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

Dr. 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, Solan 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 May 2024 under the supervision Dr. Maneet Singh (Assistant Professor(SG), Department of Computer Science & Engineering And Information Technology).

Devansh Batra	**Malay Acharya**
(201447)	**(201476)**

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

Dr. Maneet Singh

Assistant Professor (SG)

Computer Science and Engineering and Information Technology

Jaypee University of Information Technology, Solan

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, Solan is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Maneet Singh** (Assistant Professor (SG), 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.

Devansh Batra **Malay Acharya**

(201447) **(201476)**

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

Dr. Maneet Singh

Assistant Professor (SG)

Computer Science and Engineering and Information Technology

Jaypee University of Information Technology, Solan

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 **Dr. Maneet Singh, Assistant Professor**, Department of CSE Jaypee University of Information Technology, Solan. 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 **Dr. 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 noninstructing, 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 **Malay Acharya**

(201447) **(201476)**

TABLE OF CONENTS

LIST OF FIGURES

ABSTRACT

Visage Verify, 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.

Visage Verify 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 Visage Verify, allowing for rapid development and easy connection with other technologies. With Flask, the backend of Visage Verify 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.

Utilizing AWS API Gateway, Visage Verify's attendance recording system gains a potent hosting solution. With AWS API Gateway's robust features and scalability, Visage Verify ensures seamless handling of varying workloads, effectively meeting the dynamic demands of a university setting. This integration underscores the system's adaptability and reliability in managing attendance records with precision and efficiency.

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 AWS REKOGNITION

Our project stands out by leveraging AWS Rekognition for facial recognition, marking a significant technological advancement beyond conventional attendance tracking methods. AWS Rekognition, a powerful tool in the realm of computer vision, enhances the precision and reliability of our application. Not only does facial recognition bring practical advantages, but it also introduces an unmatched level of security, ensuring that attendance processes are seamless and exceptionally secure. With AWS Rekognition, both teachers and students can easily and swiftly verify their attendance, streamlining the process and providing individualized recognition.

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 TRACKING SYSTEMS

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 EVALOVING 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 ENAHNCED 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

The opening chapter of our project report, contains the overview of the whole project. The Introduction chapter includes the complete overview of the project, the opening chapter also contains the problem statement, it also contains the main usage of the project, its advantages, use cases. The motivation for the team to work on this project is also mentioned in the Introduction chapter highlighting its significance as well.

1.5.2 CHAPTER 2: LITERATURE SURVEY

The opening chapter of our project report, contains the overview of the whole project. The Introduction chapter includes the complete overview of the project, the opening chapter also contains the problem statement, it also contains the main usage of the project, its advantages, use cases. The motivation for the team to work on this project is also mentioned in the Introduction chapter highlighting its significance as well.

1.5.3 CHAPTER 3: SYSTEM DEVELOPMENT

The required technological stacks, deployment servers, programming languages etc are covered in this chapter. This chapter covers the major infrastructure details of the system, including the hardware, software, and networking components. It also provides a pseudo code of the system's main functions and a diagram of the database structure. The necessary images needed for understanding the flow of data, networking etc. are displayed in this chapter of the report. It gives us the in depth understanding of the whole technological need of the project.

1.5.4 CHAPTER 4: TESTING

This chapter covers the testing details of the system, including the methodology used and the results of the tests.

1.5.5 CHAPTER 5: RESULTS AND EVALUATION

The results of the project are shown in this section of the report. It contains all the findings and the progress of the report. In this section we have also compared our findings and work with already existing applications to give an outline of what else needs to be done in the project.

1.5.6 CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

In this part of the report the conclusion that can be derived from the work are discussed, all the findings and results are discussed here. The future scope of this project discusses what else is left to be done and the improvements that can be done in the already created project.

CHAPTER 2: LITERATURE SURVEY

2.1 Overview of Relevant Literature

In today's digital landscape, the ubiquity of smartphones and tablet PCs has led to a surge in demand for mobile applications facilitating remote data experimentation. React Native has emerged as a prominent cross-platform mobile app development framework, catering to the need for versatile solutions compatible with both iOS and Android devices. This conceptualization and implementation of an electronic mobile application for conducting online experiments using React Native present a seamless avenue for individuals to engage in experimental pursuits conveniently. [1]

The design ethos of this application revolves around simplicity, ensuring accessibility for users of varying technical proficiencies. Its core functionalities encompass:

1. **User Login and Registration**: Providing a secure gateway for users to create accounts, enabling them to save and manage their experimentation data effectively.

2. **Experiment Selection**: Offering a menu-driven interface for users to browse and select experiments of interest, facilitating a streamlined user experience.

3. **Experiment Observation**: Empowering users to track and monitor their experimental endeavours in real-time, fostering a deeper engagement with the process.

4. **Experiment Documentation**: Furnishing comprehensive documentation for each experiment, including detailed descriptions, methodologies, and results, aiding users in understanding and analysing their findings.

5. **Algorithm Download and Configuration:** Facilitating the download and customization of algorithms tailored to users' research requirements, enhancing the flexibility and adaptability of the application to diverse experimental scenarios.

By leveraging the capabilities of React Native, this application not only bridges the gap between traditional experimentation methods and modern mobile technology but also democratizes access to scientific inquiry by providing a user-friendly platform for conducting experiments remotely. Its intuitive design and robust feature set position it as a valuable tool for researchers, educators, and enthusiasts alike, facilitating the seamless integration of digital experimentation into everyday workflows.

React Native emerges as a beacon of innovation in the realm of app development, offering a compelling alternative for crafting robust applications that seamlessly traverse the diverse landscape of operating systems, notably iOS and Android. This paper delves into the intricacies of React Native specifically tailored for Android development, conducting a meticulous examination of its foundational elements, discernible strengths, and potential weaknesses, thereby providing invaluable insights for developers navigating the ever-evolving terrain of mobile app creation. [2]

Central to the allure of React Native is its adoption of a declarative programming paradigm, imbuing the development process with a sense of clarity and conciseness that fosters code maintainability and enhances developer productivity. Leveraging JavaScript, a ubiquitous programming language renowned for its versatility and ease of use, further augments the accessibility of React Native, lowering the entry barrier for developers and catalysing the adoption of this framework across the development community.

At the core of React Native's architecture lies its component-based structure, heralding a paradigm shift in app development methodology. By encapsulating discrete functionalities into reusable components, React Native not only expedites the development process but also engenders a modular ecosystem conducive to code reuse and extensibility. This inherent modularity not only enhances development efficiency but also augments code maintainability, facilitating seamless collaboration among developers.

A distinguishing hallmark of React Native is its incorporation of hot reloading functionality, a game-changing feature that revolutionizes the iterative development workflow. By enabling instantaneous updates to the application state without the need for manual recompilation, hot reloading streamlines the code debugging process and expedites iteration cycles, thereby fostering a more iterative and agile development methodology.

Furthermore, React Native's seamless integration with native Android modules empowers developers to harness the full spectrum of platform-specific capabilities, transcending the confines of cross-platform development frameworks. This symbiotic relationship with native Android APIs bestows developers with unprecedented flexibility and versatility, enabling the creation of immersive and feature-rich applications that leverage the full potential of the Android ecosystem.

In summary, React Native emerges as a transformative force in the realm of Android app development, offering a potent blend of efficiency, accessibility, and versatility that empowers developers to craft immersive and feature-rich applications with unparalleled ease and agility. While its strengths are undeniable, a nuanced understanding of its potential limitations is essential for developers to navigate the complexities of app development effectively and harness the full potential of this groundbreaking framework.

The realm of geolocation in React Native is a rich landscape, offering developers a plethora of options to tailor location-based functionalities to their specific application needs. This article provides a comprehensive overview of various geolocation APIs, ranging from the foundational native Geolocation API to third-party libraries like React Native Geolocation Service and React Native Background Geolocation, each catering to distinct use cases and requirements. [3]

At its core, the native Geolocation API serves as a foundational tool for fulfilling simple location-based tasks, offering essential functionalities to retrieve device location data. However, for developers seeking more advanced features such as background geolocation tracking and geofencing, third-party libraries like React Native Geolocation Service and React Native Background Geolocation emerge as indispensable allies. These libraries augment the capabilities of the native API, providing developers with a robust suite of tools to implement complex geolocation functionalities seamlessly.

The selection of a geolocation API hinges upon the specific needs and objectives of the application. For simple location-based tasks, the native Geolocation API may suffice, offering a lightweight and straightforward solution. Conversely, for applications requiring advanced features such as continuous background geolocation updates or geofencing capabilities, third-party libraries offer a more comprehensive and feature-rich alternative.

In conclusion, the breadth of geolocation APIs available to React Native developers underscores the versatility and adaptability of the framework in catering to diverse application requirements. While each API comes with its own set of pros and cons, the optimal choice ultimately depends on the nuanced needs and objectives of the application at hand. By leveraging the appropriate geolocation API, developers can unlock the full potential of location-based functionalities in their React Native applications, enriching the user experience and enhancing the overall functionality of their creations.

In today's dynamic landscape of React Native development, geolocation functionalities play a pivotal role in enhancing the user experience and expanding the capabilities of mobile applications. This article offers a comprehensive exploration of the diverse array of geolocation APIs available to React Native developers, encompassing both native solutions and third-party libraries such as react-native-geolocation-service and React Native Background Geolocation. [4]

The native Geolocation API serves as a fundamental building block for incorporating basic location-based features into React Native applications. With its simplicity and efficiency, it caters to straightforward use cases, providing developers with essential tools for retrieving device location data and implementing simple location-related tasks.

However, for applications requiring more advanced geolocation functionalities, third-party libraries emerge as invaluable assets. React Native Geolocation Service and React Native Background Geolocation extend the capabilities of the native API, offering a myriad of advanced features such as background geolocation tracking and geofencing. These libraries empower developers to create immersive and feature-rich applications that leverage the full potential of geolocation technology.

The selection of a geolocation API hinges upon the specific needs and objectives of the application. While the native Geolocation API suffices for simple location purposes, third-party libraries provide a more extensive suite of functionalities tailored to diverse use cases. Developers must carefully assess the requirements of their application and choose the most appropriate API to meet their needs effectively.

In conclusion, the breadth of geolocation APIs available to React Native developers underscores the platform's flexibility and adaptability in catering to diverse application requirements. Each API comes with its unique set of advantages and limitations, necessitating thoughtful consideration in the selection process. By leveraging the vast array of geolocation APIs at their disposal, developers can unlock new possibilities and elevate the functionality of their React Native applications to new heights.

In the ever-evolving landscape of web development, the utilization of RESTful APIs has become increasingly indispensable, facilitating seamless communication between client-side applications and server-side resources. This comprehensive guide delves into the intricacies of building and harnessing RESTful APIs using Flask, a lightweight and flexible Python web framework renowned for its simplicity and versatility. [5]

At the heart of effective RESTful API design lies a deep understanding of fundamental concepts such as resource-oriented URIs, standardized interfaces, and adherence to the stateless protocol. By adhering to these principles, developers can create APIs that are not only simple and intuitive but also highly manageable and scalable, laying the foundation for robust and resilient applications.

The guide delves into the nuances of Flask API development techniques, offering insights into designing APIs, creating routes, processing requests, and delivering results. From defining endpoints to handling HTTP methods and status codes, developers are equipped with the knowledge and tools necessary to architect APIs that meet the diverse needs of modern web applications.

In addition to exploring the mechanics of API development, the guide delves into best practices for designing strong and scalable APIs. Concepts such as error handling, documentation, and versioning are elucidated, providing developers with strategies for mitigating errors, enhancing API usability, and ensuring compatibility across different iterations.
Despite the myriad of frameworks available for API development, Flask has emerged as a standout choice for its lightweight nature and flexibility. Its minimalist approach empowers developers to focus on crafting elegant and efficient APIs without unnecessary overhead,

making it particularly well-suited for building RESTful APIs that prioritize simplicity and performance.

In conclusion, this guide serves as an invaluable resource for software engineers seeking to delve into the intricacies of designing and implementing RESTful APIs with Flask. By providing a comprehensive overview of design principles, development techniques, and best practices, it equips developers with the knowledge and skills necessary to create robust and scalable APIs that meet the demands of modern web applications. With a straightforward outline and practical case studies, developers are empowered to embark on their journey of API development with confidence and proficiency.

In the realm of high-performance in-memory databases, Redis++ emerges as a groundbreaking solution that addresses two critical challenges plaguing its predecessor, Redis: memory fragmentation and cache misses. By leveraging innovative strategies such as segmental memory management and a two-level hash index structure, Redis++ achieves unparalleled efficiency and performance, setting a new standard for in-memory database systems. [6]

The segmented memory management system employed by Redis++ represents a significant departure from traditional memory allocation approaches. By dynamically assigning and removing fixed-size memory blocks, Redis++ optimizes memory utilization and minimizes fragmentation, ensuring optimal resource allocation and mitigating the risk of memory waste. This strategic allocation of memory space not only enhances efficiency but also contributes to the overall stability and robustness of the database system.

The cumulative impact of these advancements is reflected in Redis++'s superior performance metrics, including enhanced memory efficiency, lower latency response times, and improved throughput. By outperforming Redis on multiple fronts, Redis++ emerges as a compelling candidate for building advanced high-speed in-memory databases, capable of meeting the demanding requirements of modern applications and workloads.

In conclusion, Redis++ represents a significant leap forward in the realm of in-memory database systems, offering a potent combination of performance, efficiency, and scalability. Its ability to effectively address memory fragmentation and cache misses positions it as a formidable contender in the landscape of high-speed database solutions. As organizations increasingly prioritize speed and efficiency in their data management strategies, Redis++ stands poised to emerge as a leading choice for powering mission-critical applications and services.

In the pursuit of scalability and reliability within the domain of in-memory key-value databases, Redis has garnered significant attention. However, as data volumes grow and the demand for data integrity intensifies, it becomes imperative to augment Redis's capabilities.

In this context, this paper delves into two novel techniques: Client-Side Key-to-Node Caching and Master-slave Semi Synchronisation, aimed at enhancing Redis's scalability and ensuring data integrity. [7]

The Client-Side Key-to-Node Caching technique represents a paradigm shift in addressing communication costs between clients and nodes. By embedding key-to-node mappings within client applications, this technique streamlines request routing, ensuring that each request is directed to its respective service node. This alleviates the overhead associated with node discovery routing, thereby enhancing overall system efficiency, particularly in scenarios involving high request volumes or complex routing configurations.

In parallel, the Master-slave Semi Synchronisation technique leverages the TCP protocol to bolster data integrity across master-slave nodes within the Redis ecosystem. By implementing semi-synchronous replication mechanisms, this technique ensures that data consistency is maintained even in the event of node failures or network disruptions. The use of TCP protocol facilitates robust data synchronization, thereby mitigating the risk of data inconsistencies and bolstering overall system reliability.

Experimental validation of these techniques underscores their efficacy in addressing scalability and reliability concerns within Redis-based systems. While Client-Side Key-to-Node Caching exhibits a notable increase in request latency and performance, particularly in resource-constrained environments or during periods of high application load, Master-slave Semi Synchronisation offers a compelling solution for ensuring data consistency with reasonable performance overhead.

