Amazon Bargain Bot

Abigail Clemens
James Park
Steven Wetten
HaoLun Zhang

Table of Contents

Chapter 1: Vision – 2

Chapter 2: Proposal – 3

Chapter 3: Intermediate Milestones – 4

Chapter 4: Final Results – 7

Chapter 5: Conclusion – 10

Epilogue: Resources – 12

Epilogue: Poster – 13

Epilogue: Resumes – 14

Chapter 1: Vision

Introducing the Amazon Bargain Bot, an innovative program that will revolutionize the online shopping experience by providing users with excellent product selection backed by a comprehensive sorting algorithm.

When searching for products on Amazon, many users are exposed to a plethora of products that just aren't what they're looking for. Whether it be unreasonable prices, terrible reviews, or those sneaky products that have a high rating due to a low review count, the Amazon Bargain Bot aims to reduce user exposure to products such as these.

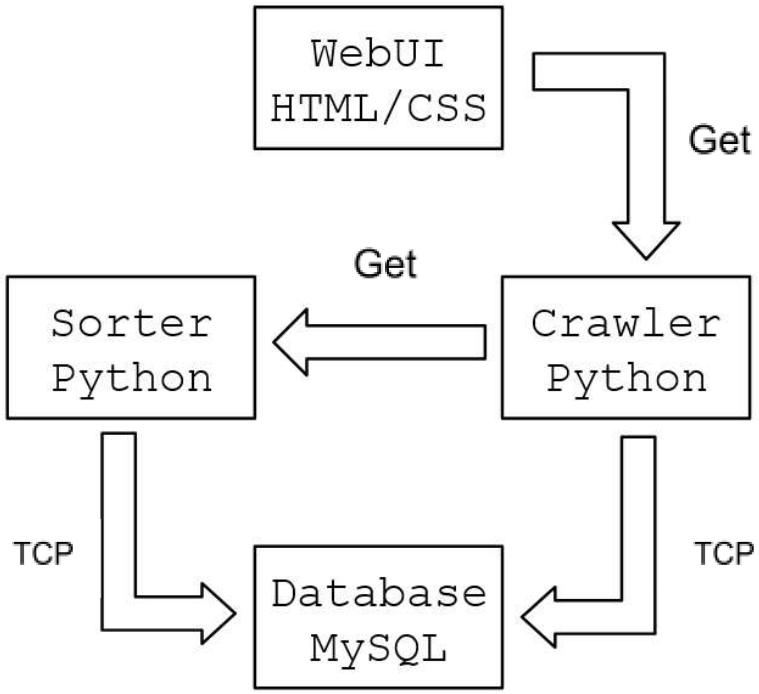
The Amazon Bargain Bot, like most online stores, will prompt users to enter their search term (like table, grill, deck chair, or any other item you could possibly want). Following this, it will scour the Amazon website and store all results into a separate database. Unlike the Amazon UI, however, the Bargain Bot will return only the items worth the user's time, based on a comprehensive sorting algorithm.

The sorting algorithm works as follows: first, all items will be retrieved from the Amazon website. Then, the Bargain Bot will find the average price of all of these items, as well as the average number of reviews. Finally, it will return only the products with; a price at or below the average price of all products; a review count greater than or equal to the average review count of all products; a rating greater than or equal to 4 out of 5 stars. This ensures that the products displayed by the Amazon Bargain Bot are only the best of what Amazon has to offer.

<u>Summary of Key Features:</u>

- User Interface capable of providing a customizable shopping experience
- Comprehensive sorting algorithm ensuring only the best items are displayed
- Removal of most "ad" products

Chapter 2: Proposal



The figure above is a diagram of the Amazon Bargain Bot Architecture, which works as follows: First, the user will interact directly with the WebUI, inputting a shopping search-term, via an HTTP connection, which should work on any chromium browser using the correct URL. Following this, the WebUI will issue a Get call to the Crawler.

The Crawler will establish a connection with the Amazon website and search for the term provided by the WebUI. Following this, the Crawler will parse each product listed on Amazon, storing the description, price, rating, number of reviews, and URL of each of them into a csv file. Then, the Crawler will establish a TCP connection to the MySQL database and import the csv file into a new table. Finally, the Crawler will issue a Get call to the Sorter.

The Sorter will first establish a connection with the database and pull all prices and review numbers for each product in the new table. After finding the averages of these two items, it will return only the items from the table that match the algorithm stated in chapter 1 to the Crawler. The Crawler will then return this to the WebUI.

Chapter 3: Intermediate Milestones

3.1: WebUI

The WebUI for the search functionality has several components that will come together in a cohesive experience that will allow the user to fine-tune their search to select the best product for their needs. These components are the search bar, the minimum rating field, the price range field, and the minimum number of reviews field. The two main aspects of the database WebUI are the ability to sort results based on various characteristics and the basic layout of the database itself.

The main part of the search functionality WebUI is the search bar. This field will contain placeholder text telling the user to search something. The other fields are used to filter out unwanted search results and give the user exactly what they are looking for. The first of these fields is the minimum rating field. This field will allow the user to select their cutoff point for how much of a positive reception a product would have to receive before they would consider buying it. The second field is the price range field. This would allow the user to make sure that they can afford the product that they're buying and that the product that they're buying isn't too cheap. The third field is the amount of review. This field is important because sometimes a product will have a five star rating but only have five reviews, which calls the quality of the product into question. With this field, a user can select a number of reviews that will guarantee that the review score is accurate.

The database WebUI also has important components. For starters, the layout of the database has several pieces of information about the products searched up. These are the name of the product, a link to the product, the number of reviews, the rating, and the price. In addition, the database allows you to sort by all of these so that the user can determine which product suits their needs the best.

3.2: Crawler

The Crawler, while priorly thought to be the most daunting portion of this project, has surprisingly seen a large amount of progress, second only to the Database. This is due partially to the fact that a premade template for the Crawler was found on Github, created by user praneethravuri. With some modernization

and modifications to both fix the script and add some functionality, the script was ready to be implemented into our project design.

The script first uses Selenium to establish a Google Chrome connection and navigates to an Amazon URL corresponding to a search term provided by user input. There, it uses the BeautifulSoup library to parse through the html source code of all pages of results generated by Amazon, separating each result. Then, it stores the description, price, rating, number of reviews, and URL of each product into a new csv file. Next, it uses the Pandas library to manipulate the csv file into a form that will be deemed acceptable by the Database. Finally, after using the MySQL connector python library to establish a connection to the Database, it imports the manipulated csv file into a new table. This script does have a completed respective Dockerfile, has been built into an image, has been pushed to a DockerHub repository, and has been successfully tested as a running container.

While all of these results may seem like an accomplishment, the Crawler still has a long way to go. First and foremost, the Crawler lacks completed yaml files, making Kubernetes deployment currently impossible. The introduction of yaml files will change the needs of the code itself, meaning that there will be more problems during the development of said files. All necessary future changes to the script, due to the introduction of yaml files, that are currently known to the group are listed below:

- The input form is going to need to change. During the deployment process of the crawler component, we encountered an EOF Error, especially when the script attempted to read input from the standard input stream. This error was found to be caused by the use of the Python "input" statement to read user input. This works fine when the script is given input directly by the user but will need to change when input is received through the WebUI. The EOF Error halted the deployment process and required modification of the input mechanism to accommodate the lack of interactive user input in the deployment environment. We are exploring alternative methods to provide input to the crawler component during deployment. One approach is to pass input parameters as environment variables.

- The naming scheme of the MySQL table is going to have to be discussed and determined. Currently, the script uses a predetermined, hard-coded name for the MySQL table. This works fine for testing purposes, but becomes a problem when the system is used more than once. There are two possible fixes for this. First, the table could be deleted by the Sorter when it is finished generating the return data. Second, the Crawler could search the Database to see if a table already exists under the name of the user's search term, and skip the data collection for this instance, sending the already-existing table to the Sorter right away. This would also speed up the system.

3.3: Sorter

The Sorter, like the WebUI, has seen minimal progress. Currently, the data type to be returned is the only glaringly unknown item, but more will become known in the future.

3.4: Database

The Database is the element of the project that has seen the most progress, as it is essentially finished.

Currently, the Database has a respective Dockerfile and respective yaml files, as well as a built image uploaded to DockerHub. The Database is open to TCP connections and has been successfully manipulated through both solitary and containerized python scripts outside of the reach of the Database. These successful tests were done while the Database was instantiated through Kubernetes, through the use of a service.

The only issues with the Database are that it currently has an unrelated user, an insecure password for said user, and a table that is not essential for this project, due to it being repurposed from an assignment outside of the scope of this project. This can, and will, be fixed by deleting the table, deleting the user, creating a new, reasonable user, then rebuilding and pushing the image to DockerHub.

Chapter 4: Final Results

4.1: Accomplishments

First and foremost, the project has reached a point where it is functional enough to call it mostly working. The WebUI is capable of taking user input, passing it to the backend, and displaying the sorted results.

4.2: Challenges

By far the greatest challenge of this project was determining the communication scheme between the components. The scheme discussed in the earlier sections of this report was deemed to be too complex and not optimal for our project. In light of this, we decided to establish a new communication scheme, demonstrated in figure 4.2-1.

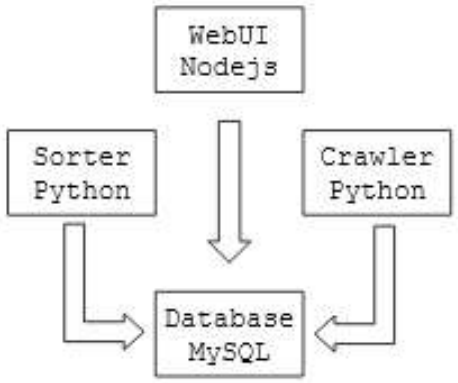


Figure 4.2-1: The final communication scheme

With this communication scheme, the database became the backbone of the entire deployment. The WebUI would establish a connection to the database and place a search term into a specific table, which would be read by the Crawler. The Crawler would then place the search results into another table, which would be read by the Sorter. The Sorter would then sort this data and place it back into a final table, which would be read and displayed by the WebUI.

Another challenge encountered was establishing a method to retrieve user input via the WebUI. You may have noticed that the WebUI box in figure 4.2-1 is

slightly different from previous examples. This is because, in order to retrieve and process data from both mysql and the user, we decided to switch to using NodeJS instead of the standard Javascript and HTML/CSS we had originally planned on implementing.

Communication was also a sizable challenge. Due to the nature of this project being course-structured, there were many times where other course-related work or even non-course related events impeded the continuation of progress. Specifically, there were times when communication between members was slow or even nonexistent due to the busy schedule that comes with attaining a bachelor's degree.

4.3: Issues

While the project is in a state that can be considered a success, there are still a number of issues that would need to be addressed to truly call it complete. For the sake of formatting, these issues are listed below:

1. The WebUI is, almost embarrassingly, barebones. It consists of a single search bar and two buttons on one page, and displays the results in a raw format, if it could even be called a format. This is a simple fix, so we decided to forego it in favor of fixing other, more technical issues.
2. The implementation of the project is slow and unoptimized. This is in large part due to the lack of error handling in both the Crawler and the Sorter. Currently, when no data is present in the search table or results table, the Crawler and Sorter crash and revert back to their original state. By introducing error handling, the Crawler and Sorter would both "complete" their operations instead of crashing, removing the need for the pods to revert back to a responsive state. This would significantly speed up the process. Again, this is a simple fix, it would just be rather time consuming.
3. Due to the slow nature of the project, displaying the results now requires the press of a button on the index page of the WebUI. Eventually, we would want to switch this to an instant load of the results after a search, but significant optimization of the backend would need to take place, first.
4. Certain search terms cause the Crawler to error out. The exact cause of this is unknown to us, but could most likely be fixed, given more time.

While the presence of these issues is not to be overlooked, we still believe we can say that our project has succeeded. This is mainly due to none of these issues being impossible to fix, only time consuming. Given an extension, we truly believe we could deliver a fully-functioning and visually-pleasing bargain bot.

Chapter 5: Conclusion

As we bring the journey of the Amazon Bargain Bot project to a close, we reflect on a path marked by significant learning, innovation, and collaborative achievement. The project began with a bold vision to enhance the online shopping experience by filtering out subpar products, using a robust algorithm to ensure users receive recommendations that are not only cost-effective but also highly rated and well-reviewed.

Throughout the development process, our team tackled numerous technical challenges, from perfecting the WebUI for optimal user interaction to refining the Crawler for effective data extraction from Amazon's vast product listings. Each milestone was met with a dedicated effort to overcome obstacles and improve our system's functionality and user interface.

The architecture of the Amazon Bargain Bot, as detailed in the proposal and subsequent chapters, exemplifies a dynamic integration of several technologies including Selenium, BeautifulSoup, Pandas, and MySQL, orchestrated via a modern Docker and Kubernetes environment. This infrastructure allowed for scalable and efficient handling of data, and the Kubernetes deployment, in particular, highlighted our commitment to a robust, cloud-native approach.

However, the journey was not without its challenges. The adaptation of the Crawler to avoid the pitfalls of non-local Amazon domains and the continuous optimization of our data sorting and handling mechanisms are areas that highlighted both the complexity of the task and the potential for further refinement.

Despite these hurdles, the final product stands as a testament to what can be achieved through persistence, technical skill, and teamwork. It is a functional prototype that demonstrates the viability of the concept and the potential for further enhancement and commercial application.

As we conclude, we acknowledge that while the project is not without its imperfections, the foundation laid is solid, and the pathways for future improvement are clear. With more time and resources, the Amazon Bargain Bot could evolve into an indispensable tool for savvy online shoppers worldwide,

embodying the original vision of a smarter, more efficient online shopping experience.

In essence, this project was not just about building a software tool but about learning and growing as developers and collaborators in the ever-evolving landscape of technology. The experiences gained, the challenges overcome, and the knowledge shared will undoubtedly serve as invaluable assets in our future endeavors in technology and innovation.

Resources:

Ravuri, P (2023) Amazon-Product-Information-Scraper (de72711) [amazon_scraper.py]. https://github.com/praneethravuri/Amazon-Product-Information-Scraper.git.

Amazon Bargain Bot

Abigail Clemens, James Park, Steven Wetten, Haolun Zhang

https://github.com/StevenWetten/Amazon_Bargain_Bot.git

Purpose

Sort products based on certain criteria (price, rating, # of reviews)

- Easier than paging through countless search results
- Made with buying products not on the first few pages

WebUI

- Node.js, HTML, and CSS
- Simple, but easy to navigate
- Sends search term to database
- Retrieves and displays sorted results

Database

- MySQL
- Stores data in three tables
- Search Term
- Search Results
- Sorted Results

Crawler

- Retrieves search term from database
- Searches amazon with search term

Sorter

- Retrieves search results from database
- Sorts the results
- Sends sorted results to database

Result

Return items with a price less than or equal to average and a rating greater than or equal to 4 stars

```
0:
  Description:  'HP 2023 Newest 17 Laptop…Silver, KKE Accessories'
  Price:        "590.00"
  Rating:       "4.2 out of 5 stars"
  Review:       "58"
  URL:          "https://www.amazon.com//…p&qid=1714650440&sr=8-17"
1:              {…}
2:              {…}
3:              {…}
4:              {…}
```

```
FROM python:3
ADD crawler.py /
RUN wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add -
RUN sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google-chrome.list'
RUN apt-get -y update
RUN apt-get install -y google-chrome-stable
RUN pip install --upgrade pip
RUN pip install bs4
RUN pip install selenium
RUN pip install webdriver_manager
RUN pip install mysql-connector-python
RUN pip install pandas
CMD ["python", "./crawler.py"]
```

