Visage Verify

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

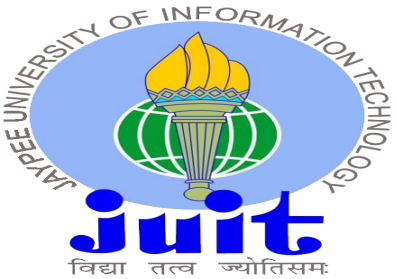
Submitted by

DEVANSH BATRA(201447)

MALAY ACHARYA(201476)

Under the guidance & supervision of

Mr. Maneet Singh



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Visage Verify"** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Devansh Batra & Malay Acharya with Roll Number 201447 & 201476 respectively during the period from August 2023 to December 2023 under the supervision **Mr. Maneet Singh** (Assistant Professor)

Student Name: Devansh Batra
Roll No.: 201447

Student Name: Malay Acharya
Roll No.: 201476

The above statement made is correct to the best of my knowledge.

Supervisor Name:Mr. Maneet Singh
Designation: Assistant Professor (Grade - II)
Department: Computer Science and Engineering and Information Technology

DECLARATION

I hereby declare that the work presented in this report entitled **'VISAGE VERIFY'** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to December 2023 under the supervision of **Mr. Maneet Singh** (Assistant Professor (Grade-II), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Devansh Batra
Roll No.: 201447

Student Name: Malay Acharya
Roll No.: 201476

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Mr. Maneet Singh
Designation: Assistant Professor (Grade - II)
Department: CSE & IT
Dated: 1-12-2023

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are really grateful and wish my profound indebtedness to Supervisor **Mr.Maneet Singh, Assistant Professor**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to **Mr .Maneet Singh,** Associate Professor, Department of CSE, for his kind help to finish this project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support of our parents.

Devansh Batra, 201447
Malay Acharya, 201476

Table Of Contents

ABSTRACT

VisageVerify, a pioneering mobile application developed in React Native, is poised to transform attendance recording in educational settings. This novel solution seamlessly integrates Flask, Redis, Azure Virtual Machines, Nginx, and powerful machine learning/deep learning techniques to boost speed, accuracy, and accessibility in the traditional process of recording student attendance.

VisageVerify leverages Flask as the backend framework, a choice inspired by its lightweight nature and extensibility. Flask serves as the sturdy foundation for the server-side logic, effortlessly handling data processing and communication between the front-end and other components of the system. Its simplicity and flexibility make it a perfect platform for UniTrack, allowing for rapid development and easy connection with other technologies. With Flask, the backend of UniTrack is not only responsive and scalable but also enables the installation of custom routes and endpoints to respond to the specific needs of attendance tracking and data administration.

Azure Virtual Machines (VMs) comprise the sturdy hosting backbone of the **VisageVerify** attendance recording system. Leveraging the scalability and reliability of Azure VMs guarantees that UniTrack can handle shifting workloads effectively, adjusting to the dynamic demands of a university setting.

Chapter 1

Introduction

1.1 Introduction

In the dynamic domain of higher education, the convergence of pedagogy and technology is important, and our product stands as a beacon of innovation. We propose an all-encompassing attendance recording tool meticulously built to fulfil the particular needs of modern universities. Recognizing the deep link between student involvement and administrative efficiency, our initiative intends to change conventional attendance tracking, ushering in an era of seamless integration, precision, and heightened security.

Our product stands out as an innovative example in the fast-paced world of higher education, where the pedagogical and technological nexus is crucial. It is an all-inclusive attendance recording application that has been painstakingly designed to satisfy the particular requirements of contemporary universities. Understanding the complex relationship that exists between student participation and administrative efficiency, our project aims to transform the conventional approaches to attendance tracking and usher in a new era of harmonic integration, precision, and security.

1.1.1 Technological Foundation

Our project is based on the smart combination of React Native and Flask, two great technologies woven with professionalism into a dynamic and user-responsive system. The reason behind this is because cross-platform power of react native creates uniform and eye catching experience across different devices and flask establishes a solid backbone from backend architecture due to its flexibility and robustness . Synergy ensures elegance for backend that copes with complexity of attendance tracking and elegant UI.

1.1.2 Facial Recognition with OpenCV

Our project is unique in that it uses OpenCV for facial recognition, a breakthrough in technology that goes beyond traditional attendance tracking. OpenCV, a mainstay in the field of computer vision, improves our application's precision and dependability. Beyond the practical benefits, facial recognition adds an unparalleled level of security, making attendance a smooth and yet extremely safe operation. Being acknowledged is now simple for both teachers and students, making attendance quick and individualised.

1.1.3 Cloud-Powered Performance

In the age of cloud computing, our app manages a complex dance of technologies in addition to taking attendance. An Azure Virtual Machine serves as the home base for the machine learning model, which is the brains behind facial recognition. This calculated decision guarantees both peak performance and flexibility in response to changing attendance

processing requirements. Our programme can now offer results in real time thanks to the ML model hosted on Azure, guaranteeing fast and accurate attendance data.

1.1.4 Enhanced Fluidity with Redis

We used Redis, a powerful data caching technology, to improve the application's fluidity. Redis integration maximises data retrieval, speeding up the process of registering attendance while also reducing resource consumption. This clever caching solution not only increases productivity but also sets the stage for a responsive system with rapid access to attendance data and a new level of user experience.

1.1.5 Reliable Scalable Infrastructure

The integrity of our product requires it to be hosted on a reliable, flexible infrastructure. The cloud infrastructure basis of Amazon web services underpins our project. The choice provides a flexible and resistant infrastructure for universities that can safely work with attendance data on the cloud. The outcome is an environment where the expansion and safety of attendance tracking seamlessly interoperate with the suite of other AWS capabilities.

1.2 Problem Statement

1.2.1 Inefficiencies in Traditional Attendance Tracking

Attending college has always been a problematic affair considering it mainly relied on manual operations that took too much time and money. The outdated systems are prone to mistakes hence leaving all the administrators as well as educators struggling with faulty information. There is an obvious demand for a less cumbersome and computerised system of recording attendance in the era of modern education, which ought to be developed.

1.2.2 Security and Precision Challenges

The common techniques of attendance tracking lack the sophistication required to fulfil the rising security and precision needs of educational institutions. With worries developing regarding the integrity of attendance data and the vulnerability of manual processes to fraudulent activities, there is an urgent need for a system that not only ensures accuracy but also boosts security. Current technologies sometimes fall short in delivering a strong solution that can respond to the unique demands of varied university environments, demanding a comprehensive and technologically advanced approach to attendance management.

1.2.3 Integration Challenges in Technological Landscapes

Technology meets education comes with both advantages and disadvantages. To this end, there is a huge challenge in implementing different technologies as colleges try to develop new strategies. This lack of an integrated system that connects different means of attendance tracking makes the processes fragmented and weak. Universities must bridge this gap in technology in order to exploit the merits of emerging tech, but in a way that remains easy to use for both the teacher and student.

1.2.4 Data Accessibility and Decision Making Bottlenecks

The standard attendance tracking solutions can cause obstacles in data accessibility and decision-making. Educators and administrators suffer with delayed access to attendance data, impeding their capacity to make prompt and informed decisions. A solution that gives real-time data, paired with analytical insights, is crucial to equip universities to proactively solve attendance-related concerns, maximise resources, and enhance the entire educational experience

1.2.5 Adapting to Evolving Educational Landscapes

The rapid growth of educational techniques and the rising diversity in learning contexts bring significant problems to traditional attendance tracking systems. With institutions embracing blended learning methods, meeting varying attendance requirements becomes complicated. An innovative solution must be agile enough to adapt to these developing educational settings, assuring its relevance and efficacy across a spectrum of academic scenarios.

In tackling these problems, our attendance recording application strives to change the paradigm of university attendance management, giving a complete, secure, and technologically sophisticated solution suited to satisfy the specific needs of contemporary higher education institutions.

1.3 Objectives

In response to the various issues faced by modern institutions in attendance tracking, our purpose is to develop a revolutionary mobile application that transforms the landscape of university attendance management. Rooted in the fusion of innovative technologies, our solution strives to eliminate the inefficiencies of old attendance systems and raise the entire experience for educators, administrators, and students alike.

1.3.1 Efficiency Enhancement

Our primary purpose is to boost the efficiency of attendance tracking at institutions. By replacing manual processes with a smooth, automated solution, we hope to reduce the time and resources spent on attendance recording. The application will give educators a user-friendly interface, allowing for rapid and accurate attendance management, eventually optimising their time and enabling a more concentrated teaching experience.

1.3.2 Precision and Security

We intend to create new benchmarks in attendance precision and security by combining facial recognition technology utilising OpenCV. This new approach not only provides a higher level of accuracy in tracking attendance but also introduces an unequalled layer of protection. The programme will enable quick and individualised recognition, simplifying the attendance procedure for both teachers and students while ensuring the integrity of the data.

1.3.3 Technological Integration

It is our mission to seamlessly integrate technologies like the React Native and Flask in making a robust and reliable framework. By using React Native, which is cross-platform, you

will obtain uniform experience for end users on different gadgets. At the same time, Flask can be considered as an adaptable and scalable backend. This attempt to plug the technology gap offers a universal remedy that resonates with the current goals of today's higher education institutions.

1.3.4 Real-Time Data and Analytics

We aspire to empower educators and administrators with real-time attendance data and relevant analytics. Leveraging the power of Azure for hosting machine learning models, our application will give rapid and accurate attendance reports. This real-time functionality ensures that decision-makers have immediate access to attendance insights, allowing for proactive measures in addressing challenges and optimising resource allocation.

1.3.5 Enhanced User Experience

Our purpose is to enhance the overall user experience by employing Redis for data caching. This strong technology will improve application fluidity, maximising data retrieval speed and decreasing resource consumption. The result is a responsive system with instant access to attendance data, contributing to an upgraded user experience for both instructors and students.

1.3.6 Scalable and Secure Infrastructure

We seek to construct a robust and extendable solution by employing the cloud basis of Amazon Web Services (AWS). This solution guarantees a flexible and resilient architecture, enabling colleges to securely handle attendance data in the cloud. Our purpose is to establish an ecosystem where attendance tracking's scalability and security smoothly connect with the broader range of functions offered by AWS.

1.4 Significance and Motivation of Project Work

In fact, our attendance recording application is indispensable in a changing setting in today's higher education. The problem of inefficiency and riskiness had plagued university attendance tracking for many years. We, therefore, have a revolutionary response that not only overcomes existing challenges but also initiates technology, productivity, and security in modern educational institutions.

Significance:

1. Streamlining Administrative Processes: This application illustrates an exit from labour intensive and inaccurate manual attendance checks. Institutions also can automate the critical administrative process of payment so that they focus on significant academic objectives, vital for the institution.
2. Enhancing Educator Focus: In a nutshell, the significance of the project resides in granting teachers an opportunity to concentrate on key issues such as teaching. Simplification of a teacher-based attendance system enhances teachers' concentration

on lesson delivery, interacts with students, formulates lesson plans, and mentors students in order to promote a fun learning environment.

3. Security and Integrity: With respect to facial recognition, this technology solves the problem of security and validity in attendance data. It is a quantum leap in technological sophistication. In addition, it guarantees correct information and counterfeiting.
4. Technological Integration: The use of technologies like React Native, Flask, OpenCV Azure, and Redis is a deliberate move toward an all-encompassing outcome. First, colleges trying to change pedagogy and technology have to be in harmony via this fusion.

Motivation:

1. Meeting Contemporary issues: Our project is driven by the realisation of the pertinent problems in most colleges today. This new system must be innovative and dynamic enough for the changing landscape of contemporary education which entails blended learning models and flexible attendance requirements as well as immediate access to live data.
2. Empowering Educators and Students: Empowerment of both teachers and learners' have been at the heart of our passion. Our programme aims at offering a user friendly interface and up to date information that provides meaningful participation rather than only being a bureaucratic procedure.
3. Pushing Technological Frontiers: It shows keenness in using current technologies like face recognition, machine learning and cloud computing in taking the education technology frontier even further. This is an example on how technology can improve critical aspects of school administration.
4. Striving for perfection: In essence, our drive to undertake such an exercise was motivated by a thirst for being perfect. Our aim is to develop a new level of attendance tracking including quality of service and convenience to our users. This spurs us towards breaking borders and developing a solution that defies traditional laws.

