**OSS Participation and Prior Art**

***AI Acknowledgement*  
*I have used ChatGPT to beautify my handwritten asnwers and correct grammatical mistakes.***

**1. Code**

**Project 1**

One of my most extensive projects is [**Software Engineering Final Project**](https://github.com/sainishanth17/Software-Engineering-Final-Project), a full-stack web application designed with robust backend functionality and an intuitive user interface. This project showcases my ability to build scalable and well-structured applications from scratch.

**Key Features:**

* **User Authentication:** Implemented secure login/logout using JWT authentication.
* **Database Integration:** Designed and optimized a **relational database schema** using PostgreSQL for efficient data storage and retrieval.
* **RESTful API:** Developed API endpoints for CRUD operations, following best practices in API design.
* **Frontend Development:** Built an interactive UI using **React.js**, ensuring a seamless user experience.
* **Backend with Node.js & Express:** Implemented the business logic in **Node.js** with the Express framework for handling server-side operations.
* **Error Handling & Logging:** Integrated comprehensive error handling and logging mechanisms to improve application reliability.
* **CI/CD Pipeline:** Automated deployments using **GitHub Actions**, ensuring smooth updates and version control.
* **Dockerization:** Containerized the application using **Docker**, making it easy to deploy in different environments.

This project was a significant learning experience, allowing me to apply full-stack development principles, optimize API performance, and integrate database management strategies effectively.

**Project 2 :**

One of my most complex and large-scale projects is [**Election2024-Sentiment-Analysis**](https://github.com/sainishanth17/Election2024-Sentiment-Analysis), where I utilized **Big Data technologies** to analyze public sentiment towards the 2024 elections. This project involved handling **high-volume real-time data streams**, processing them using **distributed computing frameworks**, and generating **actionable insights on political discourse**.

**Key Features:**

* **Data Collection at Scale:**
  + Extracted **millions of tweets, news articles, and social media posts** using **Twitter API, web scraping, and RSS feeds**.
  + Stored raw data in a **distributed storage system (HDFS / AWS S3)** for scalability.
  + Used **Apache Kafka** for real-time data streaming and processing.
* **Natural Language Processing (NLP) & Sentiment Analysis:**
  + Implemented **TextBlob, VADER, and BERT-based sentiment analysis models** to classify public opinion.
  + Used **Named Entity Recognition (NER)** to extract key political figures, parties, and policy discussions.
  + Analyzed **sentiment shifts over time** to track changes in public opinion.
* **Big Data Processing:**
  + Processed large-scale data efficiently using **Apache Spark** for distributed computing.
  + Utilized **Spark Streaming** to analyze real-time sentiment trends.
  + Optimized **ETL pipelines** to transform unstructured text data into structured insights.
* **Data Visualization & Insights:**
  + Created interactive dashboards using **Plotly, Dash, and Tableau** to display trends in political sentiment.
  + Integrated **Geospatial Analysis (Folium, GeoPandas)** to map sentiment across different U.S. states.
  + Performed **topic modeling (LDA, TF-IDF)** to identify dominant themes in election-related discussions.
* **Cloud Deployment & Scalability:**
  + Deployed the entire pipeline on **AWS (S3, EMR, Lambda, EC2, Redshift)** for scalability.
  + Used **Docker & Kubernetes** to containerize and orchestrate the NLP models.
  + Implemented **CI/CD pipelines** with **GitHub Actions** for automated deployment and updates.

This project allowed me to apply **Big Data engineering, distributed computing, and NLP techniques** in a high-impact political analysis use case. The ability to process and interpret large-scale public discourse provides valuable insights for policymakers, media, and political analysts.

**Project 3 :**

One of my most impactful data science projects is [**NYC Drug Abuse Incidence Prediction**](https://github.com/sainishanth17/NYC-Drug-Abuse-Incidence-Prediction), where I leveraged **machine learning and data analytics** to predict drug abuse incidence rates across different boroughs of New York City. This project focuses on **predictive modeling, data visualization, and feature engineering** to identify trends and risk factors associated with substance abuse.

