management module:** test cases, preconditions, executions, results and history, Jira links
- Migration of existing test cases, plus traceability with automated tests
- **AI story review:** unclear or missing acceptance criteria, testability score
- **AI regression prioritization:** risk score per feature and suggested new scenarios
- QA review screen for AI suggestions
- Pilot on a real release, with metrics
- Integration into the Cognira Test Hub

Required Skills

- TypeScript and a modern frontend framework
- REST API design
- Data modeling and databases (SQL)
- Python
- Basics of AI / LLMs and machine learning
- Git

Nice-to-have Skills

- Jira API
- NLP / embeddings, retrieval (RAG)
- QA / test management concepts
- Playwright
- UX design
- Azure

AI-based framework for Hybrid API & UI Performance Testing (k6 / xk6)

Project Description

- A custom k6 binary built with xk6, shipped as a versioned Docker image.
- A TypeScript framework with reusable API, browser and hybrid scenarios (smoke, load, stress, spike, soak).
- An in-house Go extension for k6 (auth tokens shared across virtual users).
- One configuration profile per client environment, with SLOs and load limits.
- **GitHub Actions workflows:** on demand, nightly and after each deployment.
- Grafana dashboards, an HTML report per run, and a first baseline for each client environment.
- Technical documentation and the internship report.

Expected Deliverables

- **A custom k6 distribution:** an xk6 build workflow in GitHub Actions, a versioned Docker image and a changelog.
- **A framework and scenario library:** a TypeScript core, API and browser journeys, hybrid scenarios and workload profiles.
- **An in-house xk6 extension:** Go source, unit and integration tests, and usage examples.
- **Environment profiles:** one profile per client environment, with SLOs and load budgets agreed with their owners.
- **CI/CD workflows:** on-demand, scheduled and post-deployment runs, with one job per environment.
- **Dashboards and reports:** Grafana dashboards for a single run, trends and cross-client comparison, and an HTML report per run, published next to the Allure reports.
- **A first baseline campaign:** baseline results per client environment for the critical journeys.
- **Documentation:** architecture, “add a scenario” and “add an environment” guides, a safety runbook and the internship report.

Required Skills

- JavaScript / TypeScript
- HTTP and REST APIs
- Docker and Git

- CI/CD basics
- Willingness to learn Go



Test-Impact Selection Service

Project Description

A tool that takes a code change in a Scala repository and decides which tests need to run. Today every change runs the whole test suite, even when it touches one small area. The
tool works out which classes the change affects, finds the tests that depend on them, and runs only those, with a reason for each choice.

Expected Deliverables

- A working tool that selects the tests affected by a code change and explains each choice
- A replay report on past commits showing no missed failures and less test time
- Integration into our CI
- Documentation and a final demo

Required Skills

- Scala
- PostgreSQL
- SemanticDB

Nice-to-have Skills

- Git

Human In The Loop

Project Description

Full-stack app enhancing AI development experience by assisting the developer review code changes. We are in the AI era, code is being written faster than ever, we should make use of it as much as we can but with care and control. In order to achieve said goals, the app should allow the developer to comment on different blocks of code, ask for clarification, pull request walk through...

Required Skills

- Go / Python
- PostgreSQL
- React



Semantic Similarity

Project Description

Use an LLM to read product descriptions and assess with they should be considered substitues, complements, "super-SKU" components, or how they should fit into a product hierarchy together.

Expected Deliverables

A science module that groups products into likely complements, substitutes and assembles a product hierarchy based on reading the descriptions of products.

Required Skills

- Data Analytic
- Statistics
- Python

Nice-to-have Skills

- Databricks
- Pyspark
- Knowledge of LLM techniques
- Natural Language Parsing familiarity

Data Pipelines with DuckDB

Project Description

Our data pipelines run every transformation with Apache Spark on cloud clusters, even when the data is small enough for one machine. This project moves suitable transformations to DuckDB, a fast SQL engine that runs on one machine, and lets them run on a developer's laptop as well as in the cloud. The goal is faster and cheaper runs, and a shorter cycle to develop and test.

Expected Deliverables

- A short comparison of Spark and DuckDB on a few existing transformations, used to select which ones to move.
- DuckDB versions of selected transformations, with automated tests that prove they give the same output as the Spark versions.
- A tested way to run DuckDB transformations on Databricks, our cloud data platform, including the compute type, the dependencies and the access to our data. If Databricks does not support this directly, the project delivers a workaround.
- Changes to our pipeline orchestration, the system that defines and schedules each step, so that each step declares whether it runs on Spark or on DuckDB.
- One command that runs a DuckDB transformation on a laptop, with sample data or with data read directly from cloud storage.
- One version of the code for each transformation, which runs both locally and on Databricks without changes.

Required Skills

- Python
- SQL

Nice-to-have Skills

- Databricks
- Pyspark
- DuckDB

Interactive MCP Apps for PromoAI's Agent Cora

Project Description

PromoAI is a SaaS platform that helps retailers plan, forecast and analyze their promotions. Its AI agent, Cora, answers planners' questions in plain language, like which promotions clash with an event or why a forecast looks low.
Cora can already be used from other AI apps, such as Claude, through the Model Context Protocol (MCP). There, its answers arrive as plain text. MCP Apps, a recent extension of the protocol, lets a tool return a small web app that the AI app shows inside the chat.
You will use it to make Cora's answers interactive wherever people use them. That means charts people can explore, tables they can sort and export and pickers to choose exact products and stores. You will work across a TypeScript backend and a library of small web components, and test them in real AI apps.

