ISABEL

MINIMIZE COSTS AND MAXIMIZE BENEFITS IN THE TRANSITIONARY EUROPEAN REGIONAL LABOUR MARKETS

mInimize coSts and mAximize Benefits in the transitionary European regional Labour markets

Deliverable No 3.3

Initial version of SNAP

WP3: Data Lake creation and Data Collection on the uneven effects of JCD

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

DOCUMENT ID	D3.3.
DUE DATE	30/06/2026
DELIVERY DATE	30/06/2026
WORK PACKAGE	WP3
TASK(S)	T3.3
DISSEMINATION LEVEL	PU-Public
VERSION	V2
AUTHOR/S	CERTH
FILENAME	ISABEL_D3.3_V2

Abstract:
This deliverable presents the foundational software architecture for the initial version of the SNAP (Smart oNe-stop digitAl data shoP) ecosystem, developed under Work Package 3. SNAP is an integrated, backend-centric research infrastructure designed to facilitate synthetic demographic simulation, automated academic literature discovery, and unstructured data extraction. By fusing macroeconomic labour statistics from Eurostat with granular occupational taxonomies from ESCO, the system provides a mathematically rigorous framework for modelling labour market dynamics and evaluating workforce transitions towards green and digital economies.
To support data sovereignty and compliance with European privacy frameworks, the architecture employs a "local-first" model orchestration paradigm. It leverages localised Large Language Models (LLMs) and advanced Retrieval-Augmented Generation (RAG) engines to ensure that all generative reasoning and semantic memory persistence remain on infrastructure controlled by the project, so that no labour-market data is sent to commercial AI services (see D2.5 and D5.3). This document details the multi-source data fusion mechanisms, the Medallion storage strategy, and the algorithmic logic of the toolkit's four core subsystems, ultimately delivering a secure, methodologically transparent, and reproducible computational infrastructure for the European research community.

Views and opinions expressed in this document are those of the author(s) and do not necessarily reflect those of the European Union or REA. Neither the European Commission nor the granting authority can be held responsible for them.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Contributors

NAME	ORGANISATION
Nicholas Vretos	**CERTH**
Michalis Angelou	**CERTH**
Symeon Samaras	**CERTH**
Vasileios Rekkas	**CERTH**
Vasileios Solachidis	**CERTH**

Peer Reviewers

NAME	ORGANISATION
Andre Carrascal Incera	UO
Mladen Stamenkovic	UB

Versions

Version	Date	Author / Organisation	Description
ToC	15/05/26	Simos/CERTH	Table of Contents
Version 1	08/06/26	Simos/CERTH	Initial version of D3.3
Version 2	29/06/26	CERTH	Integrate comments, update document

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Executive summary

Understanding the complex dynamics of the European labour market, particularly concerning the uneven effects of the transition towards green and digital economies, requires advanced computational infrastructures capable of synthesising multidimensional data. Deliverable 3.3 documents the initial software iteration of the SNAP research ecosystem, providing a comprehensive technical specification of its backend architecture, data orchestration pipelines, and operational capabilities.

Designed as a highly modular, API-driven computational ecosystem, SNAP deliberately decouples core processing logic from user interface layers. This architectural decision ensures that the underlying computational methodologies remain transparent, reproducible, and highly scalable. The primary objective of this infrastructure is to demonstrate a viable methodology for fusing official European Union statistical data (Eurostat) with sophisticated occupational frameworks (ESCO and ISCO). By executing probabilistic mathematical mappings, the system translates broad employment distributions into highly specific skill profiles, establishing a scientifically grounded baseline for labour market simulation.

A defining pillar of the SNAP architecture is its strict adherence to data sovereignty and privacy-preserving computation. Recognising the regulatory vulnerabilities inherent in processing sensitive demographic models and proprietary academic literature through commercial, cloud-based Large Language Models (LLMs), the ecosystem mandates a "local-first" execution paradigm. By orchestrating localised inference engines, such as Ollama and vLLM, alongside local Retrieval-Augmented Generation (RAG) architectures, the system keeps all semantic embedding, data extraction, and generative reasoning on project-controlled infrastructure; in the integrated ISABEL platform the model is served locally at CERTH (vLLM serving Mistral-Small-24B-Instruct-2501), so that no labour-market data is sent to commercial AI services (see D2.5 and D5.3).

The operational capabilities of the SNAP toolkit are distributed across three backend subsystems, which together expose the toolkit's four researcher-facing tools:

1. **The Synthetic Persona Generator:** A demographic simulation engine that ingests empirical Eurostat distributions to generate statistically valid synthetic personas. It mathematically computes green and digital skill intensities and deploys local LLMs to execute complex sociological survey roleplaying.
2. **The Research Companion:** An automated literature discovery and retrieval subsystem. It constructs persistent semantic memory stores (via ChromaDB and PaperQA) from academic papers, allowing researchers to execute citation-backed, deep-document question answering across vast literature libraries.
3. **The Unified Extraction & Analytics Pipeline:** A programmatic parsing engine and analytical layer integrated within OpenWebUI. It utilises strict prompt engineering to extract locked tabular data from unstructured PDFs and reports into normalised JSON and CSV artifacts, and seamlessly permits researchers to interrogate these newly structured DataFrames using natural language—translating queries into executable Python code while explicitly logging the output to guarantee methodological transparency.

In conclusion, Deliverable 3.3 establishes the software foundations of the SNAP project. Through robust containerisation, a strict Medallion storage strategy, and the seamless integration of empirical data with local AI orchestration, the deliverable provides a secure, sovereign, and methodologically rigorous infrastructure perfectly aligned with the research objectives of the Horizon Europe framework.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Table of Contents

1. Introduction 7

2. SNAP Architecture & Data Hub 8

2.1 Design Principles and High-Level Architecture 8

2.2 Semantic Homogenisation and Data Fusion Engine (green/digital skills) 8

2.3 Multi-Source Data Integration and Caching Layer(e.g. APIs → .parquet) 8

2.4 Storage Strategy: Medallion Architecture and Vector Stores 9

3. The SNAP Toolkit: The 3 Core Researcher Tools 10

3.1 Synthetic Persona Generator (Survey Simulator) 10

3.1.1 Population Sampling via Multivariate Distributions (Eurostat) 11

3.1.2 ISCO to 4-Digit ESCO Mapping 11

3.1.3 Semantic Skill Scoring (Green and Digital Attribution) 11

3.1.4 Identity Generation and Persona Finalisation 11

3.1.5 Dynamic Survey Configuration and Ingestion (Likert scale, free text, etc survey_questions.json) 12

3.1.6 LLM Roleplaying, Survey Execution, and Data Persistence 12

3.1.7 Visual Analytics and Survey Answers 13

3.2 Literature Deep-Dive Engine (Research Companion) 13

3.2.1 Automated Literature Discovery and Metadata Extraction (crawler arxiv) 14

3.2.2 Pre-processing and Abstract Summarisation via Local LLMs 14

3.2.3 Semantic Memory and Profile-Based Ranking (chromaDB) 14

3.2.4 Full-Text PDF Ingestion and Document Chunking (chunks) 15

3.2.5 Dual RAG Architecture: LangChain vs. PaperQA 15

3.2.6 Contextual Q&A and Academic Synthesis 15

3.2.7 Targeted Document Indexing 15

3.2.8 Citation-Backed LLM Synthesis 15

3.2.9 Contextual Q&A and Academic Synthesis 16

3.3 Unified Extraction & Analytics Pipeline 16

3.3.1 Target Document Identification and Content Retrieval 16

3.3.2 LLM-Driven Parsing and Data Normalisation 17

3.3.3 Data Cleansing and JSON-to-CSV Transformation 17

3.3.4 Dynamic Knowledge Base Creation and API Integration 17

3.4 Natural Language Data Analysis Tool 17

3.4.1 LLM-Data Integration and Agent Initialisation 17

3.4.2 Natural Language to Python Code Translation (PandasAI) 17

3.4.3 Automated Code Execution and Result Synthesis (deep analysis from pandai) 18

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3.4.4 Execution Transparency and Code Logging (for transparency log from openwebui) 18
3.5 Temporal Analytics and Forecasting Toolbox 18
3.5.1 Classical Statistical Processing Layer 18
3.5.2 AI-Driven Forecasting Layer: Transformer Architectures 19
3.5.3 Unified Execution and Output Pipeline 20
4. User Interface and Interactive Experience 21
4.1 Visual Analytics and Discovery Dashboards 21
4.2 Conversational UI and Tool Integration 21
5. Infrastructure, Security, and Model Management 23
5.1 Local-First LLM Orchestration 23
5.2 Containerisation and System Deployment 24
6. Appendices 25
6.1 Survey Configuration Example 25
6.2 Extraction Tool Query and Prompt Examples 25
6.3 Data Fusion and Semantic Mapping Visualisations 26
7. Conclusion 27
Bibliography 28