In conclusion, this paper presents actionable insights into addressing scalability and dependability challenges within Redis-based systems. By elucidating the principles and methodologies underlying Client-Side Key-to-Node Caching and Master-slave Semi Synchronisation, it equips practitioners with tangible strategies for improving the performance and reliability of Redis deployments, especially in the context of large-scale and mission-critical applications. As organizations strive to harness the power of in-memory databases for their data-intensive workloads, the adoption of these techniques represents a significant step towards realizing the full potential of Redis in modern computing environments.

Redis Enterprise stands as the commercial evolution of the renowned open-source Redis datastore, offering a suite of advanced features tailored to meet the demands of enterprise-grade use cases. By augmenting Redis's core functionalities, Redis Enterprise enhances scalability, reliability, and security, making it an ideal choice for organizations with stringent performance and latency requirements. [8]

At the forefront of Redis Enterprise's capabilities is its clustering feature, which enables horizontal scaling by orchestrating multiple Redis nodes into a cohesive cluster. This distributed architecture not only expands available memory but also accelerates data processing speeds, ensuring optimal performance even under heavy workloads.

In addition to scalability, Redis Enterprise prioritizes high availability through robust data replication and failover mechanisms. By seamlessly replicating data across nodes and orchestrating failover procedures, Redis Enterprise guarantees data reliability and continuity, mitigating the risk of data loss or downtime in the event of node failures.

Furthermore, Redis Enterprise offers comprehensive data management capabilities, including encryption, access control, and auditing functionalities. These mechanisms empower organizations to safeguard sensitive data and adhere to regulatory compliance requirements, bolstering data security and integrity across the enterprise.

To facilitate operational oversight and performance monitoring, Redis Enterprise incorporates a built-in monitoring and alerting system. This feature enables administrators to track the health and performance of Redis clusters in real-time, proactively identifying potential issues and optimizing resource utilization for enhanced efficiency.

Key findings underscore Redis Enterprise's prowess as a scalable, reliable, and secure in-memory data store solution, making it a compelling alternative for organizations seeking high-performance data storage with low latency requirements. RedisLabs complements Redis Enterprise with comprehensive training, consulting, and support services, ensuring seamless deployment and ongoing maintenance for enterprise customers.

In conclusion, Redis Enterprise emerges as a robust and highly customizable in-memory data store solution tailored for enterprise use cases. With its advanced features, including clustering, high availability, data management, and monitoring capabilities, Redis Enterprise offers organizations a powerful tool to meet their performance and latency demands while ensuring data integrity and security. As organizations continue to prioritize digital transformation and data-driven decision-making, Redis Enterprise stands poised to play a pivotal role in enabling their success.

Computer vision encompasses a myriad of critical tasks, extending far beyond its traditional boundaries, with face identification standing as a prominent example. However, conventional face identification systems often rely on hand-engineered attributes, which may exhibit vulnerabilities in terms of accuracy and robustness over time. In response to these challenges, this study proposes a novel approach to face detection utilizing Faster R-CNN, a state-of-the-art regional-based convolutional neural network (CNN) model.
Faster R-CNN represents a significant advancement in object detection systems, characterized by its dual-phase architecture comprising the Region Proposal Network (RPN) and the Fast R-CNN detector. The RPN generates potential object bounding boxes, which are subsequently classified and refined by the Fast R-CNN. Compared to earlier iterations of R-CNN, Faster R-CNN offers remarkable speed improvements while maintaining

comparable levels of accuracy, making it an attractive choice for real-world applications. [9]

In the context of face detection, the authors leverage a vast dataset comprising over 800,000 images sourced from the PASCAL VOC 2010 and WIDER datasets, along with approximately 1.5 million labelled faces. Through extensive training on this dataset, the Faster R-CNN model is primed for face detection tasks. To evaluate its performance, the model is subjected to rigorous testing using benchmark databases such as FDDB and IJB-A, widely recognized in the field of face detection evaluation.

The results of the evaluation demonstrate the efficacy of the proposed approach, with the Faster R-CNN model achieving state-of-the-art performance on both the FDDB and IJB-A benchmarks. Notably, the model achieves a false detection rate of 0.1 at a detection rate of 90% on the FDDB benchmark and a Verification Error Rate (VER) of 10.7% on the IJB-A benchmark. These results underscore the superior performance and reliability of the Faster R-CNN model in face detection tasks.

In conclusion, the study highlights the effectiveness of Faster R-CNN as a robust method for face recognition, yielding top-tier results on benchmark datasets such as FDDB and IJB-A. While acknowledging potential trade-offs in favour of newer facial detection algorithms, the authors assert the significance of their approach in pushing the boundaries of performance in face detection tasks.

Computer vision encompasses a diverse array of applications, spanning from surveillance and human-computer interaction to medical imaging analysis and beyond. Within this expansive field, face recognition plays a pivotal role, enabling the identification of individuals based on facial photographs. However, traditional face recognition mechanisms reliant on handcrafted features often exhibit vulnerabilities in terms of accuracy and robustness. [10]

To address these challenges, this study introduces an innovative method for face detection leveraging Convolutional Neural Network (CNN) architecture, which has demonstrated effectiveness in various computer vision tasks, including face detection.

The proposed CNN architecture comprises three stages: feature extraction, feature fusion, and classification. In the feature extraction stage, input images undergo convolutional layers to extract relevant information. Subsequently, features from different layers are fused to generate a discriminant representation. Finally, a fully connected layer is employed for classification, determining whether an input image contains a face.

Training the CNN model involves utilizing a large dataset of facial images, sourced from Multi-PIE and FRGC databases, comprising over 200,000 images and 300,000 labelled facial points. To enhance model robustness, the authors employ various data augmentation techniques, ensuring the model's resilience to diverse facial characteristics and environmental conditions.

Evaluation of the proposed model is conducted against established benchmarks, including FDDB and IJB-A, widely recognized standards for face detection assessment. The model achieves a false detection rate (FDR) of 0.7 and detects 90% of items on the FDDB benchmark, while achieving a Verification Error Rate (VER) of 14.1% on the IJB-A benchmark.

In conclusion, the study demonstrates the efficacy of the proposed CNN architecture in addressing the face identification problem, achieving record results on both the FDDB and IJB-A benchmarks. By pushing the boundaries of performance in facial detection applications, the proposed method opens up new possibilities and avenues for innovation in the field of computer vision.

OpenFace stands as a formidable solution in the realm of facial recognition and community organization, harnessing the power of deep learning to facilitate a myriad of applications. Its comprehensive framework comprises three integral components, each contributing to its robust functionality. Firstly, the feature extraction module serves as the foundation, delving into facsimile photos to extract deep features. These features encapsulate a diverse array of physical traits, culminating in higher-dimensional representations essential for precise facial identification. [11]

Next, the face recognition component takes center stage, meticulously analyzing facial characteristics to discern individuals within older photographs. This capability extends beyond mere recognition, enabling the verification of known identities as well as the identification of unknown individuals from extensive databases of faces.

Complementing these core functions is the community identification module, a feature-rich tool that partitions faces into cohesive groups based on an assortment of discernible features. This facet facilitates the classification of familial relations and resemblances, fostering intuitive categorization and organization.

Central to OpenFace's efficacy is its utilization of a meticulously crafted pre-trained model, meticulously honed on a vast dataset comprising over 9 million facial images. This model boasts remarkable adaptability, capable of extracting facial features even under varied lighting conditions, provided at least 70 percent of the image remains clear and discernible. Central to the prowess of OpenFace is its pre-trained model, meticulously honed on a corpus of over 9 million facial images. This repository of knowledge endows the model with unparalleled adaptability, enabling the extraction of facial features amidst the labyrinth of varied lighting conditions, provided at least 70 percent of the image remains clear and unobscured.

In conclusion, OpenFace emerges not only as a bastion of technological innovation but also as a harbinger of transformative change across diverse domains. Its fusion of deep learning principles with open-source ethos not only democratizes access to cutting-edge technology but also paves the way for a multitude of applications, spanning realms as varied as security, law enforcement, social media, and marketing. In an ever-evolving landscape, OpenFace stands as a testament to the boundless potential of human ingenuity, ushering in a new era of exploration and discovery.

From the Visual Geometry Group (VGG) at the University of Oxford, a notable advancement in convolutional neural networks (CNNs) has emerged, commonly known as VGGFace. Rooted in the foundational VGG-16 architecture, originally tailored for image classification, VGGFace has undergone meticulous refinement and enhancement specifically tailored for facial recognition tasks. [12]

VGGFace presents three primary architectures: VGGFace1, VGGFace2, and Ret VGGFace. While sharing a fundamental framework, each iteration varies in complexity and depth. VGGFace1, the most rudimentary, comprises 16 layers, while VGGFace2 delves deeper with 20 layers. Notably, Ret VGGFace represents an evolution, integrating advanced techniques like batch normalization and dropout for improved performance.

Training VGGFace necessitated vast datasets sourced from the internet, encompassing approximately 2.6 million images spanning over two thousand individuals. Evaluation against benchmark datasets such as LFW and YTF underscored the prowess of these models. The results speak volumes: VGGFace models consistently outperform benchmarks. VGGFace1 boasts an impressive face verification accuracy of 95.1% on the LFW dataset, while VGGFace2 achieves a remarkable 90.2% on facial recognition tasks using the YTF dataset.

In conclusion, VGGFace stands as a preeminent choice for facial recognition endeavors. Its lineage, rooted in the VGG16 architecture, coupled with its demonstrated state-of-the-art performance on benchmark datasets, positions it as an invaluable asset for real-world applications. Moreover, its relative ease of training renders it accessible and practical for a wide array of practical implementations.

Virtual machines (VMs) constitute a foundational pillar of cloud computing infrastructure, empowering users with the flexibility to deploy and manage applications in a scalable and cost-efficient manner. Microsoft Azure, renowned for its comprehensive cloud services, offers a diverse array of VM configurations tailored to accommodate a spectrum of user requirements and workload demands. However, navigating the myriad choices to pinpoint the optimal VM configuration entails a delicate balance between performance, cost-effectiveness, and resource utilization. [13]

Methodologically, the study adopts a systematic approach to dissect the performance characteristics of distinct VM setups. Three representative VM sizes—Standard_A2_v2, Standard_D2s_v3, and Standard_E2s_v3—are selected to encapsulate a spectrum of CPU, memory, and storage capabilities. Each VM size is further subjected to iterative adjustments in CPU core counts and RAM allocations to delineate alternative configurations. Additionally, the investigation scrutinizes the performance implications of employing both ordinary SSD and premium SSD storage variants.

The results unearth a landscape rife with divergent performance outcomes across various VM configurations. Notably, CPU cores emerge as a pivotal determinant, showcasing substantial performance gains with escalating core counts, particularly in CPU-bound workloads. Memory allocation assumes a significant role, particularly in the realm of memory-intensive applications, exerting discernible influence on overall performance metrics. Conversely, while storage type does exhibit an impact on performance, the disparities between premium SSDs and regular SSDs are relatively modest, with the former conferring incremental benefits.

In summation, this study illuminates the intricate interplay between CPU cores, memory allocation, and storage type in shaping the performance landscape of Microsoft Azure VMs. By elucidating these dynamics, it empowers users and decision-makers with valuable insights to inform judicious VM configuration choices that align with their specific application requirements and operational objectives.

Cloud computing owes much of its functionality and cost-effectiveness to the indispensable role played by virtual machines (VMs), which afford users the flexibility and affordability of deploying and managing applications. Among the leading cloud service providers, Microsoft Azure stands out, offering a plethora of VM configurations tailored to meet diverse user expectations and task requirements. Nonetheless, navigating the multitude of options to select the optimal VM configuration poses a formidable challenge, necessitating careful consideration of factors such as performance, expenses, and resource utilization. [14]

Against this backdrop, this paper embarks on an exploration of the efficacy of various VM configurations within the Microsoft Azure ecosystem, with a particular focus on the impact of CPU cores, allocated memory capacities, and disk types on application performance. Employing a renowned benchmarking tool, the study scrutinizes CPU, memory, and disk I/O workloads, delving into response times and throughput parameters to glean insights into performance dynamics.

Methodologically, the study adopts a systematic approach, meticulously evaluating VM setups to discern their performance characteristics. Three representative VM sizes—Standard_A2_v2, Standard_D2s_v3, and Standard_E2s_v3—are chosen to encapsulate a spectrum of CPU, memory, and storage capabilities. By varying the number of CPU cores and RAM allocations across each VM size, a range of alternative configurations is generated, further augmented by an examination of storage types, including conventional SSD and advanced SSD variants.

The findings underscore significant performance differentials across VM configurations, with CPU cores emerging as a pivotal determinant, particularly in CPU-bound workloads where increased core counts yield substantial advantages. Memory allocation also proves pivotal, especially for memory-intensive applications, exerting a discernible impact on overall performance metrics. Conversely, while storage type does exert influence, the disparities between premium SSDs and regular SSDs are relatively marginal, with the former demonstrating slight advantages in most scenarios.

In conclusion, the study advocates for a cost-effective approach to VM deployment tailored for delay-sensitive applications within the Microsoft Azure environment. The proposed methodology showcases the ability to mitigate latency while maintaining cost-effectiveness, rendering it a valuable resource for cloud customers seeking to implement delay-sensitive applications with optimal efficiency and performance.

In this study, the comparative analysis was focused solely on the two predominant open-source web servers, Nginx and Apache, with an aim to evaluate their performance across key metrics such as server reaction time, throughput (number of packets forwarded), and memory utilization under varying load conditions. The findings unequivocally demonstrated Nginx's superiority over Apache across all three parameters, thereby underscoring its suitability for websites and applications contending with higher traffic volumes. [15]

Key findings from the study unveiled a consistent advantage held by Nginx in terms of response time, regardless of the level of load imposed. Moreover, Nginx showcased a remarkable capability to handle more concurrent requests, leading to significantly higher throughput compared to Apache. Notably, even under conditions of heightened load, Nginx exhibited a propensity to operate with lower RAM consumption vis-à-vis Apache, further solidifying its position as a leaner and more resource-efficient option.

In conclusion, the study unequivocally establishes Nginx as the preferred choice over Apache for high-traffic websites and applications. With its superior response time rates, enhanced throughput capabilities, and efficient memory utilization, Nginx emerges as a more adept and scalable solution for web server deployments. These findings not only underscore the performance advantages offered by Nginx but also highlight its potential to deliver enhanced user experiences and operational efficiencies in demanding web hosting environments.

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.

There are several notable gaps in the current study that present opportunities for improvement and expansion. Firstly, the experimentation methodology primarily revolves around A/B testing and parameter sweeping, leaving out other valuable experiment types such as multi-armed bandits and Bayesian optimization. By incorporating these additional experiment types, the program can become more flexible and robust in its approach to experimentation. Secondly, while the study outlines the process of starting, stopping, and monitoring experiments through a mobile app, it lacks specific details and approaches for execution and monitoring. A deeper exploration of these aspects would provide a more comprehensive understanding of the app's performance during experimentation. Thirdly, while data synchronization between the mobile app and server is emphasized, the study does not delve into the synchronization approach employed. Furthermore, there is a gap in examining security measures to safeguard research trial data from unauthorized access or modification. Fourthly, although the article includes some visuals of the mobile app's user interface, there is limited discussion on user experience (UX) review. Expanding research on UI/UX, user feedback, and usability could enhance the app's user orientation and accessibility. Finally, the focus on building a mobile app with React-Native for iOS and Android systems overlooks the potential for cross-platform compatibility. Exploring interoperability with other platforms such as web-based browsers and portable applications could broaden the app's reach and appeal to a wider audience. Addressing these gaps would contribute to the overall improvement and effectiveness of the mobile app.

