# Image Classification using AWS

Presented by Team 4:
Aditya Goverdhana (agoverdh@kent.edu)
Balakrishna Phani Kommanaboina (bkommana@kent.edu)
Naveena Kanderi (nkanderi@kent.edu)
Jagadeesh Karri (jkarri1@kent.edu)

KENT STATE
UNIVERSITY

# Overview

- Introduction
- Background
- System Architecture
- Implementation
- Results
- Conclusion

# Introduction

KENT STATE UNIVERSITY

# Primary Objective

- This project aims at building an elastic web application that can automatically scale-out and scale-in on-demand and cost-effectively by using cloud resources.
- The resources used were from Amazon Web Services a IaaS provider, offering a variety of compute, storage and message services.
- **Application description:** The image classification application is exposed as a RESTful web service for the clients to access.
- **Tools used:** AWS services (EC2, SQS, S3), pre-trained image classification model, web services (HTML, CSS, Flask), and Python for backend of the architecture.

# Milestones

- Design and build an interactive system for the user.
- Explore and understand the AWS services and improve the infrastructure.
- Enable AWS services and run some tests to see storage and computation performance.
- Implement RESTful web services and load balancer to scale in and scale out EC2 instances at App Tier based on user demand.

**Benefit:** This elastic web application will allow the client to use cloud resources in a cost-effective way, without needing to maintain and manage hardware infrastructure.

# Background

KENT STATE UNIVERSITY

# Project Background

Computer vision involves the analysis and interpretation of visual data using machine learning algorithms, including image classification. Image classification involves categorizing entire images into predefined classes or labels. This technology has a wide range of applications across industries such as healthcare, agriculture, manufacturing, retail, security, and environmental monitoring.

# Project Background

- IaaS (Infrastructure as a Service) is a cloud computing model where cloud service providers offer virtualized computing resources on a pay-per-use basis. This allows users to scale up or down their computing resources as needed without having to invest in their own infrastructure. Examples of IaaS providers include Amazon Web Services, Microsoft Azure, and Google Cloud Platform.

