Over the past few months, I have dedicated my free time to developing personal projects that reflect my interest in technology applied to education, self-reflection, and professional growth. I fully developed these projects myself, from frontend to backend, all the way through to deployment on Azure App Service with automated CI/CD pipelines via Azure DevOps.

Quizzify (url: https://quizzify.eu)

I created Quizzify.me to support learning through automatically generated quizzes based on online content or PDF documents. The idea originated from a desire to make studying more active and engaging through gamification.

Technologies used:

- **Frontend:** React, TypeScript, TailwindCSS for a clean and responsive user interface
- **Backend:** Python with Flask, REST architecture to manage requests and integrate language models
- **AI & NLP:** Azure OpenAI for question generation, Langchain for orchestrating LLM pipelines, LangGraph to model conditional quiz generation workflows, LangSmith for prompt validation and performance monitoring
- **Cloud & DevOps:** Firebase for authentication and data persistence (Firestore), Azure App Service for deployment, Azure Pipelines for release automation

I handled the full lifecycle of the application—from frontend design and development, to API creation and automated question generation, to deployment and production monitoring. I paid special attention to the quality of the questions generated and the usability of the learning experience, optimizing prompts and tailoring feedback management. The app allows users to track their progress and offers an intuitive, personalized learning interface.

Parallax (url: https://parallax.chat/)

Parallax is an interactive platform designed to stimulate reflection and awareness. Users can submit a thought and receive a rewritten version from an Opposite, Neutral, or Empathetic perspective, then engage in conversation with AI agents that embody those viewpoints.

Technologies used:

- **Frontend:** Next.js, React, TypeScript, TailwindCSS
- **Backend:** Python with Flask
- **AI & NLP:** Azure OpenAI, Langchain for conversational flow orchestration, LangGraph for asynchronous agent logic, LangSmith for observability and optimization
- **Realtime:** WebSocket for continuous interaction
- **Cloud & DevOps:** Firebase Firestore, Azure App Service, Azure Pipelines for CI/CD

I designed and implemented the logic behind the three AI agents, differentiating their behavior through engineered prompts and asynchronous interactions. This project allowed me to explore the potential of conversational AI in fostering constructive dialogue.

Ask My CV

Ask My CV is a tool I designed to explore my professional skills in a more engaging and interactive way through an AI chatbot, using a knowledge base derived from my CV. The main goal is not to simulate an interview, but to experiment with advanced technologies to convert a resume into a navigable knowledge graph, and deepen my understanding of RAG pipelines and graph exploration with Neo4j. The project began as a hands-on exercise to integrate document parsing, semantic enrichment, and combined retrieval (graph + embedding) to ensure the LLM responds using only verified and controlled sources.

Through an advanced pipeline, information is:

- Expanded and clarified using LLMs on Azure OpenAI
- Semantically indexed via embedding and chunking
- Represented as a knowledge graph in Neo4j for advanced querying

Technologies used:

- **Frontend:** React, TailwindCSS
- **Backend:** Python with FastAPI and advanced modules for asynchronous processing and document handling
- **AI & NLP:** Azure OpenAI, Langchain for the conversational pipeline, LangGraph to manage state, LangSmith for response monitoring
- **Parsing:** OCR-based conversion of PDF pages into HTML, followed by summarization and semantic enrichment via LLMs
- **Cloud & DevOps:** Azure App Service, Azure DevOps with Azure Pipelines

The entire process was designed and developed by me, with a focus on structured knowledge extraction and semantic graph construction. This approach not only enhances the understanding of a candidate's skills but also enables dynamic adaptation of responses. I implemented an asynchronous workflow that transforms each PDF page into an image, converts it to HTML using LLMs, synthesizes it into coherent text, and finally splits and indexes it to be mapped into a knowledge graph. The system implements a Retrieval-Augmented Generation (RAG) approach, combining Neo4j knowledge graph queries with embedding-based search to provide accurate, source-grounded responses. This ensures high reliability and traceability of generated content.