Vend.io

Marsdin Davidson

121431159

Department

Rev: 01

Date:April 6th, 2021

Seneca

Seneca College

Of Applied Arts and Technology

Table of Contents

Code Checklist

Check list for all files that you will use in your code:

A. Webpages:
   1. addfunds.html
   2. balance.html
   3. dashboard.html
   4. gamelistings.html
   5. games.html
   6. index.html
   7. inventory.html
   8. layout.html
   9. newlist.html
   10. newwallet.html
   11. nologin.html
   12. register.html
   13. result.html
   14. resultredir.html

B. API Files:
   1. flask_main.py (All api calls are in this file)
      i. btccost & ethcost (requests.get method)
      ii. index.html ('/') calls btccost and ethcost
      iii. login.html ('/login') post/get api call. Requesting login.html via get will return nologin.html which is a custom 404 page I made. Requesting login.html via post will call select statements for my sqlite database to validate the username and password. There is a conditional on dashboard.html
      iv. Dashboard ('/dashboard') Post/Get calls. If website requested by get then return nologin.html, but if not return dashboard.html
      v. Games ('/games'). Connect to the sqlite database and store it in games and the sql table to games.html to be processed and displayed. Games.html has a for loop which runs via the python interpreter
      vi. Newlist ('/newlist') page to create a new listing to post an item. Uses the games table to generate radio buttons so the user can select which game he wants.
      vii. Nologin. Display 404 screen
      viii. Postregister – form handler to add user to sql table
      ix. Create wallet ('/createwallet'). Get the uid from the provided login and pass the uid to the wallet table when creating the listing so that it is linked to that user. Any error will be caught and an error message will pop up on the result.html page.

x. Purchase (‘/purchase’) if user select an item from the drop down they can add it to the items table which is linked via uid and view on the inventory page.
xi. Balance (‘/balance’) view the wallets that a user creates by uid. Uid is passed from the dashboard screen via GET.
xii. Inventory (‘/inventory’) select all the data from the inventory table and pass it to inventory.html.
xiii. Listbygame (‘/listbygame’) view user submitted listings and sort by game. GID is passed via GET.
xiv. Addfunds (‘/addfunds’). If this page is reached via POST then grab the live price for bitcoin or ethereum and convert user balance to the crypto type in their wallet. There is an if statement to display the balance in bitcoin or ethereum depending on which crypto they chose for that wallet. All data is passed to the html file. If there is an error, display an error message.
xv. Fundspost (‘/fundspost’) update the wallet with the dollar amount the user provides using the sql update function. This definition takes the current value and adds the user submitted value to the wallet.
xvi. Addlist (‘/addlist’) add a listing with the user submitted price. Subtract the balance from the wallet when submitting the listing then use a sql insert into function.

C. Database:
1. wallet
2. listing
3. users
4. game
5. items

vend.io

Project Overview:

This project simulates a crypto trading marketplace. Users can register, view listings by game, add simulated bitcoin or ethereum wallets and be able to post listings for items in that game.

Standards:

- Handle basic SQL and python errors.
- Create an SQL database
- Learn how to make python and sql talk to one another

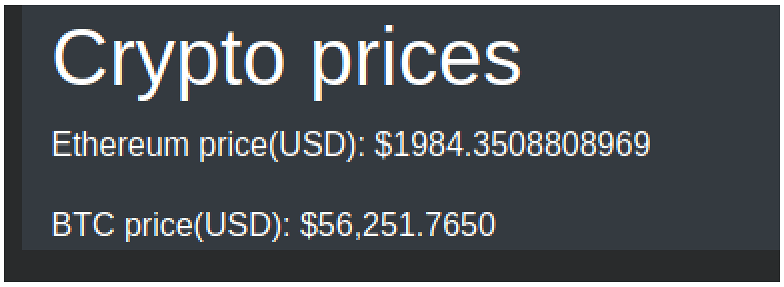
Objectives:

- Create marketplace where users can sign in, create wallets and listings.
- View supported games.
- View live crypto prices.
- Use flask to generate dynamic web pages.

