

**Course: CP108 PLUTUS/HASKELL 1****Problem statement:**

Most African governments are trying different measures to eliminate corruption, bad governance, mismanagement and lack of accountability in their countries but these efforts are mostly unsuccessful. As a tool for change, blockchain can help in solving some of these governance issues plaguing Africa.

**Task:**

With this in mind, think of a way you can implement a smart contract that can help eliminate any of these ills, then implement it using Plutus.

**Project:**

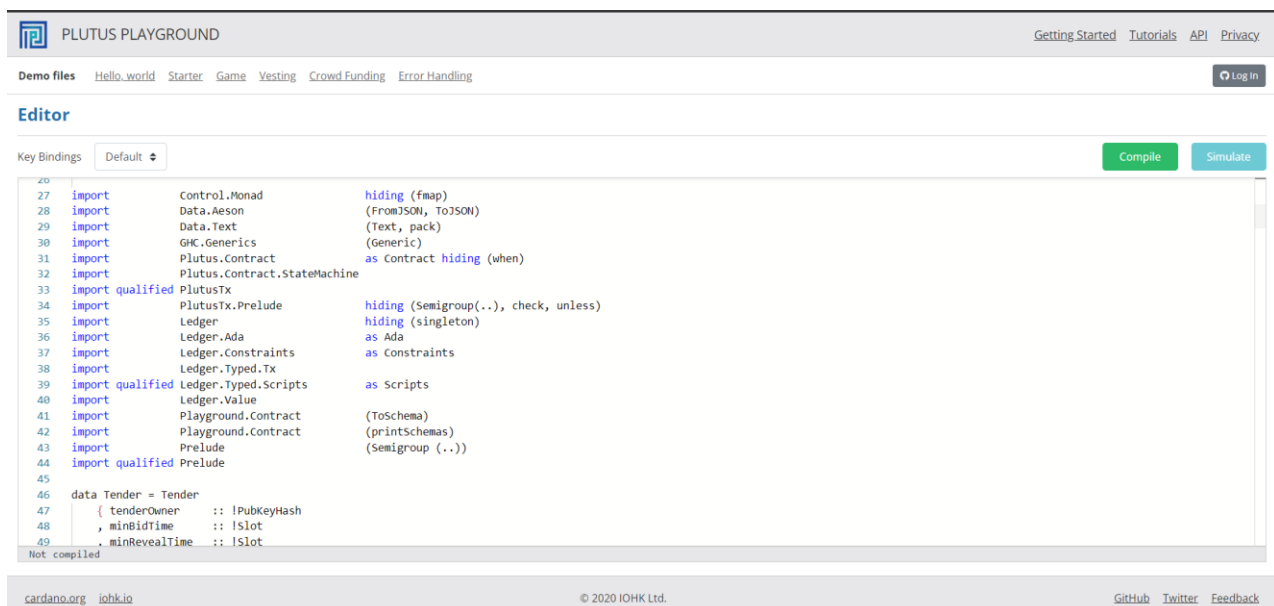
A smart contract on the Cardano blockchain that can help eliminate corruption and/or bad governance in Africa.

**Suggested solution:**

Introduce blockchain voting systems. A way that ensure that elections are done in a transparent, tamper proof and fair conduct.

The goal of this project is to create a smart contract on the Cardano blockchain that could be used as voting systems by electoral bodies. Through this we ensure secure, transparent, and above all fair elections in African countries and also all around the globe.

**Script:** the **Fabio Thomaz Molinar\_Tender.hs** can be used for this exam and can be tested on the Plutus Playground. <https://playground.plutus.iohkdev.io/>