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.

The study identifies several gaps that warrant further exploration and improvement in React Native development. Firstly, while React Native offers cross-platform capabilities, its performance can sometimes lag behind native Android apps. To address this, the article suggests investigating methods to enhance React Native speed, such as utilizing native modules, optimizing JavaScript code, and implementing caching solutions. Secondly, although React Native provides a wide range of UI elements, there may be occasions where developers need to integrate native Android UI components for aesthetic purposes and improved performance. Researching seamless integration techniques for native UI elements within React Native applications could enhance the overall user experience. Thirdly, React Native may hide certain Android-specific features and libraries, limiting developers' access to the full richness of the Android ecosystem. The paper advocates for exploring ways to facilitate controlled access to these Android-specific features and APIs within React Native apps, empowering developers to leverage the full potential of the Android platform. Addressing these gaps would contribute to the optimization and expansion of React Native development, enhancing its performance, aesthetics, and compatibility with the Android platform. 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.

The article highlights several areas where further investigation and improvement are needed in the realm of geolocation functionality within React Native apps. Firstly, while the article primarily focuses on basic geolocation services, future discussions should delve into more advanced features such as geofencing, location-based routing, and location-based notifications. Exploring these advanced functionalities would enhance the overall utility and user experience of geolocation-enabled apps. Secondly, considering the cross-platform nature of React Native development, future conversations should address factors affecting cross-platform compatibility, including variations in native geolocation APIs across different operating systems and leveraging platform-specific features for optimal performance. Thirdly, while the article briefly touches on performance issues, future discussions should provide more elaborate strategies for optimizing geolocation functionality in React Native apps. This could include techniques for conserving battery power, optimizing data access, and improving spatial data representation and rendering. Lastly, given the potential privacy implications of collecting geolocation data, future debates should focus on security and privacy considerations, such as obtaining user consent and implementing measures to protect user privacy. Addressing these gaps would contribute to the development of more robust, efficient, and privacy-respecting geolocation features in React Native apps.

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.

This article primarily focuses on fundamental geolocation functionalities, such as acquiring user coordinates and displaying them. However, there's a need for future discussions to explore advanced geolocation features like geofencing, location-based routing, and location-based notifications. Additionally, considering React Native's cross-platform nature, it's essential to delve into factors affecting cross-platform compatibility, including differences in native geolocation APIs and leveraging platform-specific enhancements. While the article briefly touches on performance concerns, future discussions should offer detailed optimization strategies, including battery conservation and efficient data processing. Furthermore, privacy and security considerations are crucial, necessitating discussions on obtaining user consent, secure data storage, and protection against unauthorized access. Lastly, the article could benefit from practical implementation examples, such as step-by-step tutorials and code snippets, to enhance understanding and application of geolocation features in React Native development. Addressing these gaps would enrich discussions and improve the development of geolocation functionalities in React Native applications.

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.

The article primarily focuses on fundamental concepts of HTTP methods and status codes within RESTful API architecture, omitting discussion on advanced HTTP concepts such as caching and content negotiation. Furthermore, while exception handling is briefly mentioned, there's a lack of detail on managing failures in RESTful APIs. Additionally, although the importance of data validation is underscored, specific techniques for validating and sanitizing user-supplied data are not outlined. Similarly, while authentication and authorization are briefly touched upon, there's a gap in discussing implementation methods like Basic Authentication, OAuth, and JWTs. Moreover, while testing and deployment are acknowledged as essential, the article lacks specific methodologies for testing RESTful APIs and deployment techniques such as containerization and cloud deployment. Addressing these gaps would provide developers with a comprehensive understanding and practical guidance on implementing, securing, testing, and deploying RESTful APIs effectively.

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.

The paper primarily focuses on the performance aspects of Redis++, particularly its in-memory capabilities, but it lacks discussion on durability and persistence of data. Future research should explore methods for storing persistent data on disk or other storage media, enhancing the overall robustness of Redis++. Additionally, while the paper discusses Redis++ performance, it overlooks implementation details for replication and high availability in distributed systems, which are crucial for fault-tolerant data management. Incorporating discussions on replication systems like master-slave and multi-master replication would provide a more comprehensive understanding of Redis++ capabilities. Moreover, while performance evaluation is conducted under specified workloads, there's a need to broaden the testing scope to include varying data volumes, request types, and concurrency levels to provide a comprehensive assessment of Redis++ performance across different environments. Furthermore, comparing Redis++ with existing in-memory databases like Memcached and Redis would offer insights into its strengths and weaknesses relative to other solutions. Lastly, the paper should address the open-source availability and community engagement of Redis++, as open-sourcing the project could foster wider adoption and collaboration among developers. Addressing these gaps would enhance the understanding and applicability of Redis++ in real-world scenarios.

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.

The paper highlights several gaps in the analysis and proposed solutions for scalability challenges in Redis. Firstly, there's a need for a thorough analysis of scalability bottlenecks, including the decentralised design of Redis and request processing costs, to understand their impact on productivity accurately. This would enable the development of tailored optimization mechanisms to address specific limitations effectively. Secondly, while the report proposes solutions like Client side K2N caching and MSS.1®, a comprehensive evaluation of these solutions under different load intensities and network conditions is necessary to identify the most effective approach. Additionally, future research should explore innovative scalability techniques beyond the presented optimizations. Moreover, there's a need for a more thorough investigation into reliability vulnerabilities, particularly regarding inconsistency-based issues in Redis's replication strategy, and the development of improved replication mechanisms and data consistency standards. Lastly, extending the evaluation to real-world production scenarios, including diverse workloads and network circumstances, would provide practical insights into Redis's performance, scalability, and reliability in actual deployment settings. Addressing these gaps would enhance the understanding and effectiveness of Redis in real-world scalability challenges and deployments.

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.

The report highlights several key gaps in the analysis and evaluation of Redis Enterprise's performance, features, cost, and real-world deployment scenarios. Firstly, there's a need for an in-depth performance analysis of Redis Enterprise under various workloads and conditions, including benchmarking against other open-source databases, to provide a comprehensive understanding of its capabilities in real-world production environments. Secondly, while the study outlines crucial elements of Redis Enterprise, a comprehensive feature comparison with other in-memory data stores like Memcached and Aerospike is necessary to assess its advantages and disadvantages thoroughly. Thirdly, there should be a rigorous evaluation of Redis Enterprise's enterprise-grade features, such as high availability, data durability, and security, to validate their value and reliability under operational conditions, including failure scenarios and security threats. Additionally, a detailed cost analysis and total cost of ownership evaluation, including transparent pricing and clarification of TCO components, would provide potential customers with insights into the cost-benefit considerations of deploying Redis Enterprise. Lastly, the report should include case studies and real-life deployment examples to illustrate how companies have effectively implemented Redis Enterprise in productive scenarios, offering crucial information and guidelines for further adopters. Addressing these gaps would provide a more comprehensive and objective assessment of Redis Enterprise's capabilities and suitability 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.

The identified gaps in the proposed face recognition technique underscore critical areas for further research and development. Firstly, while the technique shows promise in enhancing face recognition rates, its real-time performance and computational complexity remain challenges. Future studies should focus on optimizing the approach to enable real-time operation under resource-constrained conditions. Secondly, the technique's resilience to occlusions, pose variations, and changes in illumination needs improvement, particularly in complex scenarios. Alternative approaches should be explored to better address these challenges. Thirdly, adapting the technique to accommodate diverse face appearances and ethnic groups requires attention. Expanding the training dataset and addressing biases can enhance performance across different demographics. Additionally, integrating the proposed procedure with facial recognition and analysis tasks would further enhance its utility in comprehensive face processing workflows. Lastly, while the approach has been evaluated on benchmark datasets, its performance on larger and more diverse datasets remains to be assessed. Conducting thorough evaluations on such datasets will provide insights into the approach's applicability and effectiveness across a wide range of real-world scenarios. Addressing these gaps will contribute to the development of more robust and versatile face recognition techniques with broader applicability and improved performance in real-world applications.

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.

The identified gaps in the paper's evaluation of the face detection method highlight areas for further research and development. Firstly, while the method performs well under ideal conditions, its performance under challenging settings, such as poor lighting and partial occlusions, requires investigation. Future studies should explore the method's robustness in diverse real-world scenarios to ensure its practical applicability. Secondly, while the method achieves high accuracy, its computational cost may hinder real-time applications. Future research should focus on optimizing the method for real-time performance through network design optimizations and hardware acceleration techniques. Thirdly, the evaluation predominantly focuses on datasets featuring Caucasian faces, raising questions about the method's generalization to faces from diverse ethnicities and backgrounds. Ensuring the method's inclusivity and fairness requires investigation into its performance across a broad spectrum of face appearances. Additionally, exploring the integration of the method with face tracking and recognition tasks could enhance the development of comprehensive and accurate face processing pipelines. Finally, the availability of the method as an open-source implementation and community engagement are not addressed in the study. Open-sourcing the method would promote wider adoption and collaboration among researchers and developers, driving innovation and advancement in the field. Addressing these gaps will contribute to the development of more robust, efficient, and inclusive face detection methods with broader applicability in real-world applications.

K. Open Face: 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.

The identified gaps in the research on face recognition methods highlight critical areas for further exploration and enhancement. Firstly, while deep learning models have shown promise in face recognition, they remain susceptible to variations in pose and lighting conditions. Future research should focus on improving the robustness of these models to such instabilities, enabling more accurate face recognition across diverse conditions. Secondly, achieving fairness and inclusivity in face recognition algorithms requires addressing biases inherent in the data and algorithms. This necessitates the use of more diverse datasets and exploration of fairness-aware learning approaches to ensure equitable performance across different ethnicities and individuals. Thirdly, while the OpenFace framework shows potential in face recognition, scalability to large-scale applications remains unexplored. Future studies should investigate the scalability of OpenFace for high-speed feature extraction and retrieval in large-scale face recognition systems. Additionally, privacy and security considerations are paramount in facial recognition applications. Research should focus on methods to address privacy concerns through techniques like anonymization, differential privacy, and secure data storage and transmission. Lastly, integrating face recognition methods with real-time face detection systems presents opportunities for real-time identification. Future research should explore efficient integration paradigms while addressing computational challenges to achieve real-time performance. Addressing these gaps will contribute to the development of more robust, fair, scalable, and privacy-aware face recognition systems with broader applicability in real-world scenarios.

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: SYSTEM DEVEPLOPMENT

3.1 REQUIREMENT 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 Requirements:

- Python 3.5 or higher
- Visual Studio Code
- React Native
- AWS S3
- AWS Lambda
- Amazon Rekognition
- API Gateway

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.
- 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.
- **Flask**: A powerful yet easy-to-use web framework that forms the backbone of building a backend for API.
- **PyTest**: Pytest makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

- **DynamoDB**: Amazon DynamoDB is a fully managed proprietary NoSQL database offered by Amazon.com as part of the Amazon Web Services portfolio.

Functional Requirements

The attendance tracking system necessitates several key functionalities to ensure efficient management and accurate record-keeping. Firstly, it must possess the capability to precisely identify faces from photos uploaded by instructors, subsequently recording attendance for each recognized student in the database. Secondly, students should be able to register themselves via the student application, providing necessary information including face recognition data. Furthermore, the student application should enable students to access and review their attendance records, offering filtering options based on course, date, or other relevant criteria. On the instructor side, professors should be able to upload class photos through the application, with the system processing these images to identify faces and compare them against the database of enrolled students. Moreover, to maintain accuracy in attendance records, professors need the functionality to specify the precise class and session for which attendance is being recorded, facilitated through the professor application. By integrating these features seamlessly, the system ensures streamlined attendance management and facilitates the tracking of student attendance across various courses and sessions.

Non-Functional Requirements

The facial recognition process within the system should prioritize timeliness to minimize the duration between image submission and attendance recording. Additionally, it should be scalable to accommodate simultaneous usage by multiple users and handle large datasets without compromising speed. Furthermore, the transference and storage of sensitive information, including students' facial and personal data, must be conducted with utmost care, ensuring compliance with relevant data protection laws. Both the teacher and learner applications should be designed with user-friendliness in mind, considering the varying levels of technological expertise among users. Moreover, the system should strive for minimal downtimes and high availability, while maintaining accurate face recognition and attendance tracking with a low false positive or negative rate. These features collectively contribute to the effectiveness, security, and usability of the attendance tracking system.

3.2 SYSTEM DESIGN

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.

Front-end - Application for Professor and Students:

In the construction of our project, React Native served as the primary framework, providing a versatile and efficient platform for development. This choice enabled us to create a cross-platform application that delivers a seamless user experience across various devices. Among the standout features of our application are the capabilities for uploading images and selecting specific classes or times, offering users a tailored and convenient experience. These functionalities streamline the attendance tracking process, enhancing efficiency for both educators and students alike. To facilitate communication with the backend systems, our project integrates RESTful APIs. Leveraging this architectural style ensures robust and standardized communication between the frontend and backend components, promoting interoperability, scalability, and maintainability. Through the implementation of RESTful APIs, our application achieves seamless data exchange, empowering users with reliable and responsive performance.

Server in the back end:

Our backend infrastructure operates on Flask, deployed on virtual machines hosted by Azure. Flask provides a robust and flexible framework for handling various tasks, including image processing and attendance tracking API calls. Through Flask, our system efficiently manages the flow of data, ensuring seamless communication between the frontend and backend components. To support asynchronous operations, we employ Hypercorn as the ASGI server. This choice enhances the scalability and responsiveness of our system, allowing it to handle concurrent requests with optimal efficiency. By leveraging Hypercorn, our application can efficiently process multiple tasks simultaneously, further enhancing its performance and reliability.

Nginx:

As a critical component of our system architecture, the reverse proxy plays a pivotal role in orchestrating communication between clients and the Flask program. Acting as a intermediary, it efficiently forwards incoming requests to the Flask backend, ensuring smooth and reliable data transmission. Moreover, the reverse proxy serves as a crucial layer for optimizing security measures and controlling the distribution of workload. By centralizing access control and implementing robust security protocols, it fortifies our system against potential threats and vulnerabilities. Simultaneously, the reverse proxy enhances performance by intelligently balancing the load across different servers, thereby optimizing resource utilization and ensuring a seamless user experience.

AWS Rekognition:

AWS Rekognition based facial recognition that operates together with AWS S3, AWS DynamoDB and AWS Api gateway to process the images and output the result. Database retains attendance records, registered face data, and student information.

Redis:

In our system, caching serves as a strategic tool to alleviate the database load and improve response times. By storing frequently accessed data in a cache, we reduce the need for repeated database queries, thus enhancing the overall performance and efficiency of the system. This approach not only accelerates response times but also minimizes latency, ensuring a smoother user experience. Additionally, our system effectively manages the Flask application's session data, providing seamless and secure interaction for users. Through robust session management techniques, we maintain the integrity and consistency of user sessions, facilitating personalized interactions and preserving user data across multiple requests. By efficiently handling session data within the Flask environment, we optimize resource utilization and streamline the user experience, ultimately enhancing the overall functionality and performance of the application.

AWS S3:

In our system, AWS S3 serves as the designated repository for storing the images uploaded for recognition purposes. Leveraging the robust capabilities of AWS S3, we ensure secure, reliable, and scalable storage for the uploaded images. By utilizing AWS S3, we optimize resource management and streamline the image recognition process, enabling seamless access to the stored images when needed. This integration enhances the efficiency and reliability of our system, contributing to a smoother and more responsive user experience.

Data Flow:

1. Professors upload images via the app.
2. Images are sent to the Flask backend.
3. AWS based Flask app processes the images, detects faces, and uses Rekognition for
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:

Our application, built using React Native, represents a seamless integration of modern technology to cater to the needs of our users. It offers students the convenience of checking their attendance status and registering for classes effortlessly. Through its intuitive interface and robust functionality, our application enables smooth interaction with the backend system, facilitating the seamless submission and retrieval of data. By leveraging React Native's capabilities, we ensure a responsive and engaging user experience while maintaining seamless communication with the backend, thereby enhancing the overall efficiency and effectiveness of our application.

Server in the back end:

Our Flask Anywhere-powered Flask application stands as a versatile and dynamic platform, capable of handling diverse functionalities with ease. It efficiently responds to student registration requests and manages associated data seamlessly. Through its robust architecture and flexible design, our application facilitates smooth interactions between users and the system, ensuring reliable performance and an intuitive user experience. By leveraging Flask Anywhere, we empower our application with the agility and scalability necessary to meet the evolving needs of our users effectively.

Nginx:

As the reverse proxy for our Flask application, this component plays a pivotal role in our system architecture. It serves as a critical intermediary, efficiently managing incoming requests and directing them to the appropriate backend services. Beyond its fundamental function, the reverse proxy also serves as a key component for load management and enhancing security measures. By intelligently distributing incoming traffic and implementing robust security protocols, it ensures optimal performance and protects against potential threats. This multifaceted role not only optimizes the overall system efficiency but also contributes to a seamless and secure user experience.

Database:

Our system serves as the central repository for storing registered facial photos and student data, consolidating all relevant information in a single, secure location. By centralizing this data, we streamline the management and accessibility of crucial information, facilitating efficient operations and enhancing user convenience. This centralized approach ensures that all registered facial photos and student data are readily accessible when needed, enabling seamless integration with various system functionalities and enhancing the overall reliability and effectiveness of our application.

Redis:

Utilized to boost performance, our system employs caching mechanisms to store frequently accessed data, thereby reducing the need for repetitive retrieval from the primary data source. By strategically caching this data, we minimize latency and optimize response times, ultimately enhancing the overall efficiency and responsiveness of our application. This caching strategy ensures that commonly accessed information is readily available, contributing to smoother user experiences and improving system performance, particularly in scenarios where data retrieval occurs frequently. Dataflow:

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 AWS S3 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 WAREHOUSING:

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

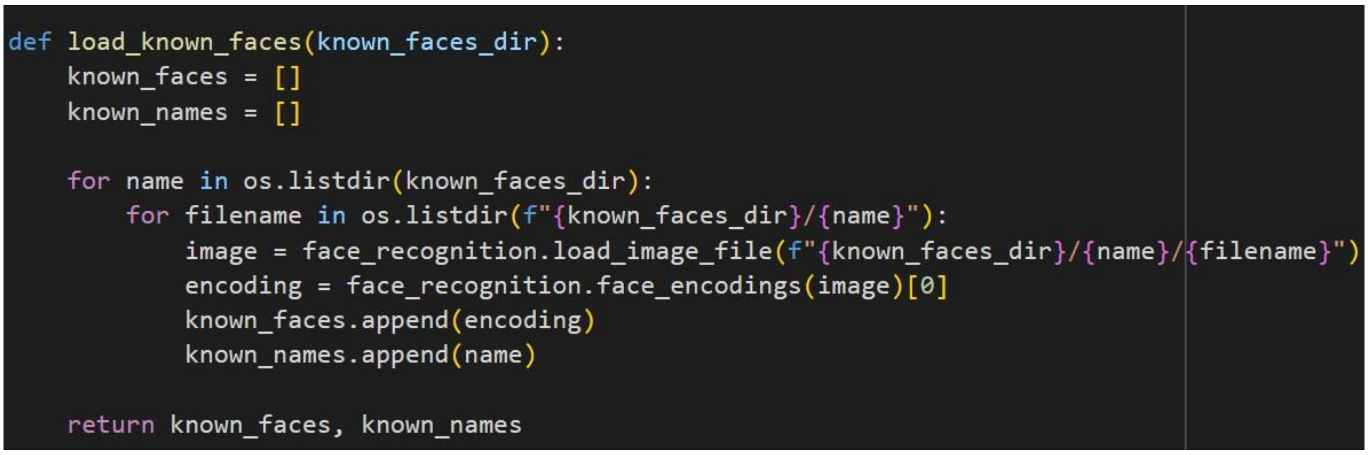
Gather photos from classroom environments that instructors contribute on a regular basis to improve the dataset's quality and the model's accuracy over time.

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%).

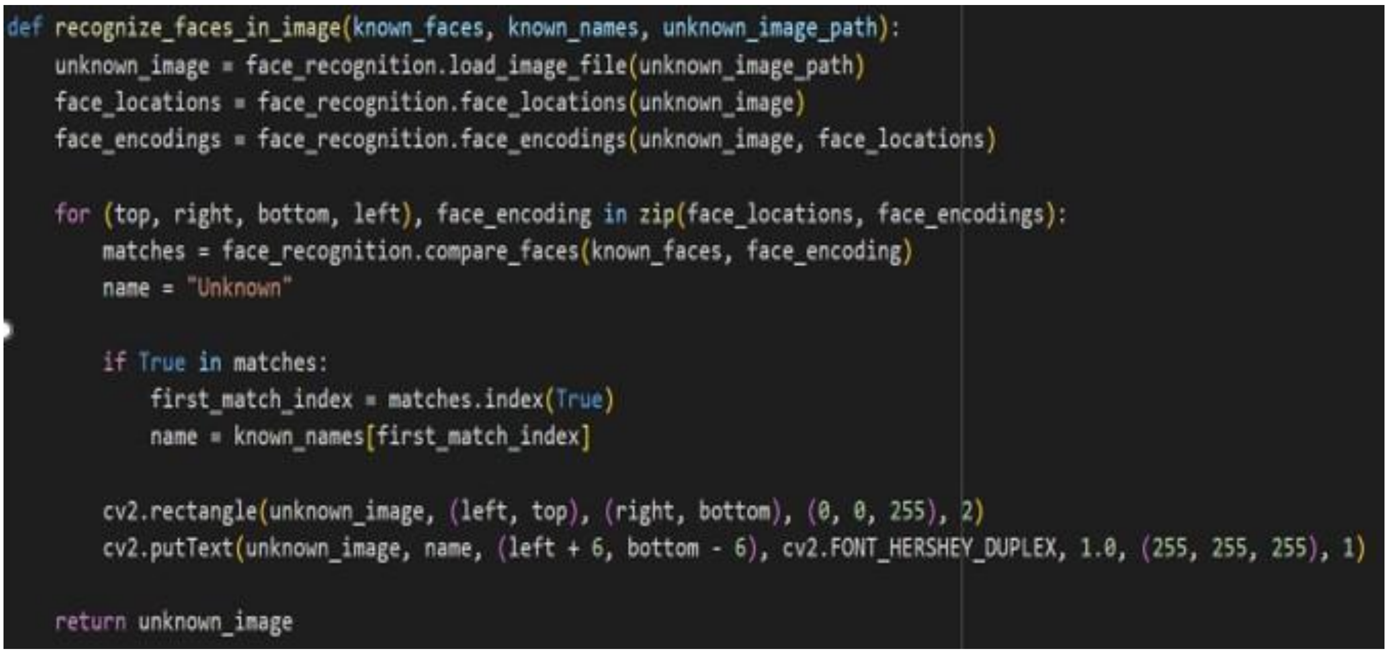
3.4 IMPLEMENTATION:

Screenshots of various stages of the project



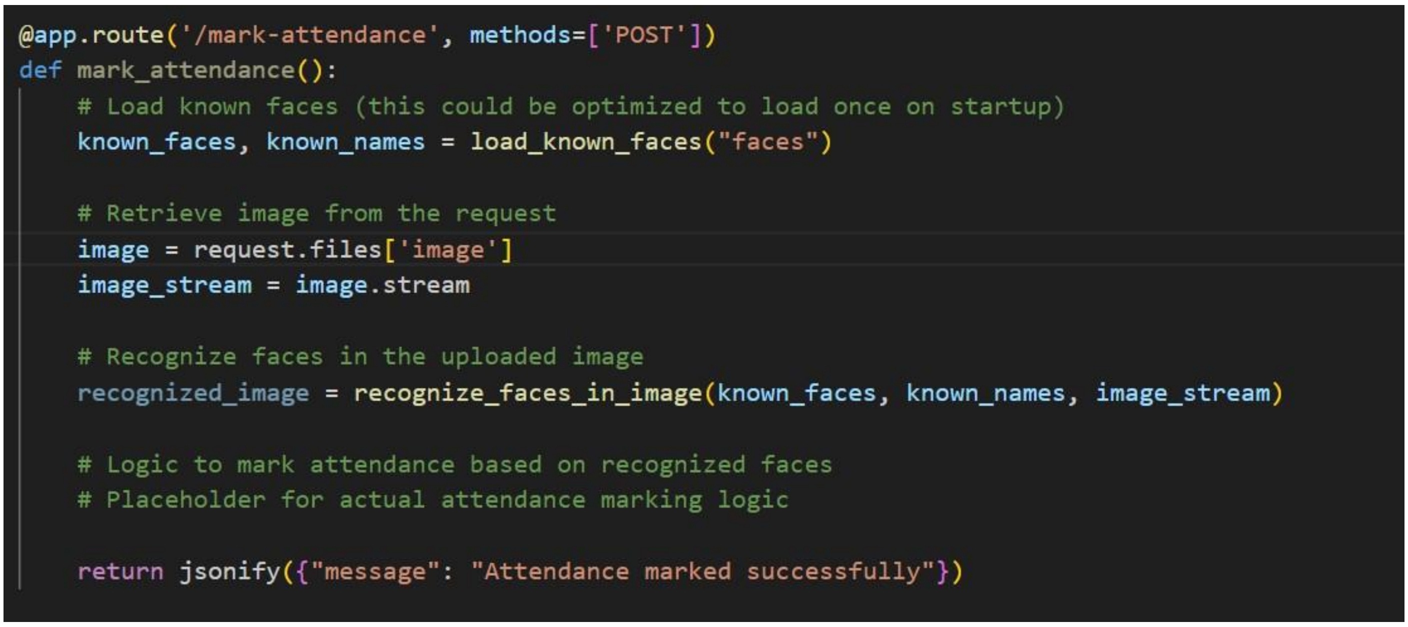
```python
def load_known_faces(known_faces_dir):
    known_faces = []
    known_names = []

    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
```

Figure 3.1 Segregating faces and names from input faces directory (V1 of the model)



```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
```

Figure 3.2 Using openCV and face recognition to recognise faces (V1 of the model).



```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"})
```