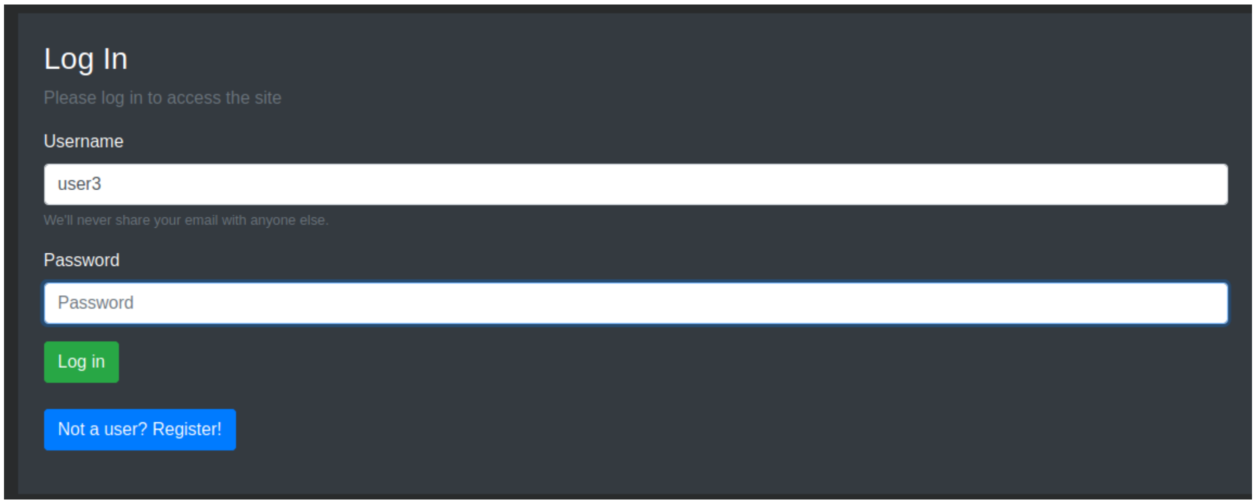
Requirements/Task(s):

Task 1 - Creating a functional website using python and a sql database.
Task 2 – Create an sql database with 5 tables or higher
Task 3 – Use 8 html pages or higher.

Project Details:

- Show the communication protocols between each module and the response page
    - btccost & ethcost (requests.get method)
    - 
      Crypto prices
      Ethereum price(USD): $1984.3508808969
      BTC price(USD): $56,251.7650
    - index.html ('/') calls btccost and ethcost
    - login.html ('/login') post/get api call. Requesting login.html via get will return nologin.html which is a custom 404 page I made. Requesting login.html via post will call select statements for my sqlite database to

validate the username and password. There is a conditional on dashboard.html

- 

Log In

Please log in to access the site

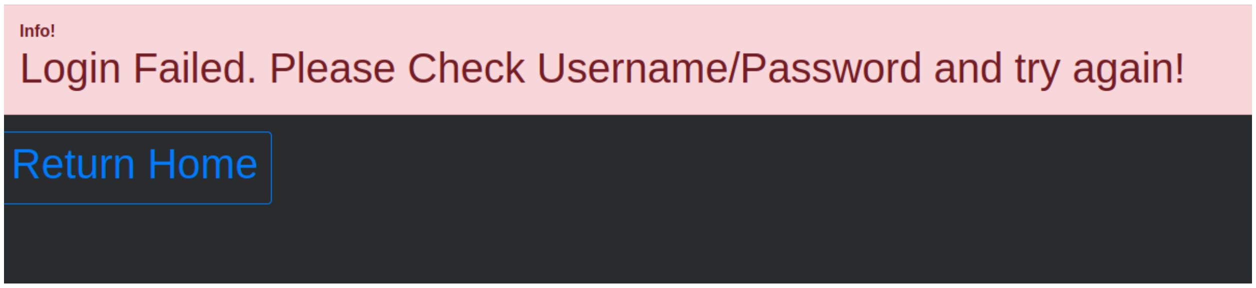
Username

user3

We'll never share your email with anyone else.

