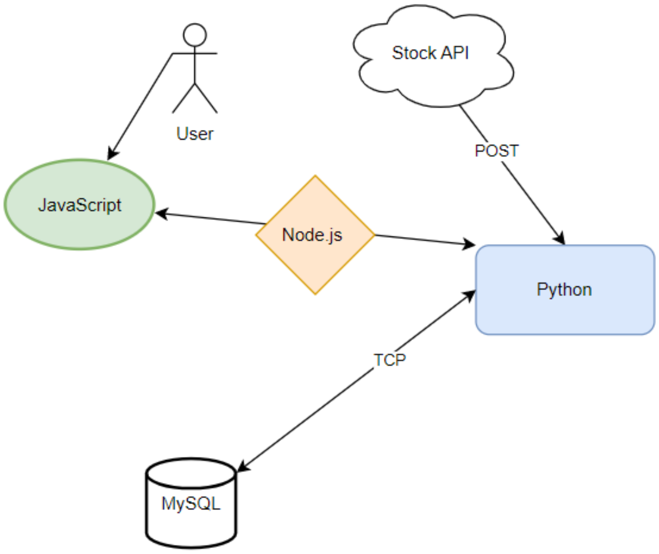
Prospera

West Chester University

Stephen Caliendo, Brian Giovinazzo, Brandon Kohler, Julia Kush & Jacob Peterson

Vision:



The idea behind this application is to make finance more personal, by providing individuals with a single platform to manage stocks and complete transactions. In the design diagram above, you can see that our web UI will be developed using JavaScript and will communicate with our Python script through Node.js. Our database will also communicate with our Python script. The web interface will depend on the Python script to gather information from the database. The API we are using is from Financial Modeling Prep, which allows us to access information such as data metrics and price data from up to 5 years ago.

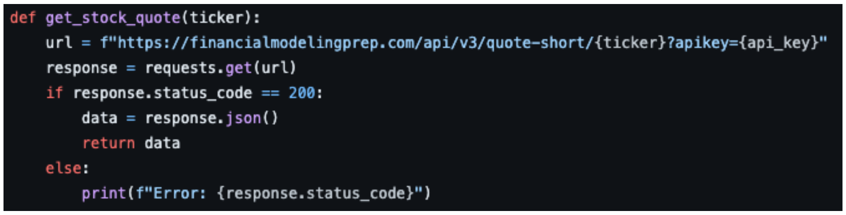
Purpose:

Our project's goal is to leverage our web UI, coded in react.js, to display information about users' accounts, including the stocks they currently own and their total market worth. Users will have the option to look up specific stocks and choose whether to buy or sell assets. One of our key objectives was to enable users to sell a portion of their shares and use the proceeds as a debit card during a transaction. Our Python workers will manage the modifications to the database and ensure that they are reflected in the web UI, ensuring smooth operation. Our frontend development will follow, and we aim to keep it relatively simple and user-friendly, so that users can navigate the website without any trouble. The web API will provide the most recent prices and quarterly reports on specific metrics, which will be valuable for users in making informed decisions about buying or selling stocks. User information, including username, password, stock holdings, and purchase prices, will be saved in our database.

Docker Build:

Worker:

The worker is the heart of this application and with that the worker is going to do all of the work behind this application. The worker is split into multiple classes with each of them ensuring things are working properly. To start off we want to first get the data of the most recent price and to do this we are going to look at the get_stock_quote method.

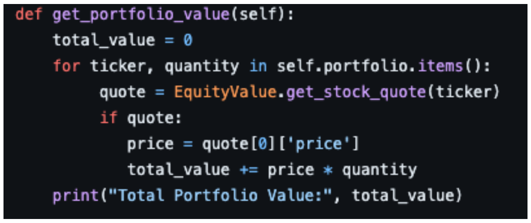


```
def get_stock_quote(ticker):
    url = f"https://financialmodelingprep.com/api/v3/quote-short/{ticker}?apikey={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        return data
    else:
        print(f"Error: {response.status_code}")
```

We are trying to get data from this URL. We imported the requests library to make this process happen. For the application we wanted the user to request information on a specific stock.

Portfolio Value:

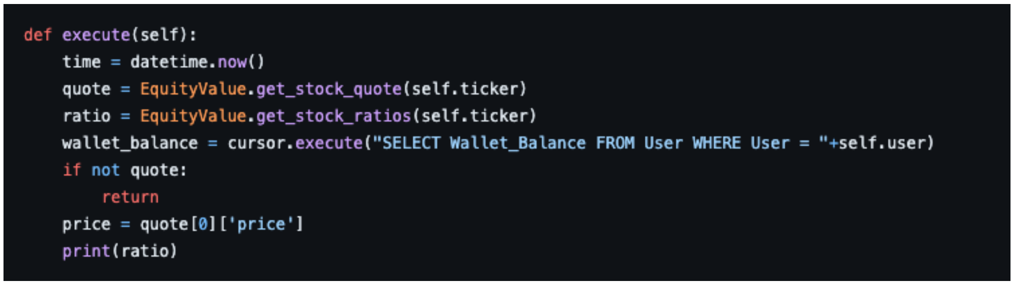
We want to get the portfolio value of all of the properties that are in our portfolio. We want to cycle through each of the names and the quantities of all of the items in the user's portfolio to generate total portfolio valuation.



```
def get_portfolio_value(self):
    total_value = 0
    for ticker, quantity in self.portfolio.items():
         quote = EquityValue.get_stock_quote(ticker)
         if quote:
            price = quote[0]['price']
            total_value += price * quantity
    print("Total Portfolio Value:", total_value)
```

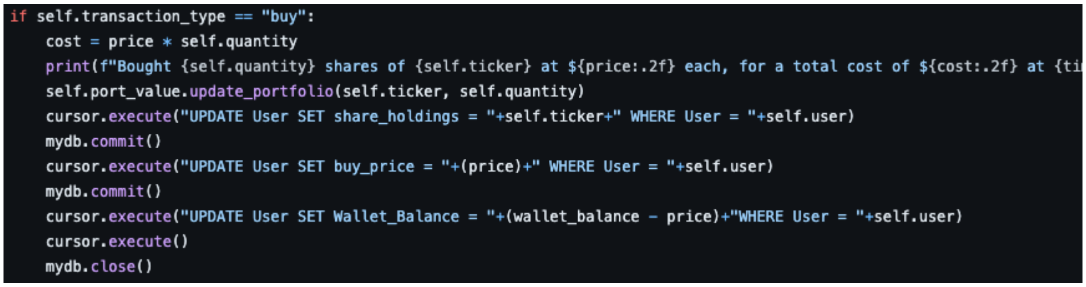
Stock Transaction:

We wanted to give the user the ability to buy and sell shares of stock in there portfolio. We start to tie everything together in our execute method.



```
def execute(self):
    time = datetime.now()
    quote = EquityValue.get_stock_quote(self.ticker)
    ratio = EquityValue.get_stock_ratios(self.ticker)
    wallet_balance = cursor.execute("SELECT Wallet_Balance FROM User WHERE User = "+self.user)
    if not quote:
        return
    price = quote[0]['price']
    print(ratio)
```

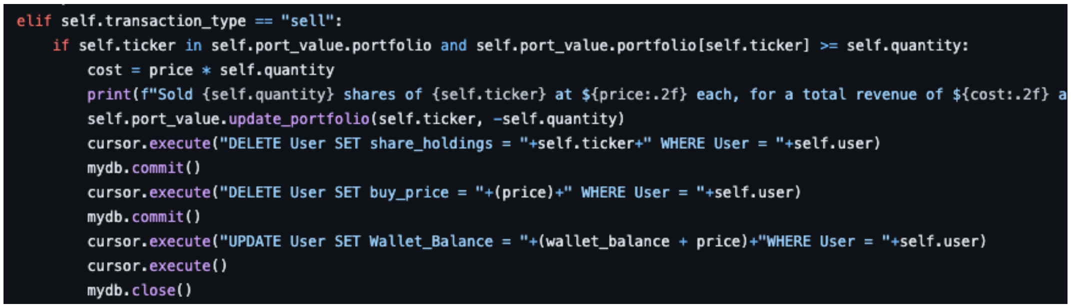
If the user decides to go with buying more shares of an individual stock it will ask the Equity Value Class what is the most recent price that AAPL is trading at. If the user decides to buy or sell, the database would then update the properties in the user's portfolio.



```
if self.transaction_type == "buy":
    cost = price * self.quantity
    print(f"Bought {self.quantity} shares of {self.ticker} at ${price:.2f} each, for a total cost of ${cost:.2f} at {ti
    self.port_value.update_portfolio(self.ticker, self.quantity)
    cursor.execute("UPDATE User SET share_holdings = "+self.ticker+" WHERE User = "+self.user)
    mydb.commit()
    cursor.execute("UPDATE User SET buy_price = "+(price)+" WHERE User = "+self.user)
    mydb.commit()
    cursor.execute("UPDATE User SET Wallet_Balance = "+(wallet_balance - price)+"WHERE User = "+self.user)
    cursor.execute()
    mydb.close()
```

Properties in the database that would be changed after Self.Transaction

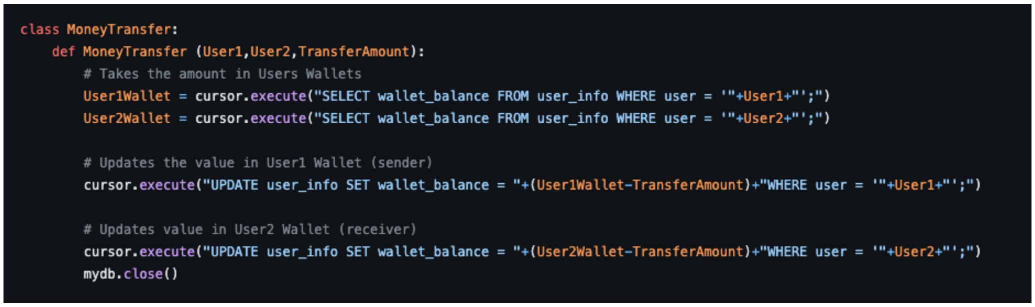
- Wallet Balance
- Portfolio
- Portfolio Valuation



```python
elif self.transaction_type == "sell":
    if self.ticker in self.port_value.portfolio and self.port_value.portfolio[self.ticker] >= self.quantity:
        cost = price * self.quantity
        print(f"Sold {self.quantity} shares of {self.ticker} at ${price:.2f} each, for a total revenue of ${cost:.2f} a
        self.port_value.update_portfolio(self.ticker, -self.quantity)
        cursor.execute("DELETE User SET share_holdings = "+self.ticker+" WHERE User = "+self.user)
        mydb.commit()
        cursor.execute("DELETE User SET buy_price = "+(price)+" WHERE User = "+self.user)
        mydb.commit()
        cursor.execute("UPDATE User SET Wallet_Balance = "+(wallet_balance + price)+"WHERE User = "+self.user)
        cursor.execute()
        mydb.close()
```

Money Transfer:

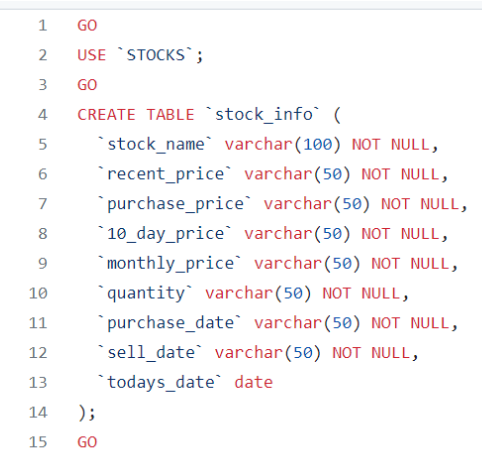
As a team we thought it would be beneficial that your cash in your portfolio would be able to be used with your peers for your everyday expenses.



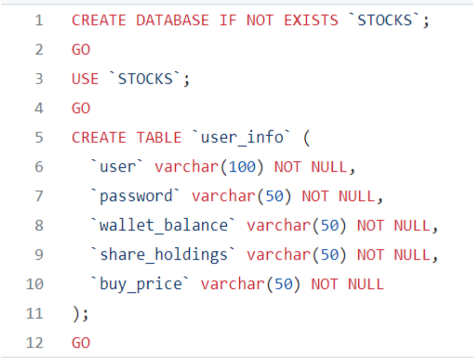
```python
class MoneyTransfer:
    def MoneyTransfer (User1,User2,TransferAmount):
        # Takes the amount in Users Wallets
        User1Wallet = cursor.execute("SELECT wallet_balance FROM user_info WHERE user = '"+User1+"';")
        User2Wallet = cursor.execute("SELECT wallet_balance FROM user_info WHERE user = '"+User2+"';")

        # Updates the value in User1 Wallet (sender)
        cursor.execute("UPDATE user_info SET wallet_balance = "+(User1Wallet-TransferAmount)+"WHERE user = '"+User1+"';")

        # Updates value in User2 Wallet (receiver)
        cursor.execute("UPDATE user_info SET wallet_balance = "+(User2Wallet-TransferAmount)+"WHERE user = '"+User2+"';")
        mydb.close()
```

Database:

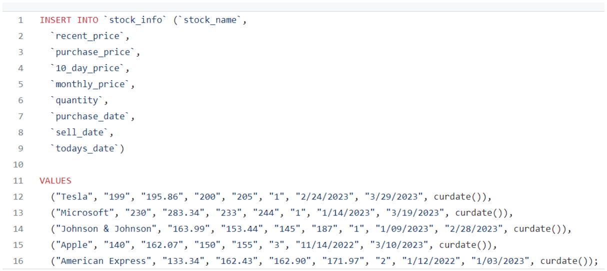
The start off we first needed to create our database. From there we created two separate tables stock_info and user_info. We then created two more scripts to enter values into these two tables.



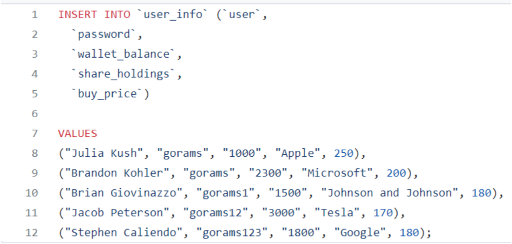
```sql
GO
USE `STOCKS`;
GO
CREATE TABLE `stock_info` (
  `stock_name` varchar(100) NOT NULL,
  `recent_price` varchar(50) NOT NULL,
  `purchase_price` varchar(50) NOT NULL,
  `10_day_price` varchar(50) NOT NULL,
  `monthly_price` varchar(50) NOT NULL,
  `quantity` varchar(50) NOT NULL,
  `purchase_date` varchar(50) NOT NULL,
  `sell_date` varchar(50) NOT NULL,
  `todays_date` date
);
GO
```



```sql
CREATE DATABASE IF NOT EXISTS `STOCKS`;
GO
USE `STOCKS`;
GO
CREATE TABLE `user_info` (
  `user` varchar(100) NOT NULL,
  `password` varchar(50) NOT NULL,
  `wallet_balance` varchar(50) NOT NULL,
  `share_holdings` varchar(50) NOT NULL,
  `buy_price` varchar(50) NOT NULL
);
GO
```

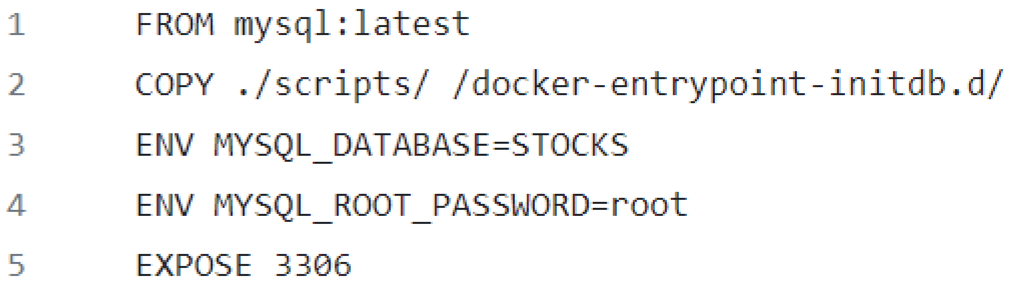


```
INSERT INTO `stock_info` (`stock_name`,
  `recent_price`,
  `purchase_price`,
  `10_day_price`,
  `monthly_price`,
  `quantity`,
  `purchase_date`,
  `sell_date`,
  `todays_date`)

VALUES
  ("Tesla", "199", "195.86", "200", "205", "1", "2/24/2023", "3/29/2023", curdate()),
  ("Microsoft", "230", "283.34", "233", "244", "1", "1/14/2023", "3/19/2023", curdate()),
  ("Johnson & Johnson", "163.99", "153.44", "145", "187", "1", "1/09/2023", "2/28/2023", curdate()),
  ("Apple", "140", "162.07", "150", "155", "3", "11/14/2022", "3/10/2023", curdate()),
  ("American Express", "133.34", "162.43", "162.90", "171.97", "2", "1/12/2022", "1/03/2023", curdate());
```



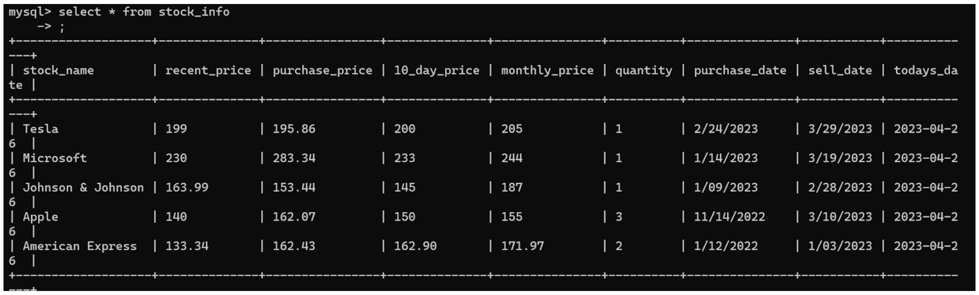
```
INSERT INTO `user_info` (`user`,
  `password`,
  `wallet_balance`,
  `share_holdings`,
  `buy_price`)

VALUES
("Julia Kush", "gorams", "1000", "Apple", 250),
("Brandon Kohler", "gorams", "2300", "Microsoft", 200),
("Brian Giovinazzo", "gorams1", "1500", "Johnson and Johnson", 180),
("Jacob Peterson", "gorams12", "3000", "Tesla", 170),
("Stephen Caliendo", "gorams123", "1800", "Google", 180);
```

The next step we took was to create a docker file to create our MySQL image from the MySQL base image. We added a copy command to copy the scripts that we used to create our tables and inserted our data into the tables. We then added environmental variables to be able to get the database we created which is named STOCKS and our MySQL root password. Lastly, we exposed the MySQL port which is 3306.

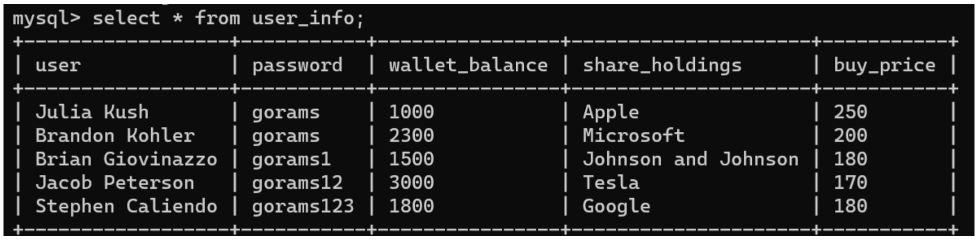


```
FROM mysql:latest
COPY ./scripts/ /docker-entrypoint-initdb.d/
ENV MYSQL_DATABASE=STOCKS
ENV MYSQL_ROOT_PASSWORD=root
EXPOSE 3306
```

Lastly, we used this docker file to build our own image for MySQL. During this step, we exposed the MySQL port which is 3306 and we set up a password to be able to connect to our database. Once we were able to connect to our database we were able to see our tables. Here you can see all of the data we inserted from our scripts in our tables.



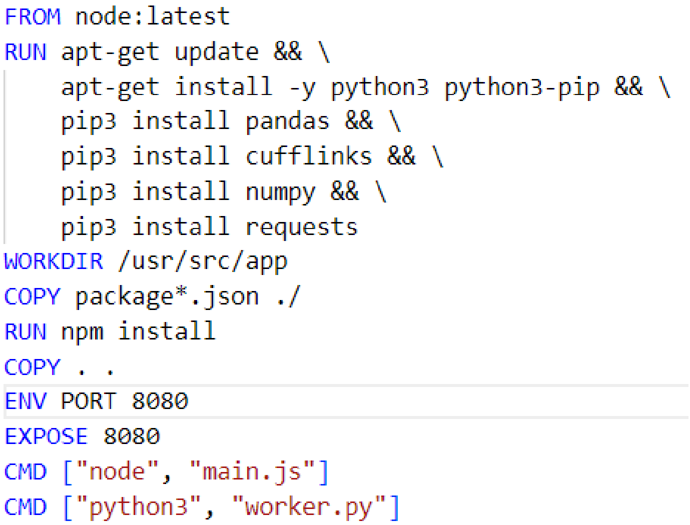
```
mysql> select * from stock_info
    -> ;
+-------------------+--------------+----------------+--------------+---------------+----------+---------------+-----------+---------
---+
| stock_name        | recent_price | purchase_price | 10_day_price | monthly_price | quantity | purchase_date | sell_date | todays_da
te |
+-------------------+--------------+----------------+--------------+---------------+----------+---------------+-----------+---------
---+
| Tesla             | 199          | 195.86         | 200          | 205           | 1        | 2/24/2023     | 3/29/2023 | 2023-04-2
6  |
| Microsoft         | 230          | 283.34         | 233          | 244           | 1        | 1/14/2023     | 3/19/2023 | 2023-04-2
6  |
| Johnson & Johnson | 163.99       | 153.44         | 145          | 187           | 1        | 1/09/2023     | 2/28/2023 | 2023-04-2
6  |
| Apple             | 140          | 162.07         | 150          | 155           | 3        | 11/14/2022    | 3/10/2023 | 2023-04-2
6  |
| American Express  | 133.34       | 162.43         | 162.90       | 171.97        | 2        | 1/12/2022     | 1/03/2023 | 2023-04-2
6  |
+-------------------+--------------+----------------+--------------+---------------+----------+---------------+-----------+---------
---+
```



```
mysql> select * from user_info;
+-------------------+-----------+----------------+---------------------+-----------+
| user              | password  | wallet_balance | share_holdings      | buy_price |
+-------------------+-----------+----------------+---------------------+-----------+
| Julia Kush        | gorams    | 1000           | Apple               | 250       |
| Brandon Kohler    | gorams    | 2300           | Microsoft           | 200       |
| Brian Giovinazzo  | gorams1   | 1500           | Johnson and Johnson | 180       |
| Jacob Peterson    | gorams12  | 3000           | Tesla               | 170       |
| Stephen Caliendo  | gorams123 | 1800           | Google              | 180       |
+-------------------+-----------+----------------+---------------------+-----------+
```

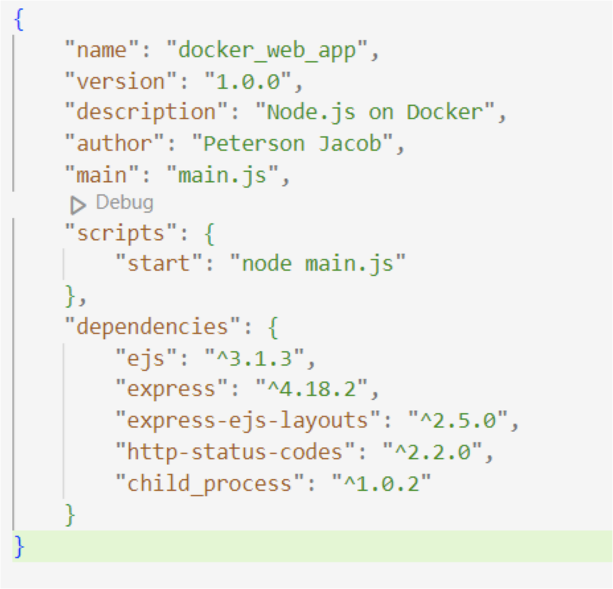
WebUI & Python:

We later decided that it would be best to integrate our Python script into the Node.js server along with the JavaScript to establish a more seamless connection. Our Docker file will manage all the dependencies for the components of the Node.js server.



```
FROM node:latest
RUN apt-get update && \
    apt-get install -y python3 python3-pip && \
    pip3 install pandas && \
    pip3 install cufflinks && \
    pip3 install numpy && \
    pip3 install requests
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
ENV PORT 8080
EXPOSE 8080
CMD ["node", "main.js"]
CMD ["python3", "worker.py"]
```

To initiate the build, we utilize the base image for Node and then proceed to install the Python dependencies. Our script makes use of the pandas, cufflinks, numpy, and requests libraries. Next, we create the working directory and copy the package.json into it. Once the dependencies are installed, we add the remaining source code, expose port 8080, and execute main.js and worker.py.



