CSC468-01 - Project Deliverable #1

GitHub Link:

https://github.com/TM826651/csc468-01_project/tree/main

<u>Group #2</u>

Christopher Calixte, Liam Daly, Tyler Martin, Alexis Nunez

Chapter #1 Project Overview: Product Price Comparison Tool

Our team will be designing a web scraping application to gain insight into prices of products hosted on Amazon (potentially others as well) to track the increase/decrease of prices over time. Comparisons will be made against products at different times of day, days of week, and beginning and end of the month. Stakeholder value would be the savings gained from knowing that they are purchasing a given product at its ideal (lowest) price. Additionally, similar products can be compared across different company web pages in order to determine which retailer offers the best possible price both historically and at the current time.

Advantages of using the app:

● **Cost savings:** Users are empowered with the information to determine whether they are getting a good deal on the goods they are purchasing by having ready access to all prices for a product over a given period of time.

● **Awareness of price fluctuations:** Users will be aware of times of day and days of the week on when the best time is to purchase a given product.

● **Price comparisons:** Users can compare a given product across multiple companies to see which one is offering the best price for a given product.
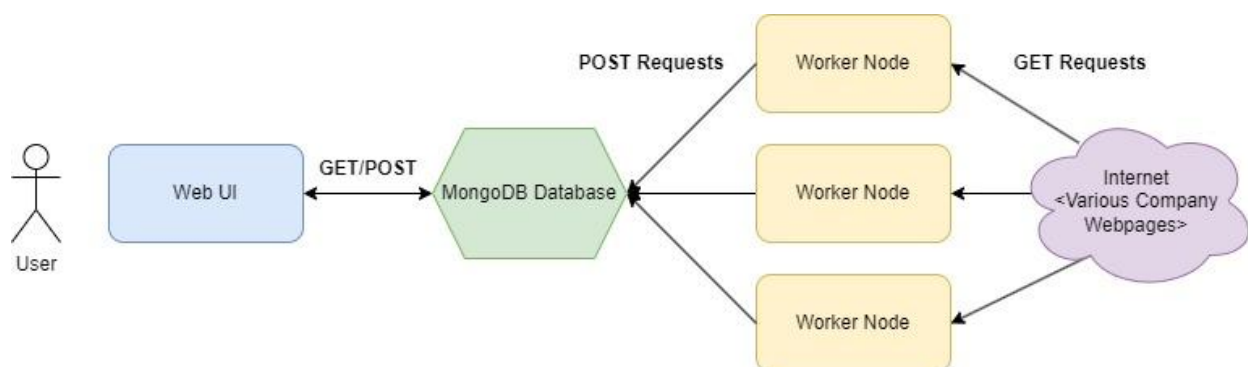
Chapter #2: Project Discussion/Implementation Plan

Our team will achieve this product through a CI/CD pipeline to handle the web scraping, data cleansing and saving to a database, and the presentation of the data through an interactive UI. Containers using a combination of Linux and Python will be utilized for web scraping and cleaning, while a lightweight Linux distribution node will handle the database management; Another node will handle the web server for the UI. By using CI/CD pipeline, we can automate the deployment of containers building the web scraping to scale and increase speed by each container performing its own data cleansing.

Web Scraping will be performed at irregular intervals between 30 seconds and 120 seconds (2 minutes) in order to avoid ban risks. Containers may be utilized with IP masks in the case of constant bans.

The Amazon API can be utilized (up to a million requests for a free account) to do direct web scraping with minimal cleaning; if that does not suffice, then alternative web scraping will need to be utilized for that specific storefront.

Step-by-step analysis of design:

1. Webscraper detects price change or pull price at irregular intervals. This will be handled through either comparison or using Python sleep at randomized intervals to avoid bot detection by the host storefront.
2. Change is pulled down to worker node. This will be performed using either a Python web scraping library or cURL requests.
3. Data is parsed for price and time. Parsing will be done with a Python script to pull the price from the data, depending on the storefront.
4. Worker node sends POST with cleaned information to MongoDB database server
5. MongoDB writes data to database.
6. User opens web UI (webserver written in React)
7. Webserver sends GET to database server to load pertinent data during query
8. Data is loaded into UI with information for user. This frontend will provide all needed information with a visually pleasing UX, as well as a suggestion on the ideal time to purchase the product in question.

