SANDHYA CHANDRAMOHAN

Los Angeles, California 92614 | (949) 522-0190 | sandhyc@uci.edu| https://github.com/s4ndhyac

EDUCATION

UNIVERSITY OF CALIFORNIA, IRVINE	**Master of Computer Science**	**CGPA: 3.60/4.00***	**Expected: Dec 2019**
INDIAN INSTITUTE OF TECHNOLOGY	**Bachelor of Technology**	**CGPA: 8.07/10.00**	**Graduated: July 2014**

EXPERIENCE

SOFTWARE ENGINEERING INTERN, Tiger Connect, Santa Monica, CA **June 2019 – Present**

- Formulating a migration plan, scripting in Ansible and executing the migration of DNS to AWS Route53 with zero downtime

SENIOR MEMBER OF TECHNICAL STAFF, Capital Float, Mumbai, India **May 2017 - Aug 2018**

- Lead Developer of Amazon Pay EMI, analogous to the Amazon.com store card in the US used by millions of users daily
- Pioneered the Publish-Subscribe Design Pattern to scale the platform, enabling it to support 10 times the existing users
- Implemented Distributed Locking using Redis to handle 100% of all race conditions in a multi-user web application
- Reduced deployment downtime by 100% adopting the Blue-Green deployment strategy, setting up a Continuous Integration / Continuous Delivery process in a distributed environment using Jenkins

MEMBER OF TECHNICAL STAFF, Capital Float, Mumbai, India **May 2016 - May 2017**

- Engineered a thread-safe concurrent application, synchronizing critical operations with Mutex objects to enable 100% real-time processing of all payments via REST API
- Spearheaded a distributed design by extracting database operations into Stored Procedures with Row-level locking
- Improved database performance by 30% and ensured 99.99% availability by setting up MySQL Master-Slave Replication
- Created a data processing pipeline enabling an ETL process for data transfers to a Datawarehouse

SOFTWARE DEVELOPER, Capital Float, Mumbai, India **May 2015 - May 2016**

- Redesigned a legacy WPF application in the Singleton pattern to a multi-tier RESTful Web Application in the MVC design pattern following OOD / OOP principles using ASP.NET Web API 2 and the WCF Framework
- Spearheaded the Unit of Work design pattern to carry out a business operation as one atomic transaction
- Ensured fast AGILE development cycles and code robustness by integrating the NUnit Unit Testing Framework
- Led the integration of an ORM - PetaPoco to abstract away database specific SQL and protect against SQL Injection

PROJECTS

BasicOS ● Contributed to the Open Source Unix-like OS xv6's kernel ● Repository **Apr 2019 – Jun 2019**

- Implemented Unix system calls to get the memory dump, support multi-threading, mutex locking, conditional variables, semaphores and linked list file-addressing to theoretically support files of infinite size
- Developed a unix shell with IO redirection, pipes and several simple unix programs like ls, cp

RDBMS ● Multi-layer relational database with optimized index and query systems ● Repository **Oct 2018 – Dec 2018**

- Devised several join algorithms such as the Block Nested Loop Join, the Index Nested Loop Join and Sort Merge Join algorithms using the External sorting algorithm for sorting
- Built an Indexing Component via a B+ Tree supporting range predicates, managing persistent indexes over structured data

Toxic Filter ● Natural Language Processing (NLP) to classify toxic comments ● Repository **May 2019 – Jun 2019**

- Pre-processed the data and performed feature engineering using NLP techniques such as removing stopwords, stemming, lemmatizing and converted clean data into word embeddings using TF-IDF vectorization and the word2vec pre-trained model
- Formulated and evaluated models such as Recurrent Neural Networks (RNN), RNN with LSTM, Convolutional Neural Networks (CNN), LGBM with Multinomial Naïve Bayes and Logistic Regression as the baseline machine learning models

Algorithms & Data Structures ● Hashing Algorithms | Tree Data Structures ● Repository **●** Repository **Jan 2019 – Mar 2019**

- Formulated optimized implementations in Java and evaluated the performance of several hashing algorithms - Linear probing, Double hashing, Chained hashing, Cuckoo hashing and Tree Data Structures - Binary Search Tree, AVL Tree, Treap, Splay tree
- Implemented the methods in the IHashingAlgorithm and ITree interface respectively for each

SKILLS

- **Languages: (Over 5000 lines)** C#, Java, Python, SQL **| (Over 1000 lines)** C++, C, JavaScript, HTML, CSS
- **Frameworks/ Libraries: (Extensive)** ASP.NET**,** Web2Py, JAX-RS, Jersey, Dropwizard **| (Comfortable)** Flask, React.js
- **Other Tools/ Technologies: (Extensive)** AWS, Docker, Git, Redis, UML
- **Relevant Courses:** Algorithms, Data Structures, Principles of Data Management, Operating Systems