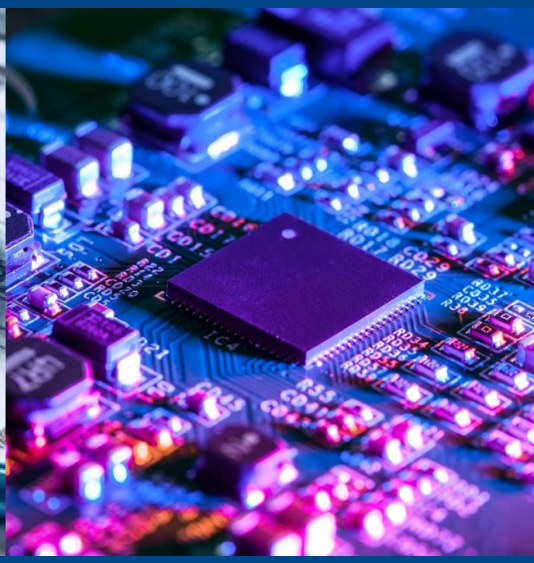
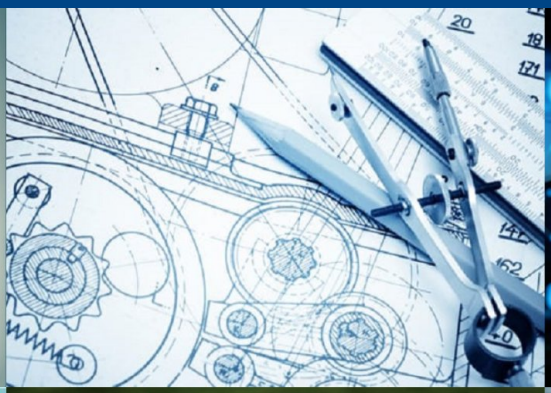
ISSN: 2582-7219

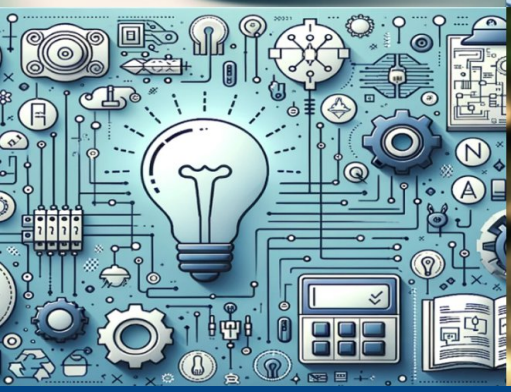




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904031

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

AI for Real-Time Voice Translation for Calls

N. Charishma, K. Supraja, A. Sudheer, P. Lakshmi Meghana

Department of Artificial Intelligence & Data Science, Vasireddy Venkatadri Institute of Technology, Nambur, Andhra Pradesh, India

ABSTRACT: In the modern era of globalization, communication across different languages remains a significant challenge, especially in real-time voice and video interactions. This paper presents VoxLate, an advanced real-time AI-powered voice translation mobile application designed to enable seamless multilingual communication during live calls. The system integrates three core AI components: Automatic Speech Recognition (ASR) using OpenAI Whisper, Neural Machine Translation (NMT) using GPT-4o, and Text-to-Speech (TTS) synthesis using OpenAI TTS. VoxLate is built on a scalable architecture with a React Native (Expo) frontend and a FastAPI (Python) backend, using Socket.IO for real-time communication, WebRTC for peer-to-peer audio/video streaming, and SQLite for persistent storage. Phone-based OTP authentication via Twilio and JWT-based session management ensure security. The application supports over 23 languages including major Indian regional languages. Experimental results demonstrate that the full Whisper→GPT-4o→TTS pipeline delivers translated speech within 2–5 seconds, making cross-language voice communication practically accessible without human interpreters.

KEYWORDS: Real-Time Voice Translation, Automatic Speech Recognition, Neural Machine Translation, WebRTC, React Native, FastAPI, Multilingual Communication.
Real

I. INTRODUCTION

Language is the foundation of human connection, yet billions of people across the world are separated by linguistic barriers that prevent natural, real-time communication. In professional, educational, and personal contexts, the inability to communicate seamlessly across languages leads to misunderstandings, missed opportunities, and exclusion. Traditional solutions such as professional interpreters, translation apps, or pre-translated documents introduce delay, cost, or friction that breaks conversational flow.