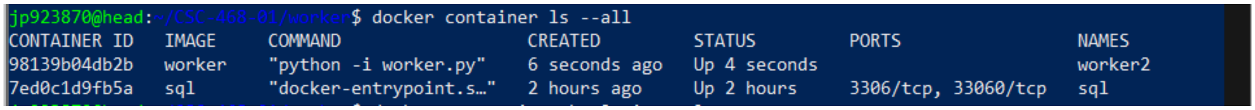
```
{
    "name": "docker_web_app",
    "version": "1.0.0",
    "description": "Node.js on Docker",
    "author": "Peterson Jacob",
    "main": "main.js",
    "scripts": {
        "start": "node main.js"
    },
    "dependencies": {
        "ejs": "^3.1.3",
        "express": "^4.18.2",
        "express-ejs-layouts": "^2.5.0",
        "http-status-codes": "^2.2.0",
        "child_process": "^1.0.2"
    }
}
```

The package.json file includes all the dependencies for the JavaScript on the Node.js server. We are utilizing the latest versions of ejs, express, express-ejs-layouts, http-status-codes, and child_process.

Dockerizing the Application:

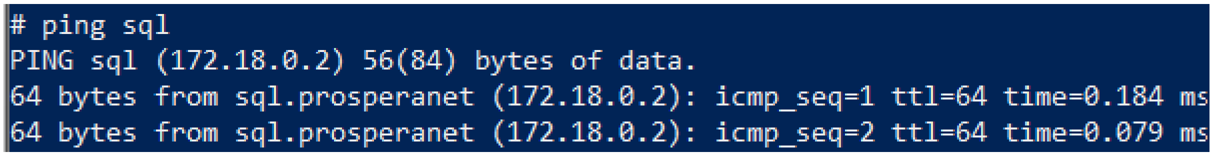
We have all the Docker images created for the components, and now we have to make sure that we are able to connect them all on the same docker network.

We created the image for both the worker and MySQL and then created a container with both of those images:



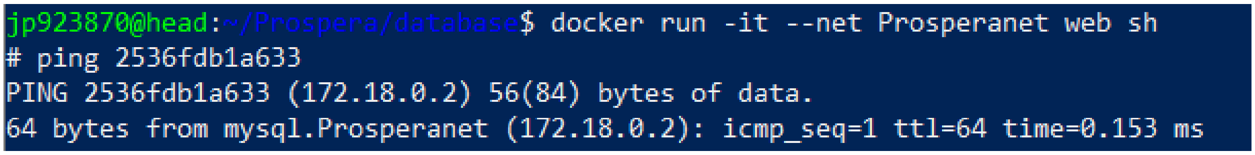
```
jp923870@head:~/CSC-468-01/worker$ docker container ls --all
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS        PORTS                 NAMES
98139b04db2b   worker    "python -i worker.py"    6 seconds ago   Up 4 seconds                        worker2
7ed0c1d9fb5a   sql       "docker-entrypoint.s…"   2 hours ago     Up 2 hours    3306/tcp, 33060/tcp   sql
```

Next, we interactively went into the worker container’s shell and pinged the MySQL container:



```
# ping sql
PING sql (172.18.0.2) 56(84) bytes of data.
64 bytes from sql.prosperanet (172.18.0.2): icmp_seq=1 ttl=64 time=0.184 ms
64 bytes from sql.prosperanet (172.18.0.2): icmp_seq=2 ttl=64 time=0.079 ms
```

We then went to the node js container and pinged the sql database on the network:



```
jp923870@head:~/Prospera/database$ docker run -it --net Prosperanet web sh
# ping 2536fdb1a633
PING 2536fdb1a633 (172.18.0.2) 56(84) bytes of data.
64 bytes from mysql.Prosperanet (172.18.0.2): icmp_seq=1 ttl=64 time=0.153 ms
```

All of our components were successfully put on the Docker network and we were able to ping each container in the network.

Assessment:

Our purpose with Prospera is to provide users with the ability to manage their investment portfolios, track the performance of their investments, and access up-to-date information on various stocks. Additionally, Prospera offers personalized transaction capabilities, allowing users to sell stocks to fund their everyday transactions.

Although each separate component has been completed and is operational, we encountered difficulties in achieving seamless integration among them. While each component had its own containers running successfully on Docker, we faced issues with the connection between the database and Python on Kubernetes. We suspect that the problem lies with the database service on Kubernetes, as the MySQL.connector in the Python script fails to locate the database on port 3306.

Since our web interface heavily relies on the Python script to retrieve information from the database, we are unable to work around this issue and consequently cannot display user information on the web UI. While each component individually met the requirements, we encountered challenges in connecting them together.

Julia Kush

Juliaekush@gmail.com

610.937.1614

EDUCATION

West Chester University
West Chester, PA

Major: BS - Computer Science *CUM GPA 3.75*
Minor: Applied Statistics
Computer Security Certification

Cardinal O'Hara High School
Springfield, PA
Graduated May 2019

SUMMARY

A talented senior majoring in computer science with a focus in cyber security and statistics. I have significant experience with multiple programming languages and advanced mathematical and analytical skills.

