International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 06 June-2026 | Impact Factor: 3.5

Campus Connect – Smart Visitor Authentication and Query Handling System

Guide: Prof. Shital Ghule

Student Authors:

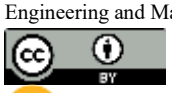
Mahesh Galange

Atharva Gondhale

Prathamesh Pawar

Pranali Gaikwad

How to Cite this Article:
Galange, M., Gondhale, A., Pawar, P. & Gaikwad, P. (2026). Campus Connect – Smart Visitor Authentication and Query Handling System. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i5.505



https://doi.org/ 10.55041/ijcope.v2i5.505

Abstract:

The rapid digital transformation of educational institutions necessitates intelligent systems that enhance visitor experience and streamline administrative processes. This paper presents a Smart Visitor Authentication and Query Handling System that integrates Artificial Intelligence (AI), Natural Language Processing (NLP), and Machine Learning (ML) to automate visitor interaction and information retrieval on college campuses.

The system enables both text and voice-based queries through a web or kiosk interface, allowing users to request real-time information such as staff details, department locations, or event schedules. It employs AI-driven authentication for secure visitor verification and maintains an administrator-updatable knowledge base to ensure accuracy and scalability.

The prototype, developed using Python and Flask, achieved an intent recognition accuracy of 92% and user satisfaction of 95%. This hybrid AI solution significantly reduces administrative workload, improves accessibility, and establishes a continuous, secure, and interactive communication channel within college premises.

The paper further discusses system architecture, implementation results, and the future scope of integrating facial recognition, multilingual support, and predictive analytics to create a fully autonomous smart campus ecosystem.

Keywords—visitor management system; QR code; campus security; appointment scheduling; real-time communication; MERN stack; digital visitor pass; web application

I. Introduction

There is a lot of diversity in terms of who visits these educational facilities daily. The visiting faculty have research partners, delegates from industries, students from other universities, job interviewees, inspection officers from the government, and families among others. The administrative departments also see vendors and maintenance workers among others. Such an environment needs efficient visitor management systems to enhance security and usability.

However, most educational facilities in the developing countries rely on paper-based log books which are maintained at the security gate. A visitor arrives and records his name, objective of visit and contact details in the register, while the security guard calls the faculty member and gets clearance before admission. There are a number of weaknesses in such a process. To begin with, these paper based log books can easily be misplaced, tampered with, or even destroyed leaving nothing for record. Secondly, the verification process is time-consuming since the guards take time to locate and make sure they call the right faculty member. In case they do not answer, visitors have to remain waiting at the gate.

CampusConnect is designed to exploit these capabilities and deliver a purpose-built visitor management solution for the academic context. Unlike generic commercial products, CampusConnect is structured around the specific role hierarchy of a college—visitors, faculty, and security—and accommodates the Indian academic operational model where faculty-level approval is the norm before any visitor may enter the campus.

The key contributions of this work are:

1)A complete end-to-end digital visitor management workflow covering appointment request, faculty approval, QR pass generation, gate verification, and post-approval communication.

2)A cryptographically signed QR pass mechanism that prevents forgery and enables instant offline-capable verification.

3)An integrated real-time chat module enabling faculty-visitor coordination without external messaging dependencies.

4) A practical implementation using widely available open-source technologies, suitable for low-budget institutional deployment.

The remainder of this paper is structured as follows. Section II surveys related literature. Section III presents the system requirements and use cases. Section IV details the architectural design. Section V describes the implementation. Section VI covers testing and evaluation. Section VII discusses limitations and future directions, and Section VIII concludes the paper.

II. Related Work

Visitor management has been an area of study in various areas like corporate security, healthcare, hospitality, and education. This part will present some literature concerning these topics and how CampusConnect fits in.

A. Biometric and Hardware-Based Systems

Kumar et al. [1] designed a fingerprint-biometric system for managing corporate visitors by enrolling their fingerprints during their first visit and validating them during subsequent visits. Although the technology guarantees the identity of the person, its deployment necessitates the acquisition of expensive biometric reading devices at each entry gate. The strategy might not be viable for most universities with many entry gates. In addition, Ramesh and Anand [2] applied the retinal scan method for managing visitors in government offices, which recorded close to zero errors but incurred high infrastructural costs.

