# SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES CHENNAI – 602105

# CAPSTONE PROJECT REPORT

## TITLE

# Serverless Computing Applications

## Submitted to SAVEETHA SCHOOL OF ENGINEERING

## By Ch. ESWAR

## 192210671

## Guided by Dr. J. CHENNI KUMARAN

# PROBLEM STATEMENT:

Problem Statement:

Serverless Computing: Develop a serverless application that leverages cloud functions to handle specific tasks or workloads without the need for managing servers.  
  
Task Management: The goal is to efficiently manage and execute tasks using serverless functions based on various criteria and triggers.  
  
The central topic in serverless computing is the efficient execution of functions for assigned workloads.

## Significance:

Serverless computing simplifies the deployment and scaling of applications by abstracting server management. It enables developers to focus on code and logic while cloud providers handle infrastructure.

# Proposed Design Work in Serverless Computing:

## Dashboard Overview:

- Start with a clean and intuitive dashboard.  
- Provide an overview of serverless functions, their current status, and invocation metrics.  
- Include visualizations (such as graphs or heatmaps) to depict real-time performance metrics.  
- Ensure that critical information is easily accessible at a glance.

## Function Management:

- Allow users to deploy, update, and manage serverless functions.  
- Provide clear labels and descriptions for each function to aid decision-making.  
- Include monitoring tools to track performance and execution metrics.

## Predictive Insights:

- Incorporate predictive models to optimize function deployment and execution.  
- Display confidence scores or probabilities to indicate the reliability of predictions.  
- Highlight any potential bottlenecks or resource constraints.

## Customization Options:

- Let users customize function triggers and parameters.  
- Allow them to adjust resource allocation based on specific requirements.

## Alerts and Notifications:

- Implement alerts for function failures or anomalies.  
- Notify users when a function's performance deviates significantly from predictions.  
- Provide actionable recommendations to mitigate issues.

## User Feedback Loop:

- Gather feedback from users regarding function performance.  
- Use this feedback to continuously improve the serverless framework.  
- Encourage users to report any discrepancies or unexpected behavior.

# IMPLEMENTATION:

## Connecting Components in Cloud:

## Resource Data Collection:

- Gather historical data on serverless functions.  
- Collect performance metrics such as execution time, memory usage, and invocation frequency.

## Feature Selection:

- Identify relevant features (QoS parameters) that impact function performance.  
- Examples: function execution time, memory allocation, trigger frequency.

## Serverless Model:

- Build a predictive model (e.g., regression, decision tree) to estimate function performance.  
- Train the model using historical data.

## Integration with Cloud Management System:

- Connect the predictive model to your cloud management system (e.g., AWS Lambda, Google Cloud Functions).  
- Ensure seamless communication between the prediction module and cloud APIs.

## Cloud Deployment:

## Resource Monitoring and Data Collection:

- Deploy monitoring agents on serverless functions.  
- Continuously collect real-time performance data.

## Prediction Module Deployment:

- Deploy the predictive model as a microservice or serverless function.  
- Ensure scalability and fault tolerance.

## API Gateway:

- Set up an API gateway to expose the prediction service.  
- Handle authentication, rate limiting, and request routing.

## Load Balancing:

- Distribute incoming requests across multiple instances of the prediction module.  
- Optimize resource utilization.

# PROJECT TESTING:

## Unit Testing:

- Test individual components (predictive model, API endpoints) in isolation.  
- Verify correctness and edge cases.

## Integration Testing:

- Test the entire system end-to-end.  
- Validate data flow, predictions, and resource allocation.

## Stress Testing:

- Simulate high loads (concurrent requests, varying workloads).  
- Assess system performance, scalability, and response times.

## Real-World Scenarios:

- Deploy the system in a production-like environment.  
- Evaluate accuracy, reliability, and user satisfaction.

# PERFORMANCE EVALUATION:

## Accuracy Metrics:

- Prediction Accuracy: Measure how well the system predicts the performance of serverless functions.  
- Common metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared.

## Resource Utilization:

- Analyze how efficiently serverless functions are utilized.  
- Calculate resource utilization ratios (execution time, memory usage) for each function.  
- Compare actual utilization with predicted utilization.

## Response Time and Latency:

- Measure the time taken to predict the performance of a function.  
- Evaluate system responsiveness during peak loads.

## Scalability:

- Assess how well the system scales with increasing workloads.  
- Test performance under various levels of concurrent requests.

## Real-World Testing:

- Deploy the system in a production-like environment.  
- Monitor performance during actual usage.  
- Gather feedback from users and stakeholders.

## Comparative Analysis:

- Compare the proposed predictive approach with other serverless management models.  
- Benchmark against existing serverless function optimization methods.

# CONCLUSION:

In the realm of cloud computing, serverless applications are pivotal for optimizing resource utilization and enhancing overall system performance. Through the utilization of advanced predictive models, we've addressed the challenge of managing serverless functions for specific tasks efficiently. By harnessing datasets containing pertinent features and target variables, our model learns patterns inherent in the data to make informed predictions. This predictive capability is instrumental in dynamically allocating resources to functions based on their requirements, thereby enhancing resource utilization and minimizing overhead costs. Moreover, the accuracy achieved underscores the reliability and efficacy of our approach. In conclusion, our efforts in serverless computing underscore the potential of predictive models to significantly improve the efficiency and performance of cloud computing infrastructures, paving the way for more agile and responsive cloud environments.