Password

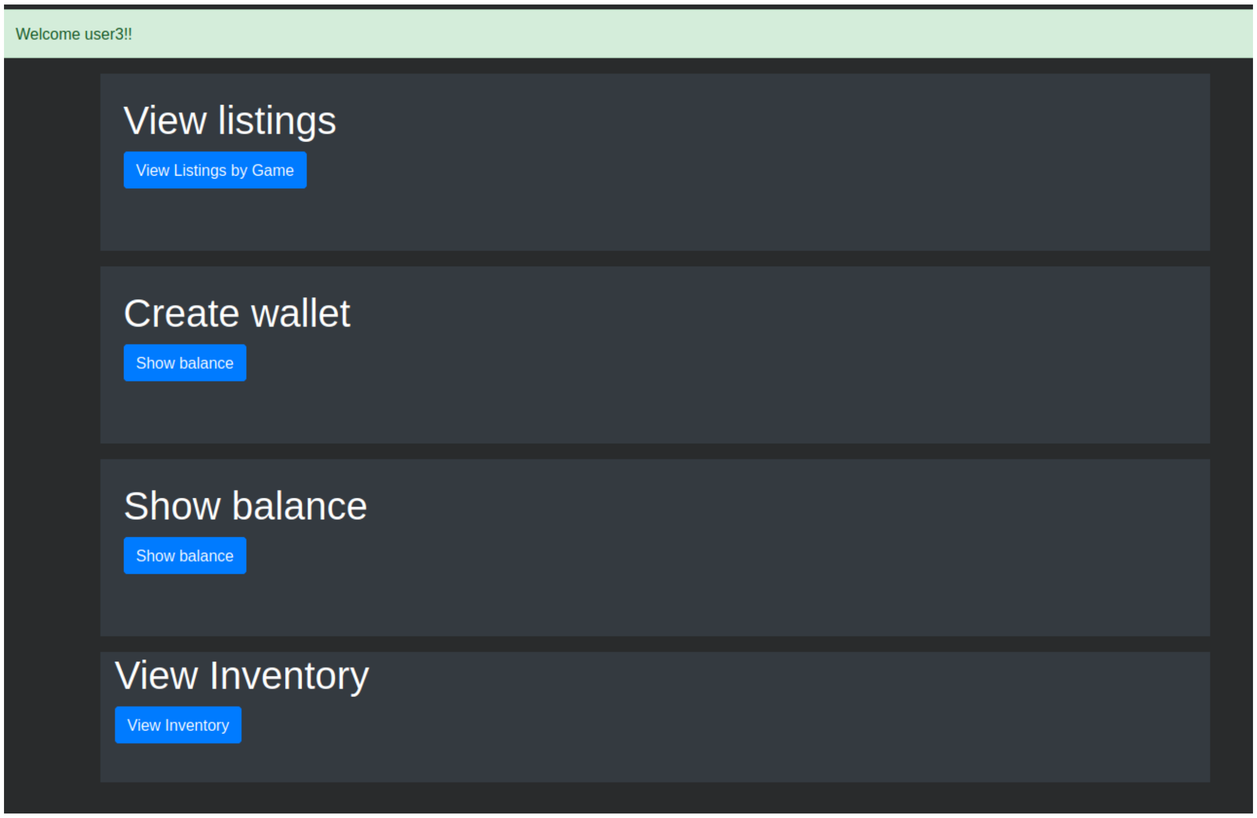
Password

- 

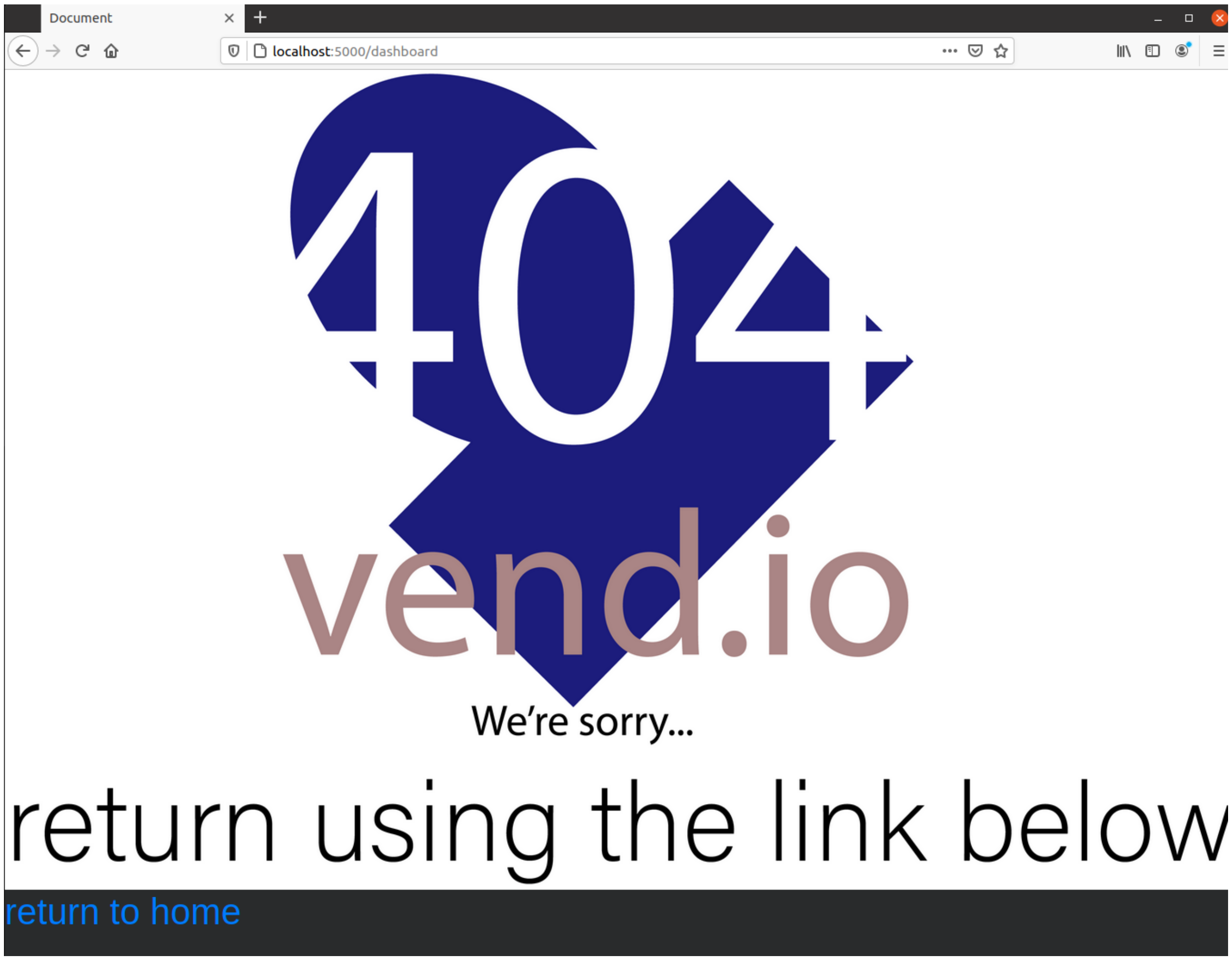
Info!

Login Failed. Please Check Username/Password and try again!

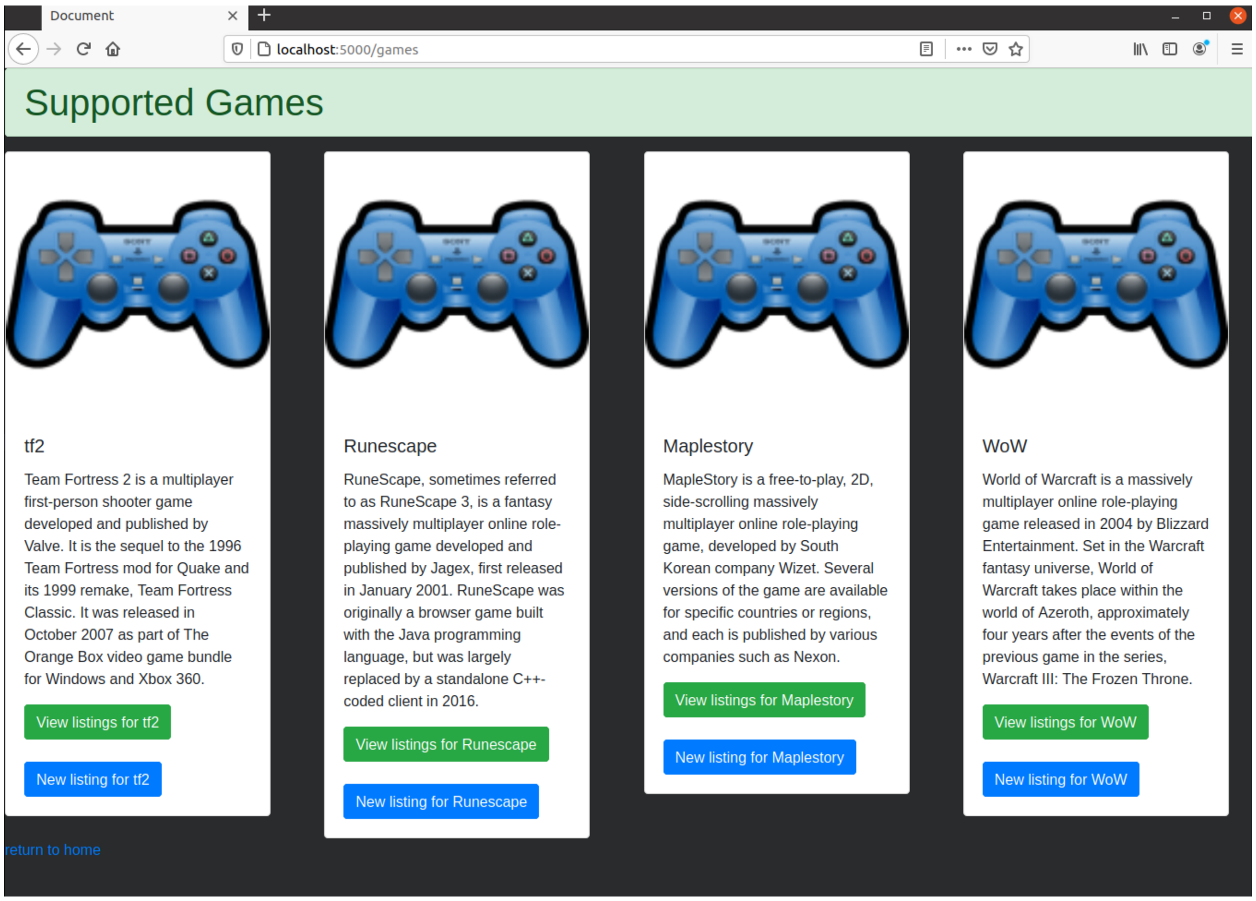
- Dashboard ('/dashboard') Post/Get calls. If website requested by get then return nologin.html, but if not return dashboard.html


Welcome user3!!
View listings
View Listings by Game
Create wallet
Show balance
Show balance
Show balance
View Inventory
View Inventory

-



-
- Games ('/games'). Connect to the sqlite database and store it in games and the sql table to games.html to be processed and displayed. Games.html has a for loop which runs via the python interpreter



Supported Games

tf2

Team Fortress 2 is a multiplayer first-person shooter game developed and published by Valve. It is the sequel to the 1996 Team Fortress mod for Quake and its 1999 remake, Team Fortress Classic. It was released in October 2007 as part of The Orange Box video game bundle for Windows and Xbox 360.

