Holy Bible 2: Tokyo Drift 2: Electric Boogaloo

Jiajie Mai, Theodore Peters, Ryan Aday

Dog Eat Dog

Description:

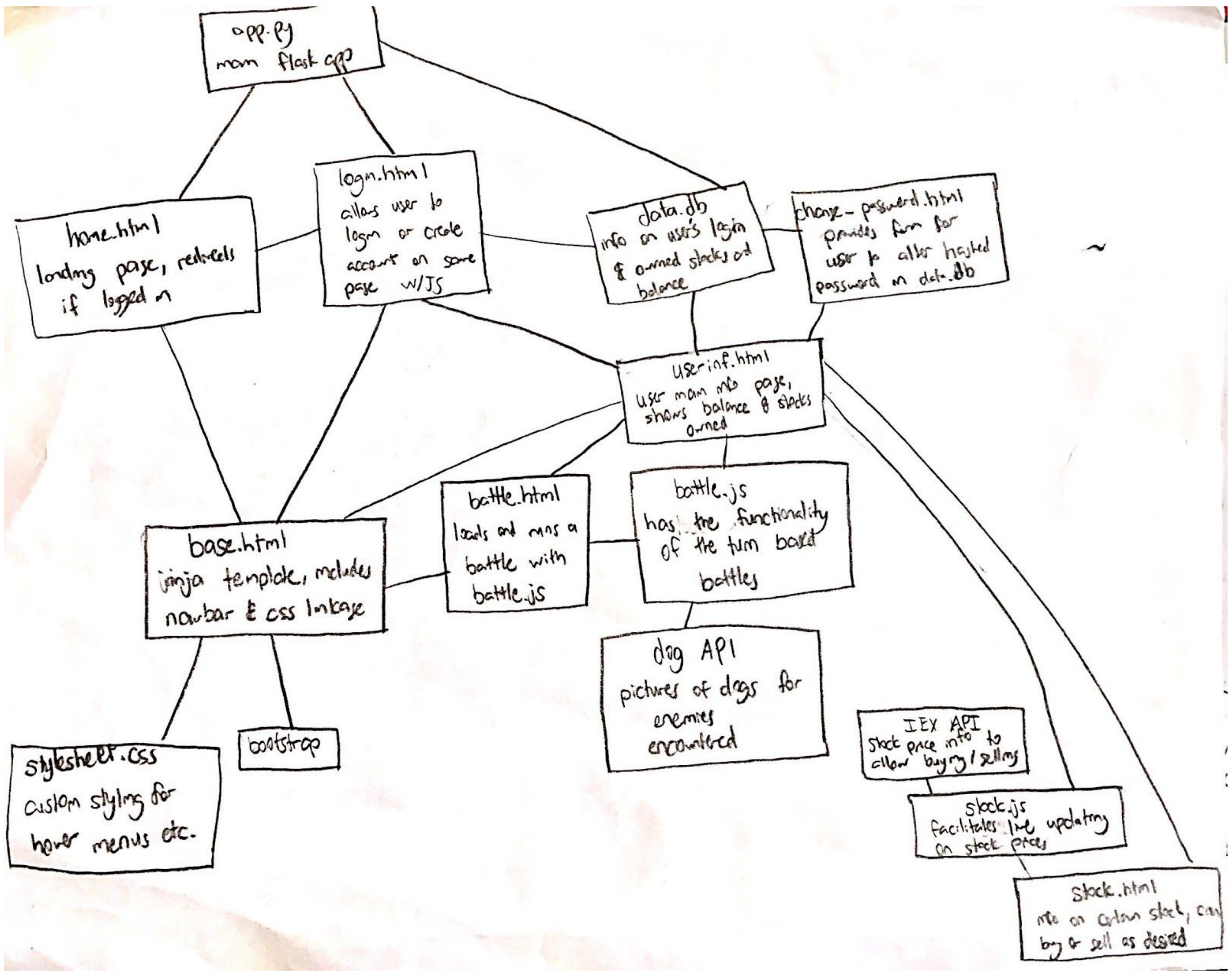
Business is a dog-eat-dog world. Why not literally play like that? Introducing, this mess-I mean our project. In this "game," you will be able to play as a dog who is fighting other dogs to get that victory royale by selling and buying stocks! (Pete would be so proud of us) We hope to use real-time stock information to create an ultimate experience and perhaps add multiplayer. *gasp!* Join us for this dumpster fire-I mean our masterpiece and do many things out on the battleground to become the number one investor: the greatest business-dog!