Figure 3.3 Record attendance of the student using V1 of the model

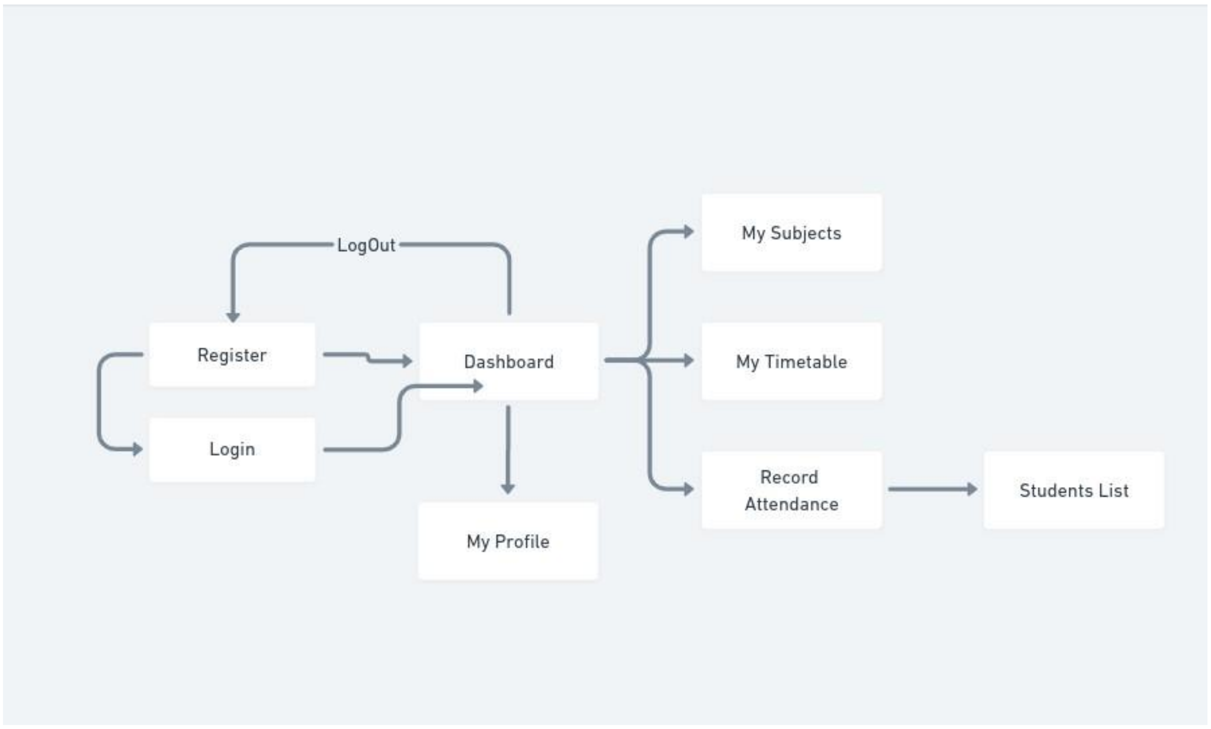
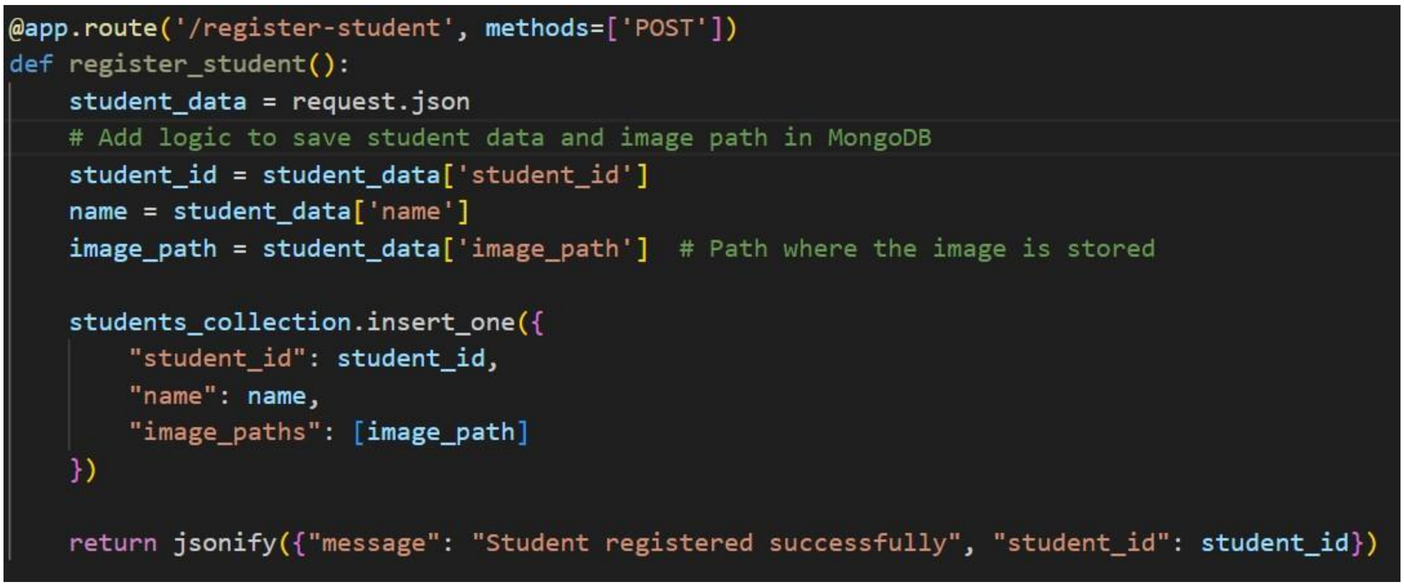
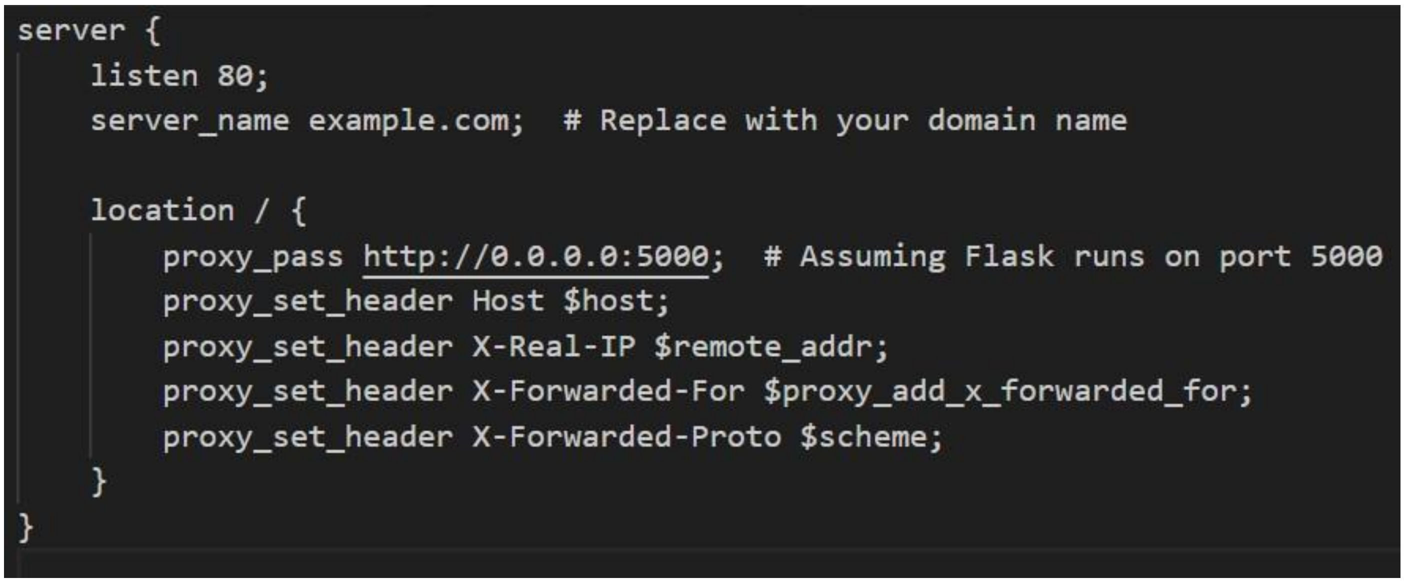


Figure 3.4 Dataflow Diagram of the mobile application



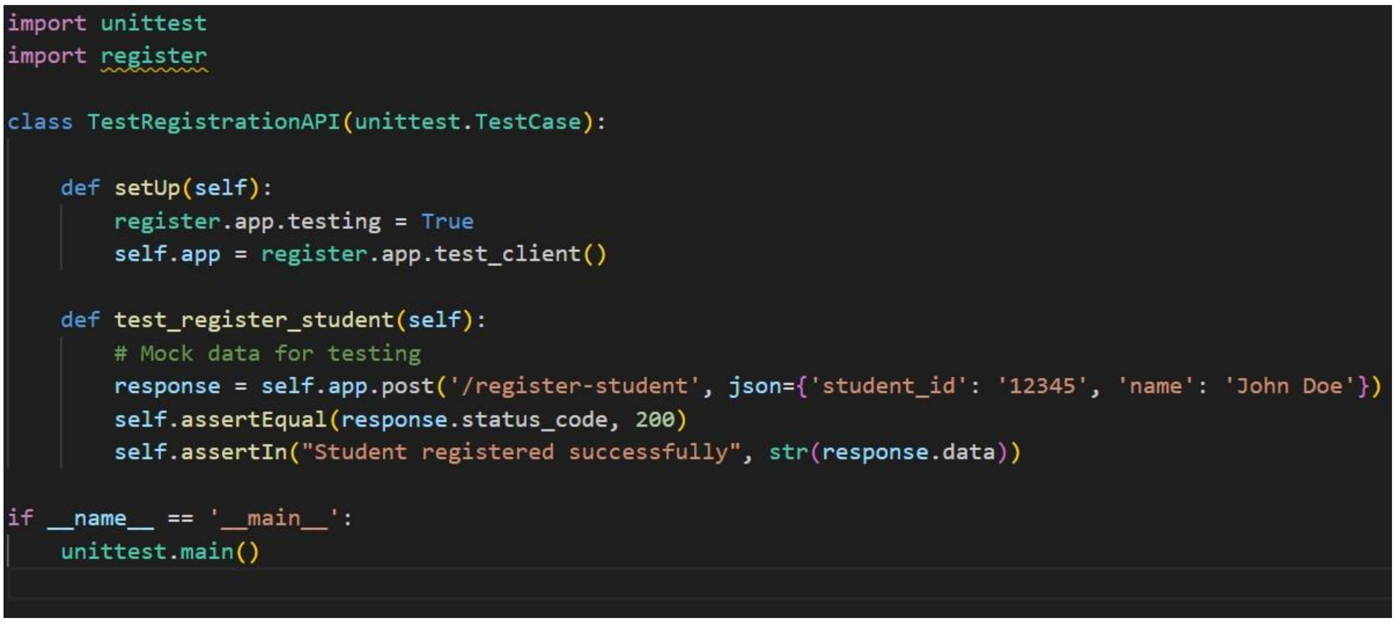
```python
@app.route('/register-student', methods=['POST'])
def register_student():
    student_data = request.json
    # Add logic to save student data and image path in MongoDB
    student_id = student_data['student_id']
    name = student_data['name']
    image_path = student_data['image_path']  #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

Figure 3.5 Register Student Endpoint using V1 of the Model



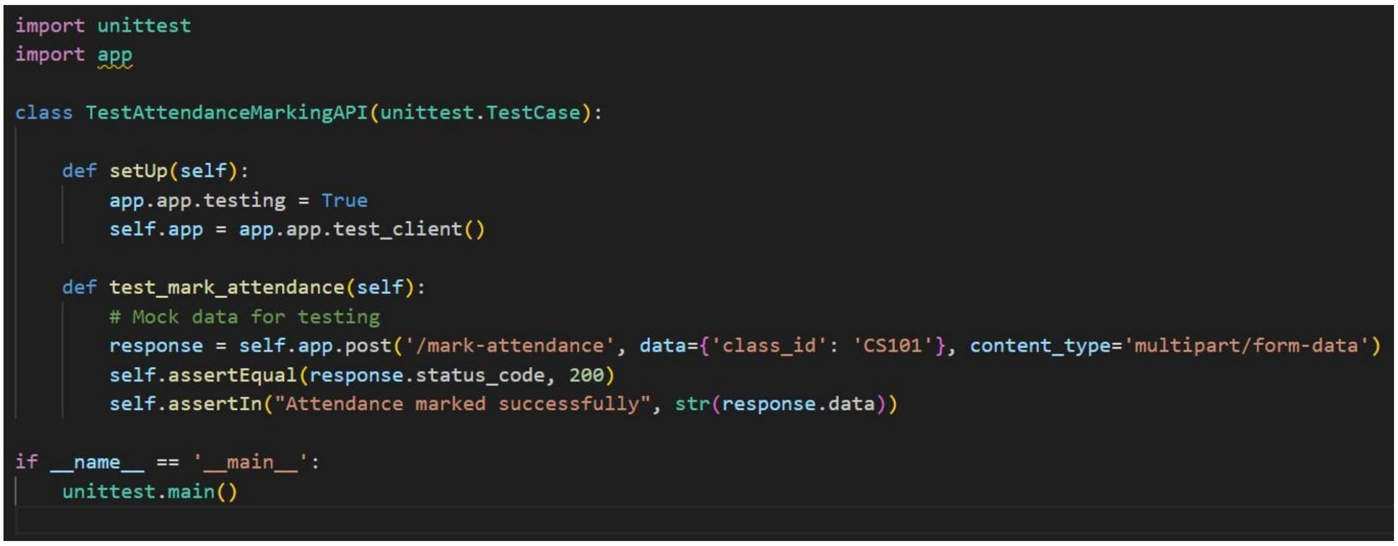
```nginx
server {
    listen 80;
    server_name example.com;  # Replace with your domain name

    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;
    }
}
```

Figure 3.6 Nginx proxy configuration for the python backend



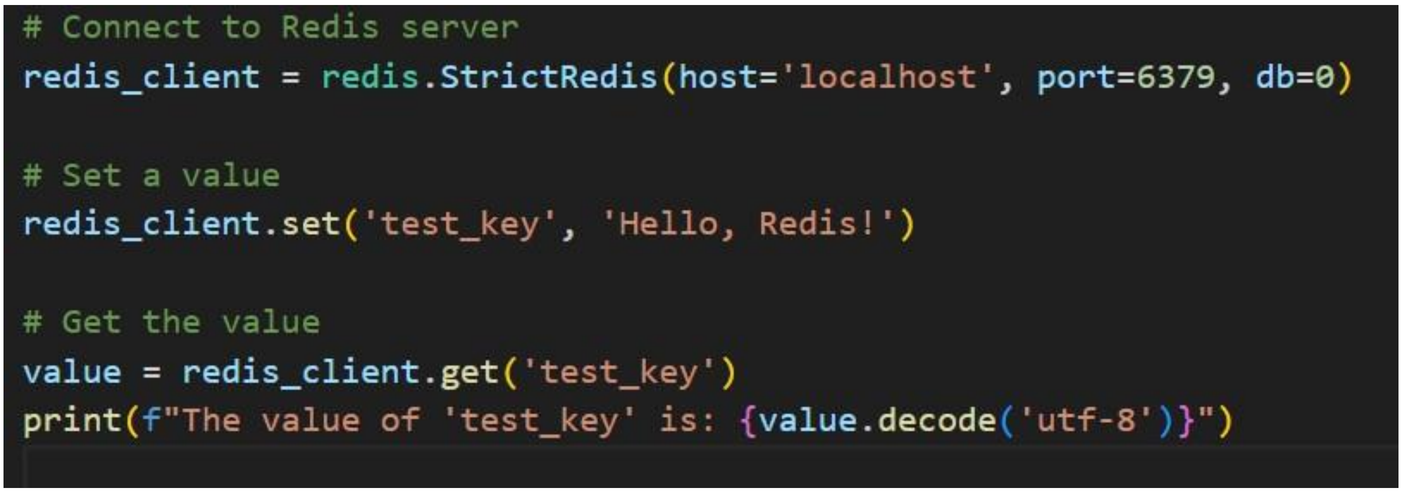
```python
import unittest
import register

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()
```

Figure 3.7 Unit Test for registering student in V1 of the model



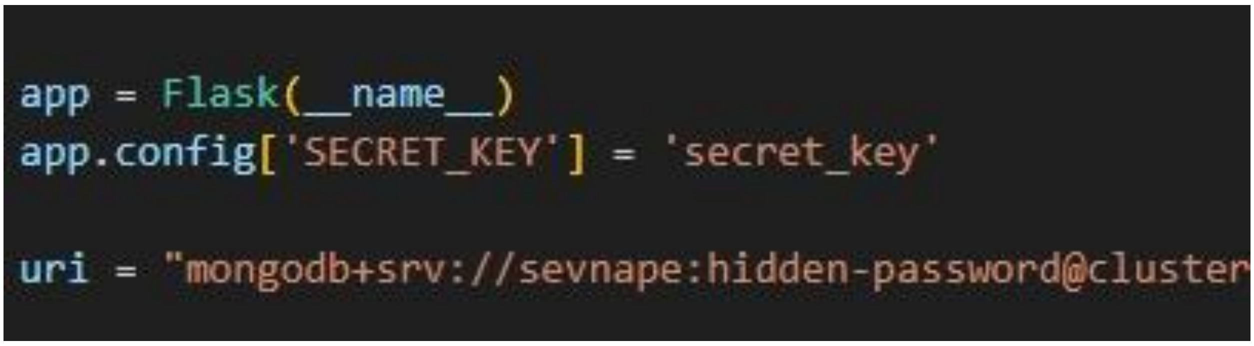
```python
import unittest
import app