List of Figures

Figure 1: Complete execution pipeline of the Synthetic Persona Generator. 10
Figure 2: JSON payload of a generated synthetic persona ("Francesca") 13
Figure 3: Three-phase architectural pipeline of the Research Companion 14
Figure 4: End-to-end data flow of the Unstructured Data Extraction and Natural Language Analysis tools 16
Figure 5: Example of a dynamic survey configuration JSON file 25

List Of Abbreviations

API	Application Programming Interface	CSV	Comma Separated Values
ESCO	European Skills, Competences, Qualifications and Occupations	EU	European Union
GDPR	General Data Protection Regulation	GPU	Graphics Processing Unit
IaC	Infrastructure as Code	ISCO	International Standard Classification of Occupations
JCD	Job Construction Destruction	JSON	JavaScript Object Notation
LLM	Large Language Model	MCP	Model Context Protocol
OCR	Optical Character Recognition	PDF	Portable Document Format
QA	Question Answering	RAG	Retrieval Augmented Generation
REA	European Research Executive Agency	REST	Representational State Transfer
SNAP	Smart oNe stop digitAl data shoP	UI	User Interface
URI	Uniform Resource Identifier	WP	Work Package

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

1. Introduction

Understanding the dynamics of the European labour market, particularly during the transition towards green and digital economies, requires new tools capable of processing complex, multidimensional data. To address this need, the SNAP project introduces an integrated software ecosystem designed for synthetic persona generation, academic literature discovery, and unstructured data extraction. This document serves as the technical documentation for the initial SNAP prototype, detailing its backend architecture, data pipelines, and core functionalities.

Designed as a modular, API-driven system, SNAP focuses on robust backend processing rather than user interfaces. This design choice ensures that the underlying logic, from ingesting raw Eurostat statistics to mapping complex ESCO skill taxonomies, remains transparent, reproducible, and easy to integrate with different front-end applications in the future. The primary goal of this architecture is to combine official European statistical data with occupational frameworks, allowing researchers to model labour market demographics and skill distributions accurately.

A defining feature of the SNAP infrastructure is its strict adherence to data privacy and local execution. Processing sensitive demographic models and academic literature through commercial, cloud-based Large Language Models (LLMs) can introduce significant data governance and compliance risks. To solve this, SNAP is built around a local-first approach. By utilising local LLM engines like Ollama and vLLM, alongside local Retrieval-Augmented Generation (RAG) techniques, the system keeps all data processing, semantic search, and text generation on project-controlled infrastructure. In the integrated ISABEL platform these models are served locally at CERTH, as described in D2.5 and D5.3, supporting compliance with EU data protection standards and shields the research pipeline from external API changes or outages.

The SNAP ecosystem is divided into three main backend subsystems, which together expose the toolkit's four researcher-facing tools, each addressing a specific research requirement:

1. **The Synthetic Persona Generator:** This module acts as the demographic simulation engine. It fetches and caches Eurostat employment data, maps broad ISCO classifications to specific ESCO skill profiles, and creates realistic synthetic personas. It then uses local LLMs to roleplay these personas and generate synthetic survey responses.
2. **The Research Companion:** Functioning as an automated literature discovery tool, this subsystem builds persistent vector databases from academic papers. By using local RAG, it allows researchers to search for specific semantic profiles across document libraries and query PDF content directly.
3. **The Unstructured Data Extraction Toolkit:** This tool bridges the gap between raw text and structured data. Using strict LLM-driven parsing, it extracts tables from unstructured documents and converts them into structured datasets. It also includes an interface to analyse this tabular data using natural language queries via Pandas.

In summary, this document outlines how these three subsystems are built and integrated. The following sections will detail the high-level architecture, the multi-source data fusion mechanisms, the storage strategies, and the technical workflows that drive the SNAP toolkit.

Relation to other ISABEL deliverables. SNAP (Smart oNe-stop digitAl data shoP) is one of the analytical modules of the ISABEL platform, whose overall system architecture is specified in D2.5 (ISABEL System Architecture) and whose first integrated release is documented in D5.3 (First ISABEL Platform). In those deliverables, SNAP denotes the platform's unified data-lake and data-access infrastructure (a Bronze/Silver/Gold Medallion lake), served locally through the OpenWebUI gateway and a vLLM-hosted Mistral-Small-24B-Instruct-2501 model at CERTH. The present deliverable describes the initial SNAP software and the researcher-facing tools built on that infrastructure.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

2. SNAP Architecture & Data Hub

The SNAP backend is built upon a highly modular, API-driven infrastructure that systematically ingests raw statistical data, applies probabilistic semantic models, and manages complex vector retrievals. Rather than operating as a monolithic application, the ecosystem is strictly partitioned into functional subsystems: demographic simulation, academic retrieval, and data extraction. This separation ensures computationally efficient execution, scientific reproducibility, and seamless interoperability between local machine learning models and EU data taxonomies.

2.1 Design Principles and High-Level Architecture

The core architectural philosophy of SNAP is driven by a local-first execution model and a robust, multi-tiered data flow. To maintain data sovereignty and mitigate reliance on external cloud providers, the architecture mandates the use of localised model hosts. This is achieved through the integration of local LLM endpoints—specifically via vLLM, Ollama, and OpenWebUI—which handle all generative tasks, semantic summarisation, and vector embeddings within the host environment.

At the macro level, the architecture orchestrates three cooperating service domains:

1. **The Synthetic Persona Pipeline:** Managed by `persona/src`, this domain handles dataset caching, configuration, ISCO-to-ESCO taxonomy mapping, and the generation of synthetic demographic profiles using robust probabilistic algorithms.
2. **The Research Companion Backend:** Managed by `research_companion/`, this domain serves as the retrieval engine. It interfaces with arXiv for document discovery, manages local chunking of PDF documents, and orchestrates dual Retrieval-Augmented Generation (RAG) pathways using LangChain and PaperQA.
3. **The Extraction and Analytics Toolkit:** Managed by `table_extractor/`, this domain interacts with the OpenWebUI backend to parse unstructured document tables into structured DataFrames, subsequently exposing them to natural language queries via PandasAI.

To ensure uniformity across these diverse domains, central configuration is maintained in files such as `persona/src/config.py`, which establishes unified directory structures (`RAW_DIR`, `CACHE_DIR`, `OUTPUT_DIR`) and system-wide default model identifiers.

2.2 Semantic Homogenisation and Data Fusion Engine (green/digital skills)

To guarantee pipeline robustness and reduce latency from repetitive external API calls, SNAP implements a strict cache-first data ingestion strategy across its subsystems.

For labour market simulations, the `EurostatLoader` fetches target datasets (such as `lfsa_egised` for demographics and `lfsa_pganws` for migrant status). The system first attempts to load these directly from local Parquet files. If absent, a live download is initiated, immediately followed by rigorous schema normalisation—converting all column headers to lower-case strings to prevent downstream joining errors. The standardisation process concludes by writing the dataset back to disk as a Parquet artifact, securing it for offline execution.

Similarly, the `EscoLoader` ingests complex occupational structures from spreadsheet matrices, extracting skill column identifiers and building precise URI-to-label dictionaries. This guarantees that textual descriptions for digital and green skills are loaded into memory efficiently, supporting high-throughput semantic embedding operations without redundant disk I/O.

2.3 Multi-Source Data Integration and Caching Layer(e.g. APIs → .parquet)

A critical innovation within the data hub is the probabilistic bridging engine, which translates broad, high-level occupational data (ISCO) into granular, actionable skill profiles (ESCO).

Implemented within the `ESCOMapper`, the system builds a baseline mapping by calculating the frequency of 4-digit ESCO codes within broader ISCO major groups. To ensure this distribution

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

reflects empirical labour market realities, the engine fetches the Eurostat `lfsa_eiisco` dataset, extracts employment weights, and dynamically scales the initial probabilities. When a persona is generated, the system stochastically assigns a specific ESCO code based on this refined, renormalised distribution.
Skill intensity is subsequently calculated through a dual-scoring mechanism:

1. **Taxonomy Scoring:** A mathematical ratio of matched skill URIs against total required job skills.
2. **Semantic Similarity Scoring:** The system encodes job skill descriptions and computes the cosine similarity against pre-loaded target embeddings for green and digital skills (using `SentenceTransformer`).

