**Elixir Code Challenge**

**The Source code based on problem statement has been developed and share(tx\_exchange.zip) with email.**

**Solution: Implementation is aligned with requirement and I enhanced solution considering below point.**

* **Supervisor: Supervisor has been implemented to make the fault tolerant system, that resilient to any error. Supervisor capable the application to restart the process if any crash occurred.**
* **Application: Application behavior being used to restart the Exchange server once we start the exchange server with “iex -S mix”.**
* **GenServer: GenServer being used to manage the state and complete the assign task in concurrent fashion.**
* **Process Name: being used in while starting the GenServer, that help us to avoid passing pid(process id) while calling the function on book library.**
* **Test case: Test case is not being covered in delivered assignment, considering the time crunch. But I am well aware with the importance of test case.**
* **I have written the help test function under TxExchange class just for development process which help me to expedite the process.**

**We can call the below set of api in the iex session to test the whole given scenario.**

**// initialize the inventory to perform the operation.**

iex> book1 = %TxBook{price: 20, quantity: 20, side: :bid, price\_level\_index: 1 }

iex> book2 = %TxBook{price: 5, quantity: 200, side: :ask, price\_level\_index: 1 }

iex> book3 = %TxBook{price: 30, quantity: 30, side: :bid, price\_level\_index: 2 }

iex> book4 = %TxBook{price: 40, quantity: 40, side: :ask, price\_level\_index: 2 }

1. iex> TxExchange.send\_instruction(book1) //it will give the duplicate error
2. iex> book1 = %TxBook{instruction: :update,price: 200, quantity: 200, side: :bid, price\_level\_index: 1 }
3. iex> TxExchange.send\_instruction(book1) // book will update in inventory
4. iex> book1 = %TxBook{instruction: :delete,price: 200, quantity: 200, side: :bid, price\_level\_index: 1 }
5. iex> TxExchange. get\_all\_book() // get the list of book added so far.
6. iex> TxExchange.send\_instruction(book1)
7. iex> > TxExchange. get\_all\_book() // get the list of book added so far.
8. iex> book1 = %TxBook{instruction: :new,price: 200, quantity: 200, side: :bid, price\_level\_index: 1 }
9. iex> TxExchange.send\_instruction(book1)// add the book in inventory
10. iex> TxExchange.order\_book(2) // get the inventory in desired format…
11. **iex:>**