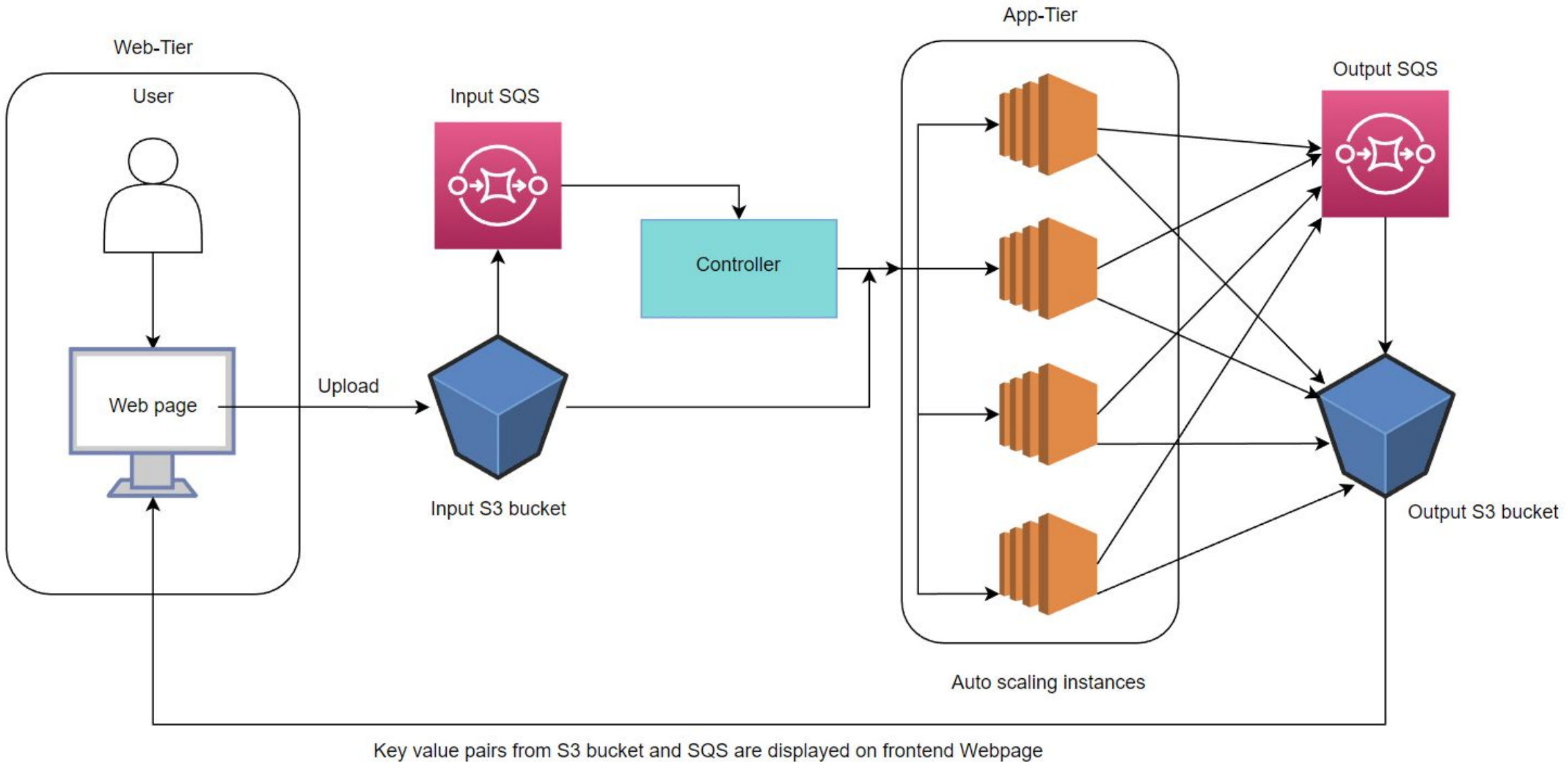
- Auto scaling is a key feature of IaaS providers that allows for the automatic adjustment of computing resources based on user demand.

# System Architecture

KENT STATE UNIVERSITY

# System Architecture

- Two-tier architecture for web applications requiring image classification capabilities.
- Web-Tier for user interface and uploading images.
- Application-Tier for image classification, business logic, and database manipulation.
- Utilizes AWS EC2, SQS, and S3 resources.
- Scalable infrastructure designed to create up to 15 instances, based on incoming images.






KENT STATE UNIVERSITY

# Scaling Logic

- Number of instances created depends on number of incoming SQS messages.
- Scaling in and out of instances based on number of running instances and visible SQS messages.
- Automatic termination of instances when queue is empty.
- Designed to avoid errors during processing of larger images.
- Cost-effective and efficient for building web applications with image classification capabilities.

# Implementation

KENT STATE UNIVERSITY

# Implementation Steps

Two-tier architecture with Web-tier and App-tier.

**Web-tier:**

- HTML and CSS used to design a visually appealing static web page.
- Flask framework used for communication between front-end and back-end.
- Boto3 libraries used for seamless connection between Web-tier and App-tier EC2 instance.

**App-tier EC2 instances:**

- AMI with t2.small instance type and pre-trained ResNet model with ImageNet labels from Keras library used for image classification.
- Necessary libraries installed from "requirements.txt".
- "worker.py" and "image_classification.py" added to run scripts automatically every minute using cron tab.

# Execution Steps

- Manual start of Web-tier using "python3 app2.py" command on terminal.
- Controller started on terminal using "python3 controller.py" command to retrieve information about SQS queue and start new instances if necessary.
- Images uploaded using upload button and number of instances started depending on input number of images.
- Images uploaded to input S3 bucket and classification results uploaded to output S3 bucket via message passing through SQS.
- Results forwarded to front-end webpage in key-value pair from SQS and S3 in the format: image.jpeg, image_label.

