1. *The business problem and who the customers/users are*

The business problem consists of an oil and gas drilling investment firm (customer), with a significant amount of money to put into work. The customer is limited to a small number of deals they can participate in, due to the lack of qualified financial analysts and off-the-self financial analysis automation solutions.

Energy industry Thought Leaders have indicated the marketplace for buying and selling oil and gas well working interest is to grow significantly in 2025 and beyond. But this marketplace will still be very competitive and fast paced, so potential investors must perform fast Return on Investment analysis, most of the time within a day or two of the deal’s offering.

It is proposed to build a custom automation software solution that will enable faster Return on Investment analysis thus potentially increasing deal participation and putting more investment money into work.

1. *How you worked with them to make sure that your solution meets their needs*

As a technology consultant, I coordinated and led recurring work sessions to discover, document/learn current processes, and demonstrate working software. All work sessions are recorded, transcribed and summarized by AI then emailed to each team member. These summaries provided quality feedback, with sentiment, that helped ensure that the solution was meeting the client’s needs.

A simple roadmap document (below) was created, as a communications tool, that identified and prioritized key tasks that could be automated.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Roadmap**

1. *Demo the solution and explain how it addresses the business problem*

This demonstration will show how this solution automated several previous manual tasks, thus reclaiming up to 20 hours to analysis, per deal, that was used complete higher value tasks. Two of these automated tasks are:

NOTE: My goal here is to highlight my python programming skills and show how AI is used to automate previous manual tasks. If time allows, as the end, I would like to explain my plan to refactor this solution into an agentic LLM application using an well-known agent orchestration framework, such as LangGraph.

* Reduce manual data entry, using OCR and AI to extract key attributes from the PDF and image files. The value-add here is the replacement of previous error prone copy-and-paste tasks of text from both PDFs and images into an Excel spreadsheet. The automated task now extracts the desired text feature and inserts them into a database to downstream processing.
* Automation of deterministic tasks, such as “Offset Well Co-development Groups” and “Child Well Risk Gun Barrel Plot”. For example,
  + “Offset Well Co-development Groups” is a report that list groups well that will be used as a substitute for the projected future cash flows of the proposed well grouping and a key input to Return on Investment calculation.
  + “Child Well Risk Gun Barrel Plot” is a Scatter Plot that indicates any surround well groups to the proposed well group that could potentially imped the future cash flows. Any identified risk is subtracted from the future cash flows calculation.

1. *Show and explain the architecture, platform, technical tools, language used, e.g., RAG model, AWS Bedrock, LangChain, python (strong preference) including which libraries, etc.*

Please see the Github repository [AFE\_Public](https://github.com/stevethomas15977/public_afe/tree/main/app) for a complete list of technologies.

The web application is built using python. The python libraries may be viewed via this [Pipfile](https://github.com/stevethomas15977/public_afe/blob/main/app/Pipfile). The UI is built with [NiceGUI](https://nicegui.io/), which is a python-base SPA (Single Page Application) web framework. Database management is provided by SQLite. Excel generation is provided by python library [XlsxWriter](https://xlsxwriter.readthedocs.io/). Current AI and OCR technologies include ChatGPT and Amazon Textract. These technologies are used to extract text from a PDF. [PDFHelper](https://github.com/stevethomas15977/public_afe/blob/main/app/helpers/anakarko_afe_helper.py)

1. *Explain how the solution was deployed, maintained, and supported*

The current version is a simple monolith web application running on AWS LightSail. LightSail is a very cost-effective compute solution for small to medium businesses. Compute components consist of a single load balancer for TLS termination and a target-group of EC2 instance running a python-based web application. See diagram below “AFE Analysis (v1.0).

The solution is deployed using [Github Actions](https://github.com/stevethomas15977/public_afe/blob/main/.github/workflows/main.yaml), [Terraform](https://github.com/stevethomas15977/public_afe/tree/main/terraform), and Linux BASH cloud-init scripting.

Maintain and support as included in a separate SLA engagement.

A diagram of a diagram of a diagram

AI-generated content may be incorrect.

**AFE Analysis (v1.0)**