**CSE 515: Multimedia and Web Databases**

**Phase 3**

**Group 17**Aaryan Gupta  
Fenil Madlani  
Krisha Gala  
Pranav Katariya  
Radhika Ganapathy  
Shivam Malviya

**Abstract**

In the final phase of this project, we perform image indexing using various clustering and classification algorithms such as Support Vector Machine, Personalized Page Rank and Decision Tree algorithms. Index structures were built using algorithms like Locality-Sensitive Hashing and VA-Files to carry out similar image search. In addition to this, we worked on relevance feedback systems to improve the nearest neighbour matches which helps account for the relevant and irrelevant results produced. Lastly, a query interface was built for the user to smoothly enter the query, retrieve results and give feedback which then produced revised results.

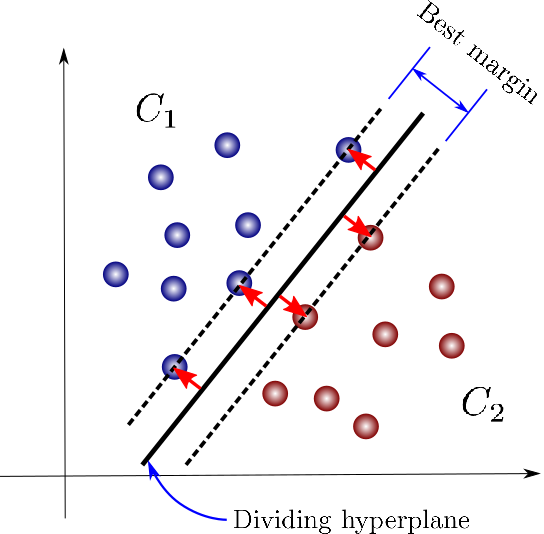
***Keywords—*** image features, image retrieval, indexing, vector models, classification, SVM, Decision Tree, Personalized Page Rank, clustering, LSH, VA Files, relevance feedback.

**1 Introduction**In the first phase of this project, we dealt with concepts of feature vectors where we extracted features based on models like Color moments, Extended Local Binary Patterns, and Histogram of Oriented Gradients to generate feature descriptors for images which were then used for comparing purposes using various similarity distance functions like the Euclidean distance, Manhattan Distance, etc.

In the second phase, we delved deeper into the concepts of multimedia retrieval where we addressed the issue of the dimensionality curse. [10] Dimensionality curse is when multimedia database systems cannot manage more than a handful of facets of the multimedia data simultaneously. We were given a dataset of facial grayscale images. The images were taken during different times, varying lighting and facial expressions. We studied and implemented the dimensionality reduction techniques like PCA, SVD, LDA, and K-Means on the given image dataset and also applied page ranking algorithms to the same.

In the final phase of this project, we study and implement classification algorithms like Support Vector Machines, Decision Trees and Personalized Page Rank. We also work with indexing algorithms like Locality-Sensitive Hashing and Vector Approximation Files to build indexing tools for efficient similar image retrieval. In addition to this we account for user feedback by building relevant feedback systems for the classification algorithms. Lastly, we build a query interface for the user to smoothly run the above-mentioned tasks.

* 1. **Terminology**
     1. **Support Vector Machines**Support-vector machines are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyse data for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). [1] It constructs a [hyperplane](https://en.wikipedia.org/wiki/Hyperplane) or set of hyperplanes in a [high-](https://en.wikipedia.org/wiki/High-dimensional_space) or infinite-dimensional space, which can be used for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis), or other tasks like outliers detection. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class.