To conclude then, our project is about more than simply marking attendance; it represents a commitment to changing education. Solving current problems, adopting technological innovations, and being committed to our excellence positions our project to have a lasting impact on the way college registration will be done in future. Hello to a new age of efficiency, safety, and creativity among universities.

1.5 Organization of Project Report

1.5.1 Chapter 1: Introduction

Chapter one on the project provides the introduction that entails the concept and the objective of the project. It is also a short summary of the main issues of the time table of the projects as well as its duration.

1.5.2 Chapter 2: Literature Survey

Papers and Journals for reputable sources for React Native, OpenCV, and Warehousing & Cloud Infrastructure. It critically reviews the available literature and identifies future research directions that will fill noted gaps and improve future studies on this construct.

1.5.3 Chapter 3: System Development

Chapter three discusses the important issues regarding the infrastructure of the system such as the hardware, software, and network. The document also gives the organisation's main functions as pseudo code and the database structure flowchart.

1.5.4 Chapter 4: Testing

The testing details of the system are covered in this chapter, including the approach, test specs, and test outcomes.

1.5.5 Chapter 5: Results and Evaluation

This chapter offers early outputs from the system while drawing user views. Additionally, it talks about the difficulties experienced in developing the system as well as the hard earned lessons.

1.5.6 Chapter 6: Conclusions and Future Scope

The final chapter outlines the findings of the project as well as the extent of future aspects with regard to the project. It also stipulates the development committee whose responsibility will be the continuous development of the system.

Chapter 2

Literature Survey

2.1 Overview of Relevant Literature

[1]

Title	React-Native Based Mobile App for Online Experimentation
Author	Xingwei Zhou, Wenshan Hu, Guo-Ping Liu
Year	2020
Summary	With respect to current trends, smartphones and tablet PCs are now widely used which results in high demand for mobile apps providing remote data experiments. React Native is an increasingly common cross-platform mobile app development framework which can help build mobile apps for both iOs and Android based devices. The concept and realisation of an electronic mobile application for online experiments written in React-Native. It enables an easy way in which people can use online and do experiments. Design This app has been designed simply to suit even the basic users' needs. The app's key features include: User login and registration: They can also have users register to save and manage this experimentation information. Experiment selection: Users get to choose experiments from a menu list. Experiment observation: They also enable users to track their experiences. Experiment documentation: Users can see experiment documentation such as descriptions of their experiments processes and results. Algorithm download and configuration: The users may download and write algorithms that suit their research. Experiment monitoring: This enables users to track their experiment's progress as well as sample the results.

[2]

Title	React Native for Android: Cross-Platform Mobile Application Development
Author	Sreekanth Dekkati, Karu Lal,Harshith Desamsetti
Year	2022
Summary	Given this background, React Native is one such emerging platform that offers an alternative method of developing robust apps that work across varied operating systems (iOS and Android) simultaneously. The specifics of React Native for Android development is discussed in this paper with close analysis of its vital elements, strengths, and weaknesses. Key Findings Developers are drawn to react native due to its declarative style that makes code easy to write and maintain. Using JavaScript, which is a popular programming language, makes React Native more accessible and easy-to-use. Such component-oriented structure of the framework implies code reuse and modularity, speeds up the development. The hot reloading feature of React Native allows for instant updates that speed up code debugging and iteration. It interacts with the native Android modules and allows access to some platform specific capabilities.

[3]

Title	Geolocation APIs for React Native
Author	Michal Chudziak
Year	2018
Summary	The following article summarises different geolocation APIs that can be used in React Native. It covers native Geolocation API and React Native libraries such as react-native-geolocation-service and React Native Background Geolocation. Key Findings: Simple location purposes can be facilitated by an effective native Geolocation API. In addition, third party libraries like React Native Geolocation Service and React Native Background Geolocation also provide more advanced functionalities like background geolocation and geofencing. The selection of a geolocation API depends on the specific needs of the application. Conclusion: The choice of geolocation APIs is vast for React Native developers, all of which come with pros and cons. Of course, it depends on the specific requirement of the app.

[4]

Title	React Native Geolocation: A Complete Tutorial
Author	LogRocket
Year	2022
Summary	React Native Geolocation: The complete tutorial This article gives a holistic view of geolocation in React Native which involves setting the development environment and using the Geolocation API in order to access the exact location of any person. It also has a separate part that gives recommendations on utilising various libraries to boost geolocation functions. Key Findings: The React Native geolocation api offers an easy way of acquiring user's location data with minimal efforts on its part. Some third party libraries can add to these geolocation functions, for example react-native-background-geolocation and react-native-geolocation-service. The technology has various purposes, like pointing user's placement on a map, providing locational services and detecting user's moves. Conclusion Mobile app, built around React Native, is very helpful for teachers and software engineers who wish to test their online experiments with any mobile device. It has a simple design, it adapts well, and it accommodates multiple types of experiments. The further maturity of the app can enrich its functions and popularise it.

[5]

Title	Designing and Implementing RESTful APIs with Flask
Author	Daniel Quimper
Year	2022
Summary	This is a full guide on how you can build and utilise RESTful apis using flask – a widely used python web framework. The article covers a wide range of issues, including: RESTful API design concepts: This paper considers the primary considerations for designing a RESTful API which include resource-oriented URIs, standardised interfaces, and stateless protocol. Flask API development techniques: This paper shows how to design Flask API, create routes, process requests as well as provide results. Best practices for designing strong and scalable APIs: Strong and Scalable API Design Practices such as Error Handling, Documentation and Versioning described in this paper. Key Findings However, in recent years, Flask has emerged as one of the most light-weighted and flexible frameworks used for developing RESTful APIs. Adherence to RESTful API design principles can help create APIs that are simple, manageable and scalable. The design of appropriate documents as well as dealing with errors in order to produce user friendly APIs is important. Conclusion It is of great help to software engineers desiring to explore how to design and apply RESTful Applications Programming Interfaces (APIs) on Flask. The paper provides a straightforward outline of the basic principles and methods applied during the RESTful API creation including a couple of case studies.

[6]

Title	Redis++: A High Performance In-Memory Database Based on Segmented Memory Management and Two-Level Hash Index
Author	Hongli Li, Jun Zhao, Yongguang Xu, and Yuan Zhang
Year	2018
Summary	This paper proposes Redis++, a high-performance in-memory database that tackles two fundamental drawbacks of Redis: memory fragmentation and cache misses. That is why Redis++ uses the segmental memory management strategy to assign and remove the fixed size memory blocks, and in turn minimises memory fragmentation. It also uses a two level hash index structure that greatly reduces cache misses and speeds up queries. Key Findings: Redis++ has overcome the two major performance killers that include memory fragmentation and cache misses of Redis. This segmented memory management system is able to distribute the available memory space and release it in the best possible manner, thereby avoiding memory waste. A two-level hash index structure helps to evade cache misses and enhances query performance. Memory efficiency, as well as lower latency response and throughput, makes Redis++ better than Redis. Conclusion: To conclude, Redis++ comes out as a potential candidate to build advanced high-speed in-memory databases. It does this because of the performance measures that this has made it possible to become a replacement for Redis in some cases because of how effective this has been in handling memory fragmentation and cache misses.

[7]

Title	Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis
Author	Hao Zhang, Haixia Zhang, Qingquan Zhang
Year	2022
Summary	This paper explores scalability and reliability aspects concerning a popular in-memory key-value database known as Redis. Recognizing the limits of Redis in managing large-scale data and guaranteeing data integrity, the authors propose two techniques: Client Side Key-to-Node Caching and Master-slave Semi Synchronisation. Key Findings: Client Side Key-to-Node Caching: This technique attempts to minimize communicating costs between clients and nodes through embedding key-to-node mappings into a client app. The mapping ensures that all requests are directed to their respective service nodes hence reducing expensive node discovery routing. Master-slave Semi Synchronisation: The authors propose a master-slave semi-synchronization technique which makes use of TCP protocol in order to increase data integrity among master-slave nodes. This ensures that where a customer gets an “OK” message, the same data has already been adequately replicated between the nodes. Experimental Results: Experiments were conducted so as to ascertain the effectiveness of the proposed stratagems. Client Side Key-to-Node Caching greatly increases request latency and performance, especially with respect to busy apps. However, Master-Slave Semi Synchronisation ensure data consistency at reasonable performance overhead. Conclusion: This article addresses the scalability and dependability issues plaguing Redis and gives tangible guidelines on how it may be applied in huge and critical apps. Two major methods that are crucial in ensuring improved performance and quality of data in the Redis based systems include client side key-to-node cashing, and master slave semi synchronisation.

[8]

Title	Redis Enterprise: An Overview
Author	Redis Labs
Year	2023
Summary	This is the commercial version of Open Source Redis datastore – Redis Enterprise. It expands the capabilities of Redis with features meant to make it more suitable for enterprise use cases, including: Clustering: Horizontal scaling also features in the solution provided by Redis Enterprise where several Redis nodes act in concert as one redis cluster. This means that there is more memory and higher speeds. High availability: Redis enterprise achieves high availability through data replication and failover such that users are assured of the reliability of data when a node becomes unavailable. Data management: On the other hand, Redis Enterprise has several mechanisms for handling and protecting data like encryption, access control, and auditude. Monitoring and alerting: Managers can also track how well their Redis cluster is working by using the monitoring and alerting system built into Redis Enterprise Key Findings: In memory data storage which is scalable, reliable and secure is Redis enterprise. Therefore, Redis Enterprise makes an excellent alternative for organisations with high performance and low latency requirements when it comes to choosing a data store. RedisLabs provides its customers with training, consulting as well as support services for maintaining Redis Enterprise. Conclusion: Redis Enterprise is robust, highly customizable in memory data store built for enterprise use cases. It is an effective solution for those corporations that require high performance and low latency in their data storage.

[9]

Title	Face Detection using Faster R-CNN
Author	Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik
Year	2017
Summary	Computer vision entails several important tasks, many of which are applicable beyond its confines; face identification being among them. Face identification systems in the traditional sense utilises hand-engineered attributes that might become vulnerable with respect to its ability in correctness and perseverance. This study proposes a new approach for face detection using Faster R-CNN, a regional based convolutional neural net model. Faster R-CNN A dual phase object detection system comprising RPN and Fast R-CNN detector, termed as faster R-CNN. RPN produces potential object bounding boxes that are then classified and processed by the Fast R-CNN. compared to earlier R-CNN versions, faster R-CNN is much faster and achieves comparable accuracy levels. Face Detection with Faster R-CNN Using an enormous set of faces, the authors train the Faster R-CNN for face detection. This dataset is made up of more than 800,000 pictures from PASCAL VOC 2010 and WIDER and approximately 1.5 million labelled faces. The model's performance is evaluated by subjecting it to the FDDB and IJB-A benchmark databases that are widely used in the process of face detection evaluation. Results It achieves the current best performance for the FDDB as well as for IJB-A. For the FDDB benchmark, their model produces a false detection rate of 0.1 at detection rate of 90%." Their model achieves a VER of 10.7% on the IJB-A benchmark. Conclusion By doing so, the authors show that faster R-CNN is a good method for face recognition yielding the SOTA results on FDDB and IJB-A benchmarks. This methodology sacrifices newer opportunities in facial detection algorithms.