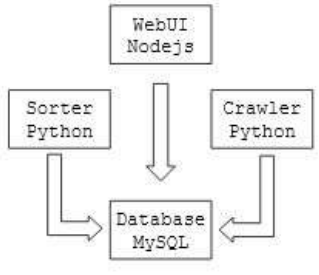
```
import os
import time
from datetime import date
import csv
from bs4 import BeautifulSoup
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException
import mysql.connector
import pandas as pd
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: crawler
spec:
  replicas: 1
  selector:
    matchLabels:
      app: crawler
  template:
    metadata:
      labels:
        app: crawler
    spec:
      containers:
        - name: crawler
          image: stevenwetten/mysql:bbotcra
          ports:
            - name: crawler-port
              containerPort: 3306
```

crawler-649d5bfc4d-b2xd7	stevenwetten/mysql:bbotcra	app: crawler, pod-template-hash: 649d5bfc4d	worker-5.stevenw-204231.edu.emulab.net	CrashLoopBa	16	1.30	209.30Mi	an hour ago
sorter-6c9d6655d9-mz77j	stevenwetten/mysql:sorta	app: sorter, pod-template-hash: 6c9d6655d9	worker-2.stevenw-204231.edu.emulab.net	CrashLoopBa	18	-	-	an hour ago
webui-7cc97cfbc4-q7mpp	stevenwetten/mysql:wbui	app: webui, pod-template-hash: 7cc97cfbc4	worker-1.stevenw-204231.edu.emulab.net	Running	4	1.00m	16.17Mi	a day ago
mysql-58cc844959-5nwc	stevenwetten/mysql:bbotdbase	pod-template-hash: 58cc844959, role: app	worker-1.stevenw-204231.edu.emulab.net	Running	0	13.00m	403.98Mi	a day ago

```
var mysql = require('mysql');
var express = require('express');
var app = express();
var http = require('http');
var fs = require('fs');
var bodyParser = require('body-parser');

var connection = mysql.createConnection({
    host: 'mysql',
    user: 'bbots',
    password: 'csc468g6',
    database: 'bargainBot',
    port: 3306,
});
```



Up Next:

- Further search customization is needed (rating, price, etc)
- A switch from WebUI button to loading screen is preferred
- Crawler and Sorter need better error handling and optimization
  - Currently, they crash if there is no data in specific tables. Error handling would allow them to "complete," speeding up the process.
- WebUI needs decoration and formatting

Abigail Clemens

215-275-1927 | AC946904@wcupa.edu

Education

West Chester University West Chester, PA

Bachelor of Science in Computer Science, Minor in Mathematics *Aug. 2020 – May 2024*

GPA: 3.79/4.0

Relevant Coursework:

- CSC 141 – Computer Science I
- CSC 142 – Computer Science II
- CSC 220 – Foundations of Computer Science
- CSC 231 – Computer Systems
- CSC 240 – Computer Science III
- CSC 241 – Data Structures & Algorithms
- CSC 301 – Computer Security & Ethics
- CSC 317 – Digital Image Processing
- CSC 321 – Database Management Systems
- CSC 335 – Data Communications & Networking I
- CSC 345 – Programming Language Concepts & Paradigms
- CSC 402 – Software Engineering
- CSC 496 – Topics in Complex Systems (iOS App Development)

Experience

Cashier October 2019 – March 2020, June 2021 - Present

Wegmans *Concordville, PA*

- Provided excellent customer service by being friendly to customers and helping them locate desired items
- Achieved a high IPM (items per minute) of 18 in order to assist customers quickly
- Received training in other areas to learn more and help benefit the company
- Learned to work in the deli

Projects

Lexical and Syntax Analyzer | *Java* August 2022 – December 2022

- Analyzed a simple language

Platformer | *Java* September 2019 – June 2020

- Developed a simple computer game that had a goal of jumping over as many moving blocks as possible and recorded a high score

Richman 4 Prototype | *Swift* August 2023 – December 2024

- Developed a simple computer game that had a goal of collecting coins and avoiding bombs

Technical Skills

Languages: Java, Python, Swift, OCaml, C, SQL, R
Developer Tools: Git, IntelliJ, Eclipse

Libraries: NumPy

James Park

jp1014506@wcupa.edu | 610-864-5133 | linkedin.com/in/james-park-0b02a6b6/

I am a product-driven individual with a wide array of professional experiences looking for a full-time position or internship that will lead to gainful employment for my last year at school. AI, Machine Learning, and Natural Language Processing are some of the areas that I would like to learn more about and work in, and I have experience in general troubleshooting for Windows and Macs, Object-Oriented Programming languages, Assembly language, Networking, and more. I am open to working in almost any industry but find work in international or political issues, food, and finance the most interesting.