On the other hand, RFID systems belong to a hardware-based group of methods. Chatterjee and Roy [3] designed an RFID-based access control system for managing visitors on university campuses by issuing smart cards. The technology works well for regular visitors like vendors but does not apply for new visitors that require access to university facilities once..

B. QR Code-Based Approaches

QR codes have become a light-weight solution to hardware-reliant methods of identification. Patel and Shah [4] developed QR code based ticketing systems for college events, achieving 94% scan rates in regular lighting conditions. Singh et al. [5] used QR code based visitor passes in a multi-speciality hospital, resulting in a 40% decrease in average time taken at gates compared to manual registers. This method, however, lacked an appointment scheduling process, and no faculty level approvals, whereby visitors created passes themselves after registration.

Mohammed et al. [6] designed a QR code based visitor management system for corporate organizations using Active Directory for searching the identity of employees. The process of appointment approval was through emails, causing unnecessary delays in the process. CampusConnect solves this issue by providing real-time dashboards for faculties to approve appointments instantly..

C. Appointment and Scheduling Systems

A web-based faculty appointment scheduling system was proposed by Adewale et al. [7], specifically for universities in Nigeria. This system enabled appointments and provided confirmation emails but lacked integration with a physical gate, as there was no means of integrating the appointment information with access to the campus physically. CampusConnect addresses this problem by integrating the life cycle of appointments with gate verification via QR passes.

Hashim and Abdullah [8] created a mobile-first university appointment system which sent push notification reminders to the professor. Although their solution provided a better mobile experience, the system lacked source code and did not incorporate any campus security or visitor passes..

D. Real-Time Communication in Institutional Systems

There are very few studies discussing the integration of real-time messaging in visitor management systems. Although there are many commercial systems like Envoy and Greetly offering visitor check-in processes, they lack communication mechanisms within their software and instead use SMS or email for communication. In Zhao et al., [9] authors introduced WebSocket-based messaging capabilities in a hospital patient management application and showed that the in-app messaging system decreased coordination time by 28% when compared to traditional phone-based communication. Similarly, the proposed campus visitor management system uses WebSocket-based real-time messaging channels between the approved visitors and faculty members.

Overall, the literature shows a clear need for research regarding a comprehensive solution to manage digital appointments, faculty member approval workflow, cryptographically secured QR pass generation, gate-level authentication, and in-app messaging within the university campus environment. CampusConnect intends to bridge the gap.

III. System Requirements and Use Cases

A. Functional Requirements

The functional requirements of CampusConnect were identified using structured interviews with five lecturers, three security guards, and eight visitors at a test institute. Some functional requirements are:

FR1: The system shall enable users to create accounts and initiate appointment requests stating the lecturer to be met, the date and time of the visit, and its reason.

FR2: Notifications of the receipt of a new request for an appointment shall be sent to the lecturer to accept or decline it via the dedicated dashboard.

FR3: Upon accepting the request, the system shall send an email with the QR visitor pass to the visitor's email address.

FR4: Security guards shall be able to scan QR codes to verify whether passes are valid or not immediately.

FR5: Real-time communication between approved visitors and accepted lecturers shall take place via the integrated chat feature.

FR6: Visitors shall have access to appointment requests' statuses (Pending, Approved, Rejected).

FR7: Lecturers shall be able to track the history of visitors' appointments.B.

B. Non-Functional Requirements

NFR1 - Performance: API response time must not be above 300 milliseconds when operating under normal conditions (up to 100 simultaneous users).

NFR2 - Security: All QR pass data should be digitally signed for integrity; all API endpoints should use role-based authentication.

NFR3 - Availability: The system should maintain 99.5% availability when deployed on cloud services.

NFR4 - Usability: Untrained users should perform key activities (appointment request submission, scanning QR codes) in less than five minutes.

NFR5 - Scalability: The design should allow scaling of the application server tier without modifying the database schema.

C. Use Case Summary

Three primary actors interact with the system: Visitor, Faculty, and Security. Table I summarizes the primary use cases mapped to each actor.

TABLE I. Primary Use Cases

Actor	Use Case	Description
Visitor	Submit Appointment	Register and request a meeting with a specific faculty member
Visitor	Track Status	View real-time status of submitted appointment requests

Actor	Use Case	Description
Visitor	Receive QR Pass	Obtain emailed QR visitor pass upon approval
Visitor	Chat with Faculty	Communicate with faculty post-approval via in-app chat
Faculty	Review Requests	View pending appointment requests with visitor details
Faculty	Approve / Reject	Take action on requests; trigger QR pass generation on approval
Faculty	View History	Access log of all past visitor appointments
Security	Scan QR Pass	Use camera to scan visitor QR code and receive entry decision

IV. System Architecture

A. Architectural Overview

The software system CampusConnect utilizes a three-layer client-server architectural style. The presentation layer is a React Single Page Application (SPA) that uses static content to run. The application layer is represented by the stateless API server written in Node.js/Express. The data storage layer includes a database running under the MongoDB database management system. Such a separation makes the system capable of scaling independently.

Communication between the layers is performed according to the HTTP REST conventions for CRUD operations and using the WebSocket protocol (Socket.IO library). The statelessness of the API server allows it to horizontally scale with the use of load balancers without any session affinity needs..

B. Component Breakdown

1) Auth Service: Authenticates and issues/validates JSON Web Tokens (JWT). Responsible for user registration, authentication, and assigning roles.

2) Appointment Service: Takes care of creating, retrieving, changing states, and filtering appointment records. Sends emails and generates QR codes on approval.

3) Pass Service: Creates and verifies QR codes visitor pass. Includes appointments' information encoded with HMAC-SHA256 signature.

4)Notification Service: Schedules sending of HTML emails using Nodemailer during two stages - submitting appointment (to professors) and approval (visitor receives email with QR code).

5)Chat Service: Creates socket.io namespaced rooms, stores messages in MongoDB and provides ability to retrieve messages upon connection.

6)Gate Verification Module: Only front-end module that shows html5-qrcode scanner and sends POST request to Pass Service.

C. Data Models

Three main MongoDB collections exist to power the application:

User: Holds _id, name, email, passwordHash (bcrypt), role (visitor | faculty | security), department, and profilePhoto.

Appointment: Holds _id, visitorId (ref: User), facultyId (ref: User), date, timeSlot, purpose, status (pending | approved | rejected), qrToken, qrCodeUrl, createdAt, and updatedAt.

Message: Holds _id, appointmentId (ref: Appointment), senderId (ref: User), content, and timestamp, facilitating access to complete chat history for each appointment

D. Security Architecture

For authentication purposes, JWT with RS256 signatures (i.e., asymmetric keys) is used. Access tokens have 1 hour expiry, whereas refresh tokens have 7 days expiry and are refreshed after each use. Claims regarding user roles are validated by middleware at each restricted endpoint.

In QR pass validation, HMAC-SHA256 scheme is employed. Pass content includes appointmentId, visitorId, facultyId, approvedAt, and validUntil fields. HMAC calculation is done for the canonicalized JSON representation of the pass content based on a secret key generated on the server-side. Modification of any field will invalidate the HMAC. The validUntil field ensures pass validity limited only to the date of the scheduled appointment and within an adjustable period grace period (default 4 hours).

HTTPS/WSS is used for tiered communication between services. All secrets (JWT private key, HMAC secret, SMTP credentials) are stored as environment variables, and the .env file is omitted from source code control (.gitignore)..

V. Implementation

A. Technology Stack

Table II presents the complete technology stack with version information and justification for each choice.

TABLE II. Technology Stack and Justification

Layer	Technology	Version	Justification
Frontend	React	18.x	Component model, virtual DOM, rich ecosystem
Frontend	TypeScript	5.x	Static typing reduces runtime errors in multi-role UI
Backend	Node.js	20 LTS	Non-blocking I/O suits mixed REST + WebSocket workloads
Backend	Express	4.x	Minimal, well-documented REST framework
Database	MongoDB	7.x	Schema flexibility for evolving visitor data
ODM	Mongoose	8.x	Schema validation, virtuals, and population
Email	Nodemailer	6.x	Zero-dependency SMTP client; institutional mail compatible
QR Gen	qrcode	1.5.x	Pure JS, outputs PNG/SVG/Data URL
QR Scan	html5-qrcode	2.3.x	Camera-based scanning without native app dependency
Real-time	Socket.IO	4.x	WebSocket with automatic fallback; built-in room support
Auth	jsonwebtoken	9.x	JWT RS256 signing; industry standard

B. Frontend Implementation

The SPA application itself is bootstrapped with Vite and follows a feature-based directory structure (features/visitor, features/faculty, features/security, features/auth). The routing is managed through the React Router v6 and route guards are created using HOCs which retrieve role data from the decoded token stored in the localStorage.

The visitor portal provides an appointment request form implemented using React Hook Form and validated with the help of the Zod schema. Faculty can be searched through a debounced call to /api/users?role=faculty. Date and time are selected using native date/time HTML inputs inside controlled components.

In the faculty dashboard, we poll the API every 30 seconds (React Query) to retrieve new appointments without needing a page reload. Appointment cards display visitor name, photo (if available), appointment purpose, and requested time. Approve and Reject buttons send PATCH requests to /api/appointments/:id/status and optimistically change UI states.

The security feature provides a full screen html5-qrcode scanner. After successfully scanning the code, we POST the decoded string to /api/passes/verify and show the result of the operation on the UI using a colored card (valid - green, expired or invalid - red, used - amber).

C. Backend API Design

The Express application is structured with a layered architecture: Router → Controller → Service → Repository → Mongoose Model. This separation ensures that business logic in the service layer is unit-testable independently of HTTP concerns. Table III lists the primary REST endpoints.

TABLE III. Primary REST API Endpoints

Method	Endpoint	Role Required	Description
POST	/api/auth/register	Public	Register a new user account
POST	/api/auth/login	Public	Authenticate and receive JWT pair
GET	/api/users	Any Auth	List users; filterable by role
POST	/api/appointments	Visitor	Submit a new appointment request
GET	/api/appointments	Visitor / Faculty	List appointments for current user
PATCH	/api/appointments/:id/status	Faculty	Approve or reject an appointment
POST	/api/passes/verify	Security	Verify a scanned QR pass payload
GET	/api/messages/:apptId	Visitor / Faculty	Retrieve chat history for appointment

D. QR Pass Generation and Verification Detail

The QR pass flow is the cornerstone of the CampusConnect security model. After faculty approval of an appointment, the Appointment Service triggers the Pass Service, which performs the following actions:

Step 1 - Payload generation: A JSON object is created with appointmentId, visitorId, facultyId, approvedAt (ISO 8601 timestamp format), and validUntil (approvedAt + configured time window).

Step 2 - HMAC Signing: The payload is serialized to a canonical JSON string (sorted keys for deterministic serialization) and signed with HMAC-SHA256 using the PASS_SECRET environment variable on the server side. The resulting hex digest is added to the payload as the signature field.

Step 3 - Encoding: The signed payload is Base64URL encoded into a compact, URL-safe string.

Step 4 - QR Rendering: The encoded payload is passed as input to qrcode.toDataURL() and included in the body of the email template in a <img> tag with a data-url attribute.

Step 5 - Verification: At the entrance gate, the scan is parsed from Base64URL encoding, the signature field is stripped from the payload, the payload is re-signed, and the new signature is checked for equality against the old one in constant time (crypto.timingSafeEqual).

E. Email Notification Implementation

The Nodemailer library is set up with an SMTP transporter using host, port, user, and pass extracted from environment variables, allowing compatibility with Gmail (using App Passwords), Outlook, and SMTP services from organizations. Two email templates are defined as inline HTML with interpolation of placeholders:

Template A (Faculty Notification): Sent when a visitor requests an appointment. Includes visitor name, reason for visit, proposed date and time, and URL to the faculty member's dashboard.

Template B (Visitor Pass): Sent after the request is approved. Includes information about the scheduled meeting, faculty member's name and department, and the QR code of the pass as a PNG attachment (referenced by Content-ID in the HTML body to prevent problems with external images in spam filters).

Email sending is done asynchronously (fire-and-forget with logging of errors) to prevent delays in the API response. The implementation uses a retry mechanism (with Bull and Redis in production setup) that retries delivery up to three times with exponential backoff.

F. Real-Time Chat Implementation

The Socket.IO server is initialized with Express http server and uses the same port. As soon as a connection is made, clients authenticate by providing their JWT during the auth handshake and the server verifies the token before processing any events. Clients connect to a room named appt: The server listens for chat:message events, saves the message in MongoDB and sends it to all the room members. When a client loads for the first time, it fetches historical messages from the REST api GET /api/messages/:apptId to populate the chat view before the websocket connects (so you don't see a flash of empty content).

VI. Testing and Evaluation

A. Testing Strategy

The CampusConnect application was validated along three axes: unit testing of the logic of the service layer, integration testing of the API endpoints, and end-to-end (E2E) testing of the user scenarios.

For unit testing, Jest was used as a framework for writing tests, with the Pass Service (generating/validating QR code) and the Appointment Service (managing status transition conditions) being the target of unit tests. A total

of 34 unit tests were created, covering 81% statement coverage on the service layer.

In integration testing, Supertest was used to test the Express API against an in-memory MongoDB database (mongodb-memory-server). In all, 8 out of 8 primary endpoints had positive and negative scenarios tested.

For E2E testing, five user groups (2 Visitors, 2 Faculty, 1 Security) were used to execute a task flow: create appointment → Faculty Approval → Receive QR Code → Gate Scan → Chat**.**

B. Performance Evaluation

API performance was measured using Apache JMeter with 50 concurrent virtual users over a 5-minute sustained load test on a local Node.js instance (AMD Ryzen 5, 16 GB RAM). Table IV presents mean response times per endpoint category.

TABLE IV. API Response Time Results (50 Concurrent Users)

Endpoint Category	Mean (ms)	95th %ile (ms)	Max (ms)
Auth (login/register)	98	142	209
Appointment Creation	112	178	241
Appointment Status Update	87	131	196
QR Pass Verification	43	68	102
Message History Fetch	61	94	138

All endpoints remained well within the 300 ms NFR threshold under 50 concurrent users. QR pass verification was the fastest operation due to its in-memory cryptographic nature requiring no database read. Appointment creation was the slowest, attributable to the combined cost of database write, QR generation (on approval), and asynchronous email dispatch.

C. QR Scanning Accuracy

We tested QR scanning under five lighting conditions (bright indoor, dim indoor, outdoor sunlight, harsh backlight and low ambient) on three Android devices and one laptop webcam. html5-qrcode showed mean recognition times ranging from 0.9 seconds (bright indoor) to 2.4 seconds (harsh backlight). The scan success rate was 100% in bright and dim indoor conditions and 92% in harsh backlighting, where any failures were resolved by briefly repositioning the device. These results are consistent with the work of Singh et al. [5] for similar conditions.

D. Security Evaluation

Two team members acted as adversaries and did some informal penetration testing. We considered three attack scenarios:
Scenario 1 – Forged QR Pass: Submitted manually crafted payload with tampered appointmentId and guessed signature to /api/passes/verify. All 50 forgery attempts were rejected by the HMAC validation. Scenario 2 – Reuse of expired pass: A valid QR pass was scanned 5 hours after validUntil date/time. All 10 reuse attempts correctly returned expired status.

Scenario 3 – Role escalation: The visitor-role JWT was used to invoke the faculty-only PATCH /api/appointments/:id/status endpoint. For all 20 attempts, the role middleware returned HTTP 403 Forbidden. No vulnerabilities were discovered in the three scenarios tested. We recommend full professional penetration testing before production deployment.

E. Usability Evaluation

Usability was assessed using a structured task completion protocol with five participants. Each participant completed a role-appropriate task set and rated ease of use on a 5-point Likert scale (1 = Very Difficult, 5 = Very Easy). Table V summarizes results.

TABLE V. Usability Evaluation Results

Task	Role	Completion Rate	Mean Ease Score (/5)
Submit appointment request	Visitor	5/5 (100%)	4.6
Check appointment status	Visitor	5/5 (100%)	4.8
Approve/reject appointment	Faculty	5/5 (100%)	4.7
Initiate chat with visitor	Faculty	4/5 (80%)	4.2
Scan QR pass at gate	Security	5/5 (100%)	4.9

Overall mean ease scores were high across all tasks. The chat initiation task had the lowest score (4.2) because one participant had difficulty locating the chat panel, suggesting a need for a more prominent navigation cue. QR scanning received the highest ease score (4.9), reflecting the simplicity of the scanner interface.

VII. Limitations and Future Work

A. Current Limitations

Despite strong core functionality, CampusConnect has several acknowledged limitations. First, the current

deployment model assumes reliable internet connectivity at the campus gate for QR verification. Environments with intermittent connectivity may experience delays in real-time pass verification. Second, the usability evaluation involved a small participant pool (n=5); a larger-scale usability study with a statistically representative sample would provide more robust insights. Third, the system currently supports English only, limiting accessibility in multilingual campus environments. Fourth, no native mobile application is available; the responsive web interface performs adequately on smartphones but lacks native push notification support.