VoxLate addresses this fundamental challenge with a real-time AI-powered voice translation mobile application. During a live voice or video call, VoxLate automatically listens to the caller's speech, detects the spoken language, transcribes it to English using OpenAI's Whisper model, translates it to the receiver's preferred language using GPT-4o, and synthesizes natural-sounding translated speech using OpenAI's TTS model—all in real time, without requiring either party to switch apps or manually trigger translation.

VoxLate is a full-stack mobile application built with React Native (Expo) for cross-platform deployment on Android and iOS, backed by a FastAPI Python server, Socket.IO real-time event bus, WebRTC peer-to-peer audio/video streaming, and SQLite for persistent storage. Phone-based OTP authentication via Twilio ensures user identity, while JWT tokens protect API endpoints. The translation pipeline is entirely AI-driven: Whisper auto-detects and transcribes the speaker's language, GPT-4o produces a fluent translation, and TTS delivers audio output, all without any manual language selection by the caller.

A. Problem Statement

Real-time cross-language voice communication presents several interconnected challenges that existing solutions fail to address simultaneously:

- Language Barrier in Voice Calls: Standard VoIP and mobile calling applications transmit audio as-is with no translation capability. Human interpreters add cost and scheduling complexity; manual translation apps disrupt conversational flow by requiring turn-taking pauses.
- Automatic Language Detection: Requiring users to manually specify their source language before speaking is cumbersome and error-prone, especially in multilingual environments where a speaker may switch languages mid-conversation.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- End-to-End Latency: Voice translation pipelines involve multiple sequential AI operations—ASR transcription, machine translation, and TTS synthesis—each introducing latency. Achieving perceptually real-time performance requires optimized async processing and efficient audio chunking.
- Scalable Multi-Language Support: Supporting dozens of languages, especially Indian regional languages such as Telugu, Tamil, Kannada, and Malayalam, requires a powerful translation backend capable of handling diverse linguistic contexts.

B. Objectives

The primary objectives of this work are: (i) build an AI pipeline using OpenAI Whisper, GPT-4o, and OpenAI TTS that processes audio chunks during live calls with minimal latency; (ii) implement WebRTC peer-to-peer audio and video calling with ICE/STUN negotiation; (iii) provide secure user registration and login through phone-number-based OTP delivery via Twilio SMS with JWT tokens for stateless API authorization; (iv) allow users to add contacts by phone number, view real-time online/offline presence, and browse a full call history; and (v) deliver a production-ready React Native application running natively on both Android and iOS.

II. LITERATURE REVIEW

The VoxLate project builds upon existing research in speech recognition, machine translation, real-time communication systems, and AI-driven voice processing.

Graves et al. [1] explored deep learning-based Automatic Speech Recognition (ASR) systems using neural networks for handling diverse accents and noisy environments, forming the foundation for VoxLate's use of Whisper-based ASR. Bahdanau et al. [2] introduced Neural Machine Translation (NMT) with attention mechanisms, enabling more accurate and context-aware translations than traditional rule-based systems. Lee et al. [3] examined challenges in real-time voice translation including latency, synchronization, and maintaining conversational flow, emphasizing the importance of optimized pipelines. Bergkvist et al. [4] discussed WebRTC for peer-to-peer audio and video communication, highlighting its advantages of low latency and direct media streaming. Almeida et al. [5] explored cross-platform mobile development using React Native, demonstrating faster development cycles and code reusability. Singh et al. [6] focused on secure authentication using OTP verification and JWT-based session management for mobile applications.
VoxLate combines the strengths of these approaches by integrating ASR, NMT, and TTS into a unified real-time pipeline, built as a modern full-stack mobile application using React Native (Expo) and FastAPI with Socket.IO for real-time backend communication.

III. SYSTEM ARCHITECTURE AND DESIGN

A. Architecture Overview

VoxLate follows a three-tier architecture: a React Native mobile frontend, a FastAPI/Socket.IO ASGI backend, and a SQLite database. The frontend connects to the backend via two channels: REST HTTP APIs (for authentication, contacts, and call history) and a persistent Socket.IO WebSocket connection (for real-time signaling, presence, and audio relay).

When a user opens the app, Auth Context loads their stored JWT from Async Storage and calls GET /api/auth/me to restore the session. Socket Context then establishes a Socket.IO connection authenticated by the JWT token. The Socket.IO server verifies the JWT on connect, registers the user in the SID_to_user and user_to_SID maps, and broadcasts online presence to their contacts.