Tech Stack:
- Frontend: Linux, React
- Backend: Linux, Python, Jenkins, Github
- Database Server: MongoDB (MongoDB Atlas)
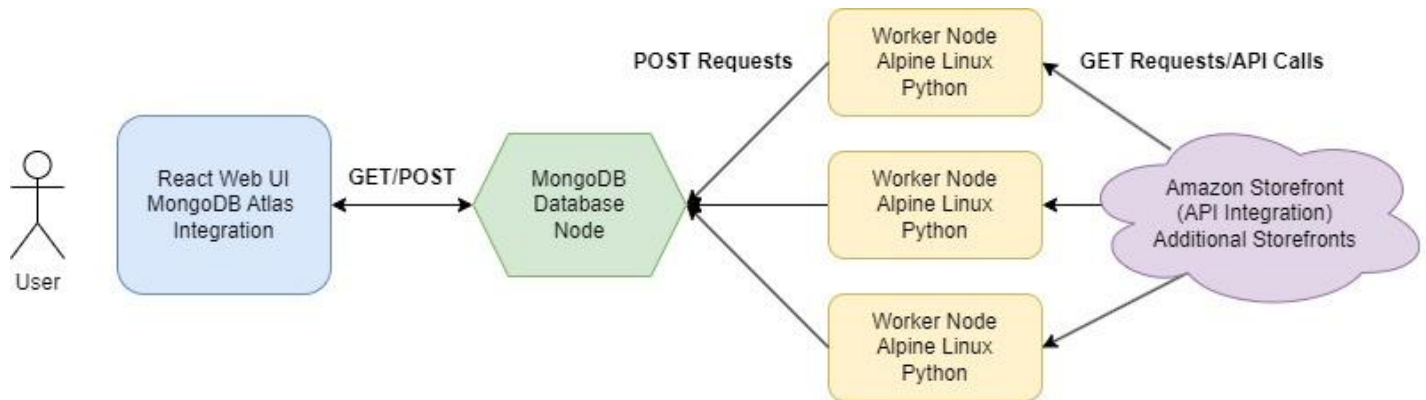
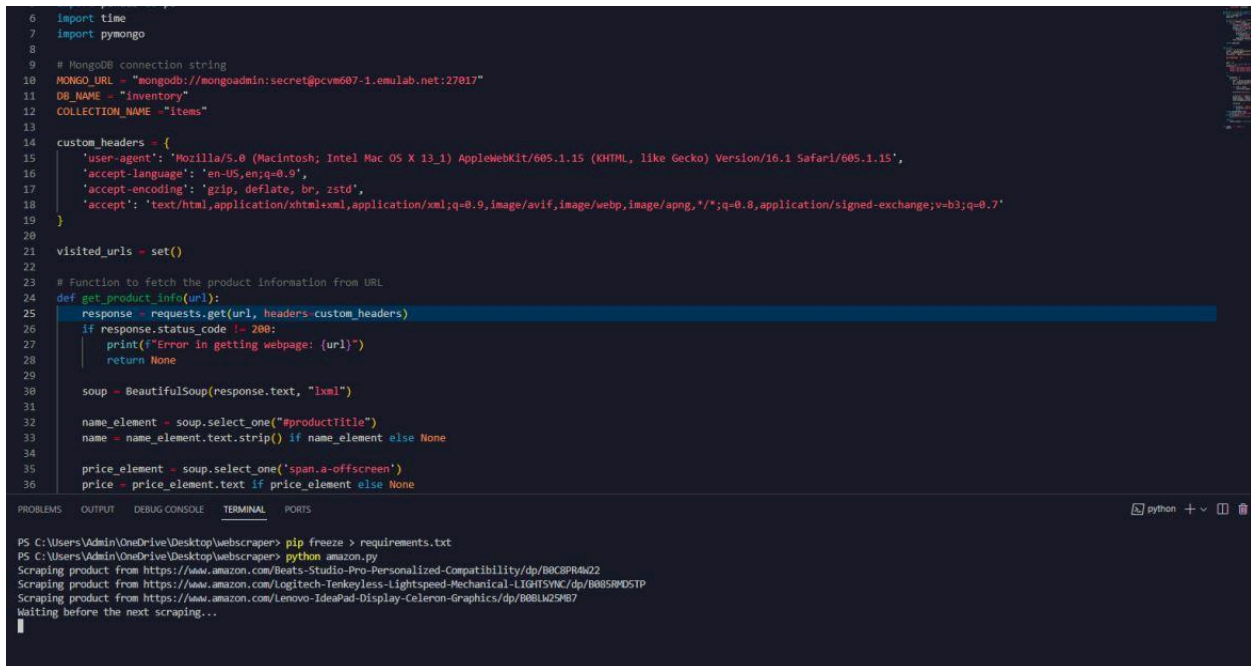**Chapter #3: Project Design, Testing Procedures, and Preliminary Results**

Amazon Storefront Product Parse Node:

The Product Price Analysis app relies on the use of Docker containers that are running Python in order to collect and process data from the Amazon storefront for a given product.The Docker image used for this container relies on Alpine Linux which is running the 3.9 version of Python. The use of Python allows for quick, simple changes to different products, as well as fast parsing speed.