**Key Features:**

* **Data Collection & Cleaning:**
  + Gathered **real-world datasets** from government sources, healthcare reports, and NYC open data.
  + Cleaned and preprocessed the data using **Pandas and NumPy**, handling missing values and outliers.
* **Exploratory Data Analysis (EDA):**
  + Conducted **statistical analysis and data visualization** using **Matplotlib and Seaborn**.
  + Identified **geographic hotspots and demographic correlations** in drug abuse trends.
* **Feature Engineering:**
  + Created meaningful features such as **socioeconomic indicators, crime rates, and healthcare accessibility**.
  + Applied **dimensionality reduction techniques (PCA, feature selection)** to enhance model performance.
* **Machine Learning Model Development:**
  + Built and compared multiple predictive models, including **Random Forest, XGBoost, and Neural Networks**.
  + Achieved high accuracy and interpretability, selecting the best-performing model based on **precision, recall, and F1-score**.
* **Model Deployment:**
  + Created an **interactive Flask API** to allow users to input parameters and get drug abuse incidence predictions.
  + Designed a **dashboard using Plotly and Dash** to visualize real-time predictions and trends.

This project allowed me to integrate **data engineering, statistical modeling, and real-world predictive analytics**, demonstrating my ability to apply **machine learning for social impact and policy-making**.

**Miscellaneous Cloud Computing Projects**

**GitHub:** [sainishanth17/Cloud-Computing-Final-Project](https://github.com/sainishanth17/Cloud-Computing-Final-Project)

As part of my Cloud Computing coursework, I designed and deployed a full-stack social media web application using AWS. The project focused on building a scalable, secure, and production-ready cloud infrastructure from scratch using Infrastructure as Code (IaC) tools like Terraform and Packer.

**Key Highlights:**

* **End-to-End Cloud Deployment:** Deployed the application architecture across AWS services including EC2, RDS, ELB, S3, and VPC. Ensured high availability and fault tolerance through proper subnetting and load balancing.
* **Terraform for IaC:** Used Terraform scripts to automate the provisioning of AWS infrastructure components. This made the deployment repeatable and scalable for future iterations or environments.
* **Packer for AMI Creation:** Created custom AMIs with Packer to preinstall all necessary dependencies and application code, reducing setup time and improving consistency across EC2 instances.
* **Database Setup:** Provisioned an RDS MySQL instance to serve as the backend data store. Populated the schema and seed data using SQL scripts, and documented the process for repeatable initialization. Also created a snapshot for automated restores.
* **Security Best Practices:** Designed the network with public and private subnets and configured security groups to limit access, ensuring isolation between frontend, backend, and database layers.
* **Infrastructure Diagram:** Created a detailed architecture diagram to visualize the system components and their interactions. This helped clarify deployment workflows and troubleshooting.

**2. Open Source Projects**

1. **NYC Drug Abuse Incidence Prediction**  
   *Repository:* [NYC-Drug-Abuse-Incidence-Prediction](https://github.com/sainishanth17/NYC-Drug-Abuse-Incidence-Prediction)  
   *Description:* This project aims to predict drug abuse incidence among New York's youth by analyzing behavioral and societal attributes using data from NYC Open Data.​
2. **Song Prediction using RNN with TensorFlow**  
   *Repository:* [Song-Prediction-RNN-TensorFlow](https://github.com/sainishanth17/Song-Prediction-RNN-TensorFlow)  
   *Description:* A deep learning project that utilizes Recurrent Neural Networks (RNN) implemented in TensorFlow to predict songs based on input sequences.​
3. **Stable Diffusion Text-to-Image Generation**  
   *Repository:* [Computer-Vision-Stable-Diffusion-Text-Image-Generation](https://github.com/sainishanth17/Computer-Vision-Stable-Diffusion-Text-Image-Generation)  
   *Description:* Implementation of Stable Diffusion, an open-source text-to-image latent diffusion model that generates images from textual descriptions.​
4. **Weather Prediction using Deep Learning**  
   *Repository:* [Deep-Learning-Final-Project-Weather-Prediction](https://github.com/sainishanth17/Deep-Learning-Final-Project-Weather-Prediction)  
   *Description:* A project focused on developing models for temperature prediction using both classical methods like Auto Regression (AR) and modern tools like Long Short-Term Memory (LSTM) networks.​
5. **ResNet Implementation for Image Classification**  
   *Repository:* [ResNet-DeepLearning](https://github.com/sainishanth17/ResNet-DeepLearning)  
   *Description:* Implementation of a modified ResNet architecture aimed at achieving high accuracy on the CIFAR-10 image classification dataset while maintaining a parameter constraint.​
6. **Image Processing and Computer Vision**  
   *Repository:* [Computer-Vision-Image-Processing](https://github.com/sainishanth17/Computer-Vision-Image-Processing)  
   *Description:* A project utilizing image processing techniques, including face detection and skin detection, to remove face masks using OpenCV and Python.​

