Arpith Paliwal — Portfolio Knowledge Base

(RAG Context Document)

---

1. Basic Introduction

My name is **Arpith Paliwal**.
I am a graduate of **Chandigarh University**, where I completed my **Bachelor's degree in Mechatronics Engineering**.

I am currently based in **Warangal, Telangana**, and I am comfortable communicating in **English, Hindi, Telugu, and Marathi**, which allows me to collaborate effectively with people from diverse backgrounds.

I have a strong engineering foundation with a focused transition into **software engineering, backend systems, and applied AI**. I enjoy working on real-world problems that require **system-level thinking, performance awareness, and clean architecture** rather than isolated features.

---

2. Professional Summary (Who I Am)

I am a **full-stack developer** with a strong emphasis on **backend architecture, real-time systems, and applied AI engineering**.

I have built:

- Production-style **real-time systems**
- **Scalable backend architectures**
- **AI-powered Retrieval-Augmented Generation (RAG) applications**
- An **autonomous robotics system** integrating hardware, software, and AI

I prefer understanding **why systems behave the way they do**, not just how to use tools. I focus on **reliability, scalability, and real-world constraints**, not just demos.

---

3. Core Strengths

- System-level thinking and architecture design
- Backend-heavy full-stack development
- Real-time systems (WebSockets, Redis, queues)
- Applied AI (RAG, LangChain, vector databases)
- Leadership, ownership, and team coordination

- Production-oriented mindset

---

4. Technologies I Am Most Comfortable With

Frontend

- React (Vite)
- TypeScript / JavaScript
- Tailwind CSS
- Redux Toolkit
- TanStack React Query
- React Router
- React Hook Form + Zod

Backend

- Node.js
- Express.js
- MongoDB (Mongoose ODM)
- REST API design
- JWT-based authentication
- Cookie-based auth flows

Real-Time & Infrastructure

- Socket.IO (WebSockets)
- Redis (caching, rate limiting)
- RabbitMQ (async messaging)
- Microservices architecture

AI & Applied ML

- Retrieval-Augmented Generation (RAG)
- LangChain
- Vector databases (Qdrant)
- Embedding pipelines
- Prompt engineering for grounded responses

Robotics & Systems

- ROS / ROS-2
- Raspberry Pi 4
- Python
- Embedded system coordination

---

5. Projects I Am Most Proud Of (High-Level)

I am most proud of four projects:

1. **Em Matladutunavu Ra** — Real-time chat application
2. **Nexora** — Video sharing platform inspired by YouTube
3. **RAG-Powered Interactive Portfolio**
4. **ARGOS** — Autonomous robotic guidance system

Each project focuses on **production-grade system design**, not academic demonstrations.

---

6. Project: Em Matladutunavu Ra — Real-Time Chat Application

Overview

Em Matladutunavu Ra is a **full-stack, real-time chat application** designed to simulate real-world messaging platforms like WhatsApp or Telegram.

It focuses on:

- Scalable backend architecture
- Secure authentication
- Optimized real-time communication
- Clean frontend UX

Architecture

The backend follows a **microservices architecture** with clear separation of concerns.

Services

- **User Service**
  Handles authentication, profile management, and user data.
- **Chat Service**
  Manages conversations, messages, and WebSocket connections using Socket.IO.
- **Mail Service**
  An asynchronous worker that sends OTP and system emails using RabbitMQ.

Communication

- REST APIs for synchronous calls
- RabbitMQ for asynchronous, event-driven workflows

Tech Stack

Backend

- Node.js + Express
- TypeScript
- MongoDB (Mongoose)
- Socket.IO
- Redis (caching, rate limiting)
- RabbitMQ
- JWT authentication
- Cloudinary for media storage

Frontend

- React (Vite)
- TypeScript
- Tailwind CSS
- Redux Toolkit
- TanStack React Query
- React Router
- React Hook Form + Zod

Key Features

- Real-time private and group chats
- Media sharing (images/videos)
- Infinite scroll message history
- Optimistic UI updates
- Unread message tracking
- Secure JWT authentication with HTTP-only cookies
- OTP-based email verification
- Redis-backed rate limiting
- Responsive UI with dark/light mode

Live Demo:
https://em-matladutunavu-ra-frontend.vercel.app/

---

7. Project: Nexora — Video Sharing Platform

Overview

Nexora is a **feature-rich video sharing platform** inspired by YouTube, designed to demonstrate **scalable API design, content management, and frontend performance optimization**.

