John Park . Jarvis Consulting

I have graduated from the University of Toronto St. George with a Bachelor's degree in Computer Science. During my studies at the University of Toronto, I was first introduced to foundational and theoretical aspects of computer science such as time complexity, data structures, and algorithmic analysis. Afterwards, I was exposed to the applicable side of computer science by taking Web and Software development courses. Since then, I endeavoured to further develop the knowledge and the skills I gained from these courses by building Web and Mobile applications on my side time. Even now, I strive to build a web or mobile application in my free time to consistently absorb knowledge and hone my skills in related fields. Furthermore, I have more in-depth knowledge than mere software development because the computer science curriculum at the University of Toronto lead me to take more specialized courses such as Operating Systems, Machine Learning, Natural Language Processing, and Networks.

Skills

Proficient: Java, Linux/Bash, RDBMS/SQL, Git, Python, Javascript, HTML/CSS, Latex, Docker, Agile/Scrum, C/C++, ReactJS

Competent: Machine Learning, Natural Language Processing, Pytorch, MATLAB, MongoDB, ExpressJs, NodeJs, Regex

Familiar: Swift, AngularJS, DevOps, Verilog, Neural Networks and Deep Learning

Jarvis Projects

Project source code: https://github.com/jarviscanada/jarvis_data_eng_JohnPark/

Cluster Monitor [GitHub]: Implemented a monitoring software for Jarvis Cluster Administration Team which automatically retrieves hardware specifications and resource usage from each machine and node of a Linux cluster. This program also initiates the database automatically and all the information retrieved from each node is stored there. The database is established using docker and PostgresSQL and the program is implemented using bash scripts. Furthermore, because the resource usages are dynamic over time from each machine or node, crontab was utilized to continually update the corresponding information on the database.

Core Java Apps [GitHub]:

- Twitter App: Created a Java application which allows a user to Create, Read, and Delete tweets. The application interacts with Twitter's REST API using HTTP requests and responses. The response from Twitter REST API is parsed and the requested information is properly returned to the user. All the relevant dependencies, classes and components are put together using SpringBoot. Junit and Mockito were used for integration and unit tests. Finally, the application is deployed using Docker and hosted on Dockerhub
- JDBC App: Implemented JDBC App which connects to the Postgres SQL database. The application is implemented using the DAO pattern thereby containing an isolated application/business layer from the database. The DAO objects created here are CustomerDAO and OrderDAO. After the application establishes a connection with the SQL database, it mounts the JDBC driver using the database connection manager. This application can Create, Read, Update, and Delete data from a Postgres SQL database containing data about customers and orders.
- Grep App: Built a java application with equivalent functionality to the Unix-based command-line utility "grep". The application takes a regex pattern to match, a path to a file, and an output file as arguments. Then, it finds the line(s) containing the inputted regex pattern from the source file and writes on the output file given. Using Maven, the application and its relevant dependencies are compiled into an Uber Jar file which was then dockerized and deployed on the Dockerhub.

Springboot App [GitHub]: Built a REST API based on microservice architecture. The API manages accounts and trade orders and updates stock information by fetching information from IEX Cloud. Java was used to develop the application and Postgres SQL was used to establish a database. Then, Spring Boot was used to consolidating all the microservice components together. Therefore, Inversion of Control through Dependency Injection was attained by the Spring framework. Maven was used to managing all the dependencies and integration tests were done through JUnit4 and Mockito. Two docker images were generated - Trading-app and Trading-psql - and were deployed on Dockerhub.

Cloud/DevOps [GitHub]: Migrated an application to the cloud using Azure. Established ad scale set and database server using Azure. The network is configured so that the database server is only accessible through the application. The application is accessible externally through the load balancer which hosts an external IP address. The application is accessible Built a Jenkins CI/CD pipeline to automate build, deploy, test etc.

Highlighted Projects

Sentimental-Analysis-Of-Stocks (Swift) [GitHub]: Built an iOS application containing a trained NLP model that predicts whether the stock market will increase or decrease based on twitter input. The NLP model is composed of Convolutional Neural Network which is trained through extensive data from Twitter and news. Once input is fed, the application will output an indication of whether the stock market is in an upwards trend or downwards trend with a confidence interval value.