Education

West Chester University of Pennsylvania West Chester, PA; *Jan 2023-Current*
-Pursuing Computer Science B.Sc. and Linguistics Minor | Expected graduation in June 2025 | Current GPA: 3.72
-Participant in: WCU Sept 2023 Programming Competition | CSTA Hackathon Sept 2023- 1st Place | WCU March 2023 Programming Competition- 3rd Place | March 2023 Python Bootcamp | April 2023 West Chester Programming Competition

George Washington University Washington, DC; *August 2008-July 2012*
-International Affairs B.A. | Chinese Language and Literature B.A.

Additional Skills

-Java, C, Python, HTML/CSS, SQL
-Multilingual in Mandarin, Korean, and Nepali
-Excellent Public Speaker

Relevant Coursework

I have taken courses on Discrete Math, Calculus, Statistic, Java, SSH and other network protocols, along with general education requirements necessary for my degree completion at WCU. These include Chemistry, Geology, & Business Writing.

Work Experience

Infinitus; Remote; *Dec 2023 - Present*
AI Operations Intern
-Developed an understanding of the healthcare insurance landscape, especially issues surrounding payments
-Operated the AI interface at the customer-interaction stage to ensure smooth and useful conversation
-Performed data-entry of patient and insurance information into specific formats for AI improvement

West Chester University Tech-Fee Computer Lab Intern West Chester, PA *May 2023 - Present*
-Monitoring the software, hardware and accessories for the over 3,000 Tech-Fee devices on WCU campus
-Providing support for the faculty/staff and students, including troubleshooting, upgrading devices, and recycling
-Negotiating solutions for issues pertaining to devices, software, and general use of the labs

West Chester University Chinese Language Tutor West Chester, PA *Jan 2023 - Present*
-Providing weekly in-person and online tutoring sessions to students enrolled in the Chinese program
-Coordinating lesson plans and student progress reports with the faculty to ensure student success

Community Academy of Philadelphia Information Technology Intern Philadelphia, PA *June 2022-Jan 2023*
-Assisted in organizing, cleaning, and maintaining inventory of computers for staff and students
-Resolved technology-related issues for faculty and students during school hours
-Organizing spreadsheets to record which devices are with whom

Dig Acres Assistant Manager Chester, NY *June 2018-Oct 2021*
-Lead manager in Food Safety and farm adherence to Good Agricultural Practices (GAP) regulations
-Managed and taught crew of 3-5 apprentices for 8-9 month through an intensive agricultural education program
-Oversaw and organized harvesting, weeding, plowing, and other activities related to day-to-day farm operations
-Facilitated Dig Supply Team and Corporate offices on various projects and operations
-Frequently held or attended events that promoted the farm and company

Peace Corps Nepal Food Security Volunteer Khasyauli, Palpa District *March 2015-June 2017*

-Trained local farmers on growing higher-quality vegetables and increasing seasonal availability, including high-yield intensive methods on small plots of land
-Co-Facilitated week-long agricultural camp for village youths, topics ranged from land-restoration to gender &caste issues
-Established a System of Rice Intensification demonstration with three model farmers, with a 100% increase in production
-Discussed nutrition-related diseases with locals and held cooking & gardening demonstrations with the goal of long-term behavior change

Steven Wetten

Lewisberry, Pennsylvania 17339
717-798-1709 stevenwtgf@gmail.com github.com/stevenwetten

Education

West Chester University of Pennsylvania **August 2020 – May 2024**
Bachelor of Science in Computer Science *West Chester, Pennsylvania*

Relevant Coursework

- Data Structures
- Software Engineering
- Cloud Computing
- Networking
- Compilers
- Computer Systems
- Software Security
- Malware Analysis

Work Experience

West Chester University of Pennsylvania **March 2023 – November 2023**
Programming Competition Judge / Writer *West Chester, Pennsylvania*

- Collaborated with a team of peers to write and solve topical and relevant Java and Python programming problems for three separate programming competitions.
- Displayed leadership throughout the development of problems of varying difficulty via coordination and delegation of work.
- Ensured perfection of problems prior to each competition, utilizing a participant/user-oriented mindset.
- Supervised and observed contestants during each competition to ensure participants could compete in a fair and equal environment.