View listings for tf2

New listing for tf2

Runescape

RuneScape, sometimes referred to as RuneScape 3, is a fantasy massively multiplayer online role-playing game developed and published by Jagex, first released in January 2001. RuneScape was originally a browser game built with the Java programming language, but was largely replaced by a standalone C++-coded client in 2016.

View listings for Runescape

New listing for Runescape

Maplestory

MapleStory is a free-to-play, 2D, side-scrolling massively multiplayer online role-playing game, developed by South Korean company Wizet. Several versions of the game are available for specific countries or regions, and each is published by various companies such as Nexon.

View listings for Maplestory

New listing for Maplestory

WoW

World of Warcraft is a massively multiplayer online role-playing game released in 2004 by Blizzard Entertainment. Set in the Warcraft fantasy universe, World of Warcraft takes place within the world of Azeroth, approximately four years after the events of the previous game in the series, Warcraft III: The Frozen Throne.

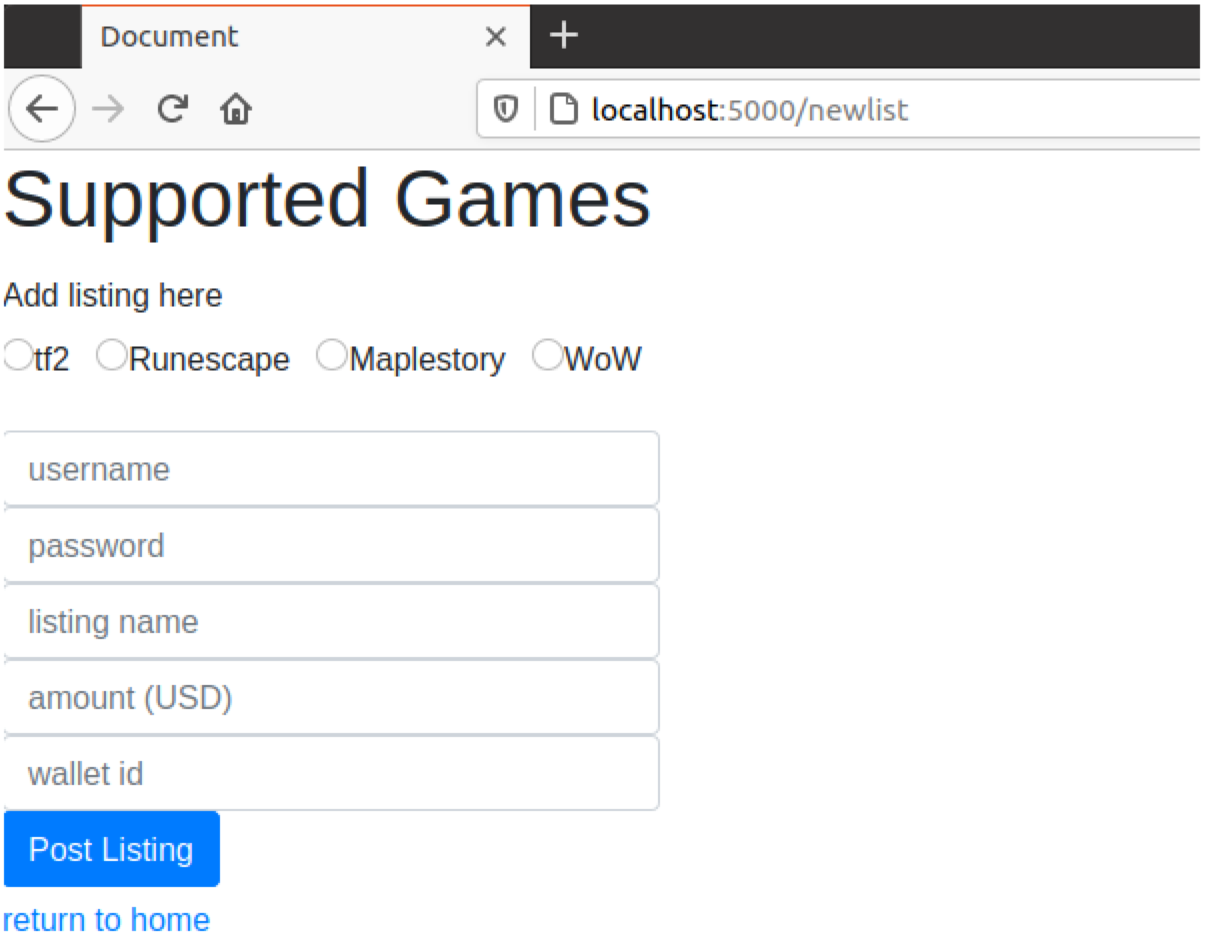
View listings for WoW

New listing for WoW

return to home

-

- Newlist ('/newlist') page to create a new listing to post an item. Uses the games table to generate radio buttons so the user can select which game he wants.