This fusion engine successfully bridges official European taxonomies with empirical labour market statistics, creating a mathematically grounded framework for skill attribution.

2.4 Storage Strategy: Medallion Architecture and Vector Stores

Data persistence across the SNAP ecosystem is modelled after a lightweight Medallion architecture, ensuring transparent data lineage from raw ingestion to highly enriched semantic knowledge bases.

- **Raw Ingestion (Bronze):** This layer consists of Eurostat files cached as binary `.parquet` files, raw ESCO matrices stored as Excel sources, and binary PDF documents temporarily downloaded from arXiv.
- **Enriched Artifacts (Silver):** This intermediate layer houses transformed structural outputs. It includes the JSON-formatted persona records populated with semantic scores, normalised probabilistic ESCO mappings, and cleaned Pandas DataFrames generated from the table extraction toolkit.
- **Semantic Knowledge Stores (Gold):** To support complex semantic search and RAG methodologies across the research companion and extraction tools, SNAP orchestrates multiple concurrent persistent stores.

The Gold layer is highly distributed based on the specific architectural need:

- **ChromaDB Collections:** Used by the LangChain RAG pipeline to store vectorised chunks of academic papers via Ollama embeddings, enabling fast similarity matching against researcher profiles.
- **PaperQA Indices:** Stores hashed `Doc` and `Text` objects, preserving citation metadata for rigorous academic provenance.

  OpenWebUI Knowledge Bases: Dynamically generated via API POST requests, these bases store the extracted CSV data from unstructured documents, transforming static tables into interactive, vectorised repositories queryable by the underlying LLM.

  Alignment with the platform data lake: in the integrated ISABEL platform (D2.5 §7.5 and D5.3 §4.2) the Medallion lake is backed by MinIO (S3-compatible) buckets for the Bronze, Silver and Gold tiers, with Docling converting Bronze files into normalised Markdown/Parquet in the Silver tier and DuckDB querying Gold-layer tables. The lightweight, file-based mapping described above is the SNAP-module view; please confirm it is described consistently with the platform-level lake.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3. The SNAP Toolkit: The 3 Core Researcher Tools

The operational capabilities of the SNAP ecosystem are distributed across four highly specialised, interoperable backend tools. Each tool is engineered to address distinct methodological challenges inherent in European labour and transition research, ranging from sociological simulation to the computational extraction of unstructured knowledge.

These four tools map onto the structure of this section as follows: the Synthetic Persona Generator (Section 3.1), the Research Companion (Sections 3.2 and 3.5, the latter covering its PaperQA pathway), the Unstructured Data Extraction Tool (Section 3.3), and the Natural Language Data Analysis Tool (Section 3.4). The extraction and analysis tools are delivered through a single OpenWebUI Pipe (exported as snap_pipe).

3.1 Synthetic Persona Generator (Survey Simulator)

The Synthetic Persona Generator represents the primary demographic simulation engine within the SNAP architecture. Located within the `persona` package, this subsystem is responsible for generating statistically robust demographic profiles, executing highly complex taxonomy alignments between broad occupational frameworks (ISCO) and granular skill matrices (ESCO), and deploying local Large Language Models (LLMs) to execute and simulate sociological survey responses. The overarching execution pipeline is orchestrated by `persona/app.py` and `persona/main.py`, ensuring all generated outputs are persistently saved to disk for methodological auditability.

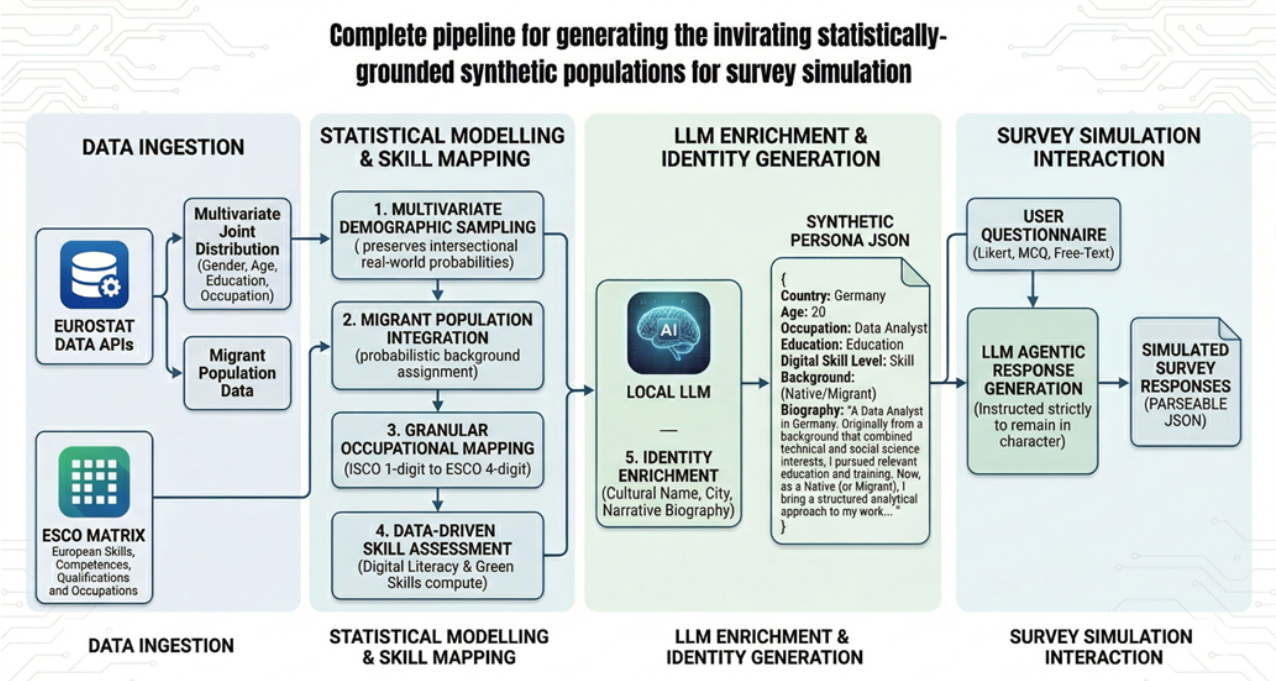


Figure 1: Complete execution pipeline of the Synthetic Persona Generator. The diagram illustrates the end-to-end data flow, starting from the ingestion of Eurostat and ESCO datasets, moving through statistical modelling and ISCO-to-ESCO mapping, and concluding with LLM-driven identity enrichment and survey simulation.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3.1.1 Population Sampling via Multivariate Distributions (Eurostat)

To ensure statistical validity, population sampling is strictly driven by empirical distributions rather than arbitrary randomisation. Ingestion is handled by the `EurostatLoader.fetch_dataset()` function, while the `persona/src/generators/demographics.py` module prepares the joint demographic datasets—specifically targeting `lfsa_egised` for core sociodemographic attributes and integrating migrant probabilities derived from `lfsa_pganws`. The generator systematically applies geographical filters, normalises categorical demographic variables, and executes weighted sampling algorithms to maintain strict statistical plausibility. The country-filtering logic, exposed within `persona/app.py` via `get_allowed_countries()`, facilitates region-limited persona generation tailored exclusively for the EU and its neighbouring states.

3.1.2 ISCO to 4-Digit ESCO Mapping

The translation of broad ISCO classifications into precise ESCO skill profiles is central to the subsystem's occupational realism. The `ESCOMapper._build_isco_to_esco_mapping()` function constructs a baseline weighted mapping from ISCO major groups to specific four-digit ESCO codes utilising the official ESCO job matrix. To ensure empirical accuracy, `ESCOMapper._build_eurostat_weights()` subsequently enhances these baseline probabilities using European employment distribution data fetched from `lfsa_eiisco`, yielding a hybrid probabilistic model. During generation, `AttributesGenerator._select_esco_code_for_group()` and `AttributesGenerator.enrich_with_skills()` utilise this model to assign a definitive ESCO occupation to each persona, explicitly logging the mathematical selection probability within the persona's metadata. A robust fallback mechanism ensures that if direct ESCO matching fails, the system safely assigns plausible, occupation-level default skills to maintain pipeline continuity.

3.1.3 Semantic Skill Scoring (Green and Digital Attribution)

The system quantifies green and digital skill intensities through a highly rigorous dual-methodology framework. Initially, `AttributesGenerator._calculate_official_score()` computes a baseline score by evaluating the taxonomic overlap between the required job URIs and official European category URIs. Concurrently, `AttributesGenerator._calculate_semantic_similarity_score()` employs a `SentenceTransformer('all-MiniLM-L6-v2')` model to evaluate the cosine similarity between textual job skill descriptions and pre-computed green and digital vector embeddings. The engine calculates a top-k average similarity, mathematically mapping these continuous similarity values onto a discrete 1–5 scale. This dual approach ensures that skill attribution is grounded in both explicit ESCO taxonomies and latent semantic alignment.