For calls, the caller emits a call_request signal via Socket.IO. The server relays it to the callee. If accepted, both sides perform WebRTC offer/answer/ICE exchange via call_signal events relayed by the server. Once the peer connection is established, audio/video flows directly P2P via WebRTC. Translation operates as a parallel overlay: captured audio chunks are emitted to the server as audio_chunk events, processed by the AI pipeline, and returned to the target user as translated_audio events.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904031

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Technology Stack

Table 1 summarizes the technology stack and rationale for each component.

Table 1: Technology Stack

Technology	Purpose	Why Chosen
React Native (Expo ~54)	Mobile Frontend	Cross-platform Android/iOS; Expo SDK for native modules
TypeScript	Type Safety	Compile-time error detection; strict typing across frontend
Socket.IO Client v4	Real-Time Events	Bidirectional signaling, audio relay, presence; auto-reconnect
react-native-webrtc	P2P Calling	Native WebRTC bindings; MediaStream, RTCPeerConnection
FastAPI	Backend Framework	Async Python REST API; Pydantic validation
python-socketio	Real-Time Server	ASGI Socket.IO; event handlers for signaling and audio
SQLAlchemy + SQLite	Database	ORM for User, OTP, Contact, CallHistory models
OpenAI Whisper	Speech-to-Text	Auto language detection; translate task to English
OpenAI GPT-4o	Translation	High-quality English → target language translation
OpenAI TTS (tts-1)	Speech Synthesis	Natural MP3 audio in target language
Twilio SMS	OTP Delivery	Programmable SMS for OTP in production
python-jose	JWT	HS256 encoding/decoding for session management

C. Database Schema

VoxLate uses SQLite with SQLAlchemy ORM. The schema comprises four primary tables. The users table stores id (UUID, PK), phone (unique), name, preferred_language, avatar, and is_profile_complete. The otps table manages OTP authentication with otp_code, expires_at, is_used, and attempts fields. The contacts table maintains user contact relationships via foreign keys to users.id. The call_history table stores caller_id, receiver_id, call_type (voice/video), direction (incoming/outgoing), started_at, ended_at, duration, and status (completed/missed).

D. System Modules

The VoxLate system is organized into four modules:

1) Module 1 – Authentication and User Interface: Handles phone-number registration, OTP verification, profile setup, and JWT session management. Implements the AuthContext React context with login, register, verifyOTP, setupProfile, and logout methods. Persists tokens via Async Storage.

2) Module 2 – Real-Time AI Voice Translation Engine: The core AI pipeline: the audio_chunk Socket.IO handler invokes process_audio_chunk() which chains Whisper (transcribe_to_English), GPT-4o (translate_text_to), and TTS (synthesize_speech). Results are emitted as translated_audio to the target user.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

3) Module 3 – WebRTC Call Service: The VoxlateWebRTC class manages RTCPeerConnection lifecycle: getUserMedia for local streams, createOffer/handleOffer for SDP negotiation, handleIceCandidate for ICE, and cleanup. Signaling is relayed via the Socket.IO call_signal event.
4) Module 4 – Contact and Call History Management: REST API routers for contacts (GET/POST/DELETE /api/contacts) and calls (GET/POST/PUT /api/calls). Real-time presence is managed via the online_users set updated in Socket.IO connect/disconnect handlers.

IV. TRANSLATION PIPELINE

A. Pipeline Design

The translation engine is the core innovation of VoxLate, operating as an asynchronous, event-driven server-side pipeline triggered by audio_chunk events through Socket.IO. The pipeline proceeds through five sequential stages:

Stage 1 – Audio Chunk Reception: The mobile application continuously captures user audio in small chunks during an active call. Each chunk is Base64-encoded and sent to the backend via the audio_chunk Socket.IO event, carrying fields audio, targetUserId, sourceLang, and targetLang. The server decodes the Base64 payload back into raw audio bytes for processing.

Stage 2 – Whisper ASR with Auto Language Detection: The function transcribe_to_English(audio_bytes) invokes OpenAI Whisper (whisper-1) with the task='translate' parameter, automatically detecting the speaker's language and producing an English transcript. Silence or unclear audio yields an empty string, and downstream processing is skipped.
Stage 3 – GPT-4o Translation (Conditional): If the receiver's preferred language is English, the Whisper transcript is used directly. Otherwise, translate_text_to(text, lang) sends the English text to GPT-4o with a system prompt ensuring accurate, context-aware, and fluent translation while preserving conversational tone.