The screenshot shows the Plutus Playground interface. At the top, there's a header with the logo and navigation links: Getting Started, Tutorials, API, Privacy. Below that, a 'Demo files' section lists various templates like Hello\_world, Starter, Game, Vesting, Crowd Funding, and Error Handling. The main area is the 'Editor' with a 'Key Bindings' dropdown set to 'Default'. On the right of the editor are 'Compile' and 'Simulate' buttons. The code in the editor is a Haskell script for a smart contract, starting with imports from Control.Monad, Data.Aeson, Data.Text, GHC.Generics, Plutus.Contract, Plutus.Contract.StateMachine, PlutusTx, PlutusTx.Prelude, Ledger, Ledger.Ada, Ledger.Constraints, Ledger.Typed.Tx, Ledger.Typed.Scripts, Ledger.Value, Playground.Contract, Playground.Contract, Prelude, and PlutusTx.Prelude. It defines a data type 'Tender' with fields 'tenderOwner' (IPubKeyHash), 'minBidTime' (ISlot), and 'minRevealTime' (ISlot). The code is not yet compiled, as indicated by the 'Not compiled' message at the bottom of the editor area. The footer of the page includes links to cardano.org and iohk.io, a copyright notice for 2020 IOHK Ltd., and links to GitHub, Twitter, and Feedback.

```
27 import Control.Monad hiding (fmap)
28 import Data.Aeson      (FromJSON, ToJSON)
29 import Data.Text       (Text, pack)
30 import GHC.Generics    (Generic)
31 import Plutus.Contract as Contract hiding (when)
32 import Plutus.Contract.StateMachine
33 import qualified PlutusTx
34 import PlutusTx.Prelude hiding (Semigroup(..), check, unless)
35 import Ledger          hiding (singleton)
36 import Ledger.Ada      as Ada
37 import Ledger.Constraints as Constraints
38 import Ledger.Typed.Tx
39 import qualified Ledger.Typed.Scripts as Scripts
40 import Ledger.Value
41 import Playground.Contract (ToSchema)
42 import Playground.Contract (printSchemas)
43 import Prelude            (Semigroup(..))
44 import qualified Prelude
45
46 data Tender = Tender
47   { tenderOwner  :: IPubKeyHash
48   , minBidTime   :: ISlot
49   , minRevealTime :: ISlot
50 }
```

The smart contract is meant to work on a base that no one including its operators can see the votes, which will be taken inform of bids, already put in ballot until the election process is over. This will help keep human interference to a minimum and ensure that whatever the results of the elections are, they are true and fair.

Here is a direct link to the code to be used herein:

[https://github.com/TechEBU/Fabio-Thomaz-Molinar-Project/blob/main/Fabio%20Thomaz%20Molinar\\_Tender.hs](https://github.com/TechEBU/Fabio-Thomaz-Molinar-Project/blob/main/Fabio%20Thomaz%20Molinar_Tender.hs)

and also an example of the same:

```
324 lines (282 sloc) | 13.9 KB
Raw Blame

1 {-# LANGUAGE DataKinds           #-}
2 {-# LANGUAGE DeriveAnyClass      #-}
3 {-# LANGUAGE DeriveGeneric       #-}
4 {-# LANGUAGE FlexibleContexts    #-}
5 {-# LANGUAGE MultiParamTypeClasses #-}
6 {-# LANGUAGE NoImplicitPrelude   #-}
7 {-# LANGUAGE OverloadedStrings   #-}
8 {-# LANGUAGE ScopedTypeVariables #-}
9 {-# LANGUAGE TemplateHaskell     #-}
10 {-# LANGUAGE TypeApplications    #-}
11 {-# LANGUAGE TypeFamilies        #-}
12 {-# LANGUAGE TypeOperators       #-}
13
14 -- TO DO: expose objects
15
16 module Tender
17   ( Tender      (..)
18   , CreateParams (..)
19   , BidParams   (..)
20   , RevealParams (..)
21   , CancelParams (..)
22   , FinishParams (..)
23   , TenderSchema
24   , endpoints
25   ) where
26
27 import Control.Monad           hiding (fmap)
28 import Data.Aeson             (FromJSON, ToJSON)
29 import Data.Text               (Text, pack)
30 import GHC.Generics            (Generic)
31 import Plutus.Contract         as Contract hiding (when)
32 import Plutus.Contract.StateMachine
33 import qualified PlutusTx
34 import PlutusTx.Prelude        hiding (Semigroup(..), check, unless)
35 import Ledger                  hiding (singleton)
36 import Ledger.Ada              as Ada
37 import Ledger.Constraints      as Constraints
38 import Ledger.Typed.Tx
39 import qualified Ledger.Typed.Scripts as Scripts
40 import Ledger.Value
41 import Playground.Contract     (ToSchema)
42 import Prelude                 (Semigroup (..))
43 import qualified Prelude
44
```

```

44
45 data Tender = Tender
46   { tenderOwner  :: !PubKeyHash
47   , minBidTime   :: !Slot
48   , minRevealTime :: !Slot
49   , txCost       :: !Integer
50   , tToken       :: !AssetClass
51   } deriving (Show, Generic, FromJSON, ToJSON, Prelude.Eq, Prelude.Ord)
52
53 PlutusTx.makeLift ''Tender
54
55 data BidInfo = BidInfo
56   { bidOwner  :: !PubKeyHash
57   , bidValue  :: !Integer
58   } deriving (Show, Generic, FromJSON, ToJSON, ToSchema, Prelude.Eq, Prelude.Ord)
59
60 -- Need to defined an instance of Eq only because I really need an instance of Ord
61 instance Eq BidInfo where
62   {-# INLINABLE (==) #-}
63   BidInfo i == BidInfo i' = bidValue i == bidValue i'
64
65 -- Defining an instance of Ord to make it easier to sort bid infos
66 instance Ord BidInfo where
67   {-# INLINABLE (<=) #-}
68   BidInfo i <= BidInfo i' = bidValue i <= bidValue i'
69
70 PlutusTx.unstableMakeIsData ''BidInfo
71
72 data TenderDatum = TenderDatum (Maybe [ByteString]) (Maybe [BidInfo]) | Finished
73   deriving Show
74
75 instance Eq TenderDatum where
76   {-# INLINABLE (==) #-}
77   TenderDatum i == TenderDatum i' = bidValue i == bidValue i'
78
79 PlutusTx.unstableMakeIsData ''TenderDatum
80
81 data TenderRedeemer = Create
82   | Bid ByteString
83   | Close
84   | Reveal ByteString BidInfo
85   | Finish
86   | Cancel
87   deriving Show
88
89 PlutusTx.unstableMakeIsData ''TenderRedeemer
90