3.1.4 Identity Generation and Persona Finalisation

The completion of the synthetic profile is executed within `AttributesGenerator._generate_identity()`. Once the demographic parameters and occupational vectors are successfully generated, the system dynamically synthesises a realistic identity, appending a plausible geographical location, a fictitious name, and a succinct biographical summary. Because these elements are appended post-enrichment, the final persona record constitutes a highly detailed, hierarchical JSON object containing demographic,

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

occupational, and identity metadata uniquely optimised for LLM prompt construction.

3.1.5 Dynamic Survey Configuration and Ingestion (Likert scale, free text, etc survey_questions.json)

To accommodate diverse methodological requirements across varied research disciplines, survey configuration is dynamic and strictly governed by external JSON schemata. The `persona/main.py` module loads questionnaire definitions (e.g., `persona/input_survey.json` or `persona/input_survey_v2.json`) via the `load_survey_config()` function. Furthermore, `persona/app.py` exposes a programmatic ingestion layer supporting the direct upload of survey configurations. The ingested JSON structures—which delineate formats ranging from discrete Likert scales to qualitative free-text fields—are rigorously validated before being translated into executable prompt instructions for the LLM.

3.1.6 LLM Roleplaying, Survey Execution, and Data Persistence

The simulation of survey responses is orchestrated by `persona/src/engine/llm_runner.py`. The `SurveyRunner.answer_survey()` method invokes a local LLM instance (via `ollama.chat()`) to roleplay the synthesised persona and generate structured output. Crucially, `persona/src/engine/prompt_builder.py` constructs persona-aware prompts that strictly mandate JSON formatting from the LLM. To mitigate generative hallucinations, the subsystem incorporates aggressive retry logic and validation parsing, ensuring all returned answers accurately map to the expected question identifiers and structural schemata. The final, validated survey responses are written to disk as JSON artifacts, creating a permanent, auditable record.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

```
{
    "persona": {
        "demographics": {
            "country": "IT",
            "sex": "F",
            "age_group": "Y_GE15",
            "age_exact": 41,
            "education_isced": "ED5-8",
            "education_label": "Tertiary Education",
            "is_migrant": false
        },
        "work": {
            "isco_major_group": "OC2",
            "job_title": "Telecommunications engineers",
            "esco_4digit_code": "2153",
            "esco_selection_probability": 0.01098901098901099,
            "core_skills": [
                "preparing financial documents, records, reports, or budgets",
                "analysing business operations",
                "gathering information from physical or electronic sources",
                "estimating resource needs",
                "performing calculations"
            ]
        },
        "skill_scores": {
            "green_skills": 3,
            "digital_skills": 5,
            "manual_skills": 1
        },
        "identity": {
            "first_name": "Francesca",
            "city": "Bari",
            "short_bio": "A 41-year-old professionals with Tertiary Education, working in specialized technical or academic field
        },
        "participant_id": "ded74f38"
    },
    "answers": {
        "q1": "5",
        "q2": "My primary concern is the high initial cost of transitioning to greener technologies.",
        "q3": "Yes"
    }
}
```


Figure 2: Complete JSON payload of a generated synthetic persona ("Francesca"), showcasing the integration of demographic data, stochastically assigned ESCO skills, semantic skill scoring, and simulated LLM survey responses.

3.1.7 Visual Analytics and Survey Answers

Although the core architecture remains deliberately backend-focused, the system proactively computes rigorous statistical aggregations over the generated persona cohorts within `persona/app.py`. The `perform_analysis()` function calculates descriptive statistics encompassing age distributions, gender ratios, educational attainment levels, mean skill intensities, and migration status percentages. Persisted personas can be programmatically loaded and filtered, allowing researchers to extract deep, cohort-level analytics seamlessly.

3.2 Literature Deep-Dive Engine (Research Companion)

The Research Companion serves as an advanced, backend-driven retrieval engine designed to facilitate academic literature discovery, semantic profile matching, and deep-document question answering (QA). Operating entirely through local model orchestration to preserve data sovereignty, the engine's capabilities are distributed across several core modules.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

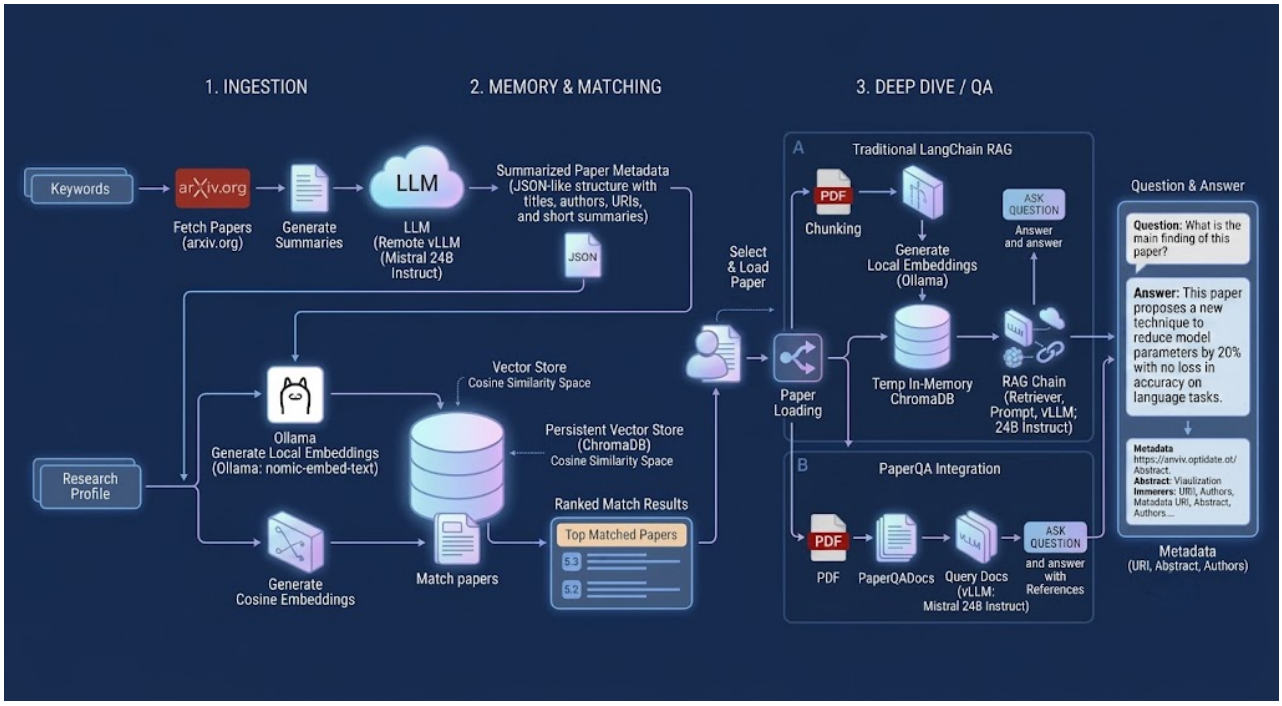


Figure 3: The three-phase architectural pipeline of the Research Companion. The diagram details (1) Automated Ingestion and Summarisation from arXiv, (2) Profile-based Semantic Memory matching via ChromaDB, and (3) The dual RAG architecture offering both Traditional LangChain and citation-backed PaperQA pathways for deep-document QA.

3.2.1 Automated Literature Discovery and Metadata Extraction (crawler arxiv)

The initial phase of literature ingestion is driven by an automated discovery pipeline utilising the arXiv Python client within `research_companion/research_companion.py`. The `fetch_papers()` function programmatically submits targeted search queries, retrieves raw metadata, and structures the findings into standardised paper records. This discovery layer forms the foundational research library, supplying essential metadata for subsequent embedding and semantic matching operations.

3.2.2 Pre-processing and Abstract Summarisation via Local LLMs

To optimise the textual data for vector representation, raw abstracts are semantically pre-processed and summarised using a local vLLM instance. Operating through an OpenAI-compatible interface within `research_companion/research_companion.py`, the system connects to the `VLLM_HOST` and executes a strict summarisation prompt configured with a low generation temperature. This deterministic process produces highly compact, researcher-friendly technical summaries that augment the baseline metadata.