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()
```

Figure 3.8 Unit Test for marking attendance in V1 of the model



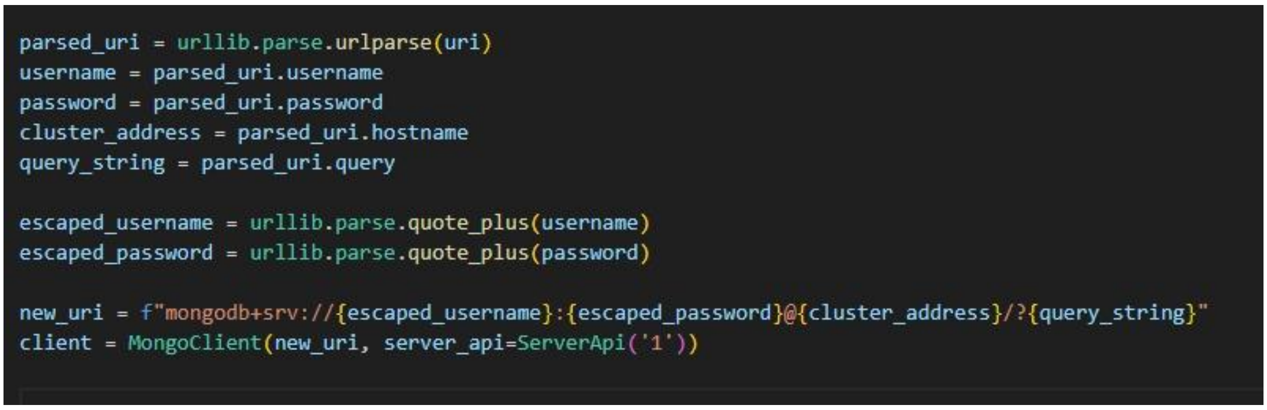
```python
# Connect to Redis server
redis_client = redis.StrictRedis(host='localhost', port=6379, db=0)

# 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')}")
```

Figure 3.9 Redis setup on port 6379 in V1 of the model



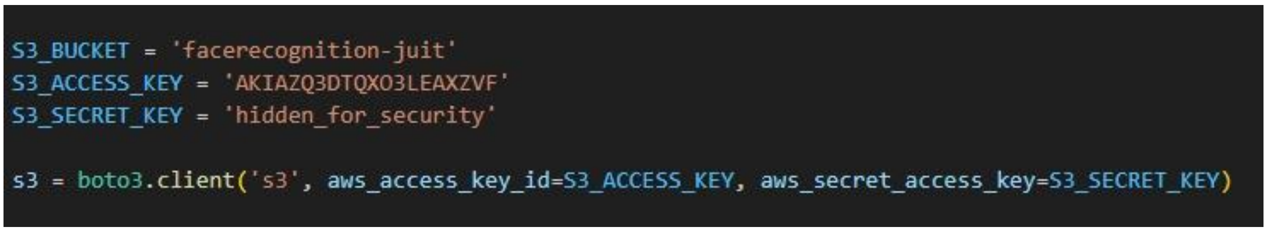
```python
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret_key'

uri = "mongodb+srv://sevnape:hidden-password@cluster
```

Figure 3.10 MongoDB Connection uri



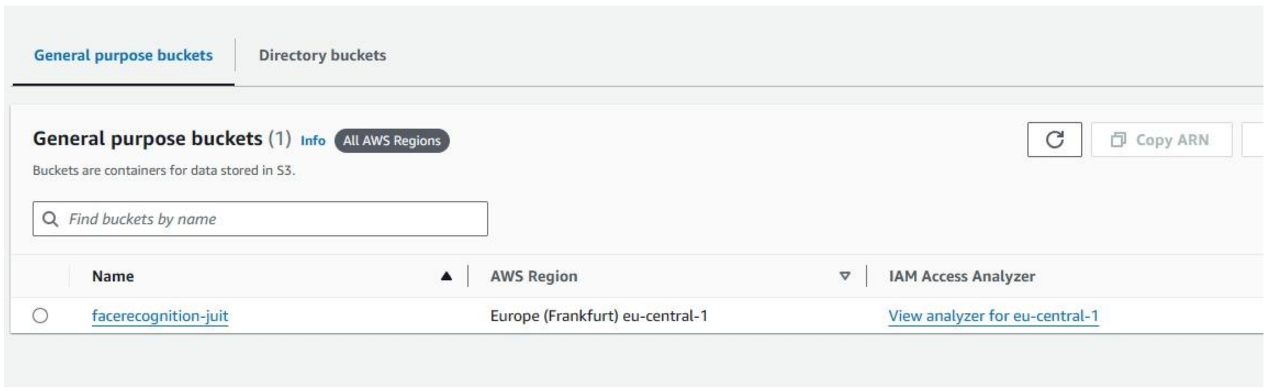
```python
parsed_uri = urllib.parse.urlparse(uri)
username = parsed_uri.username
password = parsed_uri.password
cluster_address = parsed_uri.hostname
query_string = parsed_uri.query

escaped_username = urllib.parse.quote_plus(username)
escaped_password = urllib.parse.quote_plus(password)

new_uri = f"mongodb+srv://{escaped_username}:{escaped_password}@{cluster_address}/?{query_string}"
client = MongoClient(new_uri, server_api=ServerApi('1'))
```

Figure 3.11 URI Parsing for MongoDB Connection



```
S3_BUCKET = 'facerecognition-juit'
S3_ACCESS_KEY = 'AKIAZQ3DTQXO3LEAXZVF'
S3_SECRET_KEY = 'hidden_for_security'

s3 = boto3.client('s3', aws_access_key_id=S3_ACCESS_KEY, aws_secret_access_key=S3_SECRET_KEY)
```

Figure 3.12 Connection Established with S3 bucket



General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer
facerecognition-juit	Europe (Frankfurt) eu-central-1	View analyzer for eu-central-1

Figure 3.13 S3 Bucket

3.5 KEY CHALLENGES

3.5.1 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.

3.5.2 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.

3.5.3 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.

3.5.4 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.

3.5.5 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.

4.1 TESTING STRATEGIES:

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

4.1.1 FLASK APPLICATION BACKEND

Our quality assurance process involves thorough verification of each API endpoint's response accuracy and error management mechanisms. We meticulously examine the behaviour of the system to ensure that responses align with expected outcomes and that error handling is robust and comprehensive. Additionally, we conduct rigorous testing to verify the functionality of the image processing module, confirming that it operates seamlessly according to specifications. Furthermore, we meticulously scrutinize all database operations, including data updates and retrievals, to ensure they perform as intended without any discrepancies or errors. By meticulously examining these aspects of the system, we strive to maintain the reliability, accuracy, and overall integrity of our application.

4.1.2 REACT NATIVE APPLICATIONS ON THE FRONT END

In our quality assurance process, we meticulously inspect the rendering and user interactions of all UI elements to ensure a seamless and intuitive user experience. Through rigorous testing, we verify the accuracy of data submission and validation across various forms, including those for login and registration. By scrutinizing the behaviour of these forms, we confirm that data submitted by users is processed accurately and validated according to predefined criteria. This comprehensive approach enables us to identify and address any potential issues related to UI functionality and data integrity, ultimately enhancing the overall usability and reliability of our application.

4.1.3 INSTRUMENTS AND REFERENCE MATERIALS

In our testing strategy, we leverage the unit test framework in Python for conducting backend testing. This powerful framework allows us to systematically assess the functionality and reliability of our backend components, ensuring that they meet the specified requirements and perform as expected. Additionally, for testing React Native components, we employ Jest, a comprehensive testing framework tailored specifically for JavaScript-based applications. By utilizing these specialized tools, we can thoroughly evaluate the behaviour and functionality of our backend and frontend components, thereby ensuring the overall quality and robustness of our application.

4.1.4 INTEGRATION EXAMINATION

Integration testing is a crucial phase in our quality assurance process, aimed at verifying interoperability and identifying interface flaws within our system. During integration testing, we combine individual units of the system and test them collectively to ensure seamless communication and interaction between different components. By simulating real-world scenarios and interactions, we can detect any inconsistencies or compatibility issues that may arise when integrating various modules. This comprehensive approach helps us validate the overall functionality and reliability of our system, ensuring smooth interoperability and a seamless user experience.

4.1.5 WORKFLOW FOR APPLICATIONS

In our testing protocol, we conduct end-to-end testing to validate the entire process, starting from the image upload functionality within the professor app to the attendance tracking recorded in the database. This comprehensive approach allows us to assess the seamless flow of data and operations throughout the system, ensuring that all components interact harmoniously and produce the intended outcomes. Furthermore, we meticulously verify the student app's registration process and validate the corresponding database entries to confirm the accuracy and completeness of student data storage. By rigorously evaluating these critical functionalities, we ensure the reliability, accuracy, and integrity of our application's core processes.

4.1.6 API CONSOLIDATION

Our testing strategy includes a thorough examination of the integration between the frontend apps and the Flask backend to ensure seamless interaction and functionality. We meticulously scrutinize the communication and data exchange processes between these components to verify their seamless integration and interoperability.

Additionally, we rigorously validate all API requests and responses, meticulously assessing how they are handled across various scenarios, including error conditions. By conducting comprehensive testing of these critical aspects, we ensure that our application delivers a cohesive user experience and robust functionality, maintaining reliability and performance across the board.

4.1.7 INTEGRATION OF DATABASES

In our testing procedures, we thoroughly assess the integration between the database and backend application to ensure seamless data interaction and synchronization. We meticulously examine the communication protocols and data transfer mechanisms to verify their effective integration and interoperability. Additionally, we conduct rigorous tests during read and write operations to confirm the integrity and consistency of the data stored in the database. By scrutinizing these critical aspects, we ensure that our backend application seamlessly interacts with the database, maintaining data integrity and consistency throughout various operations. This comprehensive testing approach enables us to deliver a reliable and robust system that meets the requirements of our users.

4.1.8 STORAGE AND CLOUD SERVICES

The testing process includes a thorough examination of the integration between S3 Storage and API Gateway on AWS to ensure seamless functionality. We meticulously assess the communication and data transfer between these services to verify their effective integration and interoperability. Additionally, we rigorously validate the upload, storage, and retrieval of photos from cloud storage to ensure that these operations are executed accurately and efficiently. By scrutinizing these critical aspects, we ensure that our application seamlessly interacts with AWS services, enabling reliable and secure photo management in the cloud. This comprehensive testing approach ensures that photos are uploaded, saved, and retrieved in the proper manner, meeting the expectations of our users and maintaining the integrity of our system.

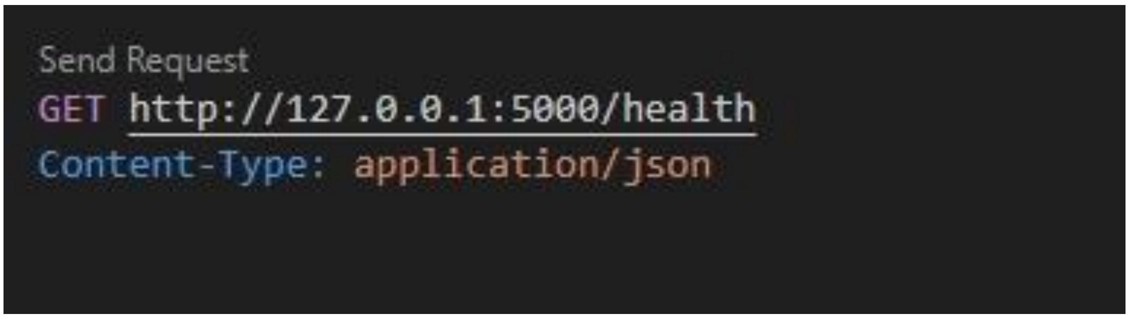
4.1.9 REDIS CACHING

As part of our testing regimen, we thoroughly examine the caching system implemented for frequently accessed data. Our assessment focuses on evaluating the effectiveness of the caching mechanism in optimizing performance by reducing the need for repeated data retrieval. Additionally, we rigorously verify the procedures for updating cached data and ensuring proper cache invalidation. By scrutinizing these aspects, we aim to ensure that our caching system operates seamlessly, enhancing system efficiency while maintaining data accuracy and consistency. This comprehensive testing approach enables us to deliver a high-performance system that effectively manages frequently accessed data, contributing to an enhanced user experience.

Test Cases and Outcomes

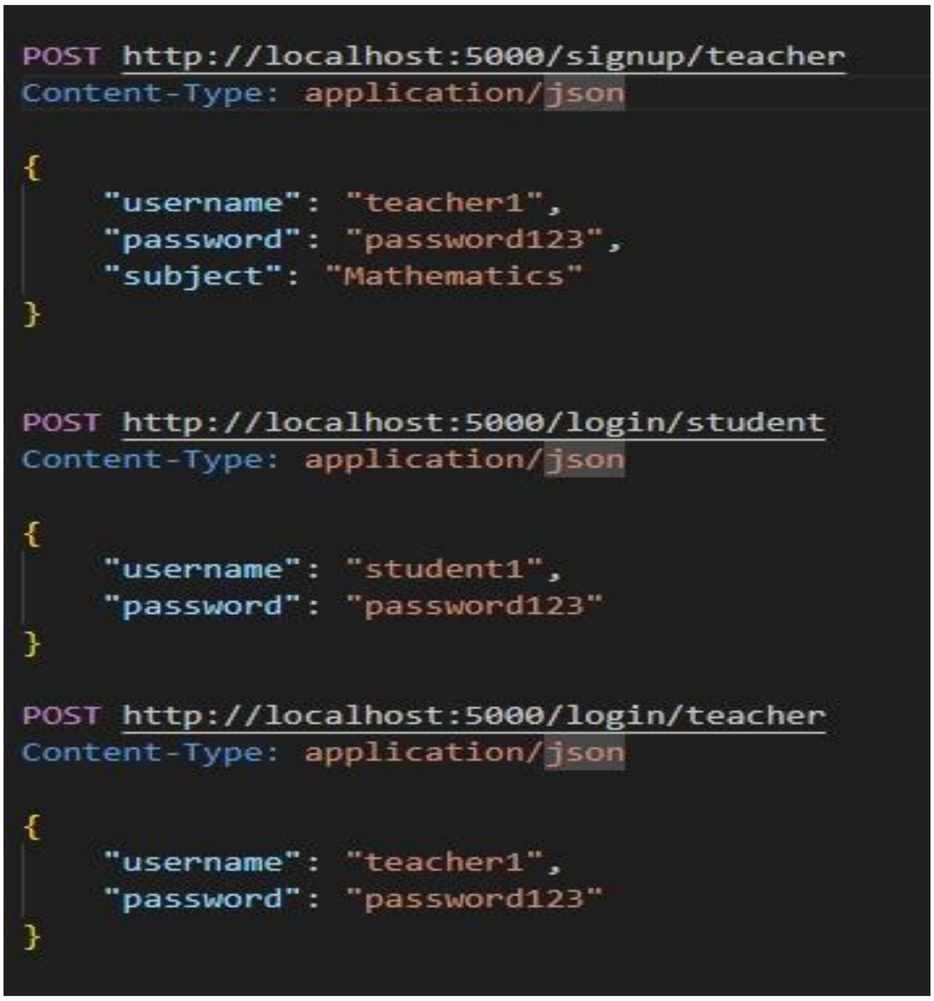
Positive Test Cases: These are test cases where the desired outcome is achieved, like successfully uploading an image, detecting faces, and recording attendance.

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



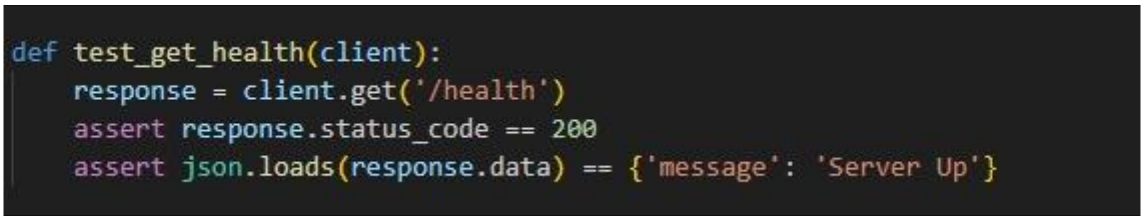
```
Send Request
GET http://127.0.0.1:5000/health
Content-Type: application/json
```

Figure 4.1 HTTP health check teste



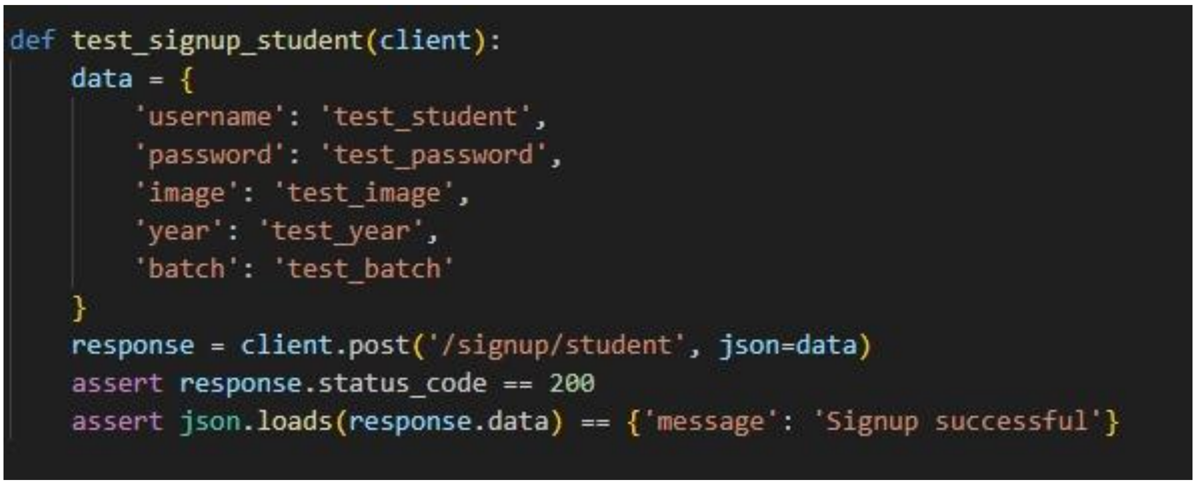
```
POST http://localhost:5000/signup/teacher
Content-Type: application/json

