**Shashank**

***Extensible experience of upwards of 7+ years as a Software Engineer in application development in Java/Python, Databases, RESTful Web Services (Large-scale Distributed System & Cloud)***

**PROFESSIONAL SUMMARY**

|  |
| --- |
| * Exceptional Computer Science Professional background in analyzing, developing, integrating, and deploying web-based applications. In which my expertise ranges in multi-disciplined teams from agile backend development, DevOps, Data mining and analytics, cloud technologies to testing, debugging, deployment and management. * Comprehensive experience in contributing to the architecture and design of current plus new systems using Python, Java, Flask, Spring Boot, MVC, J2EE, Django, JavaScript, web APIs, MySQL, PostgreSQL, MongoDB, Graph QL, SQL, database design, and normalization, JSON, HTML, CSS, and Git. * Experienced in developing applications using Agile methodologies, and Scrum in Python and Java environments. * Experience with continuous integration and delivery using Jenkins, built automation using Maven, codebase, and Version control using Git, and deployment using Docker where client-server communication is through REST API. * Experience in configuring and deploying a web application using AWS code Pipeline and its services. * Solid understanding of processes, scheduling, memory allocation, fragmentation, memory allocation, page replacement, critical section, in Operating Systems. * Knowledge of network and its services: TCP/IP protocol stack, network security (cryptography, TLS), load balancing, DNS, HTTP/s, subnetting, web cache, email, NFS, NIS, and authentication services. * Strong development experience in design and development of software applications using Java, J2EE (JSP, Servlets, JDBC), RESTful Web services, Spring Framework (MVC, REST, Cloud – AWS, cloud Kubernetes), * Experience in analyzing, designing, and developing web applications using Python, and its web services (RESTful, Requests, JSON, SOAPy, WSDL, PycURL), micro-framework (flask), full stack (Django, Pyramid). * Through understanding of Account Scoring, Salesforce analytics, CRM, data analytics using Python, SQL. * Experience in web scraping, data extraction, cleaning, indexing, crawling using Hadoop, MapReduce. * Created a research project recommended by the University of California for Autonomous Vehicle path prediction and optimization on given Time Series dataset, LiDAR using UNet CNN architecture. |
|
|
|
|
|
|

**EDUCATION**

|  |  |
| --- | --- |
| **Master of Science in Computer Science** |  |
| University of California, Riverside, CA | |
| *Relevant Coursework*: *Advanced Algorithms, Advanced OS, Computer Networks, Databases, Distributed Systems* | |
| **Bachelor of Engineering in Computer Engineering** |  |
| Devi Ahilya University, India |  |

**TECHNICAL SKILLS**

|  |  |
| --- | --- |
| **Language** | Python, Java, C++/C |
| **Development Technology** | JSP, Servlets, JMS, JDBC, Spring MVC |
| **Databases** | SQL, MySQL, PostgreSQL, AWS RDS, NoSQL (MongoDB), Graph QL, |
| **Web Technologies** | HTML, CSS, Bootstrap, JavaScript, |
| **Network/ Network Security** | OpenSSL, LibreSSL, lib TLS, root CA, TCP/IP, Socket Programming, Hashing, Bloom Filter |
| **Cloud Technologies** | AWS (RDS, S3, EC2), Heroku, Docker, Kubernetes |
| **Build/ Version Control** | Maven, Git |
| **CI/ CD** | Jenkins, AWS, Heroku |
| **Operating Systems** | macOS, Unix, Linux, Windows |
| **Big Data** | Hadoop, MapReduce, Spark |