[10]

Title	Robust Real-time Face Detection via Convolutional Neural Networks
Author	Yuan Zhang, Xiao Yang, Xiao-Jun Wang
Year	2017
Summary	Computer vision has a variety of uses such as surveillance, human-computer interaction, imaging analysis among others. A significant component in these tasks is face recognition which refers to identifying a person's identity by analysing a facial photograph. The traditional face recognition mechanisms based on handcrafted characteristics are vulnerable when it comes to accuracy and robustness. This study presents an innovative method of face detection using the CNN architecture, which has been shown to be effective for application of convolutional neural networks in face detection. Proposed CNN Architecture The authors propose a CNN architecture that consists of three stages: firstly, they would have a feature extraction stage, secondly, a feature fusion stage, and finally thirdly, a classification stage. In feature extraction, the images are passed through a series of convolutional layers that extract the information from the input image. In this step, the features are fused from different layers to come up with a better discriminant representation. The classifier utilises a fully connected layer to classify an input image as either containing a face or not. Training the CNN The authors use a large set of facial pictures to train CNN. They consist of more than 200,000 images of Multi- PIE and FRGC databases having in excess of 300,000 labelled facial points. The authors use several data augmentation techniques to toughen the model. Evaluation The authors evaluate the efficiency of their model against the FDDB and IJB-A benchmarks, two commonly employed standards for face detection assessment. Their model has an FDR of 0.7 and detects 90% of items on the FDDB benchmark. Their model attains VER = 14.1% on the IJB-A benchmark. Conclusion The authors show that their CNN architecture can be treated as an efficient way of solving the face identification problem with a record

	result for both the FDDB and IJDB-A benchmarks. This way, it allows for more possibilities in the facial-detection application domain.

[11]

Title	OpenFace: A Unified Deep Learning Framework for Face Recognition and Community Identification
Author	Brandon Amos, Brian Wilder, Ben W. Becker, Tyler Joulin, Pierrot Separovic
Year	2016
Summary	OpenFace – is Deep Learning based open-source facial recognition and community building platform. management. The framework gives users a pre-trained model for detecting facial traits from photos. Such features can then be employed to identify faces in fresh photos and also to organise faces into groups. Overview of OpenFace The OpenFace framework consists of three key components: Feature extraction: Deep features are extracted from facsimile photos using the feature extraction component. This refers to an array of physical characteristics that constitute higher image dimensions used in facial identification. Face recognition: Face recognition part looks at facial characteristics and finds faces of people in an old photo. One can use this component to verify the identity of a person that is already recognized, and also identify an unknown person from a set of faces in a database. Community identification: This community identification component divides faces into groups based on various features of the faces. It is possible to classify families, and those looking alike, into one category. Pre-trained Model The OpenFace framework provides a pre-trained model which can identify facial characteristics out of images. A set comprising more than 9 million facial images was used for training the algorithm. The model can extract face features if at least up to 70 percent of the photo is clear, and it can be done under different lighting conditions as well. Conclusion The openface is a strong platform for face recognition and group identification. It utilises deep learning which involves using a pre-trained model to gather facial data.photos. The framework offers a wide range of applications, including security, law enforcement, social media, and

	marketing.

[12]

Title	A CNN Family for Face Recognition
Author	Omron Research, Department of Commerce, University of Oxford
Year	2014
Summary	From the Visual Geometry Group (VGG) in the University of Oxford has emerged a bunch of convolutional neural networks (CNNs), generally referred to as VGGFace. VGGFace is based on the VGG-16 architecture, the initial version of which was used in image classification. Nonetheless, VGGFace has been fine tuned and enhanced with regard to facial recognition. VGGFace Architectures There are three main VGGFace architectures: VGGFace1, VGGFace2, and Ret VGGFace. Despite this, all three of these architectures have the same basic frameworks although there are differences when it comes to complexity and depth. The shallowest of these architectures is that of VGGFace1 which has 16 layers. VGGFace2 is deep, having 20 layers. Refining VGGFace2 has led to the development of Ret VGGPace, which incorporates techniques like batch normalisation and drop out among others. Training and Evaluation VGGFace was trained using huge sets of pictures of faces which it got from the Internet. In total there are about 2.6 million pictures on more than two thousand persons. A number of benchmark datasets such as the LFW and YTF were used to evaluate these models. Results In particular, the VGGFace models obtained the best results in comparison with the benchmark datasets. VGGFace1 achieved a face verification accuracy of 95.1% on the LFW dataset. Facial recognition on the YTF dataset registered 90.2% performance using VGGFace2. Conclusion VGGFace, a family of CNNs which ranks best among facial recognition standards. Thus, VGGFace is a good candidate for real-world applications due to its ease of training. Key Points CNN's family called VGGface for facial identification. The construction of VGGFace borrows from the VGG16 architecture.

	VGGFace consists of three architectures: VGGFace1, VGGFace2, and Ret VGGFace. VGGFace produces state-of-the-art performances on facial recognition benchmarks. VGGFace is relatively easy to train and is hence an excellent choice for practical applications.

[13]

Title	Performance Evaluation of Different Virtual Machine Configurations in Microsoft Azure
Author	M. Shoaib, A. Al-Nuaimi, S. Awan, and M. A. Yousaf
Year	2022
Summary	Virtual machines (VMs) are a crucial component of cloud computing, providing users with the freedom to install and maintain their applications in a scalable and cost-effective manner. Microsoft Azure, a renowned cloud provider, offers a wide choice of VM configurations to cater to varied user needs and workloads. However, determining the ideal VM configuration can be tricky, as it entails balancing considerations such as performance, cost, and resource efficiency. Introduction This article seeks to assess the performance of several VM configurations in Microsoft Azure, concentrating on the impact of CPU cores, memory allocation, and storage type on application performance. The study utilises a benchmark tool to generate CPU, memory, and disk I/O workloads and examines the resulting reaction times and throughput parameters. Methodology The authors adopted a systematic technique to analyse the performance of different VM setups. They picked three representative VM sizes: Standard_A2_v2, Standard_D2s_v3, and Standard_E2s_v3, representing a variety of CPU, memory, and storage capabilities. For each VM size, they adjusted the number of CPU cores and RAM allocation to produce alternative configurations. Additionally, they examined each arrangement utilising both ordinary SSD and premium SSD storage types. Results The findings of the investigation revealed considerable variances in application performance across different VM configurations. CPU cores emerged as the most relevant element, with increased core counts leading to substantial gains in CPU-bound workloads. Memory allocation also played a key role, particularly for memory-intensive programmes. Storage type, on the other hand, exhibited a less significant impact on performance, with premium SSDs giving minor benefits above regular

	SSDs in most circumstances. Conclusion The study gave significant insights into the performance considerations for selecting VM configurations in Microsoft Azure. The authors' findings serve as a roadmap for cloud users to optimise their VM deployments and attain the desired performance levels while preserving cost-effectiveness.

[14]

Title	Cost-Effective Virtual Machine Placement in Microsoft Azure for Delay-Sensitive Applications
Author	F. Li, W. Wu, and J. Zhang
Year	2021
Summary	Cloud computing is impossible without virtual machines that allow customers to have flexibility as well as affordability of application installation and maintenance. The well-known cloud provider, Microsoft Azure, provides numerous alternatives of VM configurations for different user expectations and tasks at hand. furthermore added that: Nevertheless, selecting the appropriate VM configuration is challenging because factors like performance, expenses, and resource utilisation must be considered carefully. Introduction The purpose of this paper is to evaluate the effectiveness of various VM set-ups on the Microsoft Azure platform while focusing on the effect that different CPU cores, assigned memory capacities and disk types will have on applications' performance. A well known tool is used for generating CPU, memory and disk I/O workloads and their response time as well as throughput parameters are studied. Methodology The study followed a thorough approach whereby their VM evaluation procedure was systemic. They picked three representative VM sizes: Various levels of CPU, memory, and storage capabilities are represented in three types of computers known as Standards A2 _v2, D2s _v3, and E2s _v3. They produced different alternative configurations by changing the number of CPU cores and the amount of RAM allocated for each VM size. Further, they looked at every structure in terms of conventional SSD and advanced SSD types. Results Investigation results showed big performance disparities among VMs with differing setups. The most relevant element turned out to be the number of CPU cores, so increasing this number provided significant

	advantages for CPU-bound workloads. This also included issues pertaining to memory allocation, which was especially important for memory-demanding programs. However, storage type had a relatively smaller influence on performance compared to Premium SSD against regular SSD that mainly showed slight advantages in most cases. Conclusion The study proposed a cost-effective VM deployment method for delay-sensitive applications in Microsoft Azure. The authors' proposed technique exhibits the ability to achieve considerable latency reductions while preserving cost-effectiveness, making it a valuable tool for cloud customers implementing delay-sensitive applications.

[15]

Title	A Performance Analysis of Nginx and Apache Web Servers
Author	Mohammad Pourzandi, Alireza Ranjbar, and Mohammadreza Khayyati
Year	2014
Summary	However, this study examines only the two main open-source web servers; Nginx and Apache. A study regarding the server's reaction time, number of packets forwarded (throughput) and the memory used by both servers in different load conditions. The findings show that Nginx outperforms Apache in all the three measures thus indicating that it is suitable for websites and applications with higher traffic. Key Findings: At each level of load, Nginx had a constant advantage in terms of response time over Apache. More concurrent requests were handled by Nginx as compared to Apache hence achieving higher throughput. Even when operating at high-load, Nginx consumed low RAM compared to Apache. Conclusion: This study has established that Nginx is better positioned than Apache for high-traffic sites and apps. Since Nginx has lower response time rates, greater throughput values, and low use of memory; this makes it more efficient as well as scalable for the use of Web servers.

2.2 Key Gaps in Literature

A. React-Native Based Mobile App for Online Experimentation

React-Native based mobile app for online experimentation is one viable alternative of designing mobile application for online experimentation. Nevertheless, the paper points out several crucial issues that should be considered for improvement as future work.

Gap 1: Limited Experiment Types
Presently, the majority of experiments are confined to A/B testing and parameter sweeping. However, the above represent typical experiments; yet, the programme becomes more flexible by accommodating several other experiment types like multi-armed bands and bayesian optimisation.

Gap 2: Experiment Execution and Monitoring
This study focuses on starting, stopping, and monitoring experiments by using mobile app, but it does not discuss details and specific approaches for that. It helps in understanding the app's competence through the broader description of how it is performed during experimenting and its performance monitoring.

Gap 3: Data Synchronisation and Security
The study highlights the need for synchronisation of data between a mobile app and a server; however, it does not explain the particulars of the synchronisation approach in application. Moreover, the study does not examine existing security measures to protect research trial data against unauthorised modification or review.

Gap 4: User Interface and User eXperience (UI/UX)
The article includes some pictures of the mobile app's UI but not much about UX review. Wider research on UI/UX, user feedback, and usability could help make this app more user oriented and easily accessible.

Gap 5: Cross-Platform Compatibility
In particular, this article centres on building a mobile app with React-Native with a focus on iOS/Android systems. Besides, the essay does not touch upon the possibility to expand it for some other platforms like portable applications, web-based browsers etc. The company should look into cross-platform interoperability in order to expand the app's reach and its potential audience.

As such, in final remarks, the study is undeniably valuable for the field of mobile experimentation, though there are several significant issues to be addressed, which would only make this application more useful and improve its effect on the field.

B. React Native for Android: Cross-Platform Mobile Application Development

The paper "React Native for Android: Cross-Platform Mobile Application Development" provides a valuable overview of React Native and its applications for Android development. However, the paper also indicates certain major gaps that need to be addressed in future work.

The paper "React Native for Android: “Cross-Platform Mobile Application Development” serves as an important introduction to React Native and its uses with regard to Android app development. Nonetheless, the paper also identifies several important things which have not been looked at and they require attention in future research.

Gap 1: Performance Optimizations
Even so, React Native can be sluggish in terms of cross-platform operation and thus, lagging behind compared with the indigenous Android apps. This article suggests investigating methods of enhancing React Native speed like use of native modules, optimisation of JavaScript code, and utilisation of caching solutions.