Core Capabilities

- Secure JWT authentication (access + refresh tokens)
- Video & thumbnail uploads (Cloudinary)
- Full CRUD for videos
- Hover-based video previews
- Infinite scrolling and lazy loading
- Subscriptions, likes, comments, playlists
- Watch history and user dashboards
- Voice-based search
- Dark / light theme support

Tech Stack

Frontend

- React (Vite)
- Redux Toolkit
- TanStack React Query
- Tailwind CSS
- Axios

Backend

- Node.js
- Express.js
- MongoDB
- JWT
- Multer
- Cloudinary
- Bcrypt

Live Demo:
https://arpithpaliwal-nexora.vercel.app/

---

8. Project: RAG-Powered Interactive Portfolio

What Problem I Solved

Traditional portfolios are static.
Traditional chatbots hallucinate.

I built a **document-grounded RAG system** where:

- Users can ask natural language questions
- The AI answers **only from my portfolio data**
- Hallucinations are explicitly prevented

RAG System Architecture

Document Ingestion

- Portfolio content ingested via LangChain loaders
- Recursive chunking with tuned size & overlap
- Semantic coherence preserved

Embeddings

- Generated using Google GenAI embedding models
- Meaning-based similarity search

Vector Storage

- Qdrant as vector database
- Abstracted vector store layer for flexibility

Retrieval Strategy

- Query embedding → top-k similarity search
- Limited retrieval to reduce noise

Prompt Engineering

- Strict instructions:
  - Answer only from retrieved context
  - Say “I don’t know” if context is insufficient

LLM Integration

- LangChain abstractions
- Groq LLMs
- Model-agnostic design

Why This Is Production-Ready

- Modular ingestion, retrieval, generation layers
- Scales with more documents
- No model retraining required
- Low hallucination risk
- Efficient token usage

One-Line Explanation:

"I built an interactive portfolio using a production-grade RAG system that allows users to query my projects and experience conversationally, with grounded responses instead of static content."

---

9. Project: ARGOS — Autonomous Robotic Guidance and Operations System

Overview

ARGOS (Autonomous Robotic Guidance and Operations System) is a **real, AI-powered autonomous robotic assistant currently deployed at my university**.
It is designed to operate in **campus and institutional environments**, where it assists users through autonomous navigation, voice-based interaction, and an interactive kiosk-style interface.

ARGOS is not a simulation or prototype. It is a **physically deployed robotic system** that integrates robotics, backend systems, networking, and AI-driven interaction into a single, cohesive platform.

The system integrates:

- Robotics using **ROS-2**
- **Lidar for mapping and navigation**
- Backend APIs for coordination and control
- Voice-based command and response
- Mobile application for monitoring and control
- Fullscreen **web-based kiosk interface** for on-site interaction

---

Core Architecture

ARGOS follows a **distributed yet centrally coordinated architecture**, with clear separation between control, communication, and interaction layers.

- **Raspberry Pi 4 (RPI-4)**
  Acts as the central control unit, coordinating all subsystems and handling communication between software, hardware, and network services.

- **ROS-2 Framework**
  Manages autonomous navigation, path planning, obstacle avoidance, sensor data processing, and motion execution.
- **API Server**
  Serves as the communication bridge between the robot, the mobile application, and external interfaces, enabling real-time command execution and status updates.
- **Mobile Application**
  Allows remote monitoring, command issuance, and system status visualization over the local network.
- **Voice Command & Text-to-Speech Modules**
  Enable hands-free interaction by accepting spoken commands and providing audio feedback to users in real time.
- **Web-Based Kiosk Interface**
  A fullscreen touch-enabled interface running in kiosk mode on the robot, used for direct user interaction, navigation guidance, and system feedback.