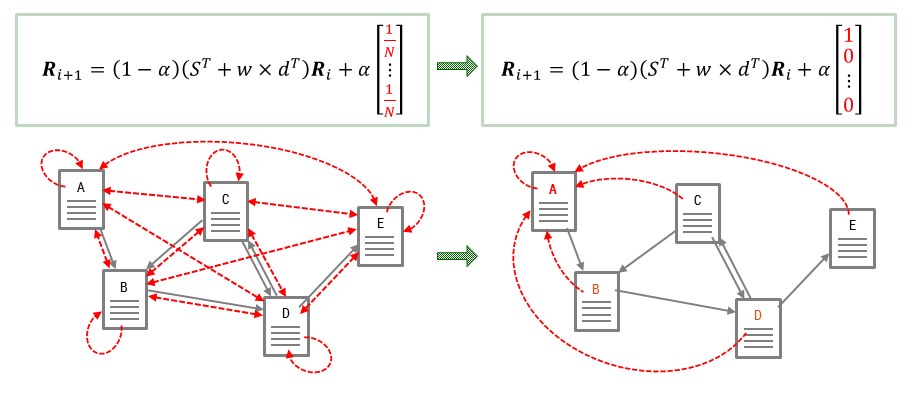
* + 1. **Decision Tree**

A Decision Tree is a is a hierarchal classification algorithm. It can be used for predicting categorical and continuous variables. Like [SVM](https://www.logic2020.com/insight/tactical/support-vector-machine-introduction), it can be used for regression or ranking as well.[2] There are two types of trees: classification decision trees and regression decision trees. The biggest advantage of decision trees is that they make it very easy to interpret and visualize nonlinear data patterns. Another advantage of classification decision trees is the possibility to improve their accuracy by setting the logic for the branches split.   
  
A decision tree consists of three types of nodes:  
-Decision nodes – typically represented by squares  
-Chance nodes – typically represented by circles  
-End nodes – typically represented by triangles



* + 1. **Personalized Page Rank**

Personalized PageRank (PPR) is a widely used node proximity measure in graph mining and network analysis. It is a standard tool for finding vertices in a graph that are most relevant to a query or user.[3] To personalize PageRank, one adjusts node weights or edge weights that determine teleport probabilities and transition probabilities in a random surfer model.



* + 1. **LSH**[Locality-Sensitive Hashing (LSH)](http://en.wikipedia.org/wiki/Locality-sensitive_hashing) is an algorithm for solving approximate/exact “Near Neighbour Search" in high dimensional spaces. LSHcan be referred to as a technique that hashes similar input items into the same “buckets” with a high probability using a hash function. It reduces the effect of dimensionality curse. [4] The key idea is to hash the points using several hash functions to ensure that for each function the probability of collision is much higher for objects that are close to each other than for those that are far apart. This will enable us to determine near neighbours by hashing the query point and retrieving elements stored in buckets containing that point.
    2. **Vector-Approximation Files**

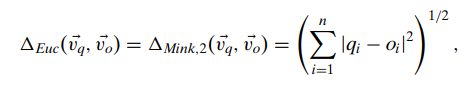
The Vector Approximation File approach based on uniform scalar quantization of feature vectors, and is a powerful technique that scales well with size and dimensionality of the data-set. VA-File partitions the space into hyper-rectangular cells, to obtained a quantized approximation for the data that reside inside the cells. Non-empty cell locations are encoded into bit strings and stored in a separate approximation file, on the hard-disk. During a nearest neighbour search, the vector approximation file is sequentially scanned and upper and lower bounds on the distance from the query vector to each cell are estimated. The bounds are used to prune irrelevant cells. The final set of candidate vectors are then read from the hard disk and the exact nearest neighbours are determined.[7]

* + 1. **Relevance Feedback**

Relevance feedback is a feature of some information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query, to gather user feedback, and to use information about whether or not those results are relevant to perform a new query.[8] Accordingly results are modified and returned after taking feedback into consideration.

* + 1. **Euclidean Distance**

The Euclidean distance is commonly used for measuring distances between points in the 3D space we are living in. It is in fact the Minkowski distance of order 2. This distance metric is most commonly used for similarity measurement in [image retrieval](https://www.sciencedirect.com/topics/computer-science/image-retrieval) because of its efficiency and effectiveness. It measures the distance between two vectors of images by calculating the square root of the sum of the squared absolute differences.



* 1. **Goal Description**

**Task 1:**

In this task, we have to implement a program in which we are given a folder of images, one of the three feature models, and a user specified value of k. We have to compute k latent semantics (if not already computed and stored), and given a second folder of images, it should associate **X** labels to the images in the second folder using the classifier selected by the user. The classifiers to be implemented are - SVM classifier, Decision-Tree classifier, Personalized Page Rank. We also have to compute and print false positive and miss rates.