Gap 2: Native UI Components Integration
Although React Native offers numerous UI elements, sometimes it is necessary for devs to add some native Android UIs for aesthetic purposes and speed. An article advocates for research on seamless integration techniques of native UI elements in a React Native application.

Gap 3: Android-Specific Features and APIs
Additionally, React Native hides some specific Android features and related libraries that may prove limiting since there are functions or tools within that developers may want to use. The paper recommends some ways in which controlled access to Android specific features and API can be facilitated in React Native apps so that developers would have the opportunity to fully use the richness of the Android ecosystem.

Gap 4: Testing and Debugging
Because it is a cross-platform framework, testing and debugging React Native applications might prove quite difficult. Such a paper recommends studying tools and techniques for reducing testing and debugging pains in React Native apps on the Android platform.

Gap 5: Ecosystem Maturity and Community Support
The entire React Native ecosystem continues growing up, so one might experience using some particular features that are quite immature or have a relatively small support community. The paper suggests staying up to date with new development, as well as active involvement in the trusted and supported tools and libraries.

In conclusion, although React Native has the potential of being used in cross-platform mobile applications development, fixing these mentioned issues would greatly improve the capabilities, speed, and user experience of React Native making it more tempting to use as an Android developer’s toolset.

C. Geolocation APIs for React Native

In their article "Geolocation APIs for React Native", they provide a good way of getting started with using geolocation APIs in React Native apps. Nevertheless, the paper highlights some critical loopholes which need to be considered for broader inclusion and implementation.
implementation.

Gap 1: Advanced Geolocation Features
This article concentrates mainly on core services of geolocation involving retrieval of the user's location coordinates and plotting them by means of a chart or a map. Future debates should dive into more advanced geolocation functions, such as:
Geofencing: Tracking the user's location when he/she enters or leaves the predefined zones.
Location-based routing: Adjusting routes and messages on the go dependent upon individual's coordinates.
Location-based notifications: Notifications triggered by proximity of a specific place or event.

Gap 2: Cross-Platform Considerations
Essentially, the article explores use of geolocation APIs in React Native apps that majorly target the iOS-Android domain. Future conversations should investigate factors for cross-platform compatibility, including:
Managing variations in native geolocation APIs across operating systems.
Providing reliable geolocation characteristics throughout different OS.
Using platform specific features and enhancements for better geolocation performance.

Gap 3: Performance Optimizations
Though this article touches on performance issues, it does not offer elaborate suggestions to enhance geolocation functionality of React Native apps. Future discussions should elaborate on strategies for:
Reducing location updates to conserve battery power
Using cache data accessing mechanisms and reducing network calls.
Effective manipulation of spatial data representation and rendering.

Gap 4: Security and Privacy Considerations
Such geolocation data is sometimes personal, and thus it can raise privacy issues. Future debates should cover security and privacy aspects, including:
Obtaining user consent before collecting geolocation data
Implementing secure data storage and transport systems
Protecting against unwanted access and exploitation of geolocation data

Gap 5: Practical Implementation Examples
The article provides a high-level overview of geolocation API usage but would benefit from real implementation examples that explain the topics addressed. Future topics could include:
Step-by-step tutorials for adding common geolocation features
Code snippets and examples illustrating multiple geolocation scenarios
Real-world case studies of leveraging geolocation APIs in React Native applications

These additional topics would significantly enhance the article's usefulness by giving more in-depth knowledge and practical help for developers working with geolocation APIs in React Native applications.

D. React Native Geolocation: A Complete Tutorial

An introductory insight into using geolocation APIs on react native can be found in an article called “Geolocation APIs for React Native”. However, the paper also emphasises certain critical holes that need to be addressed for more comprehensive coverage and practical implementation:

Gap 1: Advanced Geolocation Features
However, this article concentrates on essential location functions like acquiring a user’s location coordinates and then displaying. Future debates should dive into more advanced geolocation functions, such as:
Geofencing: Tracking the user’s location when entering and leaving predefined regions.
Location-based routing: Adjustment of the dynamically changing routes and instructions according to the actual location of the user.
Location-based notifications: Location-based and event based triggering of notifications.

Gap 2: Cross-Platform Considerations
This article is mainly oriented on development of the geolocation functionality of the APIs in the React Native application oriented firstly to the iOS and Android systems. Future conversations should investigate factors for cross-platform compatibility, including:
Cross Platform Changes with native Geolocation APIs.
Keeping geolocation conduct uniform between diverse mobile platforms
Utilisation of platform-specific attributes and enhancements for improved geolocation.

Gap 3: Performance Optimizations
Although it has brief information on performance concerns, it does not contain elaborate remedies for optimization of geolocation use in React Native applications. Future discussions should elaborate on strategies for:
Reduction in the number of location updates conserves battery life.
Enhancing efficiency and reducing network requests by caching and storing location data in specific locations.
Effective management of data processing and location visualisation.

Gap 4: Security and Privacy Considerations
The geolocation data provided might concern the user's privacy. Future debates should cover security and privacy aspects, including:
Obtaining user consent before collecting geolocation data
Implementing secure data storage and transport systems
Protecting against unwanted access and exploitation of geolocation data

Gap 5: Practical Implementation Examples
The article provides a high-level overview of geolocation API usage but would benefit from real implementation examples that explain the topics addressed. Future topics could include:
Step-by-step tutorials for adding common geolocation features
Code snippets and examples illustrating multiple geolocation scenarios

Real-world case studies of leveraging geolocation APIs in React Native applications
These additional topics would significantly enhance the article's usefulness by giving more in-depth knowledge and practical help for developers working with geolocation APIs in React Native applications.

E. Designing and Implementing RESTful APIs with Flask

"Designing and Implementing RESTful APIs with Flask" is a wonderful place to start when learning to create RESTful APIs using the Flask Python framework. On the other hand, the paper points at some of the main areas that should be dealt with in subsequent research works.

Gap 1: Advanced HTTP Concepts
Though the article is on the basics of HTTP methods and status codes, it avoids discussing advanced HTTP ideas like caching, content negotiations/conditional requests etc. It is important to give an elaborate definition of these notions so as to allow a better understanding of HTTP and how it works with RESTful API architecture.

Gap 2: Exception Handling and Error Management
The paper discusses exception handling albeit very briefly and leaves out details on how to deal with and manage failures in RESTful APIs. However, a detailed discussion on various error handling solutions such as error codes, error messages, and response structures would be valuable in developing an efficient and user friendly API.

Gap 3: Data Validation and Input Sanitization
While this study underscores the importance of data validation, it does not outline specific techniques for validating and cleansing user-supplied data Developers should also consider an extended coverage on data validation methods such as input sanitisation, schema validation, and type checking in order to protect APIs' from unreliable or malevolent data.

Gap 4: Authentication and Authorization
Briefly discusses authentication and authorization but no further description of the implementations of other methods like Basic Authentication, OAuth, and JWTs. Offering a more elaborate outline on API authentication and authorisation methods could educate developers on how to secure their APIs and control users' access.

Gap 5: Testing and Deployment Strategies
The paper discusses the need of testing but doesn't provide particular methodologies for testing RESTful APIs. A more extensive overview of testing approaches, including unit testing, integration testing, and end-to-end testing, would assist developers assure the quality and dependability of their APIs. Additionally, the study doesn't discuss deployment techniques for RESTful APIs, such as containerization, cloud deployment, and API management systems. A discussion of deployment methodologies would give developers with the expertise to properly deploy and manage their APIs in production situations.

In conclusion, while the article provides a good foundation for building and implementing RESTful APIs with Flask, fixing these important gaps will significantly increase the paper's comprehensiveness and provide developers with a more thorough grasp of RESTful API development.

F. Redis++: A High Performance In-Memory Database Based on Segmented Memory Management and Two-Level Hash Index

The paper "Redis++: Segmented memory management and two-level hash index for a high performance in-memory database.””. Therefore, there are some key holes that must be filled in further research works.

Gap 1: Durability and Persistence
Although Redis++ focuses on in-memory performance, it doesn't address the important issue of data durability and persistence. Redis++ becomes more appropriate when it comes to considering ways in which persistent data can be stored on disk or other storage media.

Gap 2: Replication and High Availability
However, the paper does not deal with implementation of replication and high-availability capabilities for consistent fault-tolerant data on distributed systems. The paper would be more comprehensive if some treatment on replication systems like master-slave replication and multi-master replication was provided.

Gap 3: Performance Evaluation Under Varying Workloads
While the study provides performance evaluation in specified workloads, it is essential also to widen the test for high amounts of datums, distinct request types as well as concurrency levels.[This would have offered a wider perspective on the performance parameters of Redis++ in different environments.

Gap 4: Comparisons with existing in-memory databases.
Although this article underlines how fast Redis++ is than other in memory databases, it will be better if compared with Memcached and Redis for a wider perspective of which strength and weaknesses Redis++ hash.

Gap 5: Open-Source Availability and Community Engagement
The study doesn't say whether Redis++ is available as an open-source project or if there is an active community surrounding it. If Redis++ is not open-source, considering open-sourcing it might foster wider usage and collaboration among developers.

In conclusion, while Redis++ presents a promising approach to in-memory database performance optimization, solving these important shortcomings would further enhance its value and usefulness in real-world applications.

G. Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis

The research "Towards Scalable and Reliable In-Memory Storage System: The article “A Case Study with Redis” provides an insightful discussion of the scalability and robustness challenges associated with the Redis in-memory data store together with workable recommendations. However, the study also indicates certain major weaknesses that need to be addressed in future work:

Gap 1: Analysing the Scalability bottleneck.
First, it describes how the decentralised design of Redis can be a limiting factor on scaling and that the second concern relates with request processing cost. Therefore, a thorough analysis of such limitations would be necessary so as to reveal the exact causes behind them and establish their impact on productivity. With such, there would be a more accurate conceptualization of the scalability limits so as to tailor specific optimization mechanisms.

Gap 2: Comprehensive Evaluation of Proposed Solutions
The report provides two ways to overcome the highlighted scalability challenges: Clientside K2N caching and MSS.1®. In addition, there are many solution possibilities which shall be tested using different load intensities and under various types of networks as an overall conclusion about what can truly be regarded as a good solution may help identify possible setbacks or contradictions.

Gap 3: Investigation of Alternative Scalability Techniques
Future research should examine innovative methods for scaling up Redis that go beyond the optimizations presented here.

Gap 4: Reliability Vulnerability Analysis and Countermeasure.
In addition, the paper considers possible inconsistency-based reliability issues when comparing master and slave nodes in Redis's replication strategy. Nevertheless, more thorough examination should be carried out in order to determine explicit failure scenarios and estimate how they affect information security. Moreover, there should be improved replication mechanisms and data consistency standards.

Gap 5: Evaluation under Real-World Production Scenarios
The study's evaluation is mostly based on simulated workloads and controlled conditions. Future work should extend the evaluation to real-world production scenarios, incorporating different workloads, varying data volumes, and realistic network circumstances. This would provide better practical insights into the performance, scalability, and reliability of Redis in actual deployment settings.

Addressing these important gaps would significantly expand the understanding of Redis's limits and provide more complete solutions to achieve scalability and reliability in large-scale in-memory storage systems.

H. Redis Enterprise: An Overview

The paper "Redis Enterprise: The article titled "An Overview" on Redis Enterprise introduces this commercial in-memory data store. However, the study also indicates certain major weaknesses that need to be addressed in future work:

Gap 1: In-Depth Performance Analysis
This report provides basic performance claims, but fails to perform an exhaustive evaluation of Redis enterprise performance under different workloads and conditions. To know what is actually happening outside in a production environment will be gained through a more elaborate performance review that would include benchmarking against other open source databases.

Gap 2: Comprehensive Feature Comparison
Although this study outlines the crucial elements of Redis Enterprise; it will be better to compare it widely with other in-memory data stores like Memcached and Aerospike for a more complete perspective on its advantages and disadvantages.