The requirements.txt pulls in libraries such as pandas, BeautifulSoup, and PyMongo to facilitate the process of pulling data down to the parse node and writing to the MongoDB database node. Amazon.py utilizes BeautifulSoup to pull down the product name, product price, URL, and timestamp of price capture. These values are critical to the app to help differentiate the timing of the product price capture in order to compare the product prices over time.

After the script pulls down these values, combs through the data for the price, and then writes this information to a pandas Dataframe. The scraping occurs at randomized intervals between 2 and 6 hours to help prevent bans by the Amazon server for bot activity. IP masking has not yet been implemented as the team has not had issues so far with bans against the parse node. Ideally, we will have multiple products tracked on Amazon, with the ability to extend this track to other storefronts and products as well. The chosen products are set within the array "product_urls" which can be updated as needed for additional products.

An example of the parse script working on the node is shown below:

This test was successful as reflected in the database results image in the following section. An additional benefit for the tester is that the URL is printed, so the price can be manually verified by going to the link in the output of the script.

MongoDB Database Design:

The MongoDB database will handle all data that is pulled down from the parse node, which will allow for the parse node containers to be brought up or destroyed as needed with no effect on data persistence.

The image used for MongoDB will be the 'latest', which ensures that we have the most up-to-date security and functionality for our database. There is currently one administrative account setup within the Docker image at runtime, with more to be added as testing continues. The Docker image also copies in a products.json file so that the database is populated with basic entries of data. The port exposed for communication from both the parse node and the frontend node will be port 27017.

From a schema standpoint, the database focuses on three core components:
1. Product url (string)
2. Product name (string)
3. Product price in $USD (float)
4. Date (string)

These datatypes are open to configuration and may change once more products are added to the database, or users request additional parameters.

The parse nodes will utilize a generic account ('mongoadmin2') which has read and write access to the database. This minimizes the potential for errors and limits the parsers to only view or update data as it comes in. The items will be written to the "Inventory" database within the "Items" collection. Since MongoDB is a non-relational database management system, the entries will be stored in .json format.

An example of the database with parsed entries shown below:

```
    _id: ObjectId('660b6f5d57c436f65b75c6b3'),
    url: 'https://www.amazon.com/Beats-Studio-Pro-Personalized-Compatibility/dp/B0C8PR4W22',
    name: 'Beats Studio Pro – Wireless Bluetooth Noise Cancelling Headphones – Personalized Spatial Audio, USB-C Lossles
Audio, Apple & Android Compatibility, Up to 40 Hours Battery Life – Black',
    price: '$199.95',
    timestamp: '2024-04-01 22:37:03'
},
{
    _id: ObjectId('660b6f5d57c436f65b75c6b5'),
    url: 'https://www.amazon.com/Logitech-Tenkeyless-Lightspeed-Mechanical-LIGHTSYNC/dp/B085RMD5TP',
    name: 'Logitech G915 TKL Tenkeyless Lightspeed Wireless RGB Mechanical Gaming Keyboard, Low Profile Switch Options,
ightsync RGB, Advanced Wireless and Bluetooth Support – Linear',
    price: '$172.11',
    timestamp: '2024-04-01 22:37:09'
},
{
    _id: ObjectId('660b6f5e57c436f65b75c6b7'),
    url: 'https://www.amazon.com/Lenovo-IdeaPad-Display-Celeron-Graphics/dp/B0BLW25MB7',
    name: 'IdeaPad 1 14 Laptop, 14.0" HD Display, Intel Celeron N4020, 4GB RAM, 64GB Storage, Intel UHD Graphics 600, Wi
11 in S Mode, Cloud Grey',
    price: '$144.70',
    timestamp: '2024-04-01 22:37:16'
}
```

These entries indicate successful pulls of product url, product name, product price, and a timestamp. As we get more time collecting data, the number of entries will increase as the pull time changes.