**Task 2:**

In this task, we have to implement a program in which we are given a folder of images, one of the three feature models, and a user specified value of k. We have to compute k latent semantics (if not already computed and stored), and given a second folder of images, it should associate **Y** labels to each image in the second folder using the classifier selected by the user. The classifiers to be implemented are - SVM classifier, Decision-Tree classifier, Personalized Page Rank. We also have to compute and print false positive and miss rates.

**Task 3:**

In this task, we have to implement a program in which we are given a folder of images, one of the three feature models, and a user specified value of k. We have to compute k latent semantics (if not already computed and stored), and given a second folder of images, it should associate **Z** labels to each image in the second folder using the classifier selected by the user. The classifiers to be implemented are - SVM classifier, Decision-Tree classifier, Personalized Page Rank. We also have to compute and print false positive and miss rates.

**Task 4:**

In this task, we have to implement a Locality Sensitive Hashing (LSH) tool, which takes as input the number of layers (L), the number of hashes per layer (κ) and a set of vectors (generated by other tasks) and creates an in-memory index structure containing the given set of vectors. In addition to this, we have to implement similar image search using this index structure. Given a folder of images and one of the three feature models, the images are stored in an LSH data structure (the program also outputs the size of the index structure in bytes). For any given image and “t”, the index tool returns the “t” most similar images. The program should also return the number of buckets searched, the unique and overall number of images considered, false positive and miss rates.

**Task 5:**

In this task we have to implement a VA-file index tool and conduct nearest neighbour search operations. We will be given a parameter “b” denoting the number of bits per dimensions used for compressing the vector data and a set of vectors (generated by other tasks) as input. The program creates an in-memory index structure containing the indexes of the given set of vectors. The program also returns the size of the index structure in bytes. We then have to implement similar image search using this index structure. Given a folder of images and one of the three feature models, the images will be stored in a VA-file data structure (the program also outputs the size of the index structure in bytes). For any given image and “t”, the tool returns the “t” most similar images. The program should also return the number of buckets searched, the unique and overall number of images considered, false positive and miss rates.

**Task 6:**

In this task we have to implement a Decision-Tree based relevance feedback system to improve nearest neighbour matches. This enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results.

**Task 7:**

In this task we have to implement a SVM based relevance feedback system to improve nearest neighbour matches. This enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results.

**Task 8:**

In this task we have implement a query interface. This allows the user to provide a query and relevant query parameters (including how many results to be returned). Query results are presented to the user in decreasing order of matching. The result interface should also allow the user to provide positive and/or negative feedback for the ranked results returned by the system. User feedback is then taken into account and a new set of ranked results are returned.

* 1. **Assumptions**
* Datasets provided are processed and can be used without further computation.
* Datasets given are consistent in terms of image ID, subject and type labels.
* All images to be used as input should be given in a folder labelled “images”.
* The query image should be provided in the format folder-name/image-name.
* The latent semantic files obtained in the previous phase for the various functions of PCA, SVD and LDA are correct.
* In the implementation of SVM, the number of clusters should be greater than equal to 2, to carry out classification task.
* Feedback given in Task 6 and Task 7 should be greater than 0 to take into account irrelevance/relevance feedback
* All user inputs are in accordance with the task input formats mention in the execution section.

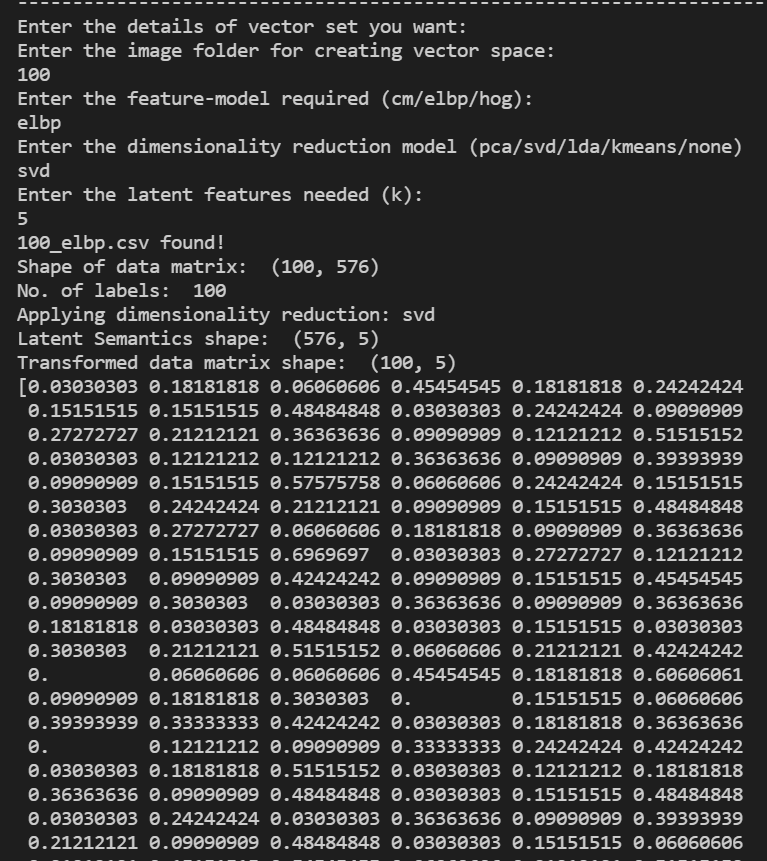
1. **Description of Proposed Solution and Implementation**