3.2.3 Semantic Memory and Profile-Based Ranking (chromaDB)

Knowledge persistence and semantic ranking are architected within `research_companion/phase2_memory_matching.py`. The `add_papers()` function transforms the academic summaries into dense mathematical vectors using local Ollama embedding models, persistently storing them in a ChromaDB vector collection. Researchers can leverage `find_matching_papers()` to compute cosine similarities against a designated

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

research profile embedding, enabling the system to return highly relevant, profile-aware rankings of the available literature based on configurable mathematical thresholds.

3.2.4 Full-Text PDF Ingestion and Document Chunking (chunks)

Deep-dive analysis necessitates the granular processing of full-text binaries. Implemented across `research_companion/phase3_deep_dive.py` and `research_companion/paperqa_rag.py`, the ingestion pipeline utilises `PyMuPDFLoader` to extract raw strings from academic PDFs. To maintain contextual integrity during vector retrieval, the text is systematically segmented using a `RecursiveCharacterTextSplitter`. This chunking strategy significantly enhances retrieval precision, enabling highly accurate local RAG over specific document segments.

3.2.5 Dual RAG Architecture: LangChain vs. PaperQA

To accommodate diverse analytical methodologies, the toolkit supports a dual Retrieval-Augmented Generation (RAG) architecture. Researchers may invoke a LangChain-based local RAG pathway (implemented within `research_companion/phase3_deep_dive.py`) for customisable, fine-grained chain-of-thought retrieval. Alternatively, they may utilise a highly specialised PaperQA-based architecture (housed in `research_companion/paperqa_rag.py`), which provides a streamlined document ingestion and answer-generation pathway focused on academic source tracking.

3.2.6 Contextual Q&A and Academic Synthesis

The synthesis of academic answers is orchestrated through the Streamlit application logic in research_companion/app.py. In the integrated ISABEL platform, these queries are exposed through OpenWebUI's REST API and the FastAPI tool server described in D2.5 / D5.3, rather than a standalone api.py module. Upon receiving a query, the system retrieves the most mathematically relevant document chunks, assembles them into a dense context window, and provides this grounding data to the local LLM prior to generation. The architecture natively supports multi-turn chat histories and returns rigorously structured answer objects tailored for downstream academic consumption.

3.2.7 Targeted Document Indexing

Rather than indexing entire cross-document libraries indiscriminately, this subsystem focuses on high-fidelity, document-level deep dives, while still allowing several documents to be indexed into a shared PaperQA store for cross-referencing. As the engine ingests a target PDF, it structures the segments into discrete text objects inextricably linked to source-tracking metadata.

3.2.8 Citation-Backed LLM Synthesis

The paramount advantage of the PaperQA integration is its rigorous synthesis protocol tailored for individual paper analysis. The engine actively traces generated assertions back to their originating text blocks within the document, ensuring that all returned answers are explicitly citation-backed. By preserving citation labels and text provenance throughout the generation pipeline, the subsystem satisfies the strict academic requirement that all LLM-synthesised arguments must be verifiable against original source materials.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3.2.9 Contextual Q&A and Academic Synthesis

The synthesis of academic answers is orchestrated through the backend endpoints defined in `research_companion/api.py` and the procedural logic in `research_companion/app.py`. Upon receiving a query, the system retrieves the most mathematically relevant document chunks, assembles them into a dense context window, and provides this grounding data to the local LLM prior to generation. The architecture natively supports multi-turn chat histories and returns rigorously structured answer objects tailored for downstream academic consumption.

3.3 Unified Extraction & Analytics Pipeline

Addressing the challenge of locked tabular data within unstructured governmental and academic reports, this subsystem systematically converts raw document content into structured CSV files and reusable OpenWebUI knowledge bases, immediately exposing them to natural language interrogation. Rather than operating as decoupled standalone scripts, this ecosystem is consolidated into a singular, highly automated OpenWebUI "Pipe" module.

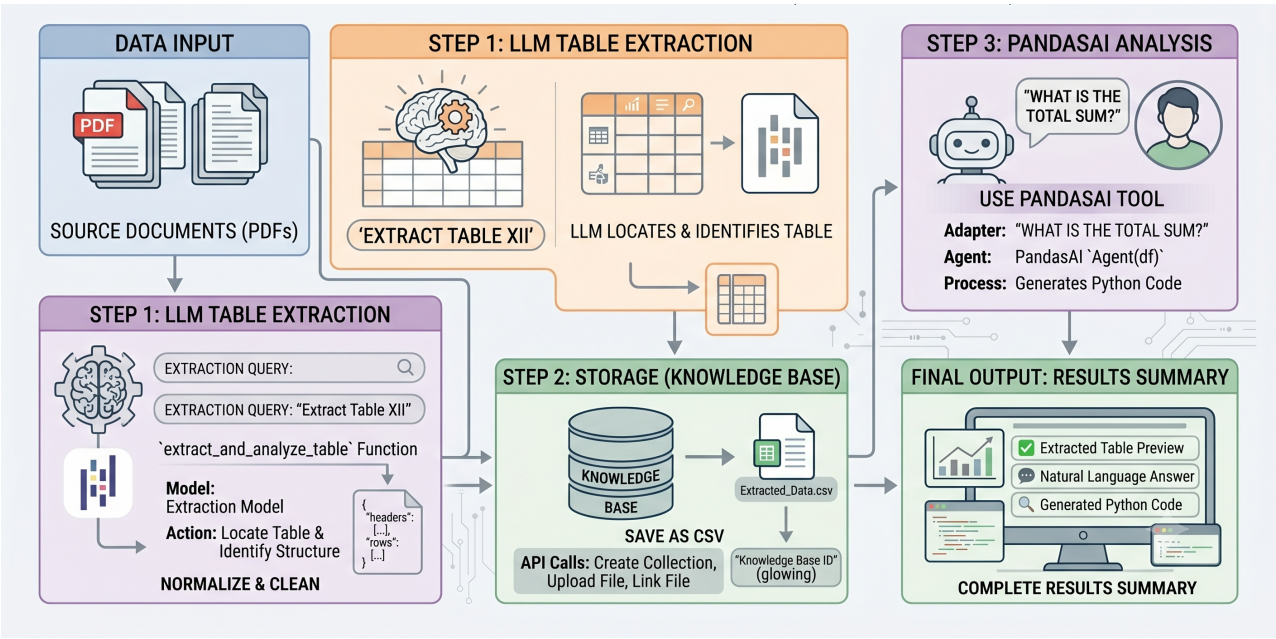


Figure 4: End-to-end data flow of the Unstructured Data Extraction and Natural Language Analysis tools. The diagram illustrates the parsing of PDF documents into strict JSON, the transformation into CSV artifacts for OpenWebUI Knowledge Base storage, and the execution of natural language queries via PandasAI

3.3.1 Target Document Identification and Content Retrieval

The extraction workflow commences by identifying target documents via their unique `file_id` designations provided in the Pipe's configuration valves within a local OpenWebUI instance. The pipeline executes authenticated backend API requests to retrieve the raw, unstructured document payload, ensuring the process originates from a verifiable, machine-readable source.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3.3.2 LLM-Driven Parsing and Data Normalisation

To coerce the generative model into outputting data structures, the Pipe employs aggressive prompt engineering. It constructs a rigid system prompt mandating "strict JSON only" responses, explicitly forbidding markdown formatting. Following extraction, the pipeline programmatically extracts the primary JSON object, rectifies any malformed outputs, and ensures precise row-to-header alignment—transforming the nested data into a structured Pandas DataFrame. To enforce absolute data provenance, a `_source_file_id` metadata tag is injected into every row. The finalised DataFrame is then exported as a CSV artifact, perfectly formatted for subsequent analytics.

3.3.3 Data Cleansing and JSON-to-CSV Transformation

Following extraction, the raw JSON payload undergoes rigorous normalisation. The unified Pipe module ensures precise row-to-header alignment, transforming the nested data into a structured Pandas DataFrame. To enforce absolute data provenance, the system injects a `_source_file_id` metadata tag into every row. The finalised DataFrame is then exported as a CSV artifact, perfectly formatted for subsequent analytics and knowledge base ingestion.

3.3.4 Dynamic Knowledge Base Creation and API Integration

To guarantee that the extracted data remains persistently searchable, the tool automatically constructs a new vector space. It executes a sequence of POST requests to the OpenWebUI backend API (specifically targeting `/api/v1/knowledge/create`, `/api/v1/files/`, and `/api/v1/knowledge/{kb_id}/file/add`), seamlessly uploading the CSV artifact. This integration transforms static, document-derived tables into interactive, reusable knowledge bases.