Additionally. MongoDB Atlas will be used to create visualizations for the user to have faster comprehension and understanding of the data. As a MongoDB product, it is functions well with our current MongoDB database setup and will provide a simple integration. Within our React node which is running the frontend, MongoDB Atlas integrations will be embedded in the frontend for live updating from the database node. These communications will be through a designated port (80801).

React Frontend Application/UI:

The React frontend will be running on a node that utilizes Node.js and an alpine version of nginx. We are using node:slim to cut down on the space taken up by this large image. This image copies the package,json file and uses it as a reference to

download Node.js package dependencies needed for React. The Dockerfile then build the Node project and uses nginx to bring up the React project on an nginx server using port 80. In preliminary testing, docker commands were utilized to expose the webserver on port 8080. As testing continues, a different port may be utilized.

The UI will feature the name of the website, and a dropdown menu will allow the user to choose a product which pulls the embedded MongoDB Atlas Charts, which will contain line graphs and scatter plots to help the user visualize the data. This simple interface will provide the user with all the information they need to make an informed decision. The UI is still being designed and implemented, keeping in mind that the user may request different functionality as more products and websites are added.

An example of the React server running:
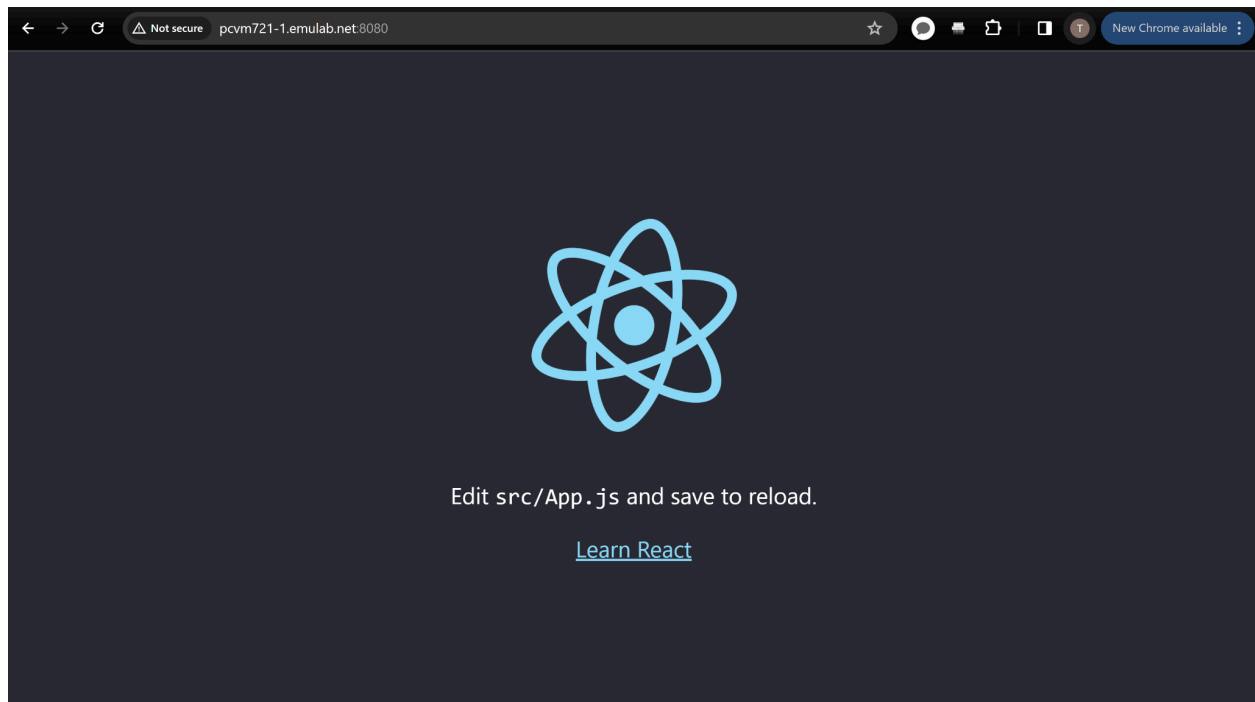
Chapter #4: Final Results

The team succeeded in creating a GUI that a customer could use to gain insights on prices of products on Amazon. Within the Kubernetes cluster, we have a node to parse data from Amazon, a MongoDB node to store data, a MongoDB Atlas node to create charts, and another simple webserver node that is running nginx.