Expected Deliverables

- **MCP Apps support in Cora's MCP server:** interactive views linked to its tools, with a complete text answer for apps that cannot show them.
- **A library of views:** answer cards with charts and tables, promotion and event cards, a forecast breakdown chart and a promotions calendar.
- Views that call the server directly, skipping the AI model: product and store pickers, an alerts inbox, and an exportable data table.
- Styling that follows each AI app's theme, in light and dark mode.
- Automated tests for the views and server changes, and a compatibility report for Claude, ChatGPT, Gemini and VS Code.
- Stretch goal: one component library shared by PromoAI's Cora own chat panel and the external AI apps.

Required Skills

Web stack familiarty :
- Typescript/Javascript
- React
- Vite, Nginx

Fundamental Knowledge of:
- LLM tool calling
- AI agents
- MCP
- Automated testing

Nice-to-have Skills

- Experience with Python
- Web security basics: iframes, Content Security Policy and OAuth.
- Building charts and data tables for the web.

Blueprint: A Visual Screen Builder with AI-Ready Design Handoff

Project Description

Our design system is documented in Storybook. Today, Product Managers and UX designers cannot assemble screens from these components without help from engineers. Handoff from design to engineering is also manual and easy to misread.
The intern will build a visual screen builder on top of our Storybook. In the builder, PM and UX users choose components from a catalog, place them in predefined layouts, and set their properties (title, variant, size, and more) through generated controls. The builder reads the component list and prop definitions directly from Storybook and TypeScript metadata, so it stays in sync with the component library.
Each screen is saved as a structured specification (JSON) that describes the component tree and the props of each component. The intern will then make this specification available to AI coding agents through an MCP (Model Context Protocol) server. Agents can then read, validate, and implement the screens with the real components and props.

Expected Deliverables

1. A builder application (a Storybook addon or a standalone app) with:
- A component catalog generated from Storybook metadata
- A canvas that renders live components inside predefined layouts
- A props panel generated from component type definitions
- Save and load of screens

2. A screen specification format:
- A JSON Schema generated from component prop definitions
- Validation of saved screens against the schema
- A Markdown export that both humans and agents can read

3. An MCP server that exposes the screens

Required Skills

- React and TypeScript
- Basic knowledge of component-based UI and design systems
- JSON and JSON Schema
- Git and code review workflow
- Ability to write clear technical documentation

Nice-to-have Skills

- Storybook (stories, args, addons)
- Experience with MCP or with LLM tool calling
- Node.js backend development
- MUI or Kendo React UI
- Drag-and-drop libraries (for example, dnd-kit)
- Vitest and Testing Library
- Interest in UX and design-to-code workflows

Cognira AI Agent for Forecast Debugging

Project Description

Use an LLM agent to investigate differences between our batch and real-time forecasts: it explores the data, configurations and features with read-only tools, and proposes the most likely root cause with supporting evidence.

Expected Deliverables

A prototype debugging agent, its toolset, and an evaluation framework measuring how often it finds the correct root cause on known issues.

Required Skills

- Python, Data Analytics
- Software Engineering
- Databricks
- PySpark,
- Scala
- LLM agents



Deep Learning for Promotional Demand Forecasting

Project Description

Benchmark modern deep learning forecasting models (pretrained time-series foundation models and transformer-based models) against our production models, and explore how richer product information can improve forecasts for products with limited promotional history.

Expected Deliverables

A reproducible benchmark of deep learning models vs production, with an analysis of where they improve promotional forecasts.

Required Skills

- Machine Learning
- Deep Learning
- Python
- Statistics
- Databricks
- PySpark
- PyTorch
- Time Series Forecasting
- LLM



Agentic DAST and LLM Red Teaming

Project Description

Cognira builds AI-powered promotion planning software for retailers. Users reach the platform in two ways: through its web APIs, and through an AI assistant that answers questions and acts on their behalf, including through the Model Context Protocol (MCP). Both paths lead to the same data, so a weakness behind one can often be exploited through the other.

Expected Deliverables

In this internship, you will build a test harness that attacks the platform through both paths and runs automatically on every release, in an isolated environment with synthetic data. You will start with authenticated API scanning in CI/CD, then generate authorization tests across users, roles and tenants. Next, you will red team the AI assistant for prompt injection, tool misuse and data leakage, including attacks hidden in the data it reads. Finally, you will build an AI attacker agent that chains multi-step attacks.

Required Skills

- Web and API fundamentals: HTTP, REST, JSON, cookies, sessions and tokens, and the difference between authentication and authorization
- Web application security: OWASP Top 10, common vulnerability classes (broken access control, injection, XSS, SSRF)
- Basic knowledge of how large language models work and how applications call them through an API
- Linux, Git, Docker and Kubernetes basics
- Good written English: you will read technical documentation and write findings that developers can reproduce



We look forward to discovering what we could achieve together.

COGNIRA