End-to-End System Flow

1. A user issues a command via **voice input**, **mobile application**, or **touch-based kiosk interface**
2. The request is received and processed by the **Raspberry Pi 4**
3. Control logic is coordinated with **ROS-2** for navigation or task execution
4. The robot performs the required action (movement, guidance, response)
5. Status updates and feedback are delivered via:
   - Audio responses (TTS)
   - Visual kiosk interface
   - Mobile application updates

This flow ensures **real-time responsiveness**, reliability, and smooth human–robot interaction.

Capabilities

- **Autonomous Navigation**
  Independently navigates campus environments using ROS-2, with obstacle avoidance and adaptive path planning.
- **Voice-Based Interaction**
  Supports hands-free operation through speech recognition and real-time command interpretation.
- **Interactive Kiosk Mode**
  Touch-enabled fullscreen web interface for direct interaction, guidance, and information display.
- **Battery-Powered Mobility**
  Operates untethered with onboard battery power, enabling flexible deployment across campus locations.

- **Remote Monitoring & Control**
  Supports real-time monitoring and command execution over Wi-Fi using a static IP setup.
- **Secure System Access**
  Provides SSH-based remote access for maintenance, debugging, and updates.
- **Remote Exposure via Ngrok**
  Enables secure external access to the local API server for testing and remote monitoring without public deployment.

Deployment & Real-World Usage

ARGOS is **actively deployed within my university campus**, where it is used to:

- Assist visitors and students
- Provide navigation and guidance
- Demonstrate intelligent automation in institutional environments
- Showcase real-world integration of robotics, AI, and web systems

The system is designed with **deployability, maintainability, and scalability** in mind, rather than as an academic demonstration.

Design Philosophy

ARGOS was built with a strong emphasis on:

- **Real-world reliability over theoretical perfection**
- **Clear separation of concerns between subsystems**
- **Scalability for future feature expansion**
- **Safe operation in dynamic human environments**

Summary

ARGOS is a **production-grade autonomous robotic system**, combining ROS-2–based navigation, backend APIs, voice interaction, mobile control, and a web-based kiosk interface. Its real-world deployment at my university demonstrates my ability to design, integrate, and operate **complex, cross-domain systems that span hardware, software, networking, and AI**.

---

10. Leadership & Team Experience

In all my major group projects, I acted as the **team lead**.

Responsibilities

- Defined system architecture
- Distributed tasks based on strengths
- Coordinated timelines
- Resolved technical and interpersonal conflicts
- Ensured successful delivery

What I Learned

- Engineering is as much about **people** as code
- Clear communication prevents most failures
- Ownership drives results

All group projects I led were **completed successfully**.

---

11. How I Learn New Technologies

I learn by **building real systems**.

My approach:

1. Understand fundamentals
2. Apply them in a real project
3. Observe trade-offs and constraints
4. Refine design based on failures

This helps me understand **when and why** to use a technology, not just how.

---

12. Why You Should Hire Me

You should hire me because I bring:

- Strong technical execution
- System-level thinking
- Leadership experience
- Ownership mentality

I focus on building **clean, scalable, and reliable systems** that work in real-world conditions. I adapt quickly, learn continuously, and take responsibility for the outcomes of what I build.

---

13. Future Goals

I see myself growing as a **backend and applied AI engineer**, designing scalable systems and contributing to high-impact products. I aim to deepen my expertise in **system architecture and distributed systems** over the next few years.

---

14. One-Line Summary (Anchor Statement)

I am an engineer who combines **strong backend systems, applied AI, and leadership experience** to build **production-grade, scalable solutions**, not just demos.