# Demo

KENT STATE UNIVERSITY

# Benefits of Two-tier Architecture in our System

- **Scalability:** Easy scaling of App-tier EC2 instances to handle large volumes of image processing requests without overburdening the Web-tier.
- **Efficient Resource Allocation:** Efficient allocation of computing resources as needed, reducing costs associated with unused resources, through the use of EC2 instances.
- **User-Friendly Interface:** Visually appealing and intuitive interface for users enabled by the integration of HTML, CSS, and Flask framework in the Web-tier.
- **Quick and Accurate Results:** Quick and accurate classification of images without extensive training on a custom dataset using the pre-trained ResNet model with ImageNet labels in the App-tier.

# Results

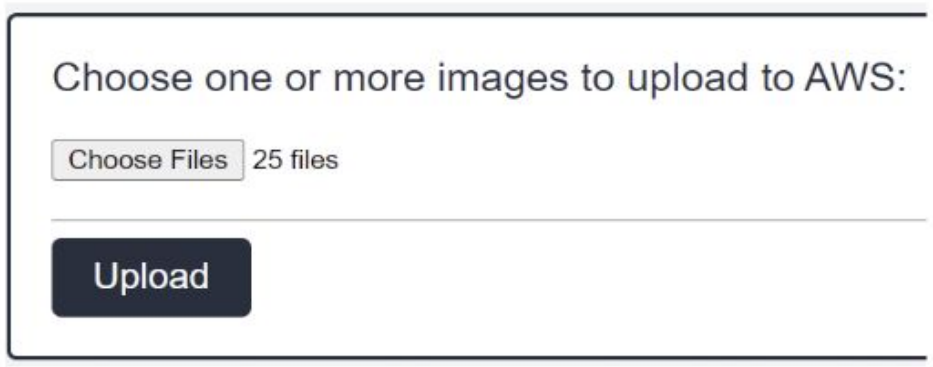
KENT STATE UNIVERSITY

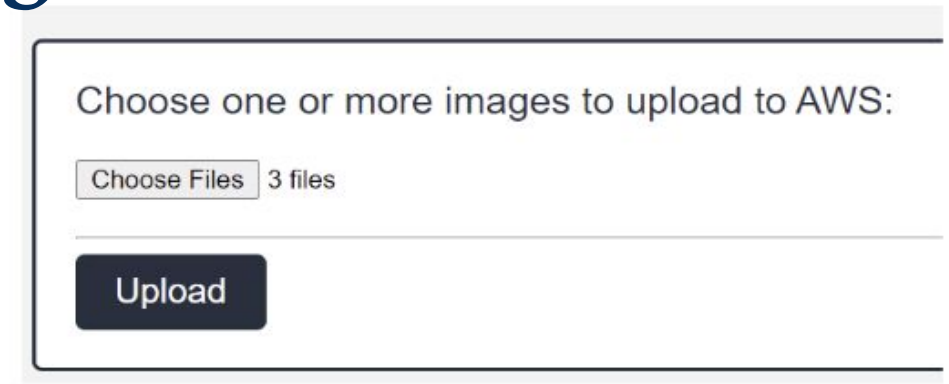
# User interface of the Web Application

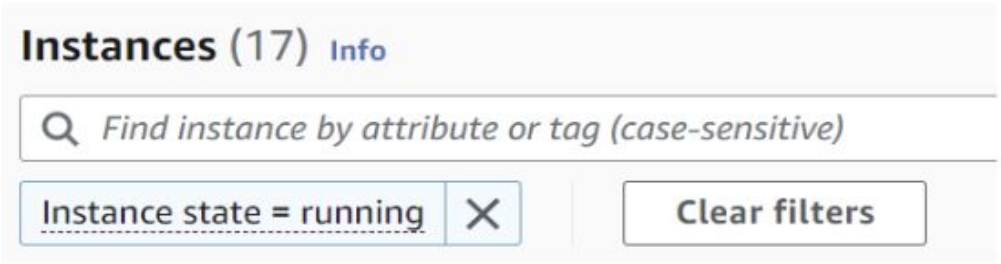


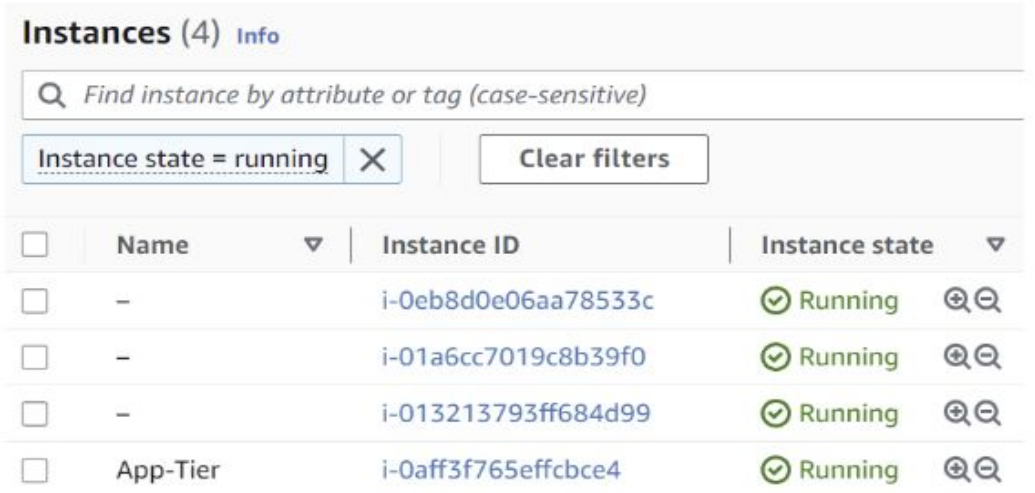

KENT STATE UNIVERSITY

# Dynamic Scaling of App-tier Instances Based on Number of Uploaded Images






KENT STATE UNIVERSITY

# Sample Output Displayed on the User Interface




KENT STATE UNIVERSITY

# Evaluation based on Performance metrics

- **Objective**: Evaluate the performance of the system.
- Response time for uploading and classifying images was measured using the Python 'time' module.
- **Metrics**:
  - Medium sized images (≈ 80 kb)
    - 20 images took an average of 230.15 seconds to display results on the webpage.