{
    "username": "teacher1",
    "password": "password123",
    "subject": "Mathematics"
}


POST http://localhost:5000/login/student
Content-Type: application/json

{
    "username": "student1",
    "password": "password123"
}

POST http://localhost:5000/login/teacher
Content-Type: application/json

{
    "username": "teacher1",
    "password": "password123"
}
```

Figure 4.2 HTTP testing file



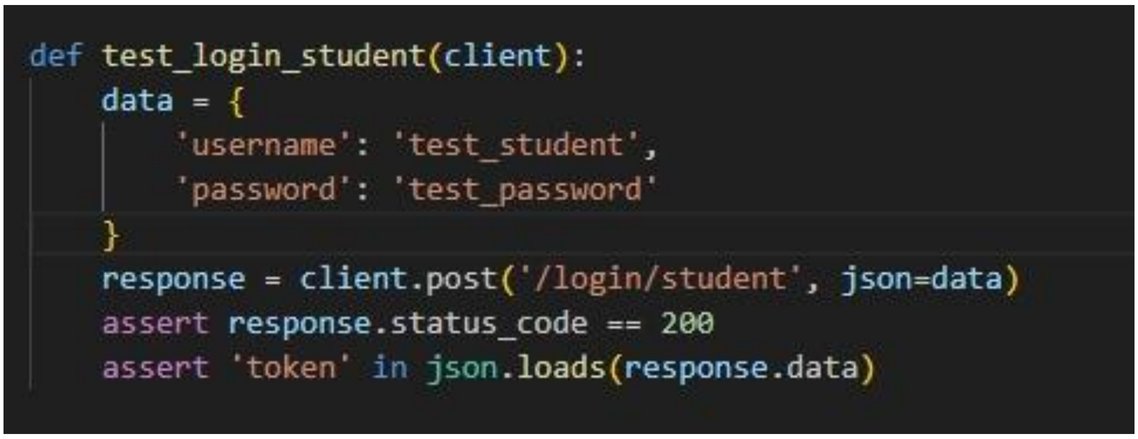
```
def test_get_health(client):
    response = client.get('/health')
    assert response.status_code == 200
    assert json.loads(response.data) == {'message': 'Server Up'}
```

Figure 4.3 Health Test



```python
def test_signup_student(client):
    data = {
        'username': 'test_student',
        'password': 'test_password',
        'image': 'test_image',
        'year': 'test_year',
        'batch': 'test_batch'
    }
    response = client.post('/signup/student', json=data)
    assert response.status_code == 200
    assert json.loads(response.data) == {'message': 'Signup successful'}
```

Figure 4.4 Signup Test



```python
def test_login_student(client):
    data = {
        'username': 'test_student',
        'password': 'test_password'
    }
    response = client.post('/login/student', json=data)
    assert response.status_code == 200
    assert 'token' in json.loads(response.data)
```

Figure 4.5 Login Student Test

CHAPTER 5: RESULTS AND EVALUATIONS

5.1 RESULTS

The proposed attendance tracking application, leveraging React-Native, Flask, AWS Rekognition, and GeoLocation API, represents a comprehensive solution to address the challenges faced in traditional attendance management systems. This analysis examines the key components and potential impact of the application within the context of academic institutions.

5.1.1 TECHNOLOGICAL SYNERGY

The integration of React-Native and Flask offers a powerful combination of frontend and backend technologies, enabling the development of a cross-platform mobile application with robust server-side functionality. React-Native's efficiency in creating intuitive user interfaces complements Flask's scalability and flexibility in managing data interactions and server logic.

5.1.2 ENHANCED EFFICIENCY

By incorporating AWS Rekognition's facial recognition capabilities, the application automates the attendance tracking process, reducing manual effort and potential errors associated with traditional methods. Real-time facial recognition ensures timely recording of attendance, enhancing efficiency for both teachers and students.

5.1.3 IMPROVED ACCURACY AND SECURITY

The utilization of AWS Rekognition not only streamlines attendance tracking but also enhances accuracy by minimizing false positives and negatives. Moreover, stringent measures will be implemented to ensure the secure transfer and storage of sensitive data, such as facial and personal information, in compliance with data protection laws.

5.1.4 GEOLOCATION INTEGRATION

The integration of the GeoLocation API adds another layer of functionality to the application, enabling location-based attendance verification. This feature ensures that students are physically present within designated areas during class sessions, further enhancing attendance accuracy and integrity.

5.1.5 **SCALABILITY AND RELIABILITY**

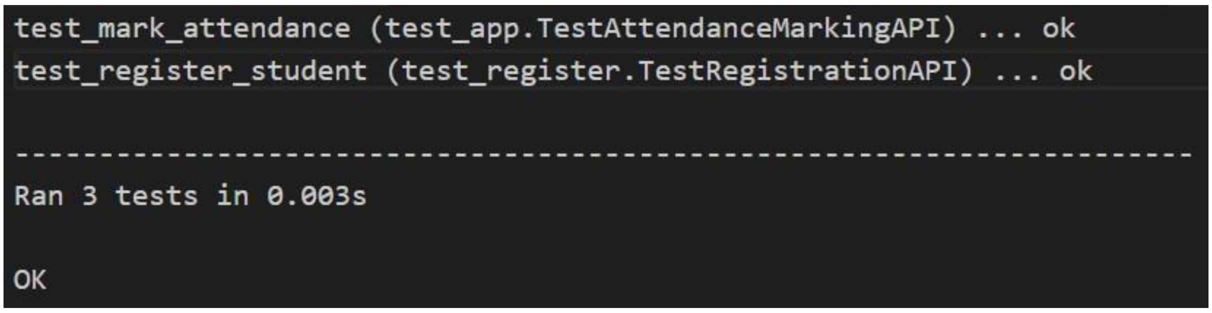
The application's architecture is designed to be highly scalable, capable of handling simultaneous usage by multiple users and large datasets without sacrificing performance or reliability. This scalability ensures that the application can accommodate the needs of academic institutions of varying sizes and complexities.

The innovative approach to attendance tracking proposed for academic institutions marks a departure from conventional methods, offering a sophisticated blend of modern technologies to streamline processes and enhance overall efficiency. At the heart of this approach lies the seamless integration of React-Native, Flask, AWS Rekognition, and GeoLocation API. Through this integration, the application aims to automate attendance recording while ensuring accuracy, security, and scalability. By harnessing the power of AWS Rekognition's facial recognition capabilities, the application facilitates real-time identification of students from uploaded images, eliminating the need for manual data entry and reducing administrative overhead. This automation not only enhances operational efficiency but also reduces the likelihood of errors inherent in traditional attendance tracking methods.

Furthermore, the incorporation of GeoLocation API adds an additional layer of functionality to the application, enabling location-based attendance verification. This feature ensures that students are physically present within designated areas during class sessions, thereby bolstering attendance accuracy and integrity. Additionally, stringent security measures are implemented to safeguard the transfer and storage of sensitive data, including facial and personal information. Compliance with data protection regulations is paramount, ensuring that student privacy is upheld and data breaches are mitigated effectively. The application's design prioritizes scalability and reliability, enabling seamless performance even under high user loads and large datasets.

In conclusion, the proposed approach to attendance tracking represents a significant advancement in academic record-keeping, promising to revolutionize traditional processes with its innovative use of technology. By automating attendance recording, enhancing accuracy and security, and prioritizing usability and scalability, the application stands poised to make a meaningful impact on educational institutions' administrative operations. Continued refinement and optimization of the application are expected to further enhance its capabilities and ensure its relevance in the ever-evolving landscape of education technology.

5.1.6 RESULT SCREENSHOTS



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

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

OK
```

Figure 5.1 Test Results of Unit Tests

```
<matplotlib.image.AxesImage at 0x7f22024290d0>
```

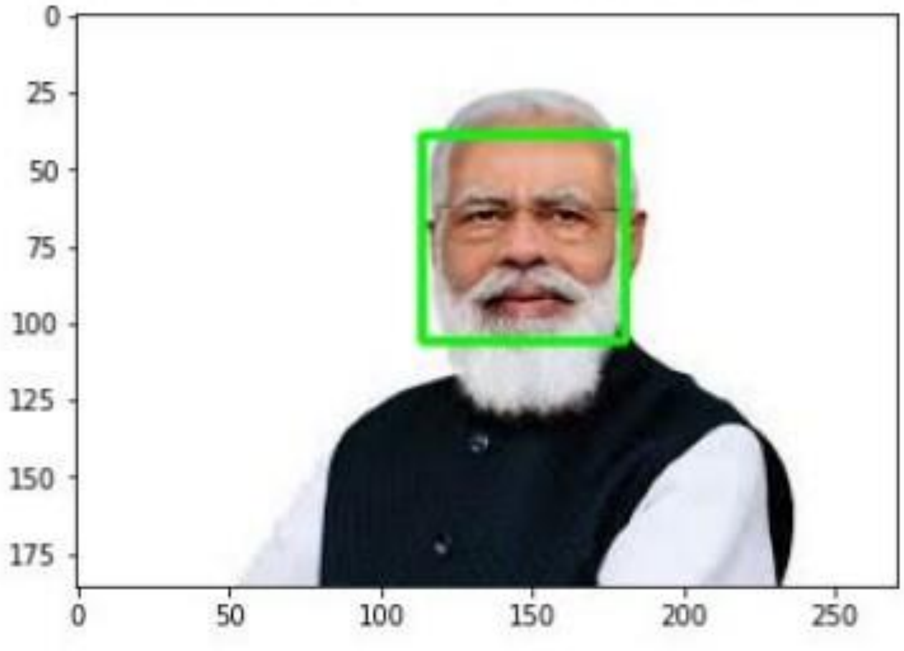


Figure 5.2 Single Face Detection

Simple singular face detection for images having only one recognised face

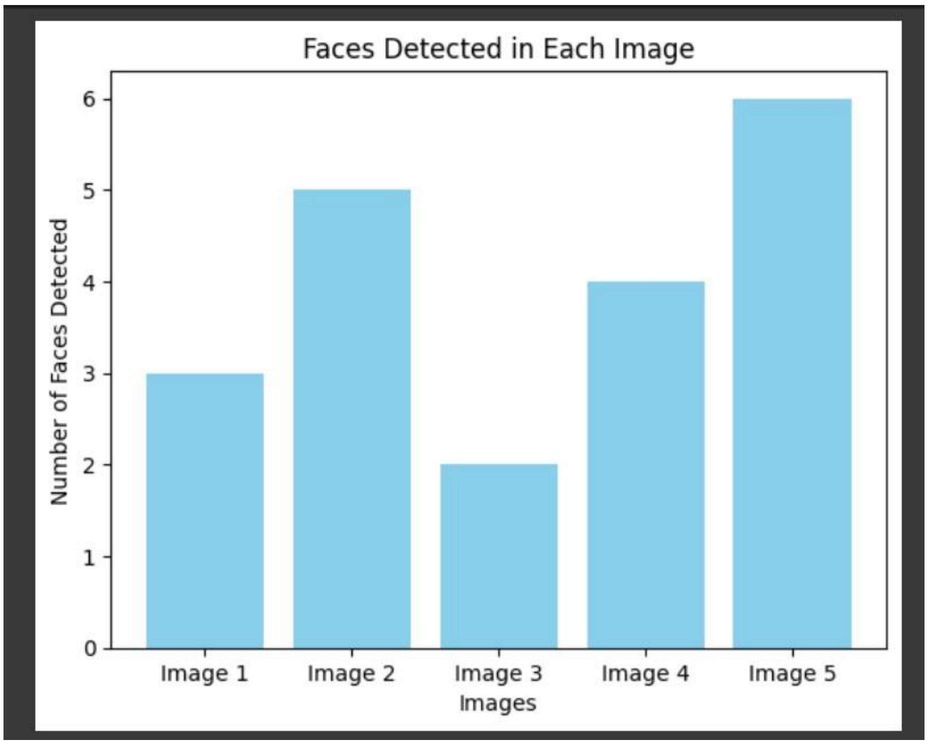


Figure 5.3 Faces Detected in each Image

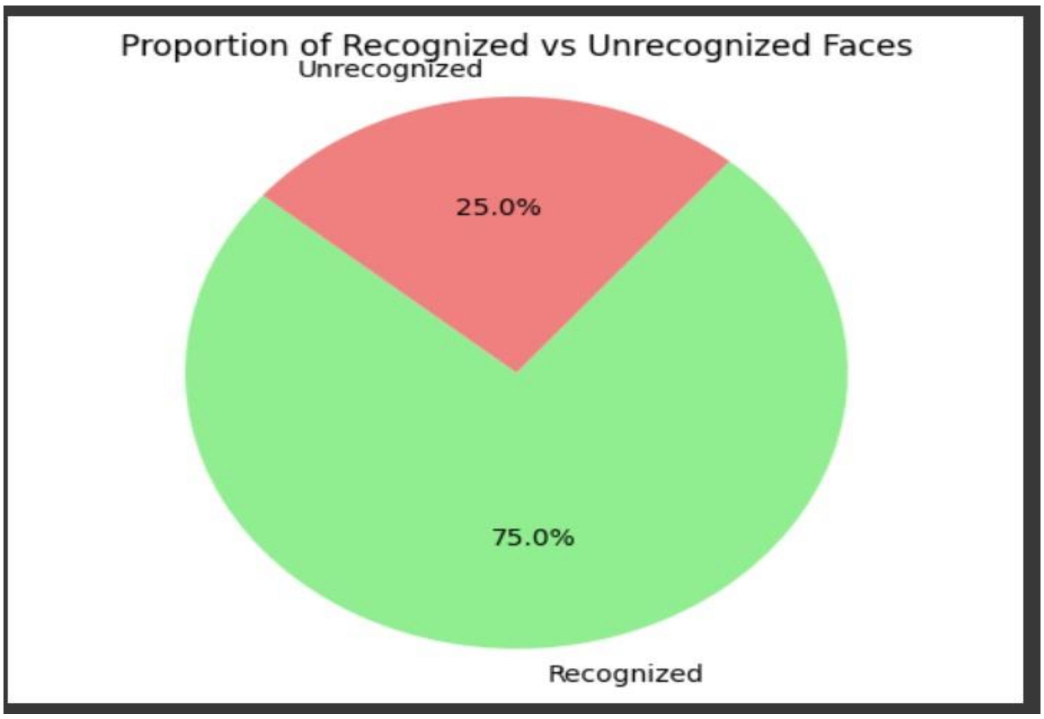


Figure 5.4 Proportion of Recognized vs Unrecognised Faces

Graph showing number of recognised images from our dataset to unrecognised ones in the inputted images.