From a stakeholder perspective, the user can gain insights on the price of an Amazon product over a given time period (data is limited to when we started scraping prices). However, even within the relatively small amount of time tracked, we did notice significant changes that could lead to savings for the user. For example, the headphones that were tracked through the app jumped from $20 USD to $40 USD within a time period of 10 hours. This was a price increase of approximately 200%; a valuable insight for a user looking to save money on a product that they want to purchase. With more time to track, the user would be able to see what times of day/week/month/year provides the most likely opportunity to save money.
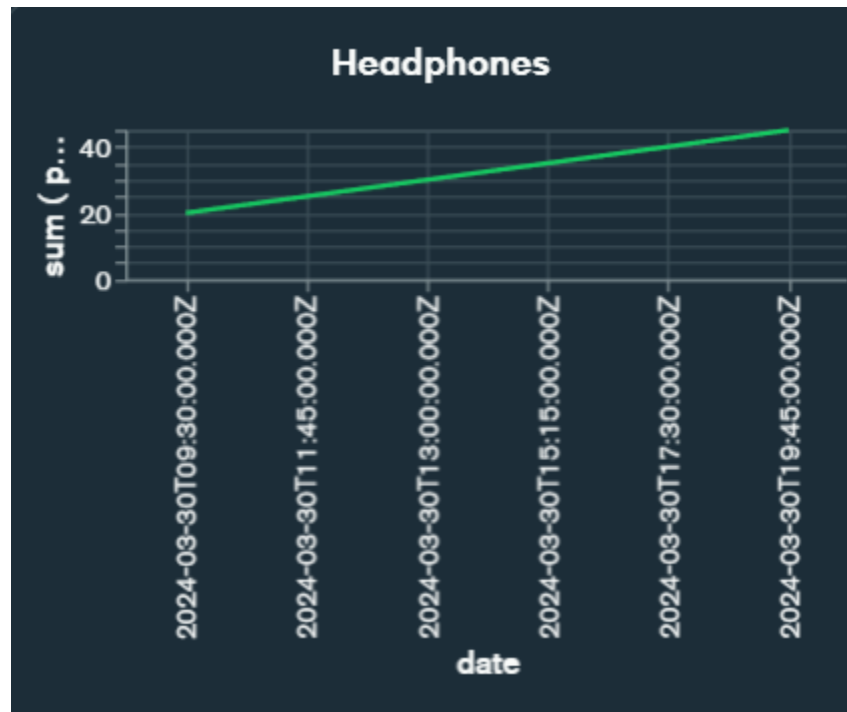

The challenges for the project were getting the webscraper to iterate consistently, connecting the MongoDB database to our MongoDB Atlas instance, and difficulties creating a React server which led to the choice to use a basic nginx server instead. The webscraper appeared to have an issue where it would only scrape once when brought up as a pod in the Kubernetes cluster.