Document
localhost:5000/newlist

Supported Games

Add listing here

tf2 Runescape Maplestory WoW

username

password

listing name

amount (USD)

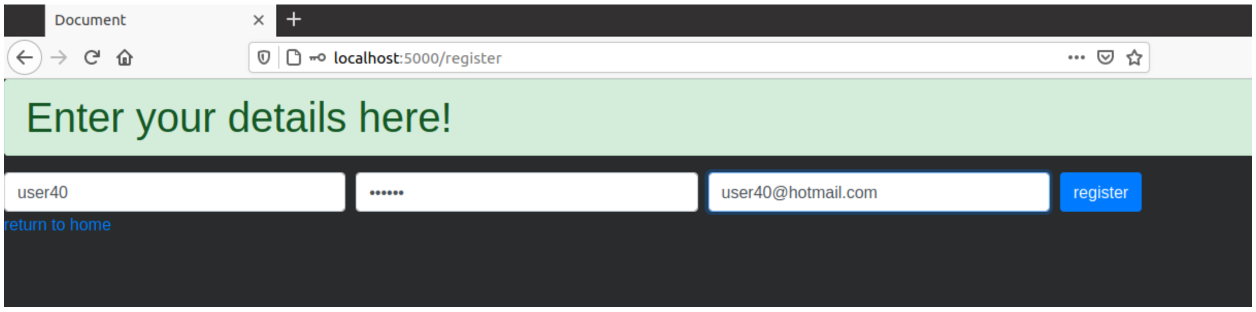
wallet id

Post Listing

return to home

-
    ▪ Nologin. Display 404 screen

- Postregister – form handler to add user to sql table. Takes values from username, password, and email and uses them in an insert into statement on the server.
- Create wallet ('/createwallet'). Get the uid from the provided login and pass the uid to the wallet table when creating the listing so that it is linked to that user. Any error will be caught and an error message will pop up on the result.html page. This route is a POST function


Document
localhost:5000/register

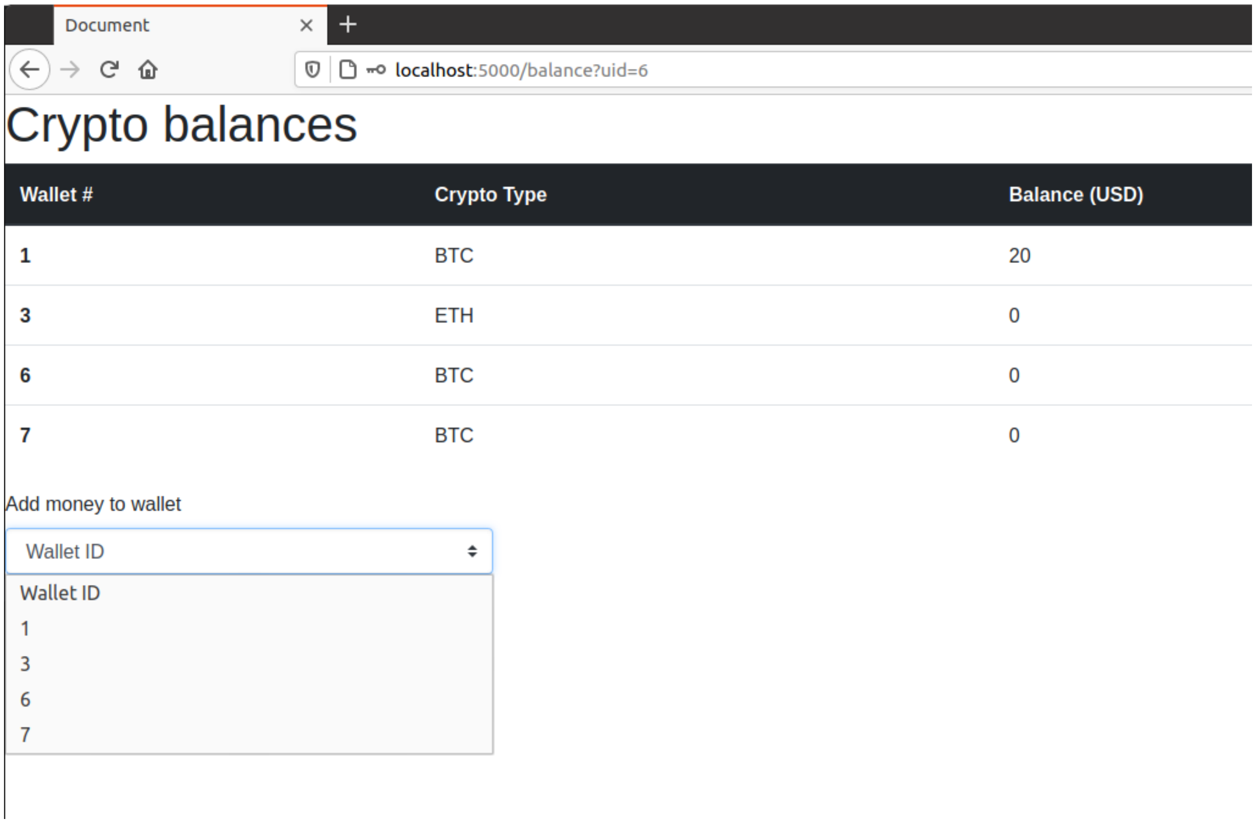
Enter your details here!

user40 •••••• user40@hotmail.com register

return to home

-

- Purchase ('/purchase') if a user selects an item from the drop down they can add it to the items table which is linked via uid and view on the inventory page.
    - Balance ('/balance') view the wallets that a user creates by uid. Uid is passed from the dashboard screen via GET.