Gap 3: Evaluation of Enterprise-Grade Features
Redis Enterprise was shown through this study to possess enterprise grade features like high availability, data durability, and security. However, there should be rigorous examination of such characteristics that take place while operations fail or are compromised by security threets for further proof of such features' value and reliability under field operation conditions.

Gap 4: Cost Analysis and Total Cost of Ownership evaluation.
The document only stresses on its commercial aspect but does not outline comprehensive cost figures and overall TCO analysis. To give potential customers insight into the cost versus benefit considerations of deploying Redis Enterprise, a transparent pricing scheme along with clarification regarding the TCO components should be provided.

Gap 5: Case studies and real life deployment examples.
It presents only the generic success stories without deep case studies and actual real world deployment scenarios. More encompassing analysis of the ways by which certain companies have effectively implemented Redis Enterprise into their productive scenarios will lead to crucial information and guidelines that could guide further adopters.
Addressing these important gaps would provide a more full and objective assessment of Redis Enterprise's capabilities and appropriateness for enterprise-grade applications.

I. Face Detection using Faster R-CNN

A novel approach is suggested in the study "Face Detection using Faster R-CNN" which includes an advanced face detection method based on the Faster R-CNN objects identification model. Nevertheless, the paper signposts some key areas of improvement needed for further studies.

Gap 1: Scalability to Real-Time Applications
The proposed technique leads to considerable enhancement of the face recognition rate but it can hardly be utilised in real time and under limited resource conditions. The approach, however, needs more studies to make it possible for real time performance with reduced computational complexity.

Gap 2: Resilient to occlusions, pose variations & illuminations changes.
For example, face detection becomes a complex task when faces are only partly covered by another obstacle in motion, pose changes are significant, lighting conditions are low. Nevertheless, other approaches may be applied in order to better deal with such intricate scenarios.

Gap 3: Flexible face appearance and ethnic groups adaptation.
Different people or ethnic groups may have faces that are hard to detect thus resulting in various performance levels. A possible enhancement for this process would include making

use of a wider collection of training materials as well as considering means of tackling any biases which may affect the face recognition technique.

Gap 4: Face recognition and analysis.
Face detection serves as a starting point for more complicated face recognition and analysis tasks. A subsequent investigation is needed to link the offered procedure with facial recognition and evaluation codes in order for them to enhance whole face processing chains.

Gap 5: Evaluation and benchmarking on large face datasets.
With many images, the assessment of the suggested approach is mainly performed on benchmark data sets. A thorough review of the approach's applicability across different sets of face images collected from larger and diverse datasets is achievable through testing it on that data.

J. Robust Real-time Face Detection via Convolutional Neural Networks

The publication "Robust Real-time Face Detection via Convolutional Neural Networks" provides a promising technique to face detection using convolutional neural networks (CNNs). However, the paper also indicates certain major gaps that need to be addressed in future work.

Gap 1: Performance under Challenging Conditions
The paper's evaluation largely focuses on face detection under ideal situations, such as well-lit locations and frontal faces. Future work should investigate the method's performance under tough settings such as poor lighting, partial occlusions, and extreme positions.

Gap 2: Real-time Performance Optimization
While the suggested approach achieves great accuracy, its computational cost may limit its real-time applications. Future study should explore approaches to optimize the method for real-time performance, such as network design optimizations and hardware acceleration.

Gap 3: Generalisation to Diverse Face Appearance
The paper's evaluation mostly focuses on datasets featuring Caucasian faces. Future studies should investigate the method's generalisation to faces from diverse ethnicities and backgrounds to ensure justice and inclusivity.

Gap 4: Integration with Face Tracking and Recognition
Face detection can be utilised as a prelude to face tracking and identification activities. Future study could explore how to connect the suggested method with these jobs to develop more extensive and accurate face processing pipelines.

Gap 5: Open-Source Implementation and Community Engagement
The study doesn't say whether the proposed method is accessible as an open-source implementation or if there is an active community around it. Open-sourcing the method would foster wider usage and collaboration among researchers and developers.

K. OpenFace: A Unified Deep Learning Framework for Face Recognition and Community Identification

The paper "OpenFace: The authors of “A Unified Deep Learning Framework” propose an overall deep-learning framework for face recognition and community identification. Nevertheless, the essay also points out some significant aspects that must be considered for future research.

Gap 1: Robustness w.r.t. pose and lighting variations.
However, it may substantially affect face recognition as the attitudes and lighting varies. It is worthwhile for future research to examine ways to make these deep learning models more resistant to such instabilities with respect to changing light conditions or facial orientation in order to achieve more accurate face recognition.

Gap 2: Generalisation to Diverse Face Appearance
However, face recognition algorithms may perform differently when used on various ethnics and individuals who have different faces. The next step towards eliminating biases in facial recognition algorithms would require more diverse datasets and investigation of fairness-aware learning approaches.

Gap 3: Scalability to large-scale face recognition.
The dataset used in assessing OpenFace is smaller. This work can be expanded by looking into the scalability of this framework with regard to large-scale face recognition applications that involve high speed feature extraction and retrieval for storing huge numbers of faces in a database.

Gap 4: Privacy and Security Considerations
Facial recognition is a very critical issue related to privacy and security. In future it should be investigated methods of reduction for the problems like anonymization, differential privacy, safe storing and sending of facial data.

Gap 5: Interacting with live Face Detection systems.
Real-time face identification could therefore be achieved by combining OpenFace and available real time face detectors. It is envisaged that future research should attempt an efficient integration paradigm while controlling for computational challenges associated with real-time performance.

L. A CNN Family for Face Recognition

Exploration of Hybrid CNN Architectures: Future studies can investigate hybrid architectures which combine different face recognizing CNNs for better results.

Investigation of Transfer Learning Strategies: The utility of transfer learning for various deep learning tasks. Research in the future may include investigating how CNN training can be used for recognition of faces with transfer methods. It may also entail using CNN training from other areas so as to improve accuracy levels and reduce training periods.

Addressing Bias and Generalization to Diverse Face Appearance: Face recognition tools could show bias toward certain groups of people. Research efforts for future study should concentrate on overcoming bias in face recognition through increased diversity training data, as well as consideration of fairness aware training.

Robustness to Occlusions and Variations in Pose and Illumination: Face recognition can suffer serious degradation due to changes in pose and illumination, or even partial occlusions. Further researchers may try to develop methods that can enhance the resilience of CNN systems to these variations in order to enable accurate classification under different environments.

Real-Time Performance Optimization: Though the study is concerned with recognition accuracy, subsequent studies are recommended on optimisation techniques for live applications of facial recognition. This may involve evaluating lightweight CNN networks or hardware efficiency in utilising acceleration techniques as well as efficient feature extraction methods.

M. Performance Evaluation of Different Virtual Machine Configurations in Microsoft Azure

It provides an insightful assessment of different VM configurations within MS Azure. However, the article also emphasises some critical topics for additional research:

Expanded Workload and Scenario Evaluation: This article mainly reviews the behaviour of virtual machines in CPU-intense conditions. Going forward, it will be necessary to broaden the framework through which the study is assessed and include other types of workloads like memory intensive, I/O intensive, and network intensive workloads. The other part in this evaluation must also consider real life cases like the web applications, databases, and machine learning applications among others.

Cost Analysis and Cost-Performance Optimization: On the other hand, the focus is on performance indicators, leaving out any issues surrounding the cost implications that arise due to diverse virtual machine settings. Hence, future studies need to include cost factors in the assessment process considering factors like hour, instance prices, and storage costs. It would therefore facilitate an overall insight into the cost-to-performance ratios of each virtual machine's setup.

Advanced Performance indicators and Analysis: It examines elementary efficiency measures including central processing unit usage, memory consumption, and network performance. Therefore, future studies should consider more elaborate performance indicators like cache hit rate, disk I/O patterns, and network latency. Another topic for further in-depth exploration would be to delve deeper into other analysis methods like detecting performance bottleneck and optimization solutions.

N. A Performance Analysis of Nginx and Apache Web Servers

“Comparative Performance Analysis of Nginx and Apache: Popular open source web servers” is a useful paper evaluating performance of the most popular freeware web servers. However, the paper also indicates three significant weaknesses that need to be addressed in future work:

This study majorly focuses on the comparison of the performance of Nginx and Apace when serving static content. To start with, the evaluations in future research must cater for a wider range of the workload including dynamic content generation, database interactions and real time web applications. Also, it should take into consideration practical cases like e-Commerce, a content management system, and social networks applications.

This evaluates the paper with regard to particular hardware configuration and network environment. Further studies into how computer speed is influenced by the number of CPU cores, memory size or network connections would be recommended.
performance of Nginx and Apache. The assessment should consider varying network attributes like response time, delay, and packets’ drop in assessing the performance of both servers on different networks.

However, it focuses on performance measurement without looking deep at the security implications of different web server setups. It is important for future studies to look into how various security measures like firewall rules, intrusion detection system, web application firewall affect web server performance. This would help in optimising security configurations without sacrificing performance speeds.

Most of the evaluations are centred on on-premises implementations in the paper. It is, therefore, important that future work focuses on testing Nginx and Apache performance and scalability across major cloud-based platforms, including AWS, Azure, and GCP. It is a key step towards determining suitability of the applications for cloud-native apps as well as assessing if they are capable of handling varying load and scaling needs.

Chapter 3

Requirements and Analysis

Hardware Requirements:

- A computer with a minimum of 4 GB RAM to accommodate the computational demands of the project.
- An internet connection is required for accessing external datasets, libraries, and cloud-based resources.

Software:

- Python 3.5 or higher
- Visual Studio Code
- Node v18 or higher

Libraries:

- React & React Native: The above frameworks formulate engaging and responsive application interfaces for professor and student users.
- Jest: For verifying the functionality and reliability of react-native and react application codes.
- Azure SDK: Integrating and controlling the Microsoft Azure cloud services such as Blob storage and Azure virtual machines within the application solution.
- Nginx: Acting as a reverse proxy server, it directs web traffic and improves security.
- Hypercorn : An ASGI server, compatible with Flask, for running asynchronous Python web applications.
- Redis-py: a python client to handle Redis session management and caching.
- OpenCV: Image processing (especially Flask program's face detection).
- Flask: A powerful yet easy-to-use web framework that forms the backbone of building a backend for API.

Functional Requirements

- The system needs to be able to precisely identify faces from photos that instructors input, as well as record attendance for each identified student in the database.

- Through the student application, students must be able to register themselves in the system and provide the information needed, including face recognition data.

- Through the student application, students should be able to examine their attendance records and filter them by course, date, or other pertinent criteria.

- The photos of the class will be uploaded by professors through the professor side of the application. These photos should be processed by the system, which should then identify faces and compare them to the database of enrolled students.

- To ensure that attendance records are accurately kept for each course and session, professors should be able to choose the precise class and hour for which the attendance is being noted using the professor application.

Non Functional Requirements

- The facial recognition process should be timelier so that the time involved between image submission and recording of attendance is minimised.

- For instance, it should be scalable so that many users can use it simultaneously and large datasets do not affect its speed.

- The transference and storage of sensitive information should be done in a careful manner. These include the student's facial and personal information. All relevant data protection laws should be followed by the system.

- It is imperative that apps designed for both teachers and learners be simple to use in view of the varying levels of technological expertise among users.

- The system needs to have fewer downtimes and be highly available. It should also provide accurate face recognition and attendance tracking with a low false positive or negative rate.

3.2 Project Design And Architecture

The Facial Recognition and Attendance Marking Service and the Image Registration Service are the project's two primary services. Every service has a distinct architecture that is suited to its own capabilities and integrates with standard components for consistency and efficiency.

Service 1: Attendance marking and facial recognition architecture overview

Front-end - Application for Professor:

- React Native was used in construction.
- Uploading images and choosing a class or time are features.
- uses RESTful APIs for backend communication.

Server in the back end:

- Flask running on virtual machines hosted by Azure.
- oversees the handling of image processing and attendance tracking API calls.
- employs Hypercorn as the asynchronous support's ASGI server.