TextYourTile (MERN stack) [GitHub]: Developed a MERN full-stack application deployed on Heroku. Thus, ReactJS was used for the frontend, Node/ExpressJS was used for the backend, and MongoDB was used for the database. It is an application where users can write on tiles laid out on the canvas. The tiles extend almost infinitely because new tiles are constantly generated as a user navigates around the tiles. Anything that is written on the tiles is not deleted and stored in the database. The user can come back to see what they have written unless other users have erased or modified it. Also, a socket is built into the application so that users can see other users making changes in real-time. Login and Sign up features are also available. Therefore, users can personalize the profile and make a separate personalized canvas.

Online-Educational-Performance-Analysis (Python) [GitHub]: Developed machine learning models that predict whether a student can answer diagnostic questions based on the student's answer to previous answers to other questions. This will help online learning platforms to predict the students' level of abilities and personalize the education service. For example, assigning the questions with proper difficulty levels. Using the data obtained from Kaggle, three different models using four ML different algorithms - K-Nearest Neighbourhood, Item Response Theory, Matrix Factorization - were developed. The accuracy of Item Response Theory (IRT) was improved by using the ensemble technique and by extending the one-parameter base IRT model to a two-parameter IRT model.

File-System (C) [GitHub]: Designed and developed a Unix-based file system using C. This program behaves very similarly to the Linux file system. However, the data structure and the format of how data are stored behind have been customized. It is consisting of a superblock, inode bitmap block, 8 contiguous data-block bitmap blocks, 4 contiguous inode table blocks and finally the contiguous data blocks. Each block has a size of 4Kilo Bytes = 4096 Bytes. Similar to a Linux file system, general information of the file system like the number of inodes, the number of data blocks, where the inode begins etc… are stored in the superblock. Likewise, the metadata of files and directories are stored in their inode. One can run this program by executing mkfs.a1fs file on an arbitrary directory. Then, it acts like there is a whole new file system on that directory. One can then perform commands like mkdir, touch, cd, ls etc…

AlphabetTilesNSeaInvaders (Java) [GitHub]: Built a mobile Android gaming application using Android Studio. This has a gaming centre which harbours three games – Sliding Tiles, Alphabet Tiles, and Sea Invaders. The goal of the sliding tiles is to align the tiles in order. This can be done by simply touching the tiles next to the empty tile. Alphabet Tiles start with A and when you combine two As you get Bs, and combining two Bs get you Cs and so on… The game demands you to get the alphabet Z. The Sea Invaders game has invaders invading from the top of the screen and moving down. The goal is to not let a single invader reach the bottom of the board. All these games have a scoring system through time. The shorter you finish the game, the better score you get. The game centre will keep track of these records and post the best-scored user. Thus, it is a competitive gaming app.

Professional Experiences

Software Developer, Jarvis (Apr 2022-present): Developed applications and automated software for Jarvis projects and team. Used Linux/Bash, Git/Github, Docker/Dockerhub, Java, PostgreSQL and Virtual Machine by Google Cloud Platform for various software development projects. Collaborated and worked with other Jarvis consultants, the scrum master, and senior developers in the Agile/Scrum environment. Participated in daily scrum meetings and biweekly scrum sprint meetings. Communicated consistently with senior developers and the scrum master for technical and behavioural improvements.

Laboratory Assistant, Department of Cell and Systems Biology, University of Toronto (May 2016 - Sept 2016): Collaborated with researchers in the lab and helped them with research projects. Prepared materials and maintained the bio-model crucial to the projects in the lab. Washed agar plates and moved agar vials to the benches in demand. Constructed the agars which are used for lab activities and food for the bio-model.

Education

University of Toronto St. George (2017-2021), Honours Bachelor of Science, Computer Science Specialist - GPA: 3.28/4.00

Miscellaneous

- Coursera - Learning Linux for LFCA Certification
- Coursera - Introduction to Docker
- Volunteer - Toronto General Hospital: Provided administrative support by organizing patient data and guided patients and visitors to the hospital to their desired location
- Soccer Player - Won Gold, Silver, Bronze medal in Men's intramural soccer hosted at University of Toronto