project_design

November 15, 2021

1 Project Design

Dante Basile and Jocelyn Hsu

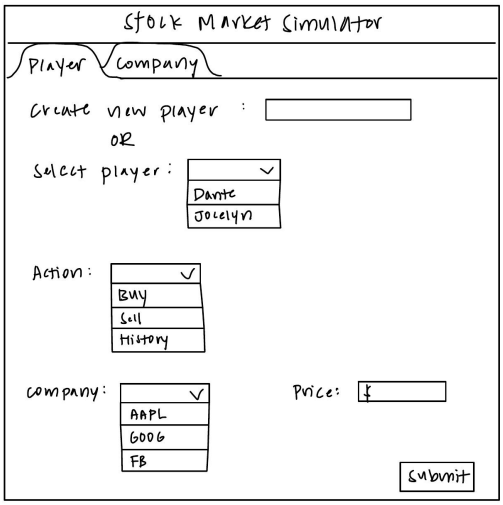
1. An overview of the purpose of the project
   - The overall purpose of this project is to provide an environment for everyone to simulate the stock market. From beginners interested in learning about the basics of trading to economist enthusiasts developing models to predict trends in the stock market and looking for a simulation environment, our project will serve as a great starting point for anyone interested in experimenting with trading.
2. A list of libraries you plan on using
   - Core: data structures
   - yojason: store game state
   - Async: API access
   - Labgtk: Graphical interface
3. Commented module type declarations (.mli files) which will provide you with an initial specification to code to

```
(*
  Contains a ticker for all companies listed on the market
  Maps ticker to list of past and present stock prices
  string -> float list
*)
type stock_price_map = float list Base.Map.M(Core.String).t

(*
  Maps ticker to past and present shares owned
  string -> int list
*)
type stock_ct_map = int list Base.Map.M(Core.String).t

(*
  Record to store funds and stocks of player
*)
type player = { funds : float list; stocks : stock_ct_map; }

(*
```

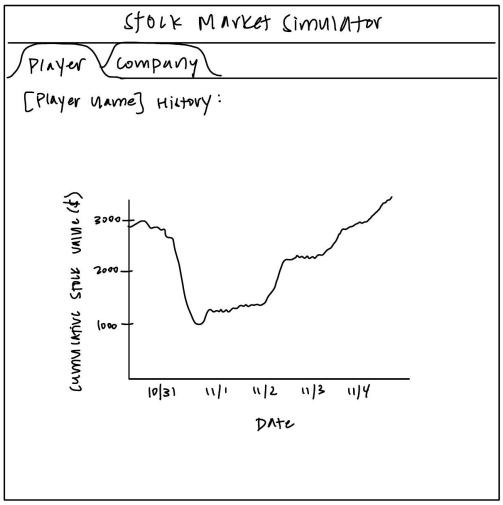
```
  Maps id to player
  string -> player
*)
type player_map = player Base.Map.M(Core.String).t

(*
  Record to store details of order
*)
type order = { id : string; ct : int; value : float; }

(*
  Maps ticker to list of relevant order tuples
  string -> order list
*)
type order_map = order list Base.Map.M(Core.String).t

(*
  Add stock to stock_price_map
*)
val add_stock : string -> stock_price_map -> (stock_price_map, string) result

(*
  Add player to player_map
*)
val add_player : string -> player_map -> (player_map, string) result

(*
  Player acquires stock from IPO
*)
val buy_ipo : string -> order -> stock_price_map -> player_map
  -> (player_map, string) result

(*
  Attempt to find an ask matching this bid and conduct the transaction
*)
val offer_bid : string -> order -> order_map -> stock_price_map -> player_map
  -> (order_map * stock_price_map * player_map, string) result

(*
  Attempt to find a bid matching this ask and conduct the transaction
*)
val offer_ask : string -> order -> order_map -> stock_price_map -> player_map
  -> (order_map * stock_price_map * player_map, string) result

(*
  Get the current price of a stock
*)
val get_price : string -> order_map -> (float, string) result
```

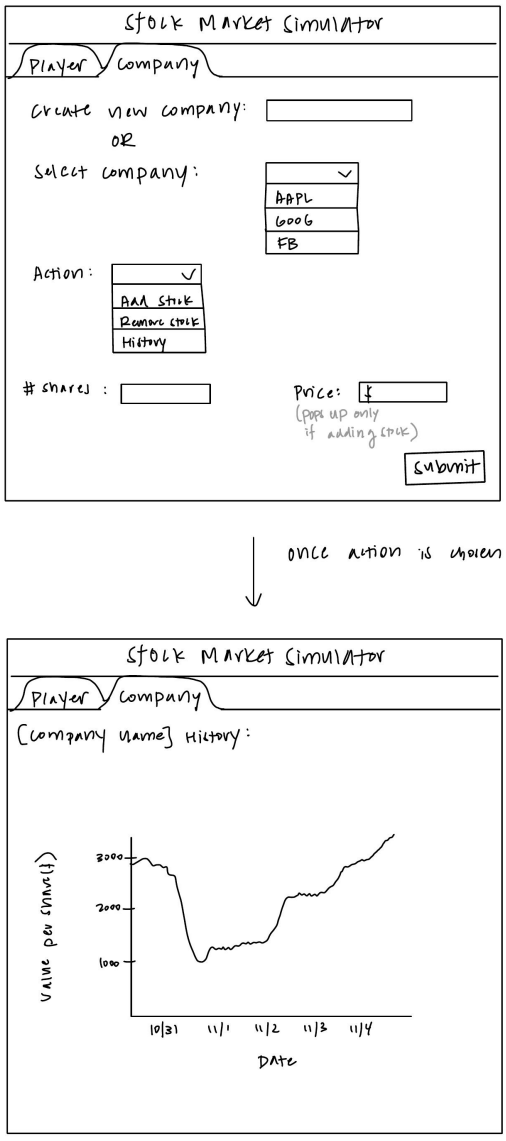
```
(*
  Get the highest bid (price a buyer is willing to pay)
  and the lowest ask (price a seller is willing to accept)
  tuple: (bid, ask)
*)
val get_bid_ask : string -> (float * float, string) result

(*
  Get the difference between the bid and ask: ask - bid
*)
val get_bid_ask_spread : string -> (float, string) result
```

4. Include a mock of a use of your application, along the lines of the Minesweeper example above but showing the complete protocol.



once action is chosen





5. Make sure you have installed and verified any extra libraries will in fact work on your computer setup, by running their tutorial examples.
   - Installed and verified all libraries, ran tutorials
6. Also include a brief list of what order you will implement features.
   1. Execution and simulation libraries (simultaneously)
   2. Plotting
   3. News feature to assess company's "popularity points"; will choose company randomly

and choose random number between -10 and 10 to adjust how popular a company is
4. Save game state

[ ]: