About Lakshya Paliwal: I am Lakshya Paliwal, currently in my third year at Manipal University Jaipur, originally from Gurgaon, India. My interests lie in building full-stack applications and exploring machine learning, with a particular focus on generative AI and the foundational aspects of large language models. I am passionate about working with AI and automation, and to gain practical experience, I am currently working at a startup since July, Oddmind Innovations (https://interviews.zariya.ai/), where I am developing an AI-powered interview platform designed to help students prepare for interviews. Developed using **Next.js** for the frontend and **Express.js** for the backend. My role involves integrating Amazon S3 for recording storage, leveraging Gemini for evaluation and feedback, and implementing LiveKit WebRTC for conducting interviews—an upgrade from our previous use of OpenAI, which struggled to maintain long conversational contexts effectively. Beyond my startup work, **Integrated Razorpay Payment Gateway** to enable secure and seamless payment processing. **Implemented a Docker-based CI/CD pipeline** on AWS — every new commit triggers an automated image build, deployment, and EC2 instance update with the latest version. **Scaled the platform** to support **100+ concurrent real-time AI-driven interviews**, ensuring high performance, reliability, and low latency. **Designed and implemented a multipart upload pipeline** for AWS S3 video recordings, reducing upload time by **90%** compared to single-file uploads. I also collaborated with senior researchers on the Chronocept project, a study aimed at giving large language models a sense of time. My contribution was focused on data annotations for two benchmarks we created to train the model. This involved defining a primary axis, where we plotted metrics like the average, maximum, and minimum time of specific events, and a secondary axis for additional categorizations, as detailed in the attached research excerpt. The full paper is available at https://arxiv.org/abs/2505.07637. In the Chronocept project, the annotation process followed a structured, multi-step protocol to ensure temporal consistency and semantic clarity. The first step, **Temporal Segmentation**, involved breaking down each text sample into coherent subsegments that preserved temporal markers—this step was crucial for maintaining an accurate timeline of events. Once segmented, each portion was assigned both a **primary axis** and a **secondary axis** through the **Axis Categorization** stage. The primary axis represented one of eight defined temporal axes—Main, Intention, Opinion, Hypothetical, Generic, Negation, Static, and Recurrent—while the secondary axis focused on **Temporal Validity Distribution Plotting**, where skewed normal distributions were annotated with parameters for location (ξ), scale (ω), and skewness (α) along a logarithmic time scale.

To maintain interpretability, distributions were anchored at t=0t = 0t=0, multimedia data was excluded, and any samples without a clear main timeline or that violated project constraints were flagged for review. Each data point was annotated independently by two annotators to ensure reliability. Rigorous quality control was in place, including periodic reviews of 10% of the annotations, a cap of 70 samples per annotator per day to prevent fatigue, and automated flagging of mismatches such as segmentation inconsistencies or large target deviations. Agreement thresholds were set high—Inter-Class Correlation (ICC) > 0.90 for numerical annotations, Jaccard Index > 0.75 for segment-level annotations, and Pπ<0.15P_{\pi} < 0.15Pπ <0.15 for segmentation consistency checks during the warm-up phase.

Ambiguous samples were either flagged for special consideration or removed, and final resolution applied a **union-based approach**—keeping all plausible interpretations to reflect the complexity of human temporal cognition. For the temporal validity parameters (ξ, ω, α), annotators' values were averaged to create probabilistic supervision targets rather than fixed labels, providing richer training data for the model.

Projects:

1. **Finfluenzz** is a retro-gaming-inspired fintech platform designed to make personal finance engaging and accessible for Gen Z users. The application gamifies money management, blending **AI-driven financial insights** with a playful arcade-style interface.

   The **mission** is to democratize financial literacy by integrating smart tools—such as an **AI Finance Advisor** powered by Groq AI and real-time stock/crypto data from Alpha Vantage and CoinGecko—into an interactive and visually appealing experience. Users can analyze stocks with candlestick patterns, track cryptocurrency prices, and get personalized investment advice enriched with technical indicators like RSI, SMA, and MACD.

   The **Investment Zone**, backed by Gemini AI, offers portfolio breakdowns, risk assessments, diversification checks, and actionable improvement recommendations. Meanwhile, the **News Terminal** aggregates stock- and crypto-specific news from Finnhub in real time. A **Budget Tracker** helps with expense management, categorization, and AI-generated summaries of spending habits.