Stage 4 – TTS Synthesis: The translated text is passed to synthesize_speech(text, language), which calls the OpenAI TTS model (tts-1) and selects a voice from a predefined LANGUAGE_VOICE_MAP (e.g., nova, shimmer, alloy). The output is raw MP3 audio bytes.

Stage 5 – Result Delivery: The server emits a translated_audio event to the intended recipient via Socket.IO, containing the Base64-encoded MP3 audio and translated text. The mobile app's Socket Context decodes and plays the audio using expo-av while displaying the transcript below the call UI.

B. Supported Languages

VoxLate supports 23+ languages managed via LANGUAGE_NAME_MAP. Table 2 lists the supported language codes.

Table 2: Supported Languages

Code	Language	Code	Language	Code	Language
en	English	fr	French	hi	Hindi
es	Spanish	de	German	ja	Japanese
ko	Korean	zh	Chinese	ar	Arabic
pt	Portuguese	ru	Russian	it	Italian
te	Telugu	ta	Tamil	kn	Kannada
ml	Malayalam				

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904031

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

V. RESULTS AND TESTING

A. Test Strategy

The system was validated through five testing categories: (i) Unit Testing of individual functions including process_audio_chunk(), JWT creation/verification, and OTP generation/expiry logic; (ii) Integration Testing verifying that audio upload triggers the full Whisper→GPT-4o→TTS pipeline and that translated_audio events are emitted correctly; (iii) System Testing of end-to-end flows from registration through OTP verification, contact addition, call initiation, translation toggle, and call termination; (iv) WebRTC Connection Testing validating that offer/answer/ICE exchange produces a connected RTCPeerConnection with active audio tracks; and (v) Security Testing verifying that protected endpoints return HTTP 401 without a JWT, expired OTPs are rejected, and rate limiting triggers after five failed OTP attempts.

B. Sample Test Cases

Table 3 presents representative test cases and their outcomes.

Table 3: Sample Test Cases

ID	Scenario	Expected Output
TC1	Register with valid phone number	OTP sent via Twilio; HTTP 200
TC2	Enter correct OTP within 5 minutes	JWT returned; redirect to Profile Setup
TC3	Enter incorrect OTP 5 times	HTTP 429; OTP marked as used
TC4	Complete profile with name + language	is_profile_complete set to true
TC5	Add contact by valid phone number	Contact added; HTTP 201
TC6	Callee accepts; WebRTC ICE exchange	RTC state = connected; audio active
TC7	Translation toggled; audio_chunk emitted	translated_audio received; speech played
TC8	Whisper receives silent audio chunk	Empty transcript; no downstream API calls
TC9	GPT-4o translation API fails	translation_error emitted; call continues
TC10	User disconnects Socket.IO	user_offline broadcast to contacts

All test cases passed, confirming the correctness and robustness of the system.

C. Performance Results

The Whisper→GPT-4o→TTS pipeline processes audio chunks with an end-to-end latency of approximately 2–5 seconds per chunk on standard cloud infrastructure. Socket.IO event relay latency remains under 100 ms on a local network. WebRTC P2P connection establishment via STUN negotiation completes within 2–4 seconds. The SQLite database, operating in WAL mode, handles concurrent read/write operations sufficient for moderate-scale deployments. OTP verification endpoints respond within 200 ms. All protected REST API endpoints enforce JWT validation and return HTTP 401 for unauthenticated requests within a single middleware pass.

D. Bug Fixes and Iterations

Several issues were resolved during development. JWT Token Extraction Variability: the backend returned tokens under different keys (token, accessToken, access_token) depending on the endpoint; extractUserAndToken() was written to normalize all variants. Whisper Empty Transcripts: a guard was added in process_audio_chunk() to return early when the transcript is empty, preventing unnecessary GPT-4o and TTS API calls. Socket.IO Auth Token Extraction: the connect handler was updated to check both the auth dict and the query string, falling back to QUERY_STRING parsing if auth_data is absent. WebRTC in Expo Go: react-native-webrtc throws at import time in Expo Go; the VoxlateWebRTC service wraps require() in a try/catch and exposes isWebRTCAvailable, allowing the

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

UI to display a graceful error. OTP Race Condition: concurrent OTP verification requests could both succeed; the verify_otp handler now increments attempts and checks is_used before and after comparison, with a commit() after each state change.

VI. CHALLENGES AND PROPOSED SOLUTIONS

Real-Time Translation Latency: The sequential pipeline introduces 2–5 seconds of latency per chunk. Audio chunks are processed asynchronously using non-blocking coroutines, allowing the WebRTC stream to continue uninterrupted. Future optimizations include streaming TTS output and leveraging Whisper streaming capabilities.