**PROFESSIONAL EXPERIENCE**

|  |  |
| --- | --- |
| **Software Engineer** | June 2020 - Present |
| **Client: Account Spark, Inc., San Francisco, CA** |  |
|  |  |
| **Account Spark Product** is one of the most powerful and promising Artificial intelligence CRM tools on the Salesforce AppExchange today. Combining the power of Salesforce with a proprietary Analytics engine, Account Spark delivers deeper insights into your accounts, driving profitable sales strategies, and fueling sales growth by letting you focus your time on the right accounts. The application allows you to inspect the account scoring of all your clients with project potential, top clients that tell you where to focus by dividing the list into 4 sections that have different levels of potential and probability. It also lets you do reinforcement learning and guide the system by taking your feedback. | |
| * Spearheaded the development of two powerful ML engines, for lookalike customers using data manipulation and data quality anomaly detection, written in Python, and plugging into a Postgres database, managed DevOps in Heroku. * Developed data integration pipeline using RESTful with simultaneous updating of data to the cloud platform, Heroku PostgreSQL database using Python, Requests, Django, including Heroku flow for building the application continuously. * Redefined the data mining model with more than 15 scripting files by following the OOP principle that fits extension without modification, managed multiple agile SDLC iterations with updating the database with various version controlling. * Drafted the Account Spark Inc. documentation and code review of all the scripting developed by the team that included more than 20 cross-functional scripts with a direct connection with software requirements and development. * Responsible for handing developer’s team GitHub branch that needs to be deployed each day with a unit test coverage and managed the master branch for the team once per week for debugging and sandbox environment execution. * Collaborated effectively with scientists on implementing account potential predictions and refining the model to fit the nuances of the consumptive Twilio cost and economic modeling solutions. | |
| **Environment: Python, Regression, RESTful, HTTP, JSON, PostgreSQL, Heroku, Github, Jupyter Lab, Salesforce** | |
|  | |
| **Software Engineer at Adroit Inc.** |  |
| **Client Project 1: Secure File Transmission Application** | January 2020 - May 2020 |
|  |  |
| **Summary:** Implemented a secure proxy application that uses a TLS protocol to provide strong authentication and secure file transmission between multiple clients and the server. A client is free to request any file from the server, where the system then directs the request to the proxy which has a separate cache of files from the last transactions. The system is capable of balancing the load by using consistent hashing which chooses the same proxy for a single file so that multiple clients don’t request the same file from different proxies. | |
| * Established an encrypted socket network using the libLTS API that makes the client, the proxy servers, and the remote server use TLS for all connections. * Authenticated the servers (proxies) from the client-side by using the CA root certificate that got proved over the TLS handshake. Thereafter sending the requested file name securely to the proxy. * Organized a consistent hashing to balance the load over the proxies that come by file requesting in such a way that all the request of one file from multiple clients goes to one proxy as this minimizes the cache reuse by 91% and stop unnecessary access to the main server. * Created a Bloom filter that keeps track of black-listed items and reduces the latency by 96% to reject a request asking for black-listed items. This can be measured by the size of the Bloom filter to the number of such items. * Implemented the server port that does not wait all time and check port for request in a while and wait for the proxy to make a TLS handshake. | |
| **Environment**: **C, OpenSSL, LibreSSL, RSA, Socket programming, Linux** | |
|  | |
| **Client Project 2: Twitter Scrapping for the product reviews** | December 2019 - January 2020 |
|  |  |
| **Summary:** A Twitter search engine that lets the user get all the tweets in a given time range that has the searched keywords in it. The frontend displays tweets in decreasing relevance order as it contains the tweet with most contains and highest appeared keywords in it. | |