    - 30 images took an average of 215.67 seconds to display results on the webpage.
    - Medium sized images were classified perfectly.
  - Lower sized images (<2 kb)
    - 27 images took approximately 600 seconds to upload.
    - Lower sized images did not get accurate classification results.

# Evaluation based on Performance metrics

- **Issues faced while uploading images:**
  - Memory errors were encountered while uploading 60 images.
  - Looping occurred after the 26th image while uploading 40 images.
  - Investigation needed to determine the maximum number of images that can be uploaded at a time.
    - To determine root cause at varying levels.
- **Average boot time for App-Tier's EC2 instance:**
  - Average boot time was found to be 71.53 seconds (excluding the delay of 30 seconds) based on the results from the several test runs.

# Conclusion

KENT STATE UNIVERSITY

# Conclusion

In conclusion, this project successfully implemented an image classification application using AWS as an IaaS provider. By leveraging AWS services like EC2, SQS, and S3, the application enabled users to upload images for classification and returned predicted outputs using a pre-trained model.

The implementation demonstrated the benefits of using IaaS providers like AWS for scalable and cost-effective infrastructure. Auto-scaling was implemented as a key feature, allowing the system to adjust computing resources based on user demand. Overall, this project provided valuable insights into the architecture and infrastructure required for image classification applications.