The Battle System:

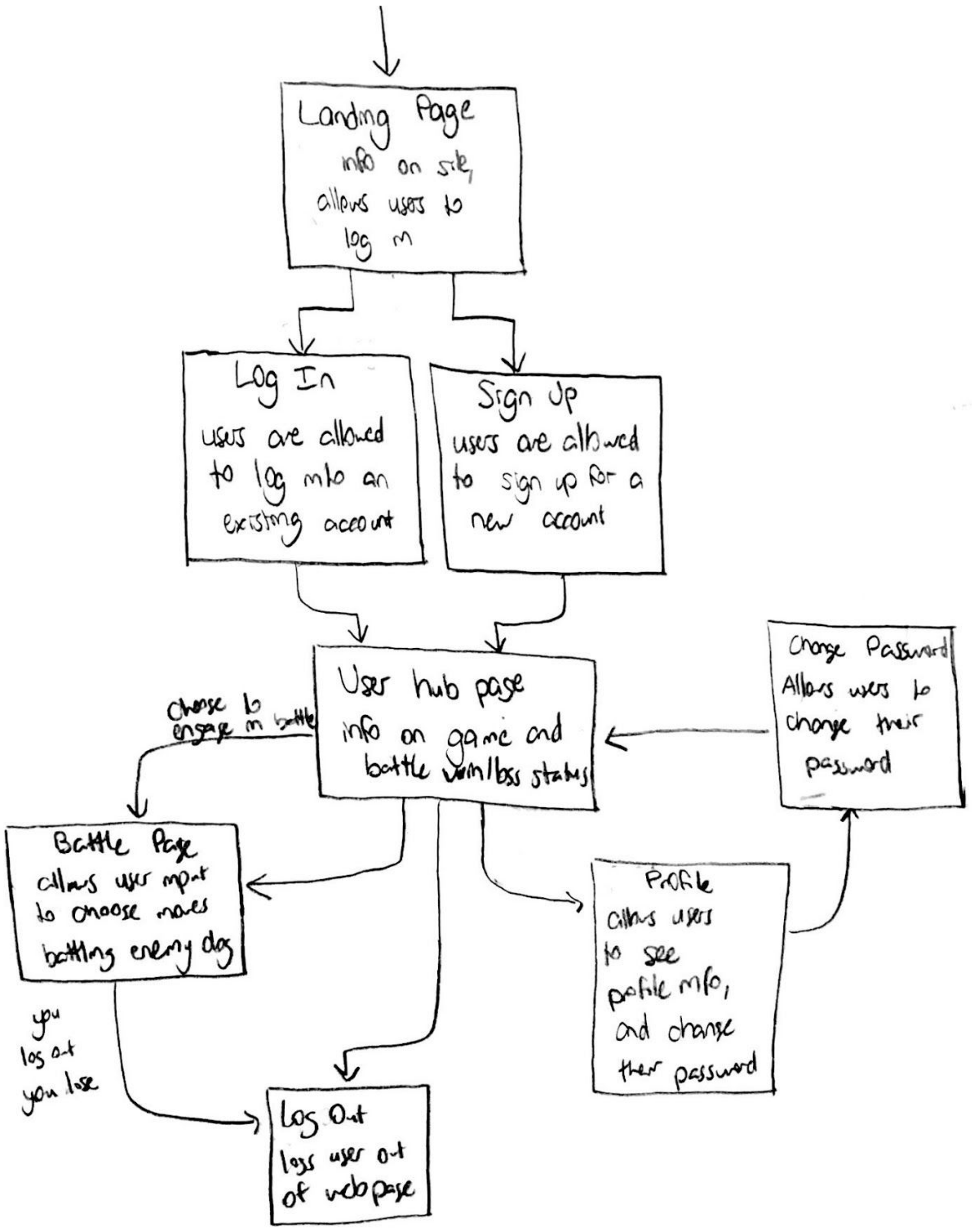
Gameplay will consist of a turn-based system where players will be presented with 5 stocks. They will choose a single stock to buy or short based on price history. Players will have a pool of money at the start of the battle that scales logarithmically with the total funds of their account. The number of shares purchased will be such that all stocks have an identical price, so the percent change after a short time period is what affects your monetary gain. After you are finished with your trading, your turn is ended and these calculations will start. As you win or lose money, the dollar amount you will be able to spend on stocks will increase, allowing you to make money more quickly. If somebody runs out of their allotted money, they immediately lose. Alternatively, whoever has the most money after 10 rounds will achieve VICTORY ROYALE (victory Quarter Pounder in the states).