WORK EXPERIENCE

Universal Health Services
King of Prussia, PA
Software Engineer - Intern *June 2022 - Present*

- Generate team project report and open incident report
- Collaborate closely with senior level developers to assist with integrated testing and documentation
- Assist with analysis on larger issues which are used by senior level developers
- Utilize SQL Server Studio to house data for projects
- Create database objects such as tables, stored procedures, and views in SQL Server Studio
- Implement SSIS packages to migrate data from data warehouse to our local data models
- Design end user reports using Power BI and SSRS Reporting tools

Telco Holdings Inc.
West Chester, PA
Computer Implementation Specialist - Intern *May 2021 – January 2022*

- Imaged windows-based computers and Laptops
- Installed applications, VPN, and management software
- Configured user domain access
- Performed testing and configuration validation

- Boxed configured devices and worked in partnership with field deployment services
- Individually contributed and collaborated with managed services desk stakeholders

Joseph Anthony Retreat Spa

Springfield, PA *Spa Attendant* *June 2018 – July 2019*

- Communicated daily with all suppliers, ensuring proper quantities of products were fresh and delivered timely
- Supported masseuses and management
- Maintained and sanitized SPA
- Replenished stock
- Tear down and set up treatment rooms after each service
- Customer-focused services

SKILLS & CERTIFICATIONS

- **Soft Skills**
  - Goal oriented, strong networking capabilities, communication and listening skills, team oriented, flexible, time-management, creative thinking, organization
- **Technical IT Skills**

  - Languages: Java, C++, MySql, R, Haskell

  - Proficient with Microsoft Suite, R-Studio, Visual Studio, GitHub, J-Grasp, Eclispe, Power BI, SSRS, SSIS, SQL Server Studio
- **Honor Roll Spring 2020, Fall 2020, Spring 2021, Fall 2021.**

Brian Giovinazzo

(209) 597-7875 | brian.giovinazzo@gmail.com

http://www.linkedin.com/in/brian-giovinazzo ·
https://github.com/Brian-Giovinazzo/School-Projects

Objective

To obtain a software engineering internship at a company that will utilize my technical skills and knowledge to effectuate impactful improvements in our world while fostering self-development.

Skills & Abilities

- Software: Java, JavaScript, Python, C, NodeJS, Express, MongoDB, CSS, HTML, XML
- Software Tools: VS Code, Eclipse, GitHub, PyCharm, Docker, Cisco Packet Tracer, SolidWorks, MATLAB, Anaconda, MongoDBCompass
- Mechanical: knockout, drill press, grinder, impact driver, blowtorch, and various hand tools
- Electrical: multimeter, crimping tool, soldering, insulation tester, DLRO, and phase rotation meter
- Personal: fast-learner, works well in a team or individual setting, detail-oriented, initiative taker, life-long learner, and prompt
- Certifications: Substation Maintenance, Circuit Breaker/Medium-Voltage, Basic Electrical Troubleshooting, Battery Maintenance and Testing, Infrared Thermography I, Electrical Safety for Industrial Facilities

Education

B.S. Computer Science | West Chester University | Estimated Graduation Date: Dec 2023 | GPA: 3.87

- Awards: Dean's List (2022), placed 3rd in the WCU CS Programming Competition in Java (Spring 2022, Spring 2023), and participated in the ICPC 2023
- Memberships: WCU Computer Science Club, Competitive Programming Club

N/A | United States Naval Academy | Sep 2007 – Sep 2010

- Mechanical Engineering and Economics majors. Commandant's List Spring of 2009. Sea Trials Leader for 2008 and 2009.

Experience

Software Intern | QSI Technologies Inc. | May 2022 - Dec 2022

- Implement front-end code in CSS, XML, and HTML to expand company capability from desktop-only website to the mobile platform
- Optimize web pages using responsive design for mobile use while prioritizing customers' requests of desired site features
- Developed enhancements that will be implemented in the next software update to service all internal and external QSI customers
- Meet with software team to present and obtain feedback on the mobile design

Hydro Maintenance Electrician | California Department of Water Resources | Nov 2014 - Nov 2021

- Supervised an Electrician Apprentice in day-to-day operations
- Led in planning and executing facility lighting project from start to finish
- Performed maintenance and repair of all electrical equipment and apparatus of a large hydroelectric installation, from 24 V to 230,000 V

Assistant Operator | Pactiv | Aug 2013 – Nov 2014

- Ensured proper operation of all machines in the production facility

Machinist's Mate and Midshipman | United States Navy | Aug 2006 – Nov 2010

- Trained to be a Mechanical Operator for submarines
- Held Secret level clearance

Stephen D. Caliendo

912 Owen Road, West Chester, PA 19380 | (484)-844-9353 | SC928713@wcupa.edu

Objective

Apply computer science skills to software engineering opportunities in the information technology industry. Expand skillset through freelance programming opportunities.

Education

B.S Computer Science | West Chester University, PA | May 2023

- Proficient in Java, C , C++, Haskell
- Skilled in writing code from scratch, and utilizing public source code, to solve complex computer science problems.
- Adept at troubleshooting and debugging code.
- CSC 468 Cloud Computing
- CSC 402 Software Engineering
- CSC 241 Data Structures and Algorithms

Experience

The Sea Pines Resort | Hilton Head SC | 2020 - Present

The Social Lounge | West Chester PA | 2018-2020

Food Runner | Line Cook | Expo | Waiter

- Maintained high level of customer service.
- Excelled in team environment.
- Responded to and resolved problems as they arose.