Document

localhost:5000/balance?uid=6

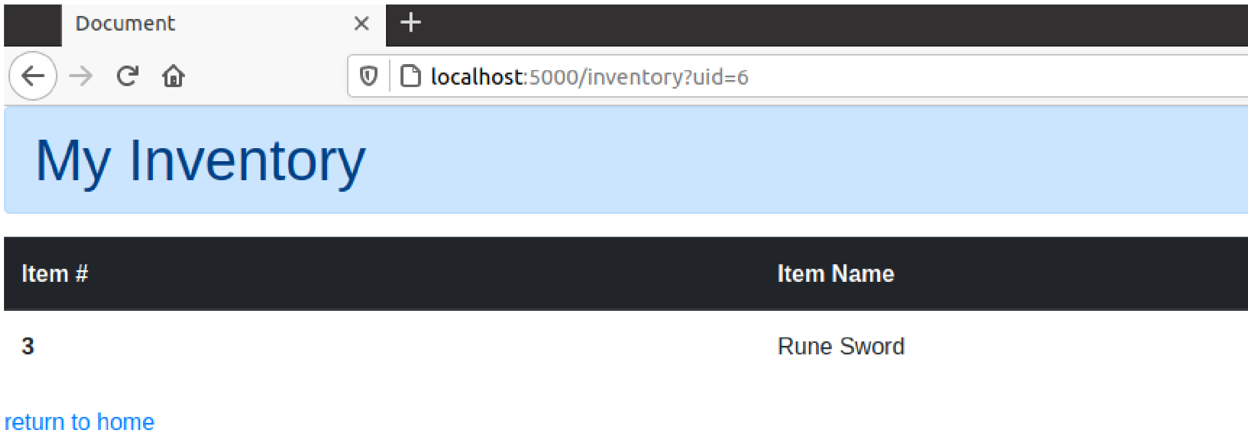
Crypto balances

Wallet #	Crypto Type	Balance (USD)
1	BTC	20
3	ETH	0
6	BTC	0
7	BTC	0

Add money to wallet

-

- Inventory ('/inventory') select all the data from the inventory table and pass it to inventory.html.
    - Listbygame ('/listbygame') view user submitted listings and sort by game. GID is passed via GET.


Document

localhost:5000/inventory?uid=6

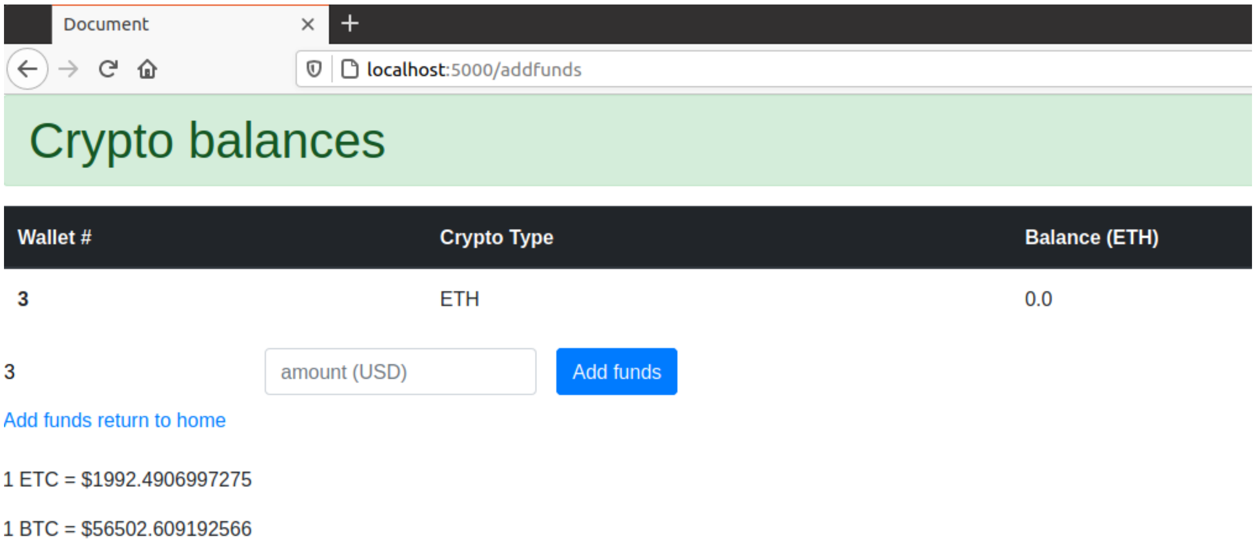
My Inventory

Item #	Item Name
3	Rune Sword

return to home

○

- Addfunds (‘/addfunds’). If this page is reached via POST then grab the live price for bitcoin or ethereum and convert user balance to the crypto type in their wallet. There is an if statement to display the balance in bitcoin or ethereum depending on which crypto they chose for that wallet. All data is passed to the html file. If there is an error, display an error message.