3.4 Natural Language Data Analysis Tool

Extending beyond basic extraction, SNAP provides a programmatic analytical layer that permits researchers to interrogate structured tables using natural language. The core execution logic for this translation and analysis workflow is integrated within the unified Pipe module.

3.4.1 LLM-Data Integration and Agent Initialisation

The analytical process initiates by establishing a secure bridge between the generative model and the tabular data. The system initialises a PandasAI Agent configured with a custom OpenWebUILLM wrapper. The previously extracted DataFrame is injected directly into the agent's contextual environment, firmly binding the LLM's reasoning capabilities to the deterministic structure of the dataset.

3.4.2 Natural Language to Python Code Translation (PandasAI)

Upon receiving an analytical query, the PandasAI agent translates the researcher's natural language prompt into executable Python syntax. The pipeline meticulously captures both the generated code and the ultimately executed script. This programmatic translation layer

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

effectively democratises advanced data analysis, enabling researchers to perform complex statistical queries without manual scripting.

3.4.3 Automated Code Execution and Result Synthesis (deep analysis from pandai)

The translated Python script is automatically executed against the in-memory DataFrame. The computational results are then evaluated by the agent, which synthesises the numerical output into a coherent natural language response. The final, consolidated output delivered to the researcher includes a preview of the extracted table, the analytical text synthesis, and the precise Python code utilised to generate the finding, guaranteeing reproducible analytics.

3.4.4 Execution Transparency and Code Logging (for transparency log from openwebui)

To eliminate the opaque nature of LLM-driven analytics, the SNAP architecture intercepts the execution layer to log computational operations explicitly. By surfacing both the `generated_code` and `executed_code` attributes within the final payload, the system provides total execution transparency. This rigorous logging ensures high auditability, allowing researchers to peer-review and validate the exact programmatic logic driving their analytical outputs.

3.5 Temporal Analytics and Forecasting Toolbox

SNAP's analytical remit extends beyond document retrieval and natural language querying to encompass structured time-series forecasting of labour market indicators. To address this, the architecture integrates a dual-layer analytics toolkit that pairs classical statistical preprocessing techniques with a suite of state-of-the-art Transformer-based deep learning models. The layered design is deliberate: traditional operations, whose behaviour is fully interpretable and reproducible, are applied prior to and in complement with the AI-driven forecasting stage, whose outputs can be benchmarked against classical baselines to ensure methodological rigour. Both layers are implemented as modular Python components and executed locally, fully consistent with the local-first execution paradigm described in Section 5.1.

3.5.1 Classical Statistical Processing Layer

The toolkit's foundational layer is constructed upon standard Python data science infrastructure. Raw transactional data ingested from `.parquet` archives via `pd.read_parquet()` is cast into a structured, time-indexed pivot matrix. Individual records are first grouped and aggregated into totals using `groupby()` in conjunction with `dt.floor('h')`. The resulting pivot table (of dimensions (time-steps × n-categories)) constitutes the primary input tensor for all downstream model architectures.

Temporal feature engineering is applied explicitly: each hourly index is decomposed into five engineered scalar features. This feature extraction mirrors the calendar decomposition routinely used in SARIMA and Prophet pipelines, and ensures that all models receive temporally grounded input. Zero-padding via `fillna(0)` handles sparse intervals, and mean-standard deviation normalisation, applied within the Reformer and Pyraformer short-term forecast branches, provides a stationary representation consistent with classical normalisation practice.

Within the Autoformer architecture, a learnable moving-average decomposition block (`series_decomp`) is applied at each encoder and decoder stage to separate trend and residual

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

components before the attention mechanism operates. Embedding this decomposition directly within the neural forward pass grounds the AI predictions in an interpretable classical operation—analytically equivalent to STL (Seasonal and Trend decomposition using Loess)—while allowing the deep learning component to model non-linear residuals that classical decomposition methods cannot capture.

3.5.2 AI-Driven Forecasting Layer: Transformer Architectures

Four distinct Transformer-based architectures are implemented as standalone PyTorch `nn.Module` classes and exposed through a unified command-line runner (`run_models.py`). Each model accepts the same tensor signature - encoded sequence, decoder initialisation, and temporal stamps - and produces short- or long-horizon forecasts of configurable length, enabling direct comparative evaluation under identical data conditions. Table 1 summarises the architectural characteristics of each model.

Table 1: Architectural summary of the four Transformer-based forecasting models integrated within SNAP

Model	Attention Mechanism	Complexity	Key Characteristic
Autoformer	AutoCorrelation	O(L log L)	Series decomposition into trend + seasonal components at each encoder/decoder layer
Informer	ProbSparse	O(L log L)	Progressive encoder distilling reduces sequence length across stacked layers
Reformer	LSH (Locality-Sensitive Hashing)	O(L log L)	Approximates the attention matrix via bucket-based random projections; encoder-only for regression
Pyraformer	Pyramidal multi-scale	O(L)	Hierarchical window structure simultaneously captures fine-grained and multi-hour patterns

Autoformer (*Wu et al., 2021*) replaces standard scaled dot-product attention with an AutoCorrelation mechanism that discovers period-based dependencies directly in the time domain. The `AutoCorrelationLayer` computes autocorrelation scores using Fast Fourier Transforms, aggregating temporal lags with the highest correlation rather than pairwise token similarity, resulting in O(L log L) complexity. The series decomposition blocks integrated throughout the encoder and decoder make Autoformer particularly well-suited to labour market time series exhibiting strong seasonal or cyclical patterns, such as monthly hiring rates or quarterly green-sector employment transitions.

Informer (*Zhou et al., 2021*) introduces ProbSparse self-attention, which selects a sparse subset of dominant queries using a KL-divergence-based criterion. A distilling operation progressively reduces the encoder sequence length across stacked `ConvLayer` modules, making the architecture tractable for long historical windows while preserving the most salient temporal features. This property is relevant for ISABEL's use case, where forecasting over multi-week or multi-month horizons from dense daily or hourly labour market feeds is required.

Reformer (*Kitaev et al., 2020*) approximates the full attention matrix using Locality-Sensitive Hashing (LSH), which groups query and key vectors into buckets and restricts attention computation to within-bucket pairs. The result is O(L log L) time and substantially reduced memory overhead compared to full attention. In the SNAP implementation the Reformer operates as an encoder-only

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

architecture for regression tasks, projecting encoded representations directly to output forecasts via a linear head, yielding a compact and memory-efficient deployment footprint.

Pyraformer (*Liu et al., 2022*) constructs a pyramidal attention graph across multiple temporal resolutions using a configurable window hierarchy (default: `[4, 4]`). The hierarchical structure allows the model to simultaneously attend to fine-grained hourly patterns and coarser multi-hour trends within a single forward pass, achieving O(L) inter-scale attention complexity. This multi-resolution design aligns naturally with labour market data, which typically exhibits concurrent short-term fluctuations and long-term structural trends.

3.5.3 Unified Execution and Output Pipeline

The unified runner (`run_models.py`) accepts a model identifier via the `--model` flag and dynamically imports the corresponding architecture at runtime. All stages—dataset construction, DataLoader batching, training (MSE criterion, Adam optimiser), inference, and output serialisation—are handled within a single executable script, eliminating configuration drift between models. Forecast arrays are serialised to `outputs/<dataset>/<model>/pred.npy` alongside a `categories_index.csv` mapping that preserves the multivariate channel order. The companion utility `view_autoformer_results.py` reads these artefacts, applies post-processing—clipping of sub-zero values and integer rounding—and renders per-category forecast sequences, providing an immediately interpretable demand projection for each monitored labour market category. This output format is directly compatible with the Natural Language Data Analysis Tool (Section 3.4), which can ingest the serialised CSVs for further PandasAI-driven interrogation.

In aggregate, the Temporal Analytics and Forecasting Toolbox provides SNAP with a methodologically layered, benchmark-ready analytical infrastructure: classical statistical operations ensure interpretability and reproducibility, while the four Transformer architectures extend SNAP's forecasting capability to long-range, multivariate, and seasonally complex labour market dynamics within a fully local, GDPR-compliant execution environment.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

4. User Interface and Interactive Experience

Although the SNAP architecture is fundamentally engineered as a decoupled backend ecosystem, validating its complex data pipelines requires functional presentation layers. To this end, the repository includes reference user interfaces (UIs) that demonstrate how external frontends can successfully interact with the underlying analytical engines. These interfaces are designed to prove the structural interoperability of the APIs, highlighting how visual analytics and conversational flows are dynamically driven by backend computations.