Bobby Rahal Honda **June 2022 – August 2023**
Porter (Seasonal) *Mechanicsburg, Pennsylvania*

- Worked with and informed multiple departments to ensure complete customer satisfaction.
- Exceeded customer and management expectations through the demonstration of strong work ethic and dedication.
- Performed a variety of customer services to maintain sales and relations.

Projects

Amazon Bargain Bot | *Python, MySQL, Kubernetes, DockerHub* **January 2024 – May 2024**

- Developed a bot using Python, MySQL, and HTML/CSS to search and display top-rated and reasonably-priced results from Amazon.
- Implemented Selenium to create an instance of Google Chrome to interact and return the correct elements of the web page.
- Created a MySQL database to store and sort the information received from the bot.
- Utilized Kubernetes to instantiate images as containers using a CloudLab resource cluster.

MiniJava-Based Compiler | *Java, Maven, Assembly* **August 2023 – December 2023**

- Created a Compiler for a MiniJava-based language called Ram23.
- Scanned, parsed, semantically analyzed, and generated assembly code for Ram23 using Java.
- Utilized IntelliJ and Maven to clean, compile, and test source code for the compiler.

Technical Skills

Languages: Java, Python, C, HTML/CSS, MySQL, Assembly, Haskell
Developer Tools: VS Code, IntelliJ, Maven, Eclipse
Technologies/Frameworks: Linux, GitHub, Kubernetes, DockerHub

Leadership / Extracurricular

Competitive Programming Club **August 2023 – May 2024**
Secretary *West Chester University of Pennsylvania*

- Hosted an award-winning university-wide programming competition.
- Informed members of club activities and events through the use of flyers and announcements.
- Recruited Computer Science and related majors utilizing various methods during recruitment campaigns.

Haolun Zhang

215-882-2692 | eric.h.zhang460@gmail.com

EDUCATION

Delaware County Community College Media, PA

Associate Degree in Computer Science GPA: 3.39 *Jan. 2020 – August. 2021*

West Chester University West Chester, PA

Bachelor's Degree of Computer Science GPA: 3.44 *Sep. 2022 – May 2024*

EXPERIENCE

Equuleus International Clu June 2021 – August 2021

Office Assistant

- Demonstrated expertise in utilizing Excel to effectively sort, filter, and manipulate large amounts of data.
- Extensive experience in using Excel for data cleansing, data validation, and data visualization tasks.
- Skilled in using Excel for data management, organization, and analysis of extensive datasets.
- Proven track record of efficiently managing and organizing large documents using Word's navigation and referencing tools.
- Strong ability to create and format tables, charts, and graphics in Word to enhance document presentation.

Alpha Computer May. 2015 – October 2015

Front Desk Assistant *Malvern, PA*

- Meet and assist incoming customers promptly
- Set up, test, and configure networks, desktops, laptops, and printers,
- Assist in basic administrative tasks, including document filing and record keeping, concerns, and issues professionally and efficiently.

PROJECTS

Database Management Project | *Python, SQL* Sep 2023 – Dec 2023

- Utilizing Microsoft Access to design and develop a database, including creating tables, and establishing relationships.
- Writing queries, and generating reports to support internal data management and information tracking within the organization.

Market Trends Analysis | *Python* Oct 2023 – Dec 2023

- Perform a thorough examination of the used car market through Python
- Utilizing the programming language for data manipulation, cleaning, and exploratory data analysis tasks.
- Using data visualization libraries to generate informative charts and graphs, effectively illustrating market trends within a designated data range.

TECHNICAL SKILLS

Languages: Java, Python, SQL, Ocaml

Developer Tools: PyCharm, IntelliJ, Jupyter Notebook

Libraries: Pandas, NumPy, Matplotlib