Document

localhost:5000/addfunds

Crypto balances

Wallet #	Crypto Type	Balance (ETH)
3	ETH	0.0

3 amount (USD) Add funds

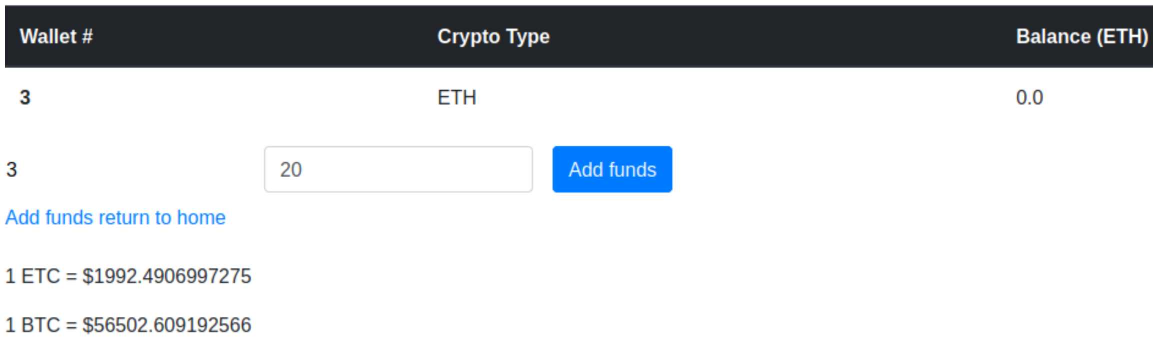
Add funds return to home

1 ETC = $1992.4906997275

1 BTC = $56502.609192566

-

- Fundspost ('/fundspost') update the wallet with the dollar amount the user provides using the sql update function. This definition takes the current value and adds the user submitted value to the wallet.



Wallet #	Crypto Type	Balance (ETH)
3	ETH	0.0

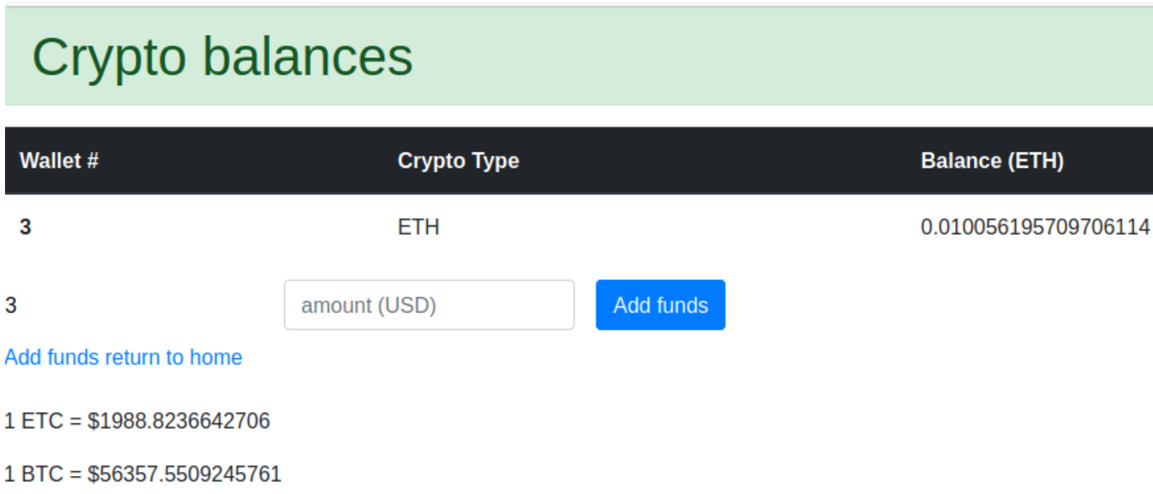
3 20 Add funds

Add funds return to home

1 ETC = $1992.4906997275

1 BTC = $56502.609192566

-



Crypto balances

Wallet #	Crypto Type	Balance (ETH)
3	ETH	0.010056195709706114

3 amount (USD) Add funds

Add funds return to home

1 ETC = $1988.8236642706

1 BTC = $56357.5509245761

-

- Addlist ('/addlist') add a listing with the user submitted price. Subtract the balance from the wallet when submitting the listing then use a sql insert into function.

- Support the illustration with a screenshot from the webpage for your project
- Show the prevented error and how you handle the project.

Outline the steps/plan for your project:

1. Illustrate database connections
2. Create database connections

3. Create sqlite database then create project.

Summarize what you learned:

I learned how to use flask to create html websites. I also learned how to use a sqlite database and learned that it was much easier than making a regular database.

Reference

https://www.youtube.com/watch?v=mqhxxeeTbu0