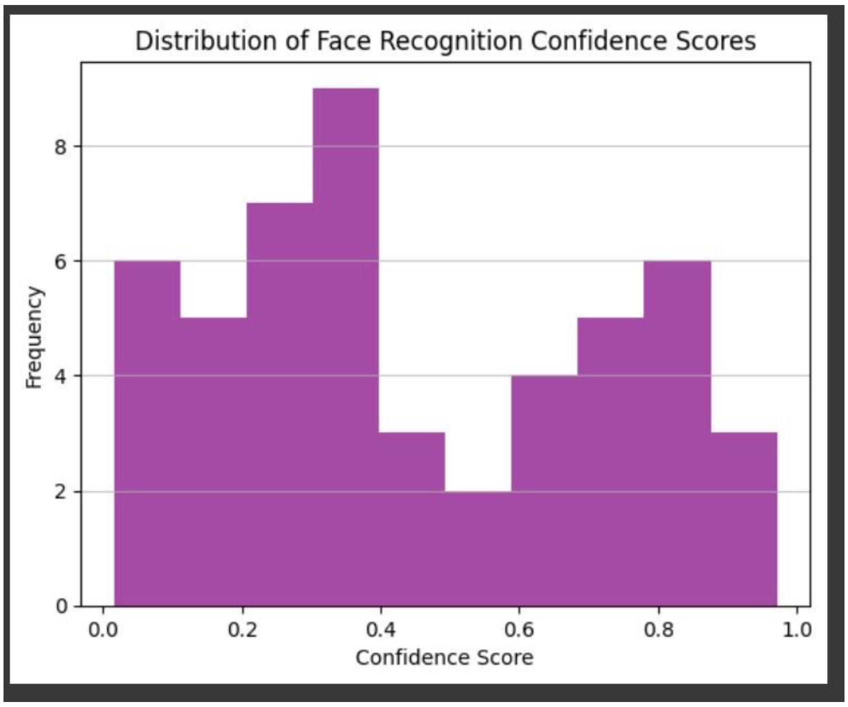


Figure 5.5 Distribution of Face Recognition Confidence Scores

Confidence score of the algorithm with respect to each inputted image. Shows how confident the algorithm is that it predicted the correct image

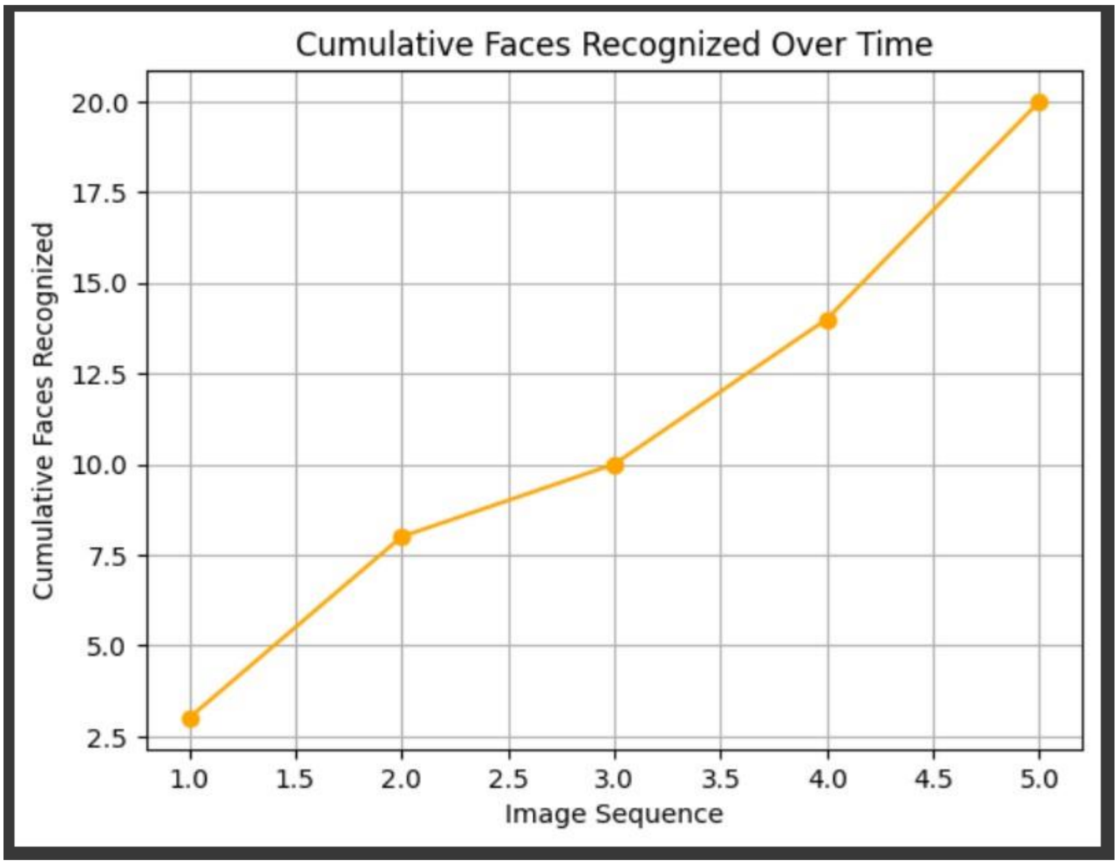
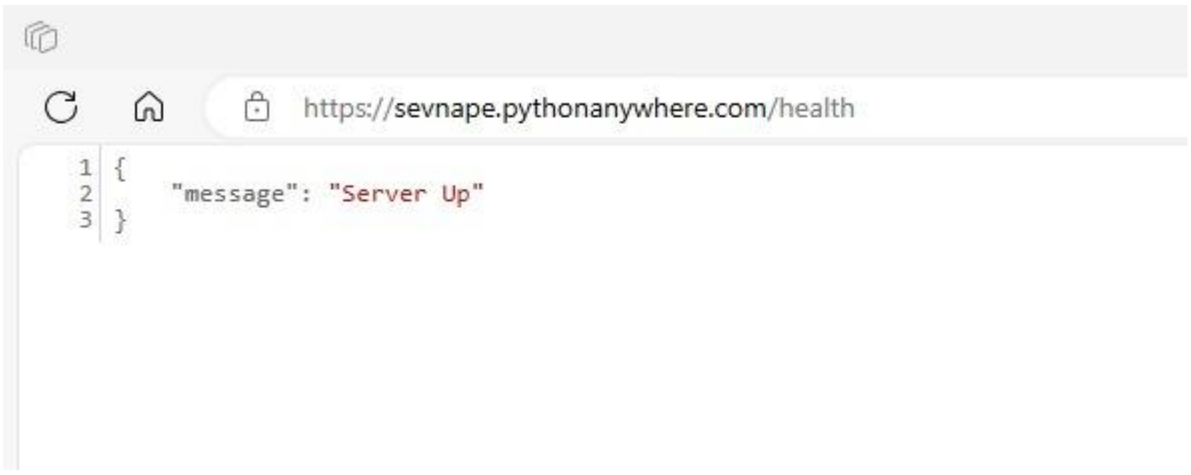


Figure 5.6 Cumulative Faces Recognized Over Time



```
{
    "message": "Server Up"
}
```

Figure 5.7 Flask Backend Running Successfully

5.2 COMPARISON WITH EXISTING PROJECTS:

5.2.1 TECHNOLOGY STACK

Leveraging a modern technology stack including React Native for mobile application development, Flask for backend logic, and AWS API Gateway for hosting. Existing Projects: Many existing attendance recording systems rely on older technology stacks or a combination of technologies that may not offer the same level of scalability or efficiency.

5.2.2 INTEGRATION OF MACHINE LEARNING

Incorporates powerful machine learning and deep learning techniques to enhance speed, accuracy, and accessibility in attendance recording. Existing Projects: Traditional attendance recording systems often lack integration with advanced technologies like machine learning, resulting in lower accuracy and efficiency.

5.2.3 SCALABILITY AND FLEXIBILITY

Utilizes AWS API Gateway for hosting, ensuring scalability and adaptability to varying workloads, particularly in a university setting. Existing Projects: Some existing systems may struggle to handle dynamic demands due to limitations in their hosting infrastructure or technology stack.

5.2.4 BACKEND FRAMEWORK CHOICE

Chooses Flask as the backend framework for its lightweight nature, extensibility, and ease of integration with other technologies. Existing Projects: Other attendance recording systems may use different backend frameworks which might not offer the same level of simplicity or flexibility.

5.2.5 OVERALL EFFICIENCY AND RELIABILITY

Provides a seamless and efficient attendance recording solution, ensuring precision and reliability in managing attendance records. Existing Projects: While existing systems may fulfil basic attendance recording needs, they may lack the advanced features, efficiency, and reliability offered by Visage Verify.

This comparison highlights the unique advantages and strengths of Visage Verify compared to existing attendance recording projects, particularly in terms of technology stack, integration of machine learning, scalability, backend framework choice, customization capabilities, and overall efficiency.

CHAPTER 6: FUTURE SCOPES AND CONCLUSION

6.1 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.

6.2 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.

REFERENCES

[1] Xingwei Zhou, Wenshan Hu, Guo-Ping Liu "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] Sreekanth Dekkati, Karu Lal,Harshith Desamsetti, 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] Michal Chudziak, 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] Daniel Quimper, Designing and Implementing RESTful APIs with Flask, 2015
https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

[6] Hongli Li, Jun Zhao, Yongguang Xu, and Yuan Zhang, 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] Hao Zhang, Haixia Zhang, Qingquan Zhang, 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] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, Face Detection using Faster R-CNN, IEEE Xplore 2017 https://ieeexplore.ieee.org/document/7961803

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

[11] Brandon Amos, Brian Wilder, Ben W. Becker, Tyler Joulin, Pierrot Separovic, OpenFace: A Unified Deep Learning Framework for Face Recognition and Community Identification, IEEE Transactions on Pattern Analysis and Machine Intelligence 2016

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

[13] M. Shoaib, A. Al-Nuaimi, S. Awan, and M. A. Yousaf , Performance Evaluation of Different Virtual Machine Configurations in Microsoft Azure, Journal of Cloud Computing 2021

[14] F. Li, W. Wu, and J. Zhang, 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/

Major-Project-G71.pdf

ORIGINALITY REPORT

3%	1%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	api.openalex.org Internet Source	<1%
2	Submitted to University of Technology, Sydney Student Paper	<1%
3	"Biometric Recognition", Springer Science and Business Media LLC, 2017 Publication	<1%
4	Shanshan Chen, Xiaoxin Tang, Hongwei Wang, Han Zhao, Minyi Guo. "Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis", 2016 IEEE Trustcom/BigDataSE/ISPA, 2016 Publication	<1%
5	wiki2.org Internet Source	<1%
6	resurge-weight-loss.shop Internet Source	<1%
7	Loubrys Lázaro Rojas Reinoso. "Um sistema de análise facial em tempo real para a	<1%

	classificação de postura de faces.", Universidade de São Paulo. Agência de Bibliotecas e Coleções Digitais, 2024 Publication	
8	Babeş-Bolyai University Publication	<1 %
9	Frederik Allotodang, Herman Tolle, Nataniel Dengen. "Design and Evaluation of Bible Learning Application using Elements of User Experience", International Journal of Advanced Computer Science and Applications, 2021 Publication	<1 %
10	Submitted to Liverpool John Moores University Student Paper	<1 %
11	bcc.ime.usp.br Internet Source	<1 %
12	www.algonquincollege.com Internet Source	<1 %
13	www.mdpi.com Internet Source	<1 %
14	Yongzheng Xu, Guizhen Yu, Yunpeng Wang, Xinkai Wu, Yalong Ma. "Car Detection from Low-Altitude UAV Imagery with the Faster R-CNN", Journal of Advanced Transportation, 2017	<1 %

Publication

15	Submitted to University of Warwick Student Paper	<1%
16	www.skyquestt.com Internet Source	<1%
17	Submitted to University of West Florida Student Paper	<1%
18	assets.researchsquare.com Internet Source	<1%
19	www.iic.jku.at Internet Source	<1%
20	abc.us.org Internet Source	<1%
21	"Computer Vision – ECCV 2016 Workshops", Springer Nature, 2016 Publication	<1%
22	Chakradhar Pabba, Praveen Kumar. "An intelligent system for monitoring students' engagement in large classroom teaching through facial expression recognition", Expert Systems, 2021 Publication	<1%
23	export.arxiv.org Internet Source	<1%

24	Gee-Sern Jison Hsu, Hung-Cheng Shie, Cheng-Hua Hsieh, Jui-Shan Chan. "Fast Landmark Localization with 3D Component Reconstruction and CNN for Cross-Pose Recognition", IEEE Transactions on Circuits and Systems for Video Technology, 2017 Publication	<1%
25	"Proceedings of Eighth International Congress on Information and Communication Technology", Springer Science and Business Media LLC, 2024 Publication	<1%
26	Bogdan Kwolek. "Face Detection Using Convolutional Neural Networks and Gabor Filters", Lecture Notes in Computer Science, 2005 Publication	<1%
27	Peng Zhang, Lichao Xing, Ninggou Yang, Guolin Tan, Qingyun Liu, Chuang Zhang. "Redis++: A High Performance In-Memory Database Based on Segmented Memory Management and Two-Level Hash Index", 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications	<1%

(ISPA/IUCC/BDCloud/SocialCom/SustainCom), 2018
Publication

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ______________________________ __Department: __________________ Enrolment No __________

Contact No. _________________________________E-mail. ____

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ___________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)   Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature   Librarian

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com