   Finfluenzz also introduces **gamified challenges**—personalized goals, achievement badges, XP systems, and leveling mechanics—to encourage better money habits. The **retro pixel-art UI** delivers a nostalgic yet modern experience, with responsive layouts, mobile optimization, and smooth micro-interactions.

   From a technical standpoint, the **frontend** is built with **React 18, TypeScript, Vite, Tailwind CSS, Lightweight Charts, Axios, and React Router**. The **backend** uses **Cloudflare Workers with the Hono framework**, **Prisma ORM**, **Neon PostgreSQL**, and **JWT authentication** with bcrypt password hashing. The platform integrates multiple APIs (Groq, Gemini, Alpha Vantage, Finnhub, CoinGecko) for AI insights and financial data.

   Security is emphasized through encrypted authentication, role-based access control, input sanitization, and secure environment variable handling. Deployment is split between **Vercel** (frontend) and **Cloudflare Workers** (backend) for global scalability.

   Ultimately, Finfluenzz is **more than just a finance tracker**—it's an educational, AIenhanced, gamified ecosystem aimed at making smart money management second nature for Gen Z. Live link: https://finfluenzz.vercel.app/

2. The Tool Router – Multilingual MCP Tool Routing System is a fully functional solution designed to accurately route user inputs in Hindi, English, or Hinglish to the most appropriate AI tool in a Multi-Component Platform environment, addressing the Puch AI interview challenge. It features a dual-routing architecture, where the primary method is a fine-tuned intent classification model based on paraphrasemultilingual-MiniLM-L12-v2 trained on over 540 examples across five intents, achieving a training loss of 0.10, over 95% validation accuracy, and 86.7% real-world routing accuracy. The secondary method is a semantic embedding similarity approach with pre-computed multilingual embeddings, reaching 73.3% accuracy and serving as a fallback when the intent model's confidence is low. The system supports eight tools in total, including five specialized ones like Leftover Chef for recipe suggestions, Nani Ki Kahaniyan for moral stories, Poem Generator, Vividh Bharti Jukebox for nostalgic

songs, and Food Locator, along with intelligent routing, diagnostic routing visualization, and a routing accuracy evaluator. It preserves cultural context, handles code-switching naturally, and uses confidence-based decision-making to avoid incorrect routing, asking for clarification when necessary. A comprehensive evaluation framework measures accuracy, precision, recall, and confusion matrices across languages, with real-world multilingual test cases. The project demonstrates not just tool routing capability but production-ready architecture with language-aware semantic understanding, cultural nuance preservation, and robust performance tracking, making it directly applicable to conversational AI in Indian household scenarios.

3. Speed-RAG is an optimized conversational Retrieval-Augmented Generation (RAG) system, built for my portfolio website, that delivers fast and accurate questionanswering over a private collection of documents through a FastAPI web application. It leverages binarized embeddings to accelerate vector searches and reduce memory usage, while a `ConversationalRetrievalChain` preserves conversational context for natural follow-up interactions. Powered by Groq's high-speed language model inference, the backend cleanly separates the RAG pipeline logic (`rag.py`) from the API server (`app.py`) for modularity and easy maintenance. On its first run, the system automatically processes and indexes PDF documents from `docs_dir/` into a Chroma vector database (`chroma_db/`) by splitting them into chunks, generating binary embeddings via a Hugging Face model, and storing them locally. During runtime, user queries are reformulated into standalone questions using chat history, the most relevant document chunks are retrieved from the database, and the Groq LLM generates a context-aware, accurate response. The application is deployed locally via Uvicorn, exposes a `/api/chat` endpoint for JSON-based requests, and is designed with a clean, modular architecture to serve as both a high-performance AI tool and a showcase project in my portfolio.

4. The **Vehicle Insurance Call AI Agent** is a bilingual (Hindi/English) voice-based AI solution built with LiveKit to handle vehicle insurance inquiries, qualify leads, and store customer information in a PostgreSQL database. It integrates a Neo4j-powered **Graph RAG system** that retrieves real-time car specifications and details, sourced from over 200 models scraped from CarDekho.com using the Firecrawl API. The system features seamless multilingual voice interaction, real-time conversations, appointment booking, intelligent data extraction, and lead scoring. Technically, it leverages Sarvam STT/TTS models for speech processing, Google Gemini 2.5 Flash for conversation generation, Silero VAD for voice activity detection, and Cloudflare Workers with Prisma ORM for backend deployment. The architecture connects LiveKit's voice interface to the AI agent, STT/TTS services, Neo4j for knowledge retrieval, and PostgreSQL for persistent storage, with scraped data processed into CSV files and ingested into the graph database. The data pipeline involves structured car data extraction, graph construction with manufacturer, body type, fuel type, and feature relationships, and real-time retrieval during calls. The conversation flow begins with language selection, followed by customer information gathering, instant car data lookup via Graph RAG, and lead qualification or appointment booking. The project includes visual graph representations of data relationships, supports multi-attribute car searches, and delivers instant, personalized

insurance recommendations. The tech stack spans Python, Node.js, TypeScript, Neo4j, PostgreSQL, Hono.js, and LangChain, with a structured codebase for agents, scraping, backend APIs, and data analysis notebooks. This project demonstrates advanced integration of conversational AI, real-time data retrieval, and voice technology for the insurance domain.

5. Snippify is a full-stack web application that enables developers to discover, create, and share reusable code snippets, blending a community-driven snippet repository with an Autonomous AI Agent capable of instantly generating, explaining, and refactoring code. Users can manage snippets through features like creating, updating, deleting, and copying, with options for public sharing or private storage for personal or team use. The platform also offers a curated component library section, allowing developers to explore and integrate pre-built, customizable UI components to enhance productivity and project quality. Built with React, TypeScript, and Tailwind CSS on the frontend, Hono and Cloudflare Workers on the backend, PostgreSQL for data storage, and powered by Groq Cloud LLMs for AI features, Snippify delivers a fast, scalable, and AI-enhanced development experience. It is accessible online at **https://snippify-zeta.vercel.app/**.

6. Project: Smart Property Finder, It is a **Model Context Protocol (MCP)** tool designed for the **Puch AI WhatsApp bot** to help users discover real-estate listings and receive intelligent property insights directly within chat. It leverages **Firecrawl** for web scraping to retrieve property data from platforms like **99acres** and **SquareYards**, and uses the **Groq LLM** for deep contextual analysis of both property details and location-specific information. Users can send queries via WhatsApp, which are processed through the MCP tool, scraped for relevant listings, analyzed for key insights, and returned as an easy-to-understand summary in the chat interface. The project supports scraping, personalized recommendations, and conversational delivery, making it a seamless realestate discovery experience.

7. **Kisaan Saathi** is an AI-powered farmer dashboard designed to empower Indian farmers with real-time data, intelligent decision-making tools, and easy access to agricultural resources. It integrates multiple AI and ML models to address key farming challenges, offering features such as AI-based crop disease detection using a custom-trained Xception model with multilingual support, hotspot mapping with DBSCAN clustering for government inspections, and a Random Forest-based water footprint calculator for precise irrigation guidance. The platform also includes a multiagent RAG-powered Farmer AI Assistant with ChromaDB vector storage for domainspecific advice, real-time market analysis with price trends and regional comparisons, a centralized hub for government schemes, accurate weather forecasting, a crop waste exchange to promote biofuel and biogas initiatives, and nearby equipment rental services to make farming tools more accessible. Built with a robust tech stack including FastAPI, TensorFlow, Scikit-learn, LangChain, and Groq LLM, alongside APIs for weather, market prices, and schemes, Kisaan Saathi delivers an interactive, data-driven experience. The platform's intuitive frontend is built with HTML5, TailwindCSS, JavaScript, and Chart.js, ensuring seamless visualization and accessibility. Designed as a comprehensive digital companion, Kisaan Saathi aims to boost productivity, improve decision-making, and strengthen the economic resilience of farmers across India.

8. The **Multi-threaded Web Server** project is a Python-based implementation of a simple HTTP server capable of handling multiple client requests simultaneously using multithreading. Built with the socket module for low-level network communication and

concurrent.futures.ThreadPoolExecutor for efficient thread pooling, the server demonstrates key concepts in scalable and responsive backend development. It listens for incoming HTTP requests, processes them concurrently via a thread pool, and responds with static HTML content while logging each request to the terminal. The

server automatically detects the host's local IP address, making it easy to access from any device on the same network. This project showcases essential backend development skills, including socket programming, the basics of the HTTP protocol, and resource management with thread pools, making it a practical example for learning how to serve multiple clients efficiently in real-world applications. This is an operating system project lakshya built for his College project.

9. Email campaign Performance Optimization: This project focuses on optimizing email marketing campaigns through data-driven insights, feature engineering, and machine learning modeling to improve engagement rates. A total of **100,000 emails** were sent, with **10,345 opened** and **2,119 clicked**, resulting in an **open rate of 10.35%**, a **click rate of 2.12%**, and a **click-through rate (CTR) of 20.48%