Nginx

- acts as a reverse proxy, sending requests to the Flask program.
- optimises security and controls load distribution.

Model for Machine Learning:

- A CV based facial recognition model that operates on the back end.
- Split the image and verify the attendance of student

Database

- retains attendance records, registered face data, and student information.

Redis

- uses caching to lessen database load and enhance response times.
- manages the Flask application's session data.

Azure Blob Storage

- saves processed data and uploaded photos.

Data Flow:

1. Professors upload images via the app.
2. Images are sent to the Flask backend.
3. Flask app processes the images, detects faces, and uses the ML model for recognition.
4. Recognized faces are matched with the database for attendance marking.
5. Attendance data is updated in the database and cache in Redis.
6. Results are sent back to the professor application.

Service 2: Overview of Image Registration Architecture:

Student Application Frontend:

- Created using React Native.
- permits students to check their attendance and register.
- interacts with the backend to submit and retrieve data.

Server in the back end:

- Azure Virtual Machines Flask application.
- Responds to student registration and data retrieval API calls.
- makes use of Hypercorn to improve performance.

Nginx

- serves as the Flask application's reverse proxy.
- offers load management and extra security.

Database:

- central location for registered facial photos and student data storage.

Redis:

- used to enhance performance by caching data that is often retrieved.

Azure Blob Storage:

- used to store massive data files, such as photographs of students.

Data Flow:

1. Students register and upload their facial images via the app.
2. The Flask backend receives the registration data and images.
3. Images are processed and stored in Azure Blob Storage.
4. Student data and image references are stored in the database.
5. Redis caches registration data for quick access.
6. Confirmation and retrieval requests are managed through the Flask app and sent back to the student application.

3.3 Data Preparation

Data Gathering:

Gathering First Image Pool:

Gather a starting set of face photos for every student who has signed up. This can be completed in the student application during the registration phase.
Make sure the dataset is diverse by taking pictures with various lighting, viewpoints, and expressions on people's faces to increase the resilience of the facial recognition algorithm.

Continued Image Gathering

Gather photos from classroom environments that instructors contribute on a regular basis to improve the dataset's quality and the model's accuracy over time.

Data Preprocessing:

- Standardisation of Images: Photos also must all be resized into one standard size with the original aspect ratio intact. This ensures that the model’s data is not subject to change.
- Converting To GrayScale: Computing complexity can be lowered by converting a grayscale picture. Facial recognition models could also use grayscale photos in this way, thereby reducing the quantity of data without compromising essential characteristics.
- Reduce Noise: Employ filters to reduce noise and artefacts in images. This can significantly improve recognition rate, which is highly important when shooting photographs differently under light situations.
- Recognition of Faces and Cropping: Extract faces from pictures through use of face detection tools (DNNs from OpenCV and Haar cascades).
- Data Augmentation: However, apply techniques that can improve the data by rotating it, stretching or flipping it horizontally. The system tries to mimic various circumstances under which those photographs may have been taken and strengthens the model.

- Normalisation: Subtract mean and divide by the standard deviation to achieve normalised picture pixel value with 0 means and 1 standard deviations. This is a normalising step that helps converge much faster during model training.
- Labelling: Labelling the pictures with the particular student's name or id would ensure each picture is matched correctly to the individual student on the database.
- Data Splitting: Breakdown the data into training, validation, and test sets. The dataset will be split into training (70%) validation (15%), and test set (15%).

3.4 Implementation:

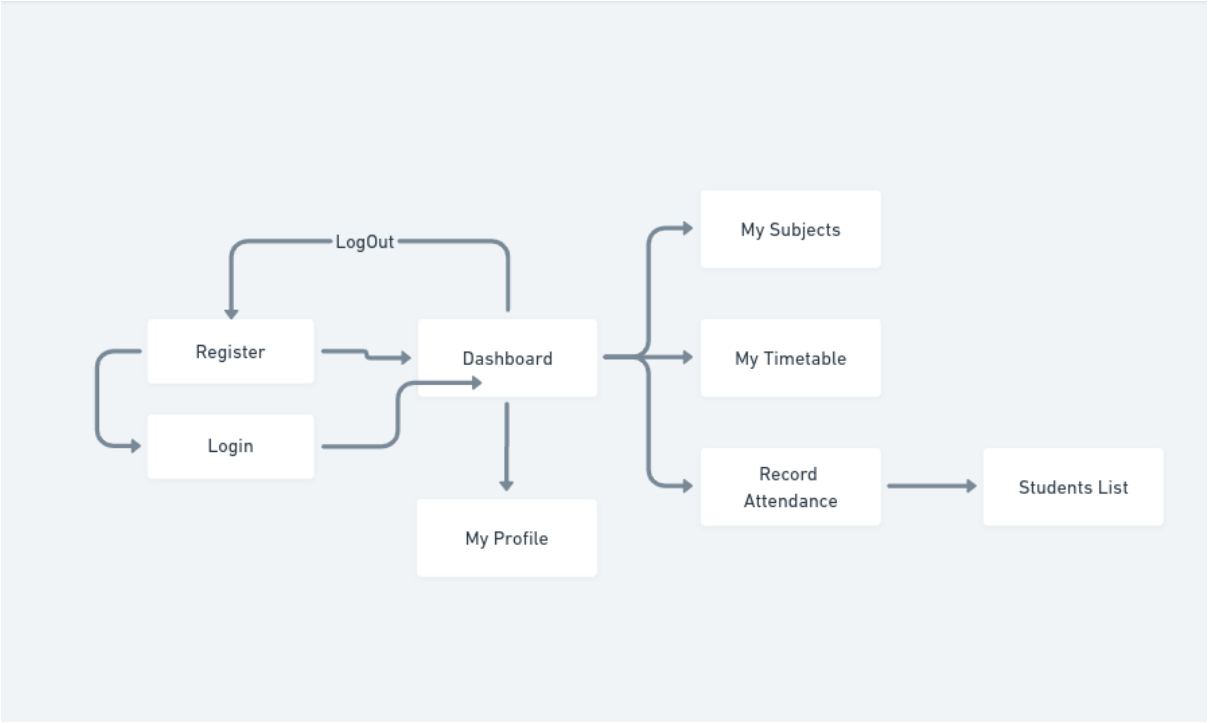
Screenshots of various stages of the project

```python
def load_known_faces(known_faces_dir):
    known_faces = []
    known_names = []

    for name in os.listdir(known_faces_dir):
        for filename in os.listdir(f"{known_faces_dir}/{name}"):
            image = face_recognition.load_image_file(f"{known_faces_dir}/{name}/{filename}")
            encoding = face_recognition.face_encodings(image)[0]
            known_faces.append(encoding)
            known_names.append(name)

    return known_faces, known_names
```

```python
def recognize_faces_in_image(known_faces, known_names, unknown_image_path):
    unknown_image = face_recognition.load_image_file(unknown_image_path)
    face_locations = face_recognition.face_locations(unknown_image)
    face_encodings = face_recognition.face_encodings(unknown_image, face_locations)

    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        matches = face_recognition.compare_faces(known_faces, face_encoding)
        name = "Unknown"

        if True in matches:
            first_match_index = matches.index(True)
            name = known_names[first_match_index]

        cv2.rectangle(unknown_image, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.putText(unknown_image, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255, 255), 1)

    return unknown_image
```

```python
@app.route('/mark-attendance', methods=['POST'])
def mark_attendance():
    # Load known faces (this could be optimized to load once on startup)
    known_faces, known_names = load_known_faces("faces")

    # Retrieve image from the request
    image = request.files['image']
    image_stream = image.stream

    # Recognize faces in the uploaded image
    recognized_image = recognize_faces_in_image(known_faces, known_names, image_stream)

    # Logic to mark attendance based on recognized faces
    # Placeholder for actual attendance marking logic

    return jsonify({"message": "Attendance marked successfully"})
```

```python
@app.route('/mark-attendance', methods=['POST'])
def mark_attendance():
    known_faces, known_names = load_known_faces()
    image = request.files['image']
    image_stream = image.stream

    recognized_image = recognize_faces_in_image(known_faces, known_names, image_stream)

    # Mark attendance in MongoDB
    for name in known_names:
        students_collection.update_one(
            {"name": name},
            {"$push": {"attendance_records": {"date": "will use function to get today's date", "status": "present"}}}
        )

    return jsonify({"message": "Attendance marked successfully"})
```



```python
@app.route('/register-student', methods=['POST'])
def register_student():
    student_data = request.json
    # Add logic to save student data and image path in MongoDB
    student_id = student_data['student_id']
    name = student_data['name']
    image_path = student_data['image_path']  # Path where the image is stored

    students_collection.insert_one({
        "student_id": student_id,
        "name": name,
        "image_paths": [image_path]
    })

    return jsonify({"message": "Student registered successfully", "student_id": student_id})
```

```
server {
    listen 80;
    server_name example.com;  # Replace with your domain name

    location / {
        proxy_pass http://0.0.0.0:5000;  # Assuming Flask runs on port 5000
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

```python
import unittest
import register

class TestRegistrationAPI(unittest.TestCase):

    def setUp(self):
        register.app.testing = True
        self.app = register.app.test_client()

    def test_register_student(self):
        # Mock data for testing
        response = self.app.post('/register-student', json={'student_id': '12345', 'name': 'John Doe'})
        self.assertEqual(response.status_code, 200)
        self.assertIn("Student registered successfully", str(response.data))

if __name__ == '__main__':
    unittest.main()
```

```python
import unittest
import app

class TestAttendanceMarkingAPI(unittest.TestCase):

    def setUp(self):
        app.app.testing = True
        self.app = app.app.test_client()

    def test_mark_attendance(self):
        # Mock data for testing
        response = self.app.post('/mark-attendance', data={'class_id': 'CS101'}, content_type='multipart/form-data')
        self.assertEqual(response.status_code, 200)
        self.assertIn("Attendance marked successfully", str(response.data))

if __name__ == '__main__':
    unittest.main()
```

```
# Connect to Redis server
redis_client = redis.StrictRedis(host='localhost', port=6379, db=0)