4.1 Visual Analytics and Discovery Dashboards

The presentation of complex, multidimensional data requires strict separation of concerns between computational logic and visual rendering. In the Synthetic Persona subsystem, this is achieved through a server-side rendered dashboard implemented via Flask (`persona/templates/index.html`). To prevent the interface layer from becoming computationally burdened, all statistical aggregations—such as cohort age distributions, skill intensities, and migrant probabilities—are pre-calculated by the `perform_analysis()` function within `persona/app.py`. The HTML template acts as a passive receiver, dynamically populating its statistical summary cards, demographic charts, and region-filtering logic based entirely on the structured JSON payloads passed from the backend.

Similarly, the Research Companion provides a reactive discovery dashboard built upon the Streamlit framework (`research_companion/app.py`). This interface is structurally partitioned into dedicated tabbed views. The "Discovery" tab interacts directly with the arXiv crawler, allowing researchers to input programmatic search queries and immediately visualise the returned metadata and generated technical summaries. Simultaneously, the "Library" view interfaces with the ChromaDB endpoints, rendering real-time database statistics and displaying the embedded research profiles. By abstracting the complex local model orchestration away from the UI, these dashboards demonstrate how researchers can seamlessly interact with high-level visual analytics while the backend securely manages the vector and statistical operations.

4.2 Conversational UI and Tool Integration

The integration of Large Language Models introduces a paradigm shift in how researchers interact with data, necessitating robust conversational interfaces. Within the Research Companion, the conversational experience is orchestrated by `research_companion/app.py`, which securely bridges user inputs with the backend Retrieval-Augmented Generation (RAG) engines.

A critical architectural challenge in conversational UIs is state management. The Streamlit application maintains an in-memory session state to track the multi-turn chat history, ensuring that the local LLM retains contextual awareness across continuous interactions. Furthermore, the interface dynamically exposes the subsystem's dual-RAG architecture; researchers can utilise UI toggles to route their queries either through the LangChain retrieval pipeline or the citation-backed PaperQA pipeline. The frontend simply packages the user's query and the selected mode flag, transmitting them to the backend API (`research_companion/api.py`), which executes the vector synthesis and returns the final academic response alongside its citation metadata.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

This conversational integration extends directly into the unstructured data toolkit. Through the PandasAI abstraction (`table_extractor/pandasai.py`), the interactive experience shifts from document QA to programmatic data analysis. The conversational UI accepts natural language questions regarding extracted tables and returns a multi-part payload: a preview of the structured DataFrame, a synthesised textual answer, and the executable Python code generated to perform the analysis. This integration ensures that the conversational interface does not merely act as a 'chatbot', but functions as a sophisticated, transparent interactive console for code-backed data interrogation.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

5. Infrastructure, Security, and Model Management

The viability of the SNAP toolkit as a European research infrastructure relies heavily upon its deployment architecture and its adherence to stringent data governance protocols. Section 5 details the underlying infr`astructural paradigms, focusing explicitly on how the system guarantees data sovereignty through localised model orchestration and ensures scientific reproducibility through advanced containerisation strategies.

Platform-level security: authentication, authorisation and transport security for SNAP are provided by the ISABEL platform and specified in D2.5 (§7.4) and D5.3 (§3.2): federated identity via Keycloak (OAuth 2.0 / OIDC), token validation at the OpenWebUI gateway, TLS 1.2/1.3 termination at an Nginx reverse proxy, and isolated PostgreSQL instances for application and identity data. This section covers only the SNAP-specific infrastructure.

5.1 Local-First LLM Orchestration

A foundational requirement of the SNAP project is the secure processing of sensitive labour market statistics and proprietary academic literature. Transmitting such data to third-party, cloud-based Large Language Model (LLM) APIs introduces unacceptable vulnerabilities regarding data privacy, intellectual property leakage, and compliance with the General Data Protection Regulation (GDPR). To systematically mitigate these risks, the SNAP architecture enforces a strict "local-first" model orchestration paradigm.

Rather than hard-coding dependencies to external commercial providers, the backend is engineered to interface exclusively with locally hosted model serving frameworks. The ecosystem relies upon three primary inference engines:

1. **Ollama:** Utilised primarily for executing the demographic survey simulations (`llm_runner.py`) and generating dense text embeddings (`OllamaEmbeddings`) for the vector stores.
2. **vLLM:** Deployed to handle high-throughput, deterministic tasks such as academic abstract summarisation within the Research Companion, leveraging its highly optimised memory management capabilities.
3. **OpenWebUI Backend:** Interfaced programmatically by the unified OpenWebUI Pipe module (exported as snap_pipe) to drive the strict JSON parsing and PandasAI analytical workflows.

To ensure architectural flexibility, the orchestration layer abstracts the specific model weights from the execution logic. The system utilises OpenAI-compatible client interfaces and `LiteLLM` wrappers, routing programmatic requests to model endpoints designated by environment variables (e.g., VLLM_HOST and OLLAMA_HOST). This design keeps all generative reasoning, semantic embedding, and data extraction on project-controlled infrastructure (see the local serving model in D2.5 / D5.3). This design ensures that all generative reasoning, semantic embedding, and data extraction remain entirely air-gapped within the host environment. Furthermore, this decoupling allows researchers to seamlessly hot-swap the underlying foundation models (e.g., transitioning from `mistral-24b` to a domain-specific European model) purely via configuration updates, without necessitating any refactoring of the Python codebase.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

5.2 Containerisation and System Deployment

To satisfy the Horizon Europe mandate for scientific reproducibility and scalable deployment, the SNAP backend avoids brittle, host-dependent installations in favour of a robust containerisation strategy. The infrastructure is codified as Infrastructure-as-Code (IaC) utilising Docker and managed via a centralised `docker-compose.yaml` architecture.

This containerised deployment model provides several critical infrastructural advantages:

1. **Dependency Isolation:** By encapsulating the Python execution environments, system-level libraries, and vector databases (such as ChromaDB) within discrete containers, the architecture guarantees that the execution context remains identical across diverse institutional hardware.
2. **Network Security and Isolation:** The Docker Compose configuration establishes isolated internal bridge networks. Subsystems communicate via internal service resolution (e.g., connecting to `host.docker.internal` or designated subnets like `isabel-network`). This prevents the inadvertent exposure of backend API endpoints or unauthenticated model hosts to external traffic, inherently reinforcing the system's security posture.
3. **Persistent Volume Management:** To sustain the Medallion storage strategy outlined in Section 2, the container architecture maps critical internal directories to persistent local volumes. By mounting the `CACHE_DIR` (housing the Eurostat Parquet files), the `OUTPUT_DIR` (containing the generated persona JSONs), and the `./data/chroma_db` directory to the host filesystem, the deployment ensures that the ingestion layers and semantic memory stores survive container restarts.

Ultimately, this containerised approach transforms the SNAP toolkit from a theoretical prototype into a highly portable, deployable infrastructure, capable of being securely instantiated on individual researcher workstations or scaled across institutional server clusters with minimal configuration overhead.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

6. Appendices

The following appendices provide structural examples of the configuration schemata, prompt engineering strategies, and analytical outputs that drive the SNAP backend ecosystem. These artifacts demonstrate the programmatic strictness required to orchestrate deterministic outputs from generative models.

6.1 Survey Configuration Example

The Synthetic Persona Generator dynamically orchestrates its survey simulations using externalised JSON configuration files (e.g., persona/input_survey.json). This design decouples the sociological survey parameters from the underlying Python execution logic in persona/main.py.

The schema strictly defines the question identifiers, the specific prompt phrasing, and the expected structural format of the response. Below is a representative structural excerpt demonstrating how a Likert-scale and a free-text question are configured:
JSON

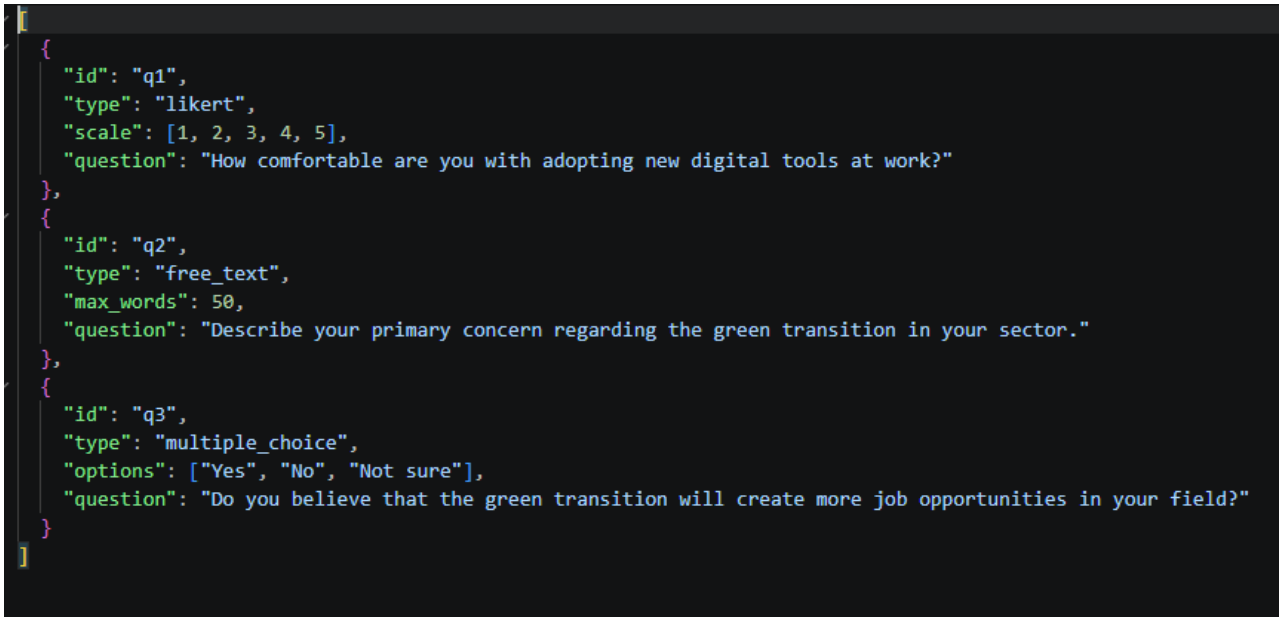


Figure 5: Example of a dynamic survey configuration JSON file, defining question types, IDs, and constraints for the LLM input.

During execution, the SurveyRunner parses this structure, injects the specific prompts into the persona-aware LLM context window, and validates that the returned JSON object perfectly matches the requested question_id keys.

6.2 Extraction Tool Query and Prompt Examples

To successfully bridge the gap between unstructured document text and structured analytical DataFrames, the unified OpenWebUI Pipe module (exported as snap_pipe) relies on aggressive prompt engineering. Generative models inherently default to conversational outputs; therefore, the system prompts must act as rigid parsers.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Below is an example of the baseline system prompt used to enforce structural compliance during the execution of _call_llm_extract():
Plaintext

SYSTEM INSTRUCTION:
You are a strict, deterministic data extraction engine. Your sole objective is to extract tabular data from the provided unstructured text and convert it into a normalised JSON structure.

CRITICAL RULES:
1. You must output ONLY valid, parsable JSON.
2. Do NOT wrap the JSON in markdown formatting or code fences (e.g., do not use ```json).
3. Do NOT include any conversational filler, greetings, or explanations.
4. If a value is missing in the source text, represent it as null.

EXPECTED OUTPUT FORMAT:
```
{
  "headers": ["column_1", "column_2"],
  "rows": [
    ["value_A", "value_B"]
  ]
}
```

This strict instructional framing is critical. By explicitly prohibiting markdown and conversational syntax, the backend extraction pipeline can programmatically ingest the LLM's response, execute json.loads(), and seamlessly transform the payload into a Pandas DataFrame without encountering parsing fatal errors.

6.3 Data Fusion and Semantic Mapping Visualisations

While visual rendering is explicitly decoupled from the core computational logic, the SNAP backend proactively calculates the mathematical vectors necessary for downstream visual analytics. The persona/skills_esteban/skills.py module and the aggregation functions within persona/app.py compute high-fidelity matrices that serve as the foundation for data fusion visualisations.

Key analytical outputs generated by the backend include:

- Semantic Similarity Heatmaps: Matrices derived from the cosine similarity calculations between ESCO job descriptions and target green/digital embeddings, visualising the precise distance between traditional occupations and transitional skillsets.
- Probabilistic Distribution Curves: Data arrays demonstrating the entropy and spread of the ISCO-to-ESCO bridging algorithm, verifying that the stochastic occupational selection aligns with Eurostat employment weights.
- Cohort Skill Histograms: Aggregated distributions of the 1–5 skill intensity scores across generated persona populations, exposing the demographic intersections of green and digital adaptability.

These pre-computed structural arrays ensure that external dashboards (such as the reference Flask UI) can render complex, multidimensional visualisations without independently executing heavy vector calculations.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

7. Conclusion

Deliverable 3.3 formally documents the initial software iteration of the SNAP research ecosystem. As a proof-of-concept backend architecture, SNAP successfully demonstrates a highly interoperable, API-driven methodology for fusing official European labour market statistics with complex occupational taxonomies. By bridging the macro-level demographic data of Eurostat with the micro-level skill requirements of ESCO, the system establishes a mathematically grounded foundation for simulating labour market transitions towards green and digital economies.

The architectural decisions detailed within this deliverable are strictly aligned with the overarching objectives of the Horizon Europe framework. By championing a "local-first" model orchestration paradigm—utilising local instances of Ollama, vLLM, and OpenWebUI—the toolkit supports data sovereignty, privacy compliance, and scientific reproducibility. Furthermore, the implementation of a Medallion storage strategy and advanced vector memory architectures (such as ChromaDB and PaperQA) ensures that both generated demographic data and extracted academic knowledge maintain auditable data lineage and content-hash-based provenance.

Moving beyond foundational infrastructure, the toolkit operationalises four core researcher utilities: the Synthetic Persona Generator, the Research Companion, the Unstructured Extraction Tool, and the Natural Language Data Analysis engine. Collectively, these subsystems democratise complex computational methodologies, allowing researchers to execute probabilistic cohort simulations, perform citation-backed literature retrieval, and interrogate unstructured data without requiring deep programming expertise.

In its current iteration, the SNAP backend serves as a robust, containerised pipeline. Future developmental phases will focus on scaling the semantic memory stores, broadening the taxonomy ingestion capabilities, and subjecting the synthetic persona outputs to rigorous empirical validation against real-world sociological surveys. Ultimately, this architecture provides the European research community with a secure, sovereign, and methodologically transparent infrastructure for navigating the complexities of the modern computational research landscape.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Bibliography

- **ESCO:** European Commission. (n.d.). *European Skills, Competences, Qualifications and Occupations (ESCO)*. Directorate-General for Employment, Social Affairs and Inclusion.
- **ISCO:** International Labour Organization (ILO). (2012). *International Standard Classification of Occupations: ISCO-08*. Geneva: ILO.
- **Eurostat LFS:** Eurostat. (n.d.). *European Union Labour Force Survey (EU-LFS)*. Datasets referenced: lfsa_egised (Employment by sex, age and educational attainment level), lfsa_pganws (Population by sex, age, migration status), lfsa_eiisco (Employment by sex, age and occupation).
- **vLLM:** Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023). *Efficient Memory Management for Large Language Model Serving with PagedAttention*. In Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23).
- **Mistral:** Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., ... & Sayed, W. E. (2023). *Mistral 7B*. arXiv preprint arXiv:2310.06825. *(Update this to the specific Mistral-Small-24B paper/announcement if applicable).*
- **Ollama:** Ollama. (2024). *Ollama: Get up and running with Llama 3, Mistral, Gemma, and other large language models locally*. Retrieved from https://ollama.com.
- **PaperQA:** Lála, J., O'Donoghue, O., Shtedritski, A., Cox, S., White, A. D., & Schwallko, P. (2023). *PaperQA: Retrieval-Augmented Generative Agent for Scientific Research*. arXiv preprint arXiv:2312.07559.
- **LangChain:** Chase, H. (2022). *LangChain*. Retrieved from https://github.com/langchain-ai/langchain.
- **PandasAI:** PandasAI. (2024). *PandasAI: Conversational Data Analysis*. Retrieved from https://github.com/Sinaptik-AI/pandas-ai.
- **SentenceTransformers:** Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.
- **ChromaDB:** Chroma. (2024). *Chroma: The AI-native open-source embedding database*. Retrieved from https://trychroma.com.
- Wu, H., Xu, J., Wang, J. and Long, M., 2021. **Autoformer**: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, *34*, pp.22419-22430.
- Kitaev, N., Kaiser, Ł. and Levskaya, A., 2020. **Reformer**: The efficient transformer. *arXiv preprint arXiv:2001.04451*.
- Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H. and Zhang, W., 2021, May. **Informer**: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 12, pp. 11106-11115).
- Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A.X. and Dustdar, S., 2021, May. **Pyraformer**: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687