Individual Contributions for Project 1:

- **Design**
    - The whole design and architecture of the application was progressively made based on the ideas of both me and Aakash. My contributions were majorly in the area of providing AWS resources for the application as well as implementation of some of the web-tier and app-tier features at coding level.
- **Implementation**
    - The application was developed using Java Spring Boot MVC from which web-tier was implemented in Spring Web and app-tier was implemented in Spring Command Line Runner. Among the coding portions, I implemented the main logic of persistence where we needed to store the input images received from the workload generator into the input S3 bucket in the web-tier as well as the output received from the classification algorithm which was needed to be stored in the output S3 bucket. Other than this, I implemented the main logic which calls the deep learning module (classification.py) and runs that algorithm on the image which was received as an input. The said image is received from the input S3 bucket after querying it using the image name which was retrieved from the input SQS queue. Also I created the logic which allows the app-tier instance to shut down automatically when it is unable to find any more inputs coming from the input SQS queue. All of the above were implemented in app-tier. In the web-tier, I helped in implementing part of autoscaling. I made the logic which allowed the web tier to take an integer as a parameter which is supposed to be the current input SQS queue size and use that number to create a number of instances which is capped at 15 to account for the instances that are either shutting down or in some other pending states. These were all of the coding contributions, other general contributions include creating AWS resources such as S3 buckets, SQS queues, custom AMI which included app-tier logic plus deep learning module, etc and creating the project report as well as giving the functional demo.
- **Testing**
    - I have thoroughly tested the functionalities at various phases of development. I performed load testing of the application by sending 100 image requests every time using both single and multi-threaded workload generators. The best times which were observed were 1 minute 57 seconds with a multi-threaded generator and 8 minutes 54 seconds with a single-threaded generator. With that I concluded that the application was working properly in good time limits.