Missed milestones were the React server; the team encountered consistent issues trying to manage the Node requirement of React when creating a Docker image. The Node library was massive (requires 5+ minutes to build) and consistently installed an incompatible version of Node into the server as the image kept installing Node/npm 12.2 version; React requires version 14+ regardless of what image was selected. This created issues where the team needed to revise our project in order to find an alternative way of having the MongoDB Charts viewable through a webserver, Fortunately, MongoDB Charts take care of all data analysis and graph management tasks, so the team was able to simply embed the charts directly into the index.html file used by the webserver. As anticipated, the functionality of MongoDB Charts was a lifesaver as it removed a large part of the configuration for the graphs.
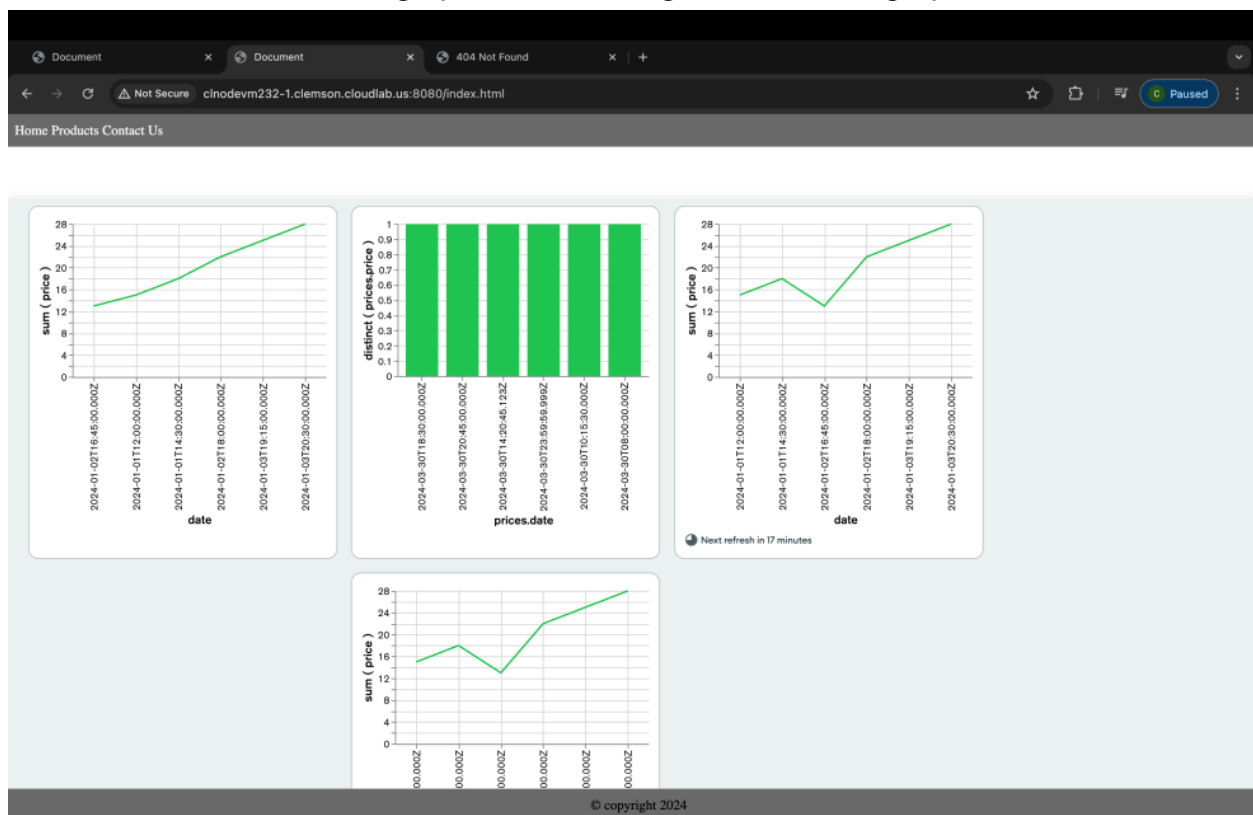
Additionally, the team was not able to add functionality for selecting products from a dropdown and more user-friendly elements such as user logins to have favorite products tracked per user. While the fundamental goal was achieved of allowing the user to gain insight on the product prices, these functions could be added to streamline the user experience which would be beneficial if the app were scaled with an official release. With more time available outside of the project window, these would be feasible.
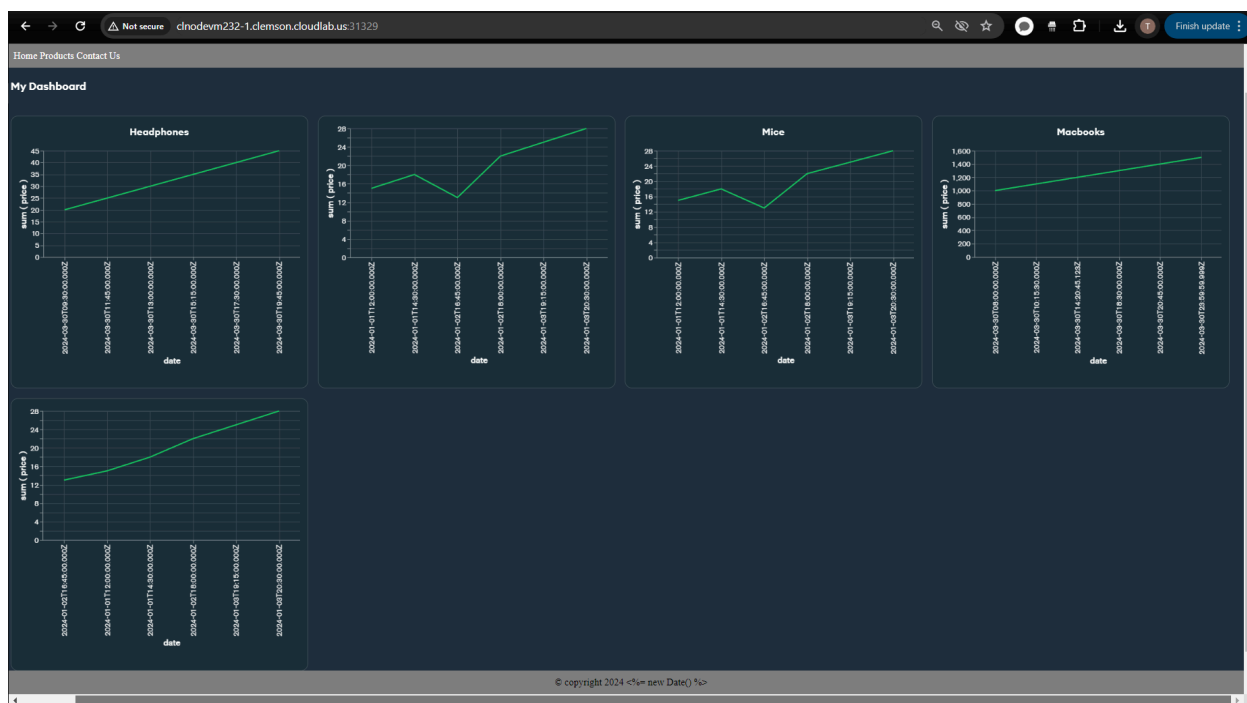
Chapter #5: Conclusion