```

```

140     where
141         token :: Value
142         token = assetClassValue (tToken tender) 1
143
144     {-# INLINEABLE final #-}
145     final :: TenderDatum -> Bool
146     final Finished = True
147     final _       = False
148
149     {-# INLINEABLE bidInfoToBS #-}
150     bidInfoToBS :: BidInfo -> ByteString
151     bidInfoToBS bi = sha2_256 (appendByteString (getPubKeyHash $ bidOwner bi) (consByteString $ bidValue bi))
152
153     {-# INLINEABLE check #-}
154     check :: ByteString -> TenderDatum -> TenderRedeemer -> Bool
155     check bs (TenderDatum (Just bs) _) (Reveal bs' bi') = bidInfoToBS bi' == bs
156
157     {-# INLINEABLE tenderStateMachine #-}
158     tenderStateMachine :: Tender -> ByteString -> StateMachine TenderDatum TenderRedeemer
159     tenderStateMachine tender bs = StateMachine
160         { smTransition = transition tender
161         , smFinal      = final
162         , smCheck      = check bs
163         , smThreadToken = Just $ tToken tender }
164
165     {-# INLINEABLE mkTenderValidator #-}
166     mkTenderValidator :: Tender -> ByteString -> TenderDatum -> TenderRedeemer -> ScriptContext -> Bool
167     mkTenderValidator tender bs = mkValidator $ tenderStateMachine tender bs
168
169     type Tendering = StateMachine TenderDatum TenderRedeemer
170
171     tenderStateMachine' :: Tender -> StateMachine TenderDatum TenderRedeemer
172     tenderStateMachine' tender = tenderStateMachine tender bs
173
174     tenderInst :: Tender -> Scripts.ScriptInstance Tendering
175     tenderInst tender = Scripts.validator @Tendering
176         ( $(PlutusTx.compile [| mkTenderValidator |])
177         `PlutusTx.applyCode` PlutusTx.liftCode tender
178         `PlutusTx.applyCode` PlutusTx.liftCode bs)
179     $$(PlutusTx.compile [| wrap |])
180     where
181         wrap = Scripts.wrapValidator @TenderDatum @TenderRedeemer
182
183     tenderValidator :: Tender -> Validator
184     tenderValidator = Scripts.validatorScript . tenderInst
185
186     tenderAddress :: Tender -> Ledger.Address

```

With this smart contract we eliminate corruption in electoral bodies especially on presidential candidate seats. Then African countries are able to see the change in their countries with their fairly elected leaders. This will enhance including the economic and general development in African countries.