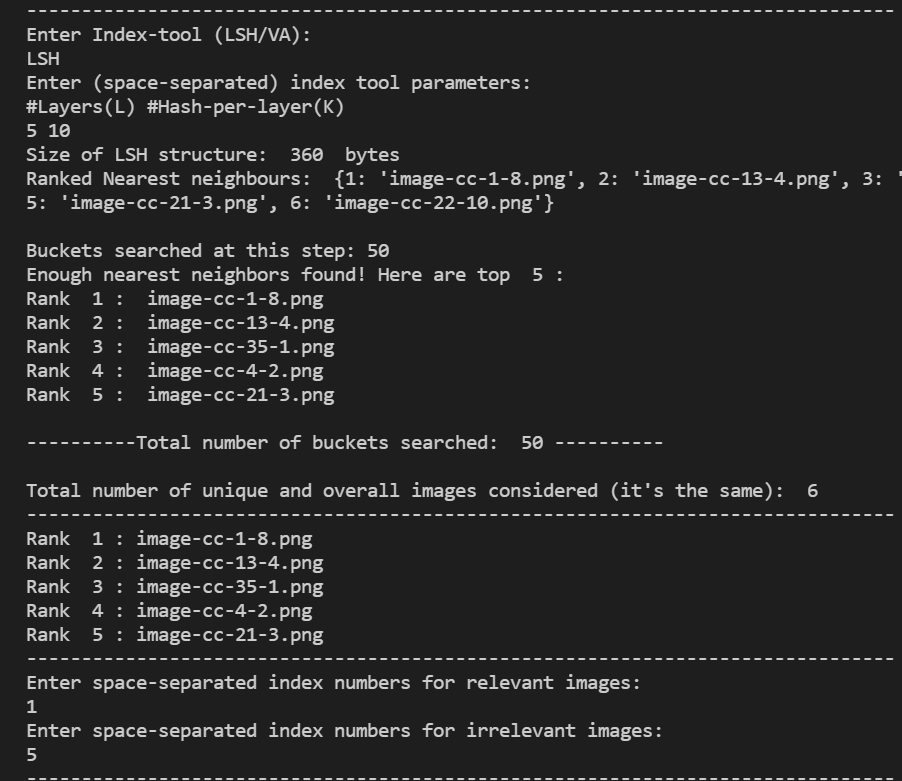
**Task 1  
Task 2  
Task 3**

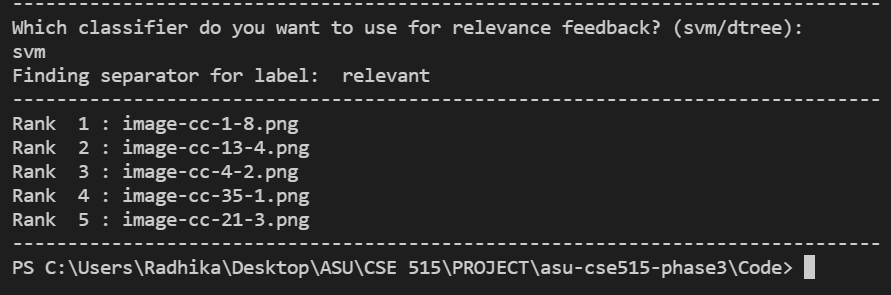
**Task 4**

In task 4, we are given the number of bits, image folder, feature model and k as input. Based on the number of bits given by the user, we partition each dimension into 2^b+1 partition points. For every image in the given folder, we check whether the given dimension of the image lies in a particular region and accordingly assign a binary value to it. Similarly, each dimension of the image is represented in the form of binary bits which forms the vector approximation file for that image. We then compute the VA file for the given query image. To find the “k” nearest images we use the VA-SSA algorithm which finds the lower bound for each image. We then calculate L3 norm distance between the lower bound of the images in the folder and our query image and return top k images in the filtered region.

**Task 5  
Task 6  
Task 7   
Task 8**







1. **Interface Specification**

This phase of the project uses the basic command line interface for taking the inputs and displaying outputs of every task. The inputs are passed as command line arguments to each task as specified in the next section under execution instructions. The outputs are displayed in stdout and screenshots of these outputs are displayed in the section above. All the task files have to be run using python 3.6 or above.

1. **System Requirements and Execution Instructions**

In this phase of the project, we use Python as our programming language and MongoDB to store our database. In addition to this, we utilize several libraries such as NumPy, Sys, PIL, Pymongo, sklearn, SciPy, etc. to carry out the given tasks.

* 1. **Environment Set up**

We have used the PyCharm IDE. PyCharm is a programming integrated development environment that focuses on the Python programming language. Make sure Python 3.6 or higher version of Python is downloaded in your system. Based on the operating system of the computer, the corresponding version of PyCharm or a similar IDE can be easily downloaded from the website.

Ensure that you install all the necessary packages. Some of the packages installed are listed below:

NumPy - A package providing high-performance multidimensional array object

Sys - The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment.

OS - The OS module in Python provides functions for interacting with the operating system.

PyMongo - The PyMongo library allows interaction with the MongoDB database through Python.

PIL - Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different images of different formats.

Sklearn - An open-source library providing efficient tools for various classification, decomposition, regression, and clustering algorithms.

* 1. **Execution**

Run the commands in the following manner for each task, changing the command-line arguments as needed.

Task 1: python p3\_task1.py <train\_folder><feature\_model><k><test\_folder>< classifier>  
*Eg: python p3\_task1.py 100 elbp 30 100 dtree*

Task 2: python p3\_task2.py<train\_folder><feature\_model><k><test\_folder>< classifier>  
*Eg: python p3\_task2.py 100 elbp 30 100 dtree*

Task 3: python p3\_task3.py<train\_folder><feature\_model><k><test\_folder>< classifier>  
*Eg: python p3\_task3.py 100 elbp 30 100 dtree*

Task 4: python p3\_task4.py <n-layers><hash-per-layer><image-folder><feature><query>  
*Eg: python p3\_task3.py 5 10 100 svm 5*

Task 5: python p3\_task5.py <bits><image-folder><feature-model><k>  
*Eg: python p3\_task3.py 5 10 100 svm 5*

Task 6, Task 7, Task 8: python p3\_task8.py   
*Eg: python task8.py*

1. **Related Work**

In [4], authors Alexandr Andoni and Piotr Indyk give an overview of efficient algorithms for the approximate exact near neighbour search problem. They survey a family of nearest neighbour algorithms that are based on the concept of Locality-Sensitive Hashing. They also describe a recently discovered hashing-based algorithm, for the case where the objects are points in the d-dimensional Euclidean space. In the end, they prove that the performance of the newly discovered algorithm is near-optimal in the class of the locality-sensitive hashing algorithms.