Things the team learned from the project were an appreciation for project scope, more technical skills, and the need for a stronger design plan. We elected to choose a more fundamental concept so that we could get the minimum viable product together and then add additional functionality later. This allowed all team members to contribute ideas since we were all familiar with what we were trying to accomplish, and allowed us to get started on parts such as the webscraper earlier in the process. Additionally, the smaller scope gave us more confidence on picking up individual portions as the MongoDB database. All the team members gained new skills (such as Python and MongoDB), and even those that were familiar with these languages/tools had to learn how they functioned within a Kubernetes cluster. In retrospect, the need for a more actionable design plan would have been beneficial to break down the project into even more defined tasks. While we did create an action plan early in Phase 2, it would have saved more time to have a defined list during Phase 1. This would be a lesson learned for the next project our team would work on.

Some reflections for this project were that the team worked well together; we all stepped in where needed and helped with parts that others had trouble with. New ideas and contributions were evaluated and all team members felt satisfied with what they were adding to the project. At the conclusion of the project, each team member was pleased with what they were able to accomplish given their available time. Other concerns such as server bans due to webscraping never came into play, so while that functionality was not used, it is still a possible avenue if it becomes an issue in the future.

Potential future extensions and added functionality for the project are user logins, availability of other products/storefronts to be added, and more robust user control of product data tracked and graphed. Ideally, any user would be able to view the frontend, login with credentials, and be brought to a webpage that displays their favorite products. The user would also be able to readily add products to be tracked, and the parser would be adaptive to different storefronts. Additionally, the user would be able to configure the frontend to show whatever data is desired by the user (different time periods, data points, and comparisons across sites).

Tyler Martin

Malvern, PA 19355 | inquiries.tmartin@gmail.com

SUMMARY

Growth oriented Computer Science student with 7+ years of experience in the arts, 1.5+ years of experience in e-commerce, and impressive customer service abilities making a career change to technology. Passionate about providing technical solutions and developing collaborative relationships through mentorship and education.

SKILLS

- Gitlab/GitHub/GitBash
- Python
- Wordpress
- Conflict Resolution

- Docker
- Java
- Public Speaking
- Problem Solving

- Kubernetes
- Shopify
- Small Group Management
- Team Building

EDUCATION

West Chester University of Pennsylvania, West Chester, PA
Bachelor of Science - Computer Science - (Anticipated Completion) - 2024

Montgomery County Community College, Blue Bell, PA
Associate of Science – Computer Science – 2021
Associate of Arts – Liberal Studies – 2019

PROFESSIONAL HIGHLIGHTS

Zenescope Entertainment, Horsham, PA
Web Development and Marketing Associate, May 2016 – October 2017
- Helped drive over $500,000 of annual comic book sales through development of e-commerce webstore
- Used Mailchimp email services to market products to 10,000+ customers
- Created content for Wordpress blog for customer contact, marketing, and product display
- Migrated 100s of products from Pinnaclecart to new Shopify web storefront
- Utilized HTML/CSS to develop Shopify storefront to improve UX/conversions
- Collaborated with team members to develop campaigns and sales strategies

Upper Moreland High School, Willow Grove, PA
Part-Time Music Teacher, June 2018 – November 2019
- Oversaw development of teamwork and musical skills for a group of 10 high school students within a larger ensemble
- Created rehearsal strategies and used interpersonal communication skills to educate and mentor 9th – 12th grade students

- Helped compose, adjust music, and collaborated with other staff members to create a positive member experience