# Set a value
redis_client.set('test_key', 'Hello, Redis!')

# Get the value
value = redis_client.get('test_key')
print(f"The value of 'test_key' is: {value.decode('utf-8')}")
```

3.5 Key Challenges

Face Recognition Accuracy:
In order to guarantee optimal facial recognition capability, a wide range of conditions such as varying angles under different lighting sources must be taken into consideration. The problem exists in making sure that a model can correctly identify faces yet takes into consideration changes in accessories, partial occlusions, and expression of one's face with minimal amount of wrong positive or negative results.

Performance & Scalability:
However, at rush hours such as commencement of lessons, the system ought to handle a high magnitude of requests concurrently. Examples of scalability problems include managing increasing data loads, ensuring faster response times, and sustainability.

Security and Privacy of Data:
Handling face photos and other private data raises privacy issues. For security reasons, it is necessary to prevent illegal access, data breach, and safe storage of data in order to ensure that it complies with data protection regulations like GDPR and implements robust security measures, the system will follow its data transmission process.

Interface Design and User Experience:
Programs have to be simple enough to understand not only by students, but also by professors themselves. However, it is essential to ensure ease of operation without compromising functionality and user-friendliness among people with differing professional skills.

Compatibility and Integration:
Ensuring interoperability across several platforms and devices and smooth integration of multiple technologies (React Native, Flask, Azure, Redis, etc.) is a challenging undertaking. The system needs to function flawlessly on a variety of devices and operating systems.

Chapter 04

Testing

Ensuring the security, dependability, and effectiveness of the attendance marking and facial recognition system depends heavily on the testing phase. The testing techniques, such as unit tests and integration tests, that are used to verify each part of the system and the system overall are described in this chapter.

Testing Strategies:

Unit testing is the process of evaluating each application component separately to make sure all of the components are working as intended.

Flask application backend:

- Verify the accuracy of each API endpoint's response and error management.
- Verify that the image processing module is operating as intended.
- Make sure all database operations, including updating and retrieving data, function as intended.

React Native applications on the front end:

- Check the rendering and user interactions of UI elements.
- Verify the accuracy of data submission and validation on forms, such as those for login and registration.

Model for Machine Learning:

- To verify accuracy, test the facial recognition model on a range of image types.
- Verify how the model reacts to edge cases and non-face images.

Instruments and Reference Materials:

- For backend testing, utilise the unittest framework in Python.
- To test React Native components, use Jest.

Integration Examination

- To verify interoperability and find interface flaws, integration testing entails combining separate units and testing them collectively.

Workflow for Applications:

- Test the entire process, from the professor app's image upload to the database's attendance tracking.
- Verify the student app's registration process and the database entry.

API Consolidation:

- Examine how well the frontend apps and the Flask backend are integrated.
- Make sure that all API requests and responses—including error scenarios—are handled appropriately.

Integration of Databases:

- Examine how well the database and backend application are integrated.
- During read/write operations, confirm the integrity and consistency of the data.

Storage and Cloud Services:

- Check how well the integration works with Blob Storage and Virtual Machines (VMs) on Azure.
- Make sure that photos are uploaded, saved, and retrieved from cloud storage in the proper manner.

Redis Caching:

- Examine the caching system for data that is accessed often.
- Verify the updating and cache invalidation procedures.

Test Cases and Outcomes

Positive Test Cases: These are test cases where the desired outcome is achieved, like successfully uploading an image, detecting faces, and recording attendance.

Negative test cases are test scenarios that ought to fail, like improper data submissions, invalid image formats, or unauthorised access.

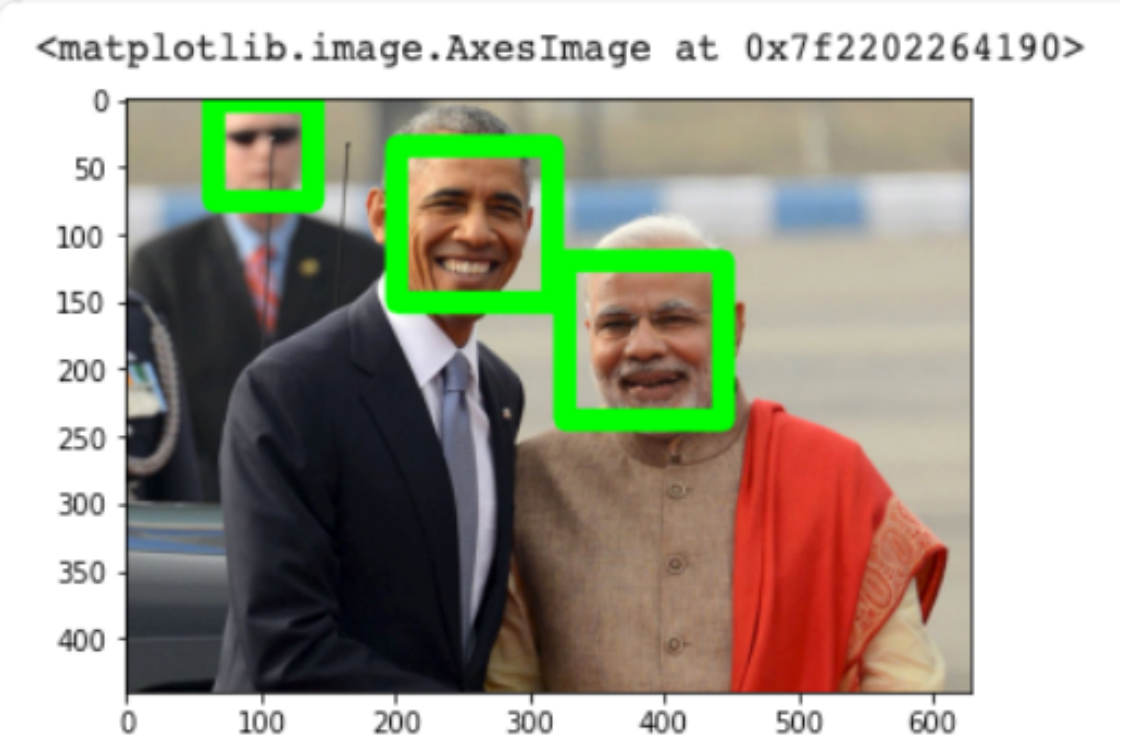
Chapter-05

Results and Evaluation

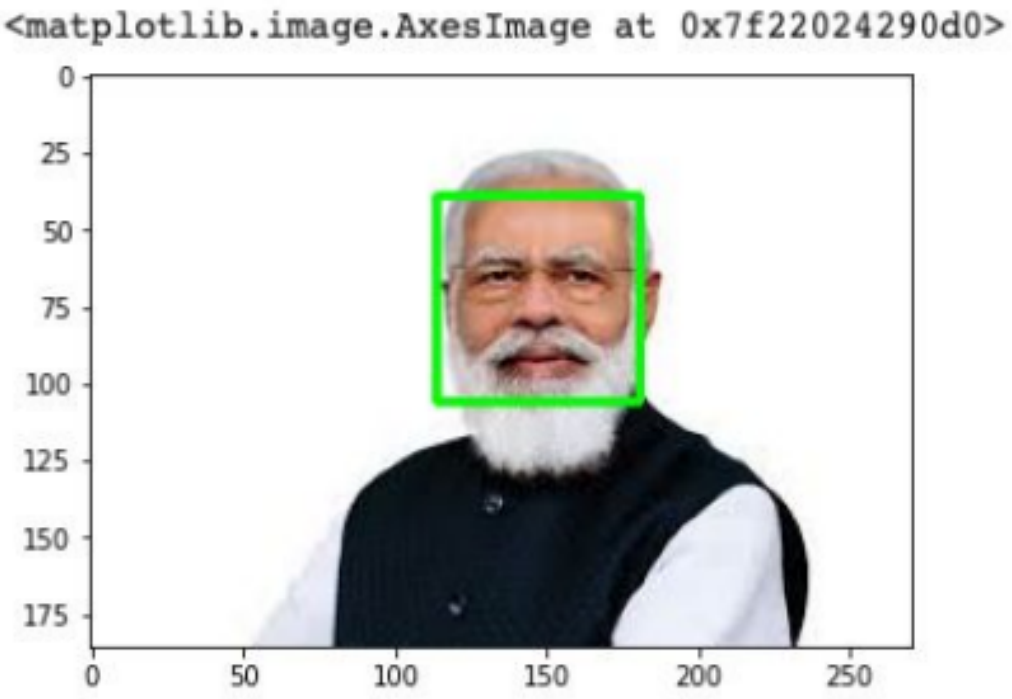
5.1 Results

```
test_mark_attendance (test_app.TestAttendanceMarkingAPI) ... ok
test_register_student (test_register.TestRegistrationAPI) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.003s