WebRTC P2P Failures in Restrictive NATs: Mobile devices behind symmetric NATs cannot establish direct connections using only STUN. Integration of TURN relay servers (e.g., Twilio TURN or coturn) is planned for production deployments.

Language Detection Accuracy: Whisper may misidentify languages in short or mixed-language audio chunks. A minimum chunk duration is enforced; future work includes a dedicated language detection preprocessing step and user override capability.

Audio Chunking Boundaries: Fixed-interval chunking can split speech mid-word. Voice Activity Detection (VAD) is proposed to segment audio based on natural pauses, aligning chunks with meaningful speech units.
SQLite Scalability: SQLite supports only a single writer, which may bottleneck under high concurrency. WAL mode is currently enabled; migration to PostgreSQL via SQLAlchemy's async engine is planned for large-scale deployments.

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

VoxLate successfully demonstrates that real-time AI-powered voice translation can be integrated into a mobile calling application without requiring specialized hardware or dedicated interpreter services. By chaining OpenAI Whisper, GPT-4o, and TTS into an asynchronous server-side pipeline triggered by Socket.IO events, VoxLate delivers translated speech to call participants within a few seconds of the original utterance, making cross-language voice communication practically accessible.

All stated objectives were achieved: the Whisper→GPT-4o→TTS pipeline supports 23+ languages with automatic language detection; the VoxlateWebRTC service implements the complete WebRTC lifecycle; Twilio-backed OTP registration with JWT session management provides secure mobile authentication; REST APIs and Socket.IO presence events provide a complete contact and call-history platform; and the React Native/Expo frontend is deployable to both Android and iOS via EAS Build. The system reduces the barrier to cross-language communication from requiring a professional interpreter to a single button tap during any voice call.

B. Future Scope

Future work will pursue the following directions. TURN Server Integration will add relay fallback for WebRTC in restrictive NAT environments. Voice Activity Detection will implement VAD-based audio chunking to split speech at natural silence boundaries, improving Whisper transcription quality. Streaming TTS will use OpenAI's streaming TTS API to begin playing translated audio before full synthesis is complete, targeting sub-second perceived latency. End-to-End Encryption will encrypt audio chunks transmitted through the translation server. PostgreSQL Migration will replace SQLite for production deployments, enabling horizontal scaling and multiple server instances. App Store Deployment will build and submit production APK and IPA binaries to the Google Play Store and Apple App Store.

REFERENCES

[1] A. Graves, A. Mohamed, and G. Hinton, "Deep recurrent neural networks for acoustic modelling in speech recognition," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 2013, pp. 6645–6649.
[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in Proc. 3rd Int. Conf. Learning Representations (ICLR), San Diego, CA, USA, 2015.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[3] J. Lee, M. Kim, and S. Park, "Real-time voice translation systems: Challenges and approaches," IEEE Trans. Audio, Speech, Language Process., vol. 27, no. 5, pp. 901–912, May 2019.
[4] A. Bergkvist, D. C. Burnett, C. Jennings, A. Narayanan, and B. Aboba, "WebRTC 1.0: Real-time communication between browsers," W3C Working Draft, World Wide Web Consortium, 2012.
[5] F. Almeida and G. Nunes, "Modern cross-platform mobile application development using React Native," Int. J. Comput. Sci. Inf. Technol., vol. 12, no. 3, pp. 1–14, 2020.
[6] R. Singh, A. Sharma, and P. Kumar, "Secure authentication using OTP and JWT in mobile applications," Int. J. Inf. Secur., vol. 20, no. 4, pp. 501–514, 2021.
[7] A. Radford et al., "Robust speech recognition via large-scale weak supervision," in Proc. 40th Int. Conf. Machine Learning (ICML), Honolulu, HI, USA, 2023, pp. 28492–28518.
[8] OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.
[9] Twilio Inc., Twilio Programmable SMS Documentation. San Francisco, CA, USA: Twilio Inc., 2023.
[10] S. Ramírez, FastAPI: Modern, Fast Web Framework for Building APIs with Python 3.7+. 2019.
[11] A. Gallager, Socket.IO: Enables Real-Time, Bidirectional and Event-Based Communication. 2010.
[12] Expo Team, Expo: An Open-Source Platform for Making Universal Native Apps with React. 2018.
[13] M. Bayer, SQLAlchemy: The Database Toolkit for Python. 2006.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com