Empwrd Mind | San Diego CA | 2022-2022

- Implemented opensource libraries in complex code products.
- Worked with other interns to complete deliverables on time.

Skills

§ Data analysis

§ JGrasp, Virtual Studio Code

§ Refactoring code, Object oriented design, Algorithm optimization

Brandon Kohler

267-975-1813 | BrandonKohler2000@gmail.com | linkedin.com/in/brandon-kohler-1788391b7/ | github.com/Kohler123

Education

West Chester University West Chester, PA
Bachelor of Science in Computer Science *Sept 2019 – Aug 2023*

- CSC496 iOS Development: Language basics, SwiftUI, SpriteKit, MVC Pattern
- CSC468 Cloud Development: Cloud Computing, Virtualization, Containerization, and Orchestration
- CSC402 Software Engineering: Object Orientated Design Principles, Refactoring, and Design Patterns
- CSC321 Database Management: Table Design and Normalization, UML Design, Web-based Database System
- CSC231 Computer Systems: Memory hierarchy, Optimizing program performance, Operating Systems and Network functions
- STA 200 Statistics II: Normal Distribution, Hypothesis Testing, and Linear Regression Analysis of Variance, Multiple Regression, Categorical Data and Chi-square Analysis and Nonparametric

Experience

Information Technology Sector Head Sep. 2020 – Present
West Chester Investment Group *West Chester, PA*

- Manages a group of analysts, with weekly meetings and teaching of appropriate topics over a span of a semester.
- Utilizing Financial Accounting concepts, Monte Carlo, VAR, Risk Analysis, Stochastic processes, Approximations, and Sentiment Analysis
- Using Bloomberg Terminal to create risk models, financial forecasting, and fundamental and technical valuation, to optimize portfolio allocation and performance

West Chester University Competitive Programming Contest March 2022-Current
West Chester University Competitive Programming Club *West Chester, PA*

- Students were presented with a number of problems with a limited amount of time.
- Encouraged students to strengthen coding skills

Office Assistant June 2021 – August 2021
Moreland Development LLC *Philadelphia, PA*

- Determined the length of time to maintain various documents such as tax returns, human resource records, leases, sale documents, and construction documents.
- Implemented this plan based on relevant documents by sorting through documents and reorganizing the files by project and year.

Projects

iOS Game | *Swift, SwiftUI* Dec 2022

- Created a Game using Swift that utilizes Swift's contact detection, SpriteKit's, and Swift's Game Physics
- Used API for Game images that were used in UI

Financial Scraping | *Python* March 2022

- Created a project for West Chester Investment Group that utilized APIs and Pandas data frame functions to get the most recent prices as well as ratios to perform fundamental analysis
- Used Pandas and Numpy for tasks

Website Portfolio Project | *JS, CSS, HTML* June 2022

- Created a website over the summer using CSS/HTML/JavaScript to demonstrate development skills.
- Learned basic JS and utilized Grid and Column Structure using HTML/CSS

Technical Skills

Languages: Java, Python, C, JavaScript, HTML/CSS, SQL, VBA,
Frameworks: React, Node.js
Developer Tools: Bloomberg Terminal, Excel, SPSS, MatLab Git, Docker, Kubernetes, Jenkins, Visual Studio, PyCharm, Eclipse
Libraries: Pandas, NumPy, Matplotlib
Certificates:[Linkedin: Business Statistics using Excel, Statistics and Probability], [Code Academy: Python 3, SQL, Data Structures, Sorting Algorithms, Linear Structures]

JACOB PETERSON

Address: 94 Hornbean Ave, Swedesboro, New Jersey 08085 **Phone:** 484.326.4896 **Email:**
jakepeterson2001@gmail.com

- PROGRAMMER –

EDUCATION

West Chester University of Pennsylvania, West Chester PA
B.S. Computer Science
Anticipated May 2023
GPA 3.6

WORK EXPERIENCE

Amazon, Logan Township, New Jersey

WAREHOUSE ASSOCIATE	**2020-Present**

- Ensuring all customer parcels are carefully packaged and shipped efficiently
- Organizing specific packages by businesses and destination
- Collaborative with team members and management
- Perfect attendance and punctuality for 2 years

Kohl's, Media, Pennsylvania

FLOOR ASSOCIATE / CUSTOMER SERVICE	2018-2020

- Helping customers find items
- Recommending certain products based on professional experience
- Using knowledge of the POS system to help customers purchase, return or exchange items in a quick mannerly fashion
- Educate customers on the return policies of the company and help them choose the outcome that most benefits their experience
- Cross trained in Customer service, Back stocking, Inventory, Homegoods, and Shoes.

ACHIEVEMENTS AND CERTIFICATIONS

- Computer Security Certificate
- Advisor to Computer Science department hiring process
- Deans list 2 semesters

SKILLS AND CODING LANGUAGES

- Proficient in Java
- Proficient in C
- Proficient in Python
- Proficient in Linux terminal environment
- Familiar with Microsoft Office Suite
- Familiar with HTML