**LIVEBOOK AND ITS POWER.**

Scoping Document.

In this project, an overview of Livebook is given and later on an in-depth walk-through.

Livebook is a web application for writing interactive and collaborative code notebooks, just as Jupyter notebook works with Python. Livebook and its features work well with Elixir and beam to best bring out the core elixir features.

The elixir features to be demonstrated in this project via livebook could also be considered as the benefits of using Elixir and they include;

1. Scalability - an application program would be scalable if it could be moved from a smaller to a larger operating system and take full advantage of the larger operating system in terms of performance (user response time and so forth) and the larger number of users that could be handled.

- This is seen on the portability and being able to version control .livemd files, even using third party applications such as GitHub.

1. Concurrency - the ability to perform two or more tasks apparently at the same time.

- This is demonstrated by being able to run multiple nodes that use Elixir BEAM on Livebook . The teaming feature unfortunately is still under development. Bidirectional live counter is one good example.

1. Fault- tolerance - Elixir has built-in support for fault tolerance, including features like supervision trees and fault-recovery mechanisms. Livebook does support supervision and child processes using Kino.
2. Functional programming - this will be a simple code snippet showing Elixir’s functional programming and the output.
3. Erlang compatibility - You can directly code Erlang on Livebook side by side with Elixir and this will be demonstrated by a simple piece of code in both erlang and Elixir.
4. OTP Distribution-

Elixir allows us to communicate across these nodes via a few different mechanisms, this is equally possible in Livebook and can be demonstrated using a portal game

Some of the Livebook features that will be learned during the demonstration include but not limited to;

* Markdown Support
* Inline Documentation and code formatting
* Interactive results such as tables , data visualization and maps
* Automation using Smart cells
* Collaboration with multiple users
* Custom runtimes in terms of connecting to nodes and having different runtimes.

By the end of the project should be a clear understanding of,

* Elixir core features implemented on Livebook
* Process and OTP handling in Livebook
* How ecto is integrated using Kino
* Data visualization using Mermaid and VegaLite.
* Nodes concurrency and key application data.

A list of tutorials, websites, repositories and videos used to explain Live book and its power will be included for any future references needed.