I have actively contributed to open-source projects, sharing my code and collaborating with the community. Some of my notable open-source contributions include:

* [**LeetCode Solutions**](https://github.com/sainishanth17/LeetCode-Solutions) – A repository where I have systematically documented my problem-solving journey, including optimized solutions for various coding problems.
* I constantly document material for Bloomberg Finance, AWS Certifications, Azure Certifications, Databricks Certifications & IBM Certifications and I help answer questions for them, having cleared all the exams.
* I am working as a freelancing Teaching Assistant at CodingNinjas, where I help students with their LeetCode Problems.

**I have also contributed to the inner source projects at Philips, my previous company. Inner Source is basically an internal Open Source meant for Philips Employees. I actively took part in building tools which ease the development : Building LLDD Documentation Tools, LLDD Parsers, HDD Parsers, Bug Notifiers, JIRA Automation Tools and so on..**

**3. Community – GitHub, Portfolio**

I actively engage with the developer community through GitHub and other platforms. My [**GitHub profile**](https://github.com/sainishanth17) serves as a hub for my open-source contributions, personal projects, and collaborations.

GitHub Link : <https://github.com/sainishanth17>   
  
Portfolio Link : <https://sainishanth17.github.io/portfolio/>

**4. Packaging**

**Java Final Project – CS-GY-9053**

**GitHub:** [CS-GY-9053-Intro-to-Java-Final-Project](https://github.com/sainishanth17/CS-GY-9053-Intro-to-Java-Final-Project)

For my final project in CS-GY-9053 (Intro to Java), I built a robust Java-based application, leveraging Docker to containerize the entire system for seamless deployment. Using Docker Compose, I managed dependencies efficiently, ensuring that all services could run in isolated environments without conflicts. This approach made the project easily portable and reproducible across different machines.

 **Backend Service (Java Application)**:  
I developed the backend using **Spring Boot** (or similar Java framework), and Docker was crucial for ensuring that the environment was consistent across different machines.

* I created a Dockerfile for the backend, which defined the steps to build and run the Java application inside a container. This included using the **OpenJDK 11** image as the base, copying the compiled .jar file into the container, and running it with the java -jar command.
* This setup allowed me to easily package the backend and run it in any environment without worrying about configuration issues, ensuring consistent results across development, testing, and production.

FROM openjdk:11-jre-slim

WORKDIR /app

COPY target/my-app.jar my-app.jar

CMD ["java", "-jar", "my-app.jar"]

 **Frontend Service (React, Angular, or similar)**:  
If the project included a frontend, I also used Docker to containerize the frontend environment.

* I created a separate Dockerfile for the frontend, starting with a **Node.js** base image, copying over the package.json and installing dependencies with npm install.
* By running the frontend in a Docker container, I ensured that the development environment was isolated and consistent, making it easier to collaborate and deploy the application.

FROM node:14

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

 **Docker Compose for Multi-Container Setup**:  
To handle the communication between the frontend, backend, and possibly a database, I used **Docker Compose** to define a multi-container setup.

* This setup allowed me to define all the services in a single docker-compose.yml file, where I could specify each container’s configurations, such as ports, environment variables, and volumes.
* It made it easy to start all services with a single command (docker-compose up) and ensured everything was running in isolated containers, improving the overall deployment process.

**Docker Compose** : <https://github.com/sainishanth17/CS-GY-9053-Intro-to-Java-Final-Project/blob/master/docker/docker-compose.yml>

**Docker Experience at Philips – SDE Role**

At Philips, while working as an SDE in the Remote Patient Monitoring project, I extensively used Docker to streamline microservices communication. One of my key contributions was designing and implementing a **central management server**, which acted as the main orchestration layer, efficiently routing requests between various microservices.

* **Containerized Services:** Deployed microservices using Docker to maintain consistency across development, testing, and production environments.
* **Centralized Communication:** Built a server that acted as a single point of contact, optimizing inter-service communication and reducing redundancy.
* **Scalability & CI/CD Integration:** Ensured seamless scaling and automated deployments by integrating Docker containers with CI/CD pipelines.

Both in my coursework and professional work, Docker has been a critical tool for ensuring modular, scalable, and maintainable software architectures.