The most epic gamers will quickly realize that they can gain advantages over their opponent through legally questionable gifts of dosh (cash money in the states) to the referee who presides over these momentous clashes.

Component Map:


app.py
main flask app
home.html
landing page, redirects if logged in
login.html
allows user to login or create account on same page w/JS
data.db
info on user's login & owned stocks and balance
change_password.html
provides form for user to alter hashed password in data.db
userinf.html
user main info page, shows balance & stocks owned
base.html
jinja template, includes navbar & css linkage
battle.html
loads and runs a battle with battle.js
battle.js
has the functionality of the turn based battles
dog API
pictures of dogs for enemies encountered
stylesheet.css
custom styling for hover menus etc.
bootstrap
IEX API
stock price info to allow buying / selling
stock.js
facilitates live updating on stock prices
stock.html
info on certain stock, can buy or sell as desired

Site Map:


Landing Page
info on site, allows users to log in
Log In
users are allowed to log into an existing account
Sign Up
users are allowed to sign up for a new account
User hub page
info on game and battle win/loss stats
choose to engage in battle
Change Password
Allows users to change their password
Battle Page
allows user input to choose moves battling enemy dog
Profile
allows users to see profile info, and change their password
you log out
you lose
Log Out
logs user out of webpage

Database Schema:

<u>Users-</u>

<u>Game Data-</u>

ID (Integer)	Username (String)	Password (String)
Stores identity of the account.	Stores the username for login and in-game identification.	Stores the password to be used as authorization.

ID (Integer)	Money (Integer)	Stocks Owned (json String)
Stores identity of the account.	Stores the amount of money the account has.	Stores the different stocks the account has as well as value.

Roles:
Jiajie- Project manager, and some HTML and CSS work.
Theodore- Bootstrap front-end, JavaScript, and CSS work
Ryan- Flask backend and SQLite database work

Front-end Reasoning:
Bootstrap will be used as the front-end because of its flexibility and large community behind it. This way, if we would require help, the solution is probably one Google search away. The coder behind the front-end stuff also would like to learn and get familiar with this framework.

Deadlines:
2019-01-08: Deliverables and “doc” files are finished and submitted.
2019-01-09: Get our algorithms and combat systems worked out. Work on REST API solidification.
2019-01-10: Flask backend (Ryan), SQLite database (Ryan), Bootstrap front-end (Theodore), JavaScript (Theodore), HTML (Jiajie), and CSS (Theodore and Jiajie)
2019-01-13: Finish each component and combine all of them
2019-01-14: Finish everything, review, and test

Changes: