Tech Stack Overview

Purpose

The main goal of this document is to explain and list each technologies and framework that will be used in the project, helping users and collaborators understand the reason behind each choices.

Context

The project we are working on is named GeneWeb it is an open source genealogy software written in OCaml which comes with a Web interface. We have been asked by our client CoinLegacy Inc to make this code compliant with current standards. They asked us to use Python for the API.

API Stack

Tech	Python	Mandatory
Framework	FastAPI	Faster calculations
Unit & Functional Testing	PyTest	Fastest tests execution
Python code formatter	black	Adapted for Python
Comprehensive Python linter	Pylint	Adapted for Python

Tech - Python

The client requested Python for the API. Python’s readability, extensive ecosystem, and strong support for web frameworks like FastAPI make it a reliable choice for building scalable and maintainable APIs.

Framework - FastAPI

We evaluated both FastAPI and Django for the project. FastAPI was chosen because it offers higher performance, better memory efficiency, and a modern, async-friendly architecture. Its concise syntax and automatic API documentation also make development faster and maintenance easier.

Unit & Functional Testing - PyTest

PyTest was chosen as the testing framework due to its simple setup, automatic test discovery, and ability to run tests in parallel, which reduces overall execution time. Its flexibility makes it suitable for both unit and functional tests, ensuring code quality and reliability throughout the backend.

Code formatting - Black

- **Deterministic formatting**: Black produces identical output regardless of who runs it, eliminating formatting debates in teams
- **AST-based**: Uses Python's Abstract Syntax Tree for parsing, ensuring it never breaks code semantics
- **Minimal configuration**: "Opinionated" by design - reduces decision fatigue and configuration overhead
- **Fast execution**: Written in Python but optimized for speed, can format large codebases quickly
- **Line length optimization**: Intelligently breaks lines to improve readability while respecting the 88-character default
- **String normalization**: Consistently handles quotes (prefers double quotes) and string concatenation

Linting - PyLint

- **Comprehensive analysis**: Performs static code analysis, not just style checking
- **Error categories**:
  - **(E) Errors**: Likely bugs in code
  - **(W) Warnings**: Style/minor issues
  - **(R) Refactor**: Code smells/complexity issues
  - **(C) Convention**: PEP 8 compliance

- **Configurable severity**: Can tune which checks to enforce
- **Plugin architecture**: Extensible for framework-specific checks (Django, Flask, etc.)
- **Metrics reporting**: Provides code quality scores and complexity measurements

---

Web App Stack

Framework	Vue.js	Great for interactive interface
Linter	Prettier & ESLint	Standard for Vue projects
E2E Testing	Cypress	Faster Response, and modern
Unit & Functional Testing	VTU & Vitest	Native, less dependencies

Framework - Vue.js

Vue.js was chosen for its excellent documentation and high performance. It excels at building highly interactive and responsive user interfaces, making it ideal for dashboards and dynamic web applications.

Linter - Prettier & ESLint

We chose Prettier and ESLint to enforce code style and consistency, as they are standard in Vue projects. Prettier automatically formats the code, while ESLint ensures compliance with coding rules and best practices. Together, they improve code quality, readability, and maintainability.

E2E Testing - Cypress

We evaluated both Cypress and Selenium for end-to-end testing, and Cypress was chosen because it provides several advantages. Unlike Selenium, Cypress runs inside the browser, which results in faster execution and more accurate testing of modern web applications. Its setup for CI/CD pipelines is simpler, and the interactive test runner makes debugging more efficient, allowing developers to inspect tests step by step.

When to Use Each Approach

Method	Use Case
UI Testing	Test user experience, visual feedback, interactions
Intercept & Stub	Test how UI handles errors, edge cases, loading states
Intercept & Verify	Ensure frontend sends correct data, handles responses properly
Direct API (`cy.request`)	Test backend logic, set up test data, verify database state

Unit & Functional Testing - VTU & Vitest

We chose VTU & Vitest, the official testing framework for Vue, because it minimizes additional dependencies while providing efficient memory usage and fast test execution. It integrates seamlessly with Vue components, making it easy to write reliable unit and functional tests for the web app.