Philadelphia Eagles, Philadelphia, PA
Gameday Entertainment/Performer, September 2014 – Present
- Performed at all Philadelphia Eagles regular-season and post-season games in stadium and adjacent lots for tailgates
- Created dynamic musical and visual entertainment for over 60,000+ fans per game
- Traveled to perform for small groups, parties, parades (Macy's Thanksgiving Day Parade)
- Maintained equipment as needed and prepared multiple pieces of music

Christoper Seth Calixte

Lansdale, Pa | LinkedIn | Github

EDUCATION

West Chester University of Pennsylvania   **West Chester, Pa**
Bachelor's of Science, Computer Science   Cumulative GPA: 3.4
Anticipated Graduation Spring 2025
Experience

Baraka Grocery   **Philadelphia, PA**
Assistant Manager   May 2019 - July 2021
- Handled sensitive information and money
- Developed skills in customer service and satisfaction
- Analyzed inventory and kept records of current stock

PROJECTS

School Projects
- **Calculator Application**-Created a simple calculator using JFrame GUI
- **BlackJack-**Created a BlackJack application that mimics the game
- **Sudoku solver**- Created a complex sudoku program to generate and solve a sudoku puzzle efficiently
- **Articles Website-**Designed and deployed a simple html and CSS article for an an anti-demagogic
- **DatabaseManagement**-Implemented a program to communicate with a sql database
- **GRSH**-Created a golden rams shell terminal
- **Personal Website**-Designed a personal website with html and css

CLUBS AND PARTICIPATION

Competitive Programming Club
Treasurer
- Participated in the International Collegiate programming contest to solve complex coding problems in Java with a team

Computer Science Club
Member
Cybersecurity Club
Member
Beta Alpha Psi & Wipfli Ethics case competition
Determined and developed a solution to a complex ethics problem with a small team
C.C.S.T. Hackathon
Won 1st place in CCST Hackathon with a mobile application

TECHNICAL SKILLS

Languages: Java, C/C++, Python, MySql, Html, Css, Javascript, Sql
FrameWorks: WordPress
Developer Tools: Git, Docker, VS Code, IntelliJ, Eclipse, MySql, Pycharm, Microsoft access

Relevant Coursework

Computer Systems**,** Operating Systems, Data structures and Algorithms, Java I, II, III, Database Management

References

Available upon request

Alexis Nunez

Kennett Square, PA 19348   alexisnunez5664@gmail.com

Independent, initiative-taking, and flexible fourth year computer science student who is ready to take on the challenges of a tech job. Excellent listener and prioritizes time for different tasks and enthusiastic to gain experience by completing projects and providing innovative ideas.

Education:

West Chester University   West Chester, PA

- o   Bachelor of Science in Computer Science
    - **o   Current GPA: 3.73**   **(Dean's List)**
- o   Expected graduation date: May 2024

Projects:

Pokémon 2D RPG Game

- o   Collaborated with a group to implement different game features
- o   Constructed a 2D game environment using SpriteKit framework

Experience:

Red Sombrero   Kennett Square, PA   June 2020 - Present

Cashier responsible for customer service and cleaning duties

- o   Recognized for learning job duties quickly and demonstrates teamwork
- o   Managed leadership by training new employees

Mushroom Festival   Kennett Square, PA   Sept. 2017 & Sept. 2018

Volunteer responsible for helping with various stands

- o   Recognized for being punctual and enlisted help when needed
- o   Strengthened a good relationship with others and was able to work under pressure

Skills:

Languages: Java, Python, Swift

Developer Tools: Visual Studio Code, GitHub, XCode

Liam Daly

West Chester, PA 19380
liamdaly1003@gmail.com

EDUCATION

Souderton Area High School, Souderton, PA

September 2016 - June 2020

Graduated: June 2020

Montgomery County Community College, Blue Bell, PA

August 2020 - May 2022

West Chester University, West Chester, PA

August 2022 - Present

Expected Graduation: 2025

WORK EXPERIENCE

GIANT Food Store, Harleysville, PA *- DIRECT Manager and Customer Service*

June 2019 - May 2022

- Manage customers, orders, and deliveries to store
- Track inventory and maintain database of financial and performance metrics

Planet Fitness, Lansdale, PA *- Customer Service*

May 2022 - August 2022

- Register, sign-in, and membership enrollment for customers
- Dust and clean facilities

VOLUNTEER EXPERIENCE

Lamancha Animal Rescue, Coatesville, PA

September 2022 - January 2023

- Animal care for dogs and cats
- Complete daily and weekly tasks