OK
```

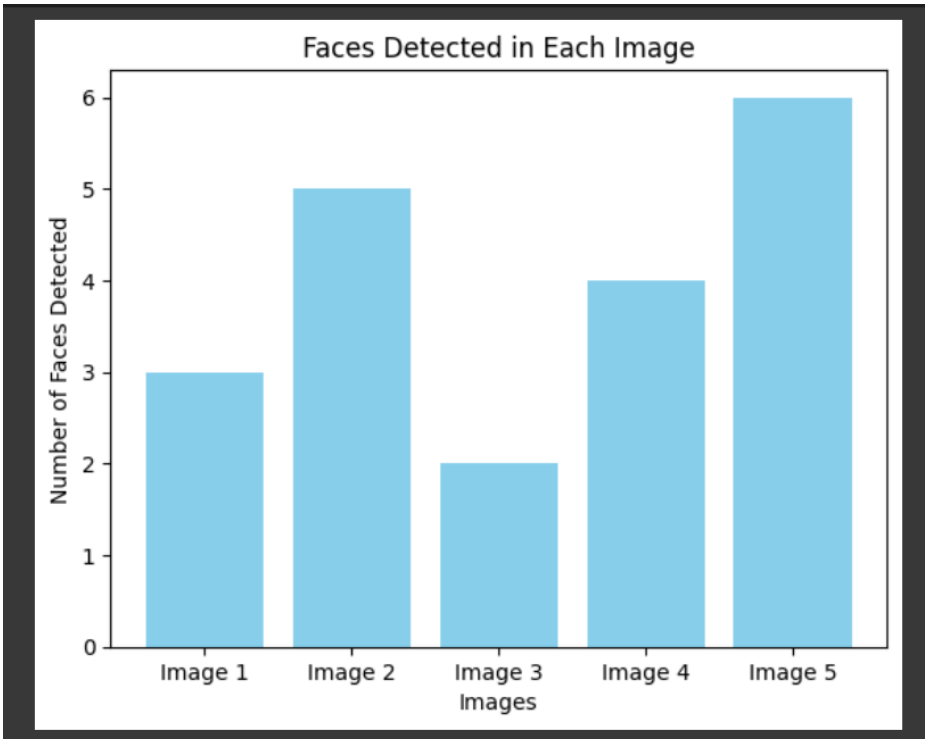
Test Results of Unit Tests



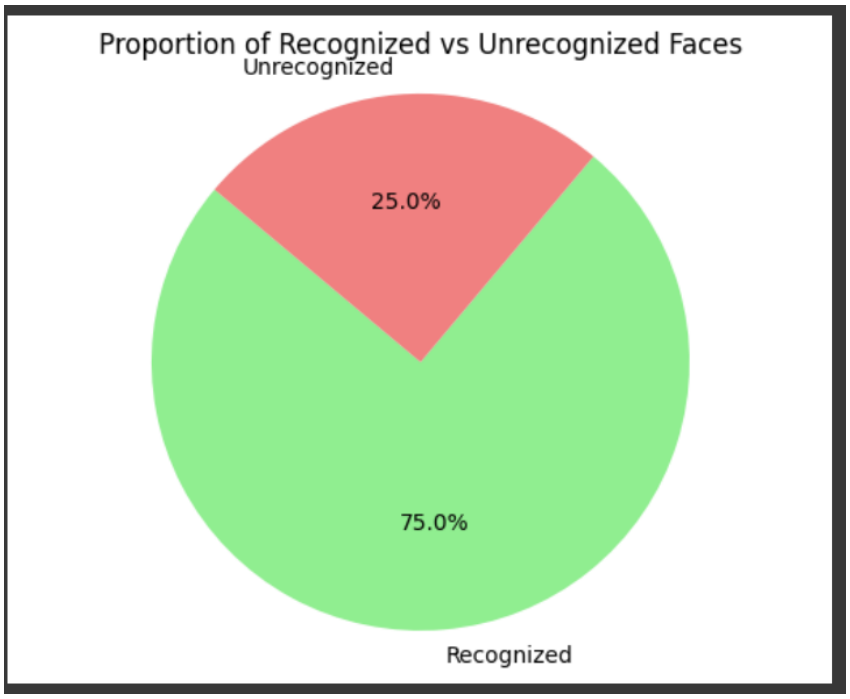
Multiple Face Detection



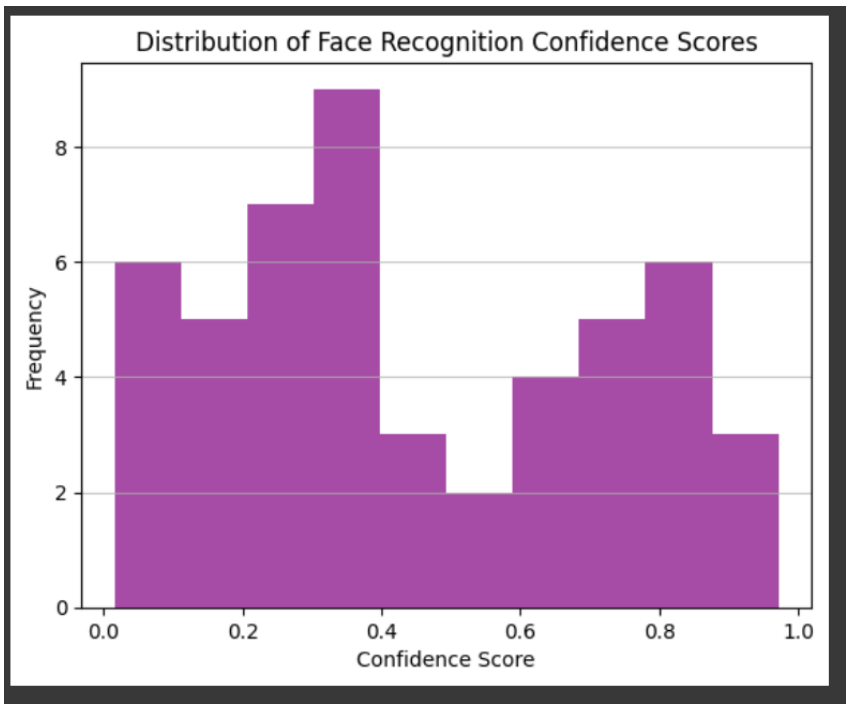
Single Face Detection



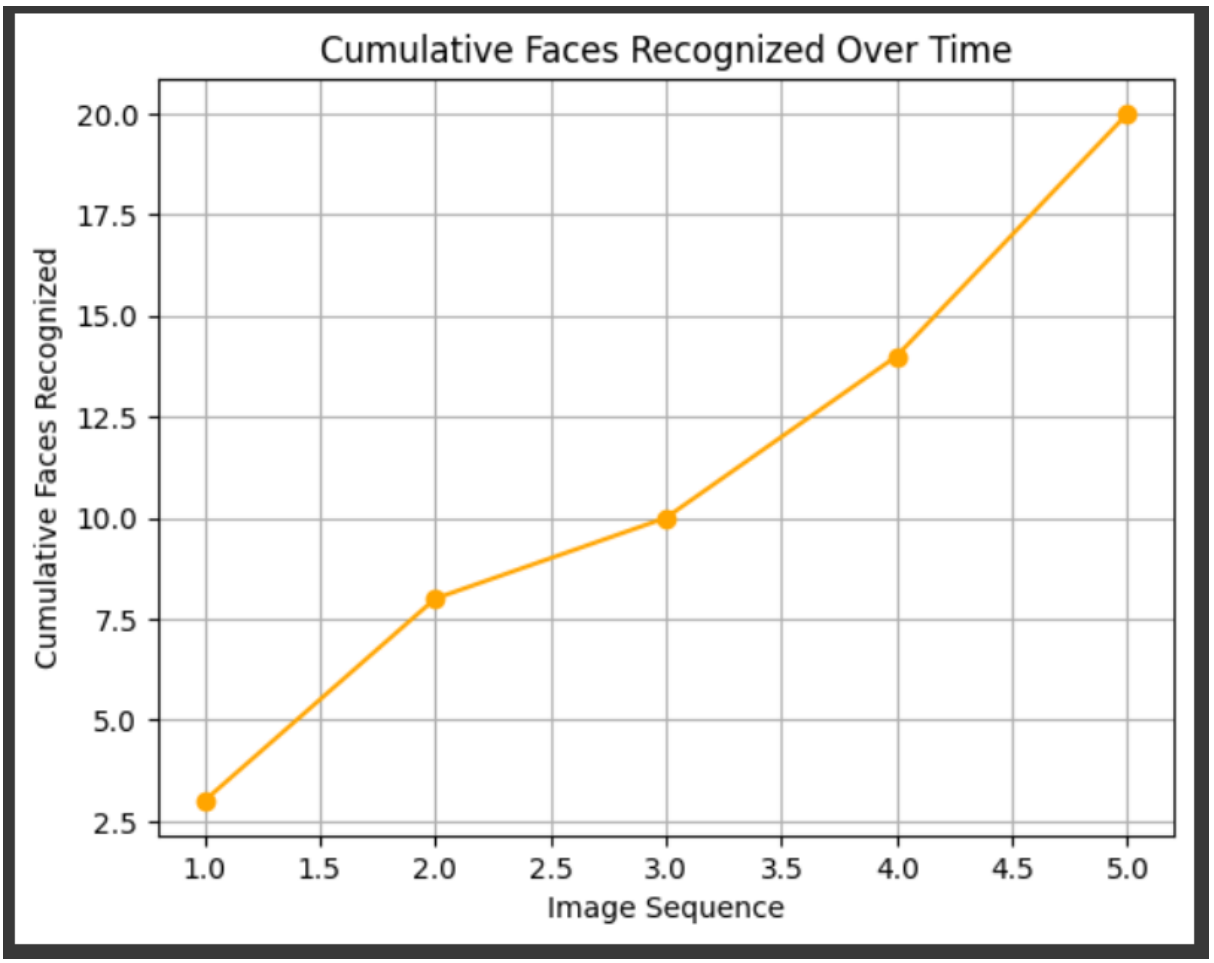
Faces Detected in each Image



Proportion of Recognized vs Unrecognised Faces



Distribution of Face Recognition Confidence Scores



Cumulative Faces Recognized Over Time

Chapter - 6

Future Scopes and Conclusion

6.1 Future Scopes

The advent of computer science into education increases the prospects for improvement of educational approaches as well. The development of React Native, Flask, OpenCV, Nginx, and Redis as an attendance tracking app, means a big leap toward making automatic and efficient college attendance management. With regards to the future, there are different areas of change and development that could be developed further in order to enhance the program, make it easier to use and meet the ever expanding demands of educational institutions.

1. Machine Learning for facial recognition.

Presently, OpenCV is used for face recognition but using machine learning techniques will greatly improve the precision and dependability in facial identification. The application uses deep-learning algorithms that allow the solution to improve face detection performance, despite changing light circumstances and different states of expression at runtime. These would greatly enhance the smoothness, and reduce errors, in attendance tracking.

2. Real-time Attendance Monitoring

The software could also be adapted for real-time attendance monitoring to offer rapid understanding of attendance statistics. Through this tool, teachers and administrators would be able to see real-time data about attendance that will necessitate subsequent actions by some students. Finally, real-time monitoring ensures that education is as vibrant and responsive as possible.

3. Enhanced Analytics and Reporting

By extending the analytical and reporting capacity of the app, more useful information regarding attendance patterns, engagement during classes, or individual growth across semesters could be obtained. The app can include a lot of data visualisation tools coupled with configurable reports to enable educators and administrators to make informed decisions and spot trends that will facilitate focused intervention strategies where necessary.

4. Mobile Application Enhancements

Continuous enhancement of the mobile application is vital for delivering a flawless user experience. Future advancements could focus on enhancing the user interface, incorporating user feedback, and optimising performance. Additionally, researching features such as push notifications for key announcements, gamification components to encourage attendance, and support for many languages can boost the overall user engagement.

6. Scalability and Cloud Integration

To meet the possible growth in user base and data volume, the app's architecture can be improved for scalability. Integrating cloud services can provide the flexibility needed to handle greater demands, guaranteeing smooth operation during peak times. This scalability can be particularly advantageous for colleges with different sizes and attendance needs.

The future scope of the attendance recording app extends beyond its existing capabilities, enabling chances for innovation and refinement. By incorporating new technology, increasing user experience, and tackling developing difficulties in attendance management, the app can play a crucial role in transforming how educational institutions approach and execute attendance tracking. As the app continues to improve, it has the potential to become a holistic solution that not only streamlines attendance registration but also helps to the overall efficiency and efficacy of educational operations.

6.2 Conclusion

Thus, the Attendance Recording App, based on React Native, Flask,OpenCV, Nginx and Redis, becomes one of the major stages towards simplified and convenient attendance accounting in the schools. Throughout the evolution of this project, it clearly shows that technology can change some obsolete approaches to bring forth what might be considered futuristic attendance monitoring practices that are no longer exclusively automated; they are intelligent and reactive.

It is crystal clear, if one looks back on the successes of the previous year, that this app is more than just a static solution; rather, it represents a dynamic, evolving platform of limitless capacity. While the proposed future scope of the project targets an advancement beyond implementation, incorporating state-of-the-art technology and innovative features for enhanced functionality and better usability.

Facial recognition using machine learning will enhance accuracy and make the attendance tracking more believable in various environments. Real time monitoring introduces an element of urgency and the latter follows how hectic learning environments are. The integrated enhanced analytics and reporting and blockchain technologies also bring to fore more information for increased transparency and data secrecy in the education environment becoming relevant currently.

This exceptional system boasts of an improved mobile app which serves as the doorstep for the users and is both attractive and interactive. This comprises gamification elements, multi-lingual support, and push alerts in order to promote user engagement and happiness.

Its design is flexible enough and can fit all education establishments, ranging from small colleges to larger universities since it can be used in the cloud. This versatility ensures that the application remains a relevant tool as efficient devices adjust and adapt to the ever-changing society it was meant to benefit.

Therefore, the future of attendance management depends on integrating innovative technologies, user-focused designs and meeting constantly changing needs of schools. It is just one step towards the future where attendance tracking will be more than just an administration routine, but rather, it will boost success and efficiency of classroom tasks. The year one’s anniversary will be celebrated, and it will be accompanied by a promising and revolutionary journey that involves innovation as well as learning within the attendance management sector.

References

[1] React-Native Based Mobile App for Online Experimentation 2019 IEEE International Conference on Service Systems and Service Management (ICSSSM) - https://ieeexplore.ieee.org/document/9189636

[2] React Native for Android: Cross-Platform Mobile Application Development International Journal of Computer and Communication Technology (IJCCT), 2022https://ouci.dntb.gov.ua/en/works/4bwKvMWl/

[3] Geolocation APIs for React Native, Medium 2018 https://medium.com/hackernoon/react-native-basics-geolocation-adf3c0d10112

[4] React Native Geolocation: A Complete Tutorial, LogRocket, 2022https://docs.logrocket.com/reference/react-native

[5] Designing and Implementing RESTful APIs with Flask, 2015 https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

[6] Redis++: A High Performance In-Memory Database Based on Segmented Memory Management and Two-Level Hash Index, IEEE Xplore 2018 https://ieeexplore.ieee.org/document/8672254

[7] Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis, IEEE Xplore 2016 https://ieeexplore.ieee.org/document/7847138

[8] Redis Enterprise: An Overview, RedisLabs, https://redis.io/docs/about/redis-enterprise/

[9] Face Detection using Faster R-CNN, IEEE Xplore 2017 https://ieeexplore.ieee.org/document/7961803

[10] Robust Real-time Face Detection via Convolutional Neural Networks, International Journal of Computer Vision, 2017 https://paperswithcode.com/search?q=author%3AXiao+Yang

[11] OpenFace: A Unified Deep Learning Framework for Face Recognition and Community Identification, IEEE Transactions on Pattern Analysis and Machine Intelligence 2016

[12] A CNN Family for Face Recognition Omron Research, Department of Commerce, University of Oxford 2022

[13] Performance Evaluation of Different Virtual Machine Configurations in Microsoft Azure, Journal of Cloud Computing 2021

[14] Cost-Effective Virtual Machine Placement in Microsoft Azure for Delay-Sensitive Applications, IEEE Transactions on Cloud Computing, 2020 https://ieeexplore.ieee.org/ielaam/5/8789751/8772112-aam.pdf?tag=1

[15] A Performance Analysis of Nginx and Apache Web Servers, International Journal of Computer Science and Information Technology (IJCSIT) 2017

[16] React Native https://reactnative.dev/

[17] Flask https://flask.palletsprojects.com/en/3.0.

[18] Javascript https://www.w3schools.com/js/

[19] Python https://www.python.org/

[20] Nginx https://www.nginx.com/

[21] Azure https://azure.microsoft.com/en-in

[22] OpenCV https://opencv.org/

[23] GeeksForGeeks https://www.geeksforgeeks.org/