In [5], authors Stephen Blott and Roger Weber introduce the algorithm of Vector-Approximation file (VA-file) for similarity search in high dimensional vector spaces. They discuss how this method overcomes the dimensionality curse by adopting a filter-based approach of signature files instead of the traditional method of data partitioning. They go onto evaluating the performance of VA files on the basis of real and semi-synthetic vector data characterizing colour features of a moderate-sized image database. For large databases, the VA-File outperforms a well-tuned scan by a factor of up to four. They have also shown that performance does not degrade, and how it improves slightly with dimensionality. It is possible to integrate search in VA-File structures over multiple vector spaces, and also over signature files for non-metric spaces. Such searches are necessary to support conjunctive queries, which are frequent in multimedia databases. Further, they conclude that both the VA-File and signature methods are relatively straight-forward to parallelize and distribute.

In [6], the authors studied the impact of dimensionality on the nearest-neighbour similarity-search in high dimensional vector spaces from a theoretical and practical point of view. Under the assumption of uniformity and independence, they established lower bounds on the average performance of Near Neighbour-Search for space, data-partitioning, and clustering structures. They prove that these methods are out-performed by a simple sequential scan at moderate dimensionality (i.e. d = 10). Further, they have shown that any partitioning scheme and clustering technique must degenerate to a sequential scan through all their blocks if the number of dimensions is sufficiently large. Experiments with synthetic and real data were conducted to show that the performance of R\*-trees and X-trees are outperformed by a sequential scan if dimensionality becomes large. They go on to conclude that at moderate and high dimensionality (d > 6), the VA-File method can out-perform any other method and that performance for this method even improves as dimensionality increases.

In [7], author Poonam proposed an overview of an efficient indexing method for high-dimensional databases using a filtering approach known as vector approximation approach which supports the nearest neighbour search efficiently. A cluster distance bound based on separating hyper planes, that complements the index in electively retrieving clusters that contain data entries closest to the query. In high dimensional, data-sets exhibit significant correlations and non-uniform distributions. Hence, indexing with the VA-File, by performing uniform, scalar quantization, is suboptimal. She then went on to propose an indexing method, based upon principles of vector quantization instead, where the data set is partitioned into Voronoi clusters and the clusters are accessed in order of the query-cluster distances. The cluster-distance bounds can be tightened by optimizing the clustering algorithm so as to optimize the cluster distance bounds.

1. **Conclusions**
2. **Bibliography**

[1] <https://en.wikipedia.org/wiki/Support-vector_machine#Definition>  
  
[2] <https://www.logic2020.com/insight/tactical/decision-tree-classifier-overview>  
  
[3] <https://changuk.github.io/algorithm/2013/08/28/pagerank.html>  
  
[4] Andoni, Alexandr & Indyk, Piotr. (2008). Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. Commun. ACM. 51. 117-122. 10.1145/1327452.1327494.  
  
[5] Blott, Stephen & Weber, Roger. (1998). A Simple Vector-Approximation File for Similarity Search in High-Dimensional Vector Spaces.  
  
[6] Weber, R., Schek, H., & Blott, S. (1998). A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. *VLDB*.  
  
[7] Sahu, Mridu and Poonam Yerpude. “Vector Approximation File: Cluster Bounding in High-Dimension Data Set.” (2011).  
  
[8] https://en.wikipedia.org/wiki/Relevance\_feedback  
  
[9] Elmore, Kimberly L., and Michael B. Richman. "Euclidean distance as a similarity metric for principal component analysis." Monthly weather review 129.3 (2001): 540-549.  
  
[10] K. Seluk Candan and Maria Luisa Sapino. Data Management for Multimedia Retrieval. Cambridge University Press, New York, NY, USA, 2010

1. **Appendix**

|  |  |
| --- | --- |
| **Group Member** | **Contributions** |
| Aaryan Gupta |  |
| Fenil Madlani |  |
| Krisha Vijay Gala |  |
| Pranav Katariya |  |
| Radhika Ganapathy |  |
| Shivam Malviya |  |