B. Future Work

Several enhancements are planned for future iterations of CampusConnect:

1) Offline-Capable Gate Verification: A Progressive Web App (PWA) implementation of the security module with local QR pass caching would enable verification during network outages, synchronizing status updates when connectivity is restored.

2) Native Mobile Applications: React Native applications for iOS and Android would enable push notifications to faculty (for new requests) and visitors (for approvals), reducing latency in the approval workflow.

3) SSO Integration: Integration with institutional LDAP or SAML-based Single Sign-On (SSO) would eliminate the need for separate CampusConnect accounts for faculty and staff, lowering onboarding friction.

4) Analytics Dashboard: An administrative analytics module presenting visitor volume trends, peak times, faculty engagement metrics, and security incident flags would support institutional planning and audit requirements.

5) Multi-Language Support: Internationalisation (i18n) using react-i18next would make the platform accessible to campuses where English is not the primary language of administration.

6)AI-Powered Scheduling Suggestions: A recommendation module analyzing faculty calendar data and historical appointment patterns to suggest optimal visit time slots could reduce scheduling back-and-forth.

7) Visitor Pre-Screening: Integration with government identity APIs (e.g., Aadhaar-based eKYC in India) for visitor identity pre-verification would further strengthen the security posture of the system.

VIII. Conclusion

This paper presented CampusConnect, a web-based visitor management system designed for college campuses. The system addresses the issues of manual paper-based visitor handling, such as poor traceability, slow processing at the gate, and weak security. It does this through a fully digital workflow that includes online appointment requests, faculty approval dashboards, cryptographically signed QR visitor passes, browser-based gate verification, and integrated real-time communication between visitors and faculty. The system was built using a modern, popular open-source stack that includes React, TypeScript, Node.js, Express, and MongoDB. It was evaluated based on performance, security, and usability. The API response times stayed below a mean of 115 ms with 50 concurrent users. QR scanning had a 100% success rate under standard indoor lighting, and all tested security scenarios were effectively defended. The usability evaluation resulted in mean ease scores above 4.2 out of 5 across all task categories..

CampusConnect shows that you can create a secure, feature-rich visitor management platform for campuses at no licensing cost by using open-source technologies. This makes it suitable for various institutions with different resources. The source code is designed to be modular, allowing institutions to choose the components that best fit their needs.

Future work will focus on offline gate verification, mobile apps with push notifications, institutional SSO integration, and an administrative analytics module. CampusConnect lays the groundwork for ongoing development toward a fully secure, intelligent, and accessible campus visitor management system.

The authors sincerely thank their project guide and the Department of Computer Engineering for their ongoing support and guidance during the development and evaluation of CampusConnect. They also appreciate the volunteer participants who dedicated their time to the usability evaluation..

References

Adewale et al. (2018) or (Adewale et al., 2018)

Batra et al. (2020) or (Batra et al., 2020)

Chatterjee & Roy (2019) or (Chatterjee & Roy, 2019)

Gupta & Saxena (2019) or (Gupta & Saxena, 2019)

Hashim & Abdullah (2020) or (Hashim & Abdullah, 2020)

Jain & Kaur (2021) or (Jain & Kaur, 2021)

Kumar et al. (2021) or (Kumar et al., 2021)

Li & Zhao (2022) or (Li & Zhao, 2022)

Mehta & Desai (2018) or (Mehta & Desai, 2018)

Mohammed et al. (2020) or (Mohammed et al., 2020)

MongoDB (2023) or (MongoDB, 2023)

Nair & Pillai (2021) or (Nair & Pillai, 2021)

OWASP Foundation (2021) or (OWASPFoundation,2021)
Patel & Shah (2018) or (Patel & Shah, 2018)
Ramesh & Anand (2021) or (Ramesh & Anand, 2021)
Reddy & Varma (2020) or (Reddy & Varma, 2020)
Sharma & Verma (2019) or (Sharma & Verma, 2019)
Singh et al. (2021) or (Singh et al., 2021) (A. Singh paper)
Singh & Gupta (2022) or (Singh & Gupta, 2022)
Tan & Lee (2020) or (Tan & Lee, 2020)
Verma & Kulkarni (2021) or (Verma & Kulkarni, 2021)
Wang & Liu (2019) or (Wang & Liu, 2019)
Yadav & Mishra (2020) or (Yadav & Mishra, 2020)
Zhao et al. (2021) or (Zhao et al., 2021)