| * Created a program for generating an inverted-indexed word occurrence in a large number of documents that reduce the scanning and counting time by 43% when compared to a program without such indexing by using MapReduce distribution of the task. * Implemented crawler and extraction of tweets from a large database of Twitter and sorting them by word frequency occurrence matched with searched keywords. * Carried out the ranking by comparing the key-value frequency and document location, and then injecting in the front end of the system developed using JavaScript. * Capable of configuring Lucene that indexes the multi-threaded crawled tweets used particularly with search engine designs that search keywords using indexes in the text/document. * Actively participated in the lead activities to gather technical requirements, improve the design of system and API, code reviews, minor mentoring, and support. * Maintained the sorting of relevance by values of keyword/s searched using Java where the hash table stores word as a key and has a list of occurrence of the key in all documents as value. | |
| **Environment: Java, Hadoop, Map Reduce, Apache Lucene, Maven, Eclipse** | |
|  | |
| **Client Project 1: Sawari**- **A Car Rental Service -** *Java Development* | January 2019 - November 2019 |
|  |  |
| **Summary:** An Award-winning web application that allows users to book car service without any restriction to trip duration and its distance. The system has minimal relation between distance traveled and time taken. The price formula is such that, it outperforms every other market value with more than 30% cheaper and the same profits. Given multiple car ranges to choose from with different base prices that are then added with base prices for total trip time and distance and then with additional costs to balance extra miles and hours on top of base settings.  Additionally, the system is user specific. While a customer can view and book, encrypting their details, admin can manage availability, price range, look of my system. Developer gives access to add and/or remove services by adding a code. | |
| * Accomplished best user feedback as measured by comparing presented price ranges for each class and different parameters (total required trip time, and total required distance) for rental booking based on Spring MVC architecture following Agile SDLC methodology. * Developed end-to-end web application based on JSP, JavaScript, HTML, CSS, bootstrap for single page application featuring with the whole viewer of the system for front-end. * Designed and developed 3 modules with user activity-specific where each user has different authority over the actions using session checker and EKS clustering to give multiple access even at the same time. * Maintained a thorough MVC architecture to maximize the code reuse using Controller class, and a seamless integration of Viewer to the servlets for resource transfer connecting with JDBC. * Spearheaded the analysis and designing of software requirements with a small team of 6 incorporating defining, evaluating, and examining newer v/s existing requirements using Agile methodology. * Analyzed system design by estimating the database cost, storage capacity, time trade-off for services that tell which services can be compromised to fasten other services of the system. Results in showing up of a listing of cars faster than updating the availability of cars. * Used AWS code Pipeline for CI/CD with Maven, Docker to build, and AWS plugins to deploy the application onto AWS Cloud Kubernetes environment. * Faster results were achieved by distributing the database hit request, which later turned into hitting with the RESTful API to get and update resources in the AWS databases RDS, and then using stateless connections to maintain transfers. * Developed REST services using Java, Spring boot, Spring MVC to handle REST actions and scalability securely. | |
| **Environment: Java, JSP, Spring MVC, HTML, JavaScript, CSS, Bootstrap, JSON WSDL, MySQL, AWS S3 REST, RESTful API, HTTP web, AWS CLI, Agile methodology, Eclipse** | |
|  | |
| **Software Developer,**  **Affimintus Technologies, Indore, India** | November 2014 - December 2018 |
|  |  |
| **Doctor Appointment Software:** Offers services to schedule, reschedule or cancel one or multiple appointments made by a user after getting authenticated. It allows a user to make appoint for anyone else as well by first verifying self-identity. It gives exceptional services to pull previous appointments, reports, feedback. More than 5 hospitals are using the software for over more than a total of 60 doctors. The backend handles all the real-world complexities including overlapping of the appointment slots, cancellation of it from both ends, authenticating network, no-show of a patient gets available soon, rebook an appointment if a doctor is unavailable on confirmation. | |
| * Participate in requirement & design discussions, release planning, daily standups, backlog grooming, iteration planning, and Demo sessions regularly through Agile. * Developed REST services using Java, Spring Boot, and Spring MVC. Used spring core for dependency injection for interacting with the database, MySQL, later implemented on AWS RDS. * Responsible for developing and customizing on changing requirements front-end using JSP, HTML, CSS, JavaScript for Single Page application side. * As a DevOps, implemented frequently and reliably continuous integration, deployment using AWS code Pipeline, AWS code Deploy, Jenkins for integration, AWS S3 for storage, AWS EC2 for deployment, and Git for the codebase. * Evaluated Overlapping requests with multiple to one doctor availability in real-time. Solved using scrapping abs in Timestamp. * Worked cross-functionally between planning, production, DevOps team and delivered efficiently requirements, development, and deployment. * For more than 3 months, got involved with the team lead, and managed a team of 4 to produce and deploy an improved overlapping algorithm to solve the real-time issue of multiple appointments for a single user/doctor. | |
| **Environment: Java, Spring, REST, JSON, HTML, JavaScript, Boot Strap, Jenkins, JUnit, MySQL, AWS RDS, S3, EC2, Eclipse, HTTP Client, API Gateway** | |
| **Global Wheat Detection -** *Computer Vision* |  |
| **Summary:** Something granular as Wheat, identifying and classifying them becomes more complex and expensive to improve target. By exploring the dataset for various grains, the quality of pixel matching, and then picking the best features, leads to faster achieving an impressive 65% accuracy. Moving further, to accomplish the best score, using a feature pyramid on top of CNN made feature selection more promising on each down sampling resulted in gaining a 4% improvement beating 75% of the world record. | |
| * By using the bounding boxes, the predictive model eliminated the multiple classifications on a grain head and localized the class with the most score. This follows the pattern You Only Look Once. * Implemented training by KFolds cross-validation with 4 to 6 folds of learning chosen randomly for each iteration, in it the model uses RCNN FPN with a clear improvement from using Fast RCNN as result were 37% better in time to gain the same accuracy, and then at convergence give 2% better accuracy. * Data preprocessing includes mosaic augmentation to give a concrete pixel learning as it gives a perfect combination of classes in the data. Followed by MixUp and cleaning augmentation. * Used Test Time Augmentation to improve testing accuracy as it does synthesis and feed the learned model with test data. | |
| **Performance analysis of Multipath TCP** |  |
| Summary: When sending packets in the network, TCP uses a single path that does the communication be that of two-way or request-response. Pushing it to optimization, by using multipath TCP saving round trip time on communication is achieved. From the second Transmission, a ping path is being followed that forms a table of rules saving source and destination from multiple paths to required destinations. Optimization is measured by recording 43% time efficiency compared to a single path from the same subnetting and IP address settings. | |
| * By improving TCP routing traffic via multiple paths, obtained average RTT reduction by ~50% using OpenFlow, learner, and controller which sets all the flow rules by the first ping. Redirects the network to the most optimized path. * The topology that was tested has 8 switches, 5 routers, and 3 hosts with limited bandwidth of 50 Mbps * Measured traffic diagnosis performance by using Wireshark, iPerf and implemented packet flooding to fill rule table for incorrect paths. | |
| **Environment: Virtual network setup - python, C++, Virtual Machine, Wireshark, Mininet, Xterm emulator** | |
| **Toxic Comment Classification Challenge -** *Natural Language processing* |  |
| **Summary:** The machine learning engine is capable of multi-classifying the Twitter tweet into 5 categories of how toxic it is. From threat to identity-based hate. Now the challenge is to determine such tweets that do not spread toxicity in direct words but use harmonious words such as “Gas the non-black as Skypes.” Certainly, it’s hated but cannot be detected by regular software that just tracks words. This model learns such thing by correlation of strong words with each word in that sentence were expressing that word with its other meaning. | |
| * Scored 97% accuracy in predicting the classes of tweets using the NLP model based on Naive Bayes with Support Vector Machine and using word2vec to mark each word with its other meaning which helps in improving the accuracy of indirect toxicity by 90%. * Structured the data preprocessing techniques, like fixing structural errors, tf-idf for priors as the model needs to get rid of such words that don’t mean anything to one’s intention but are used most number of times in a tweet, like *‘is’*,*’ the’.* * Analyzed the data comprehensively to understand useful features using correlation among target variables, character distribution, merging the labels of word categories that mean same but written differently. * Solved the multi-classification complexity by disintegrating the problem into 5 and training model for all classes separately on each class as binary classification. It improved the accuracy of 2 classes by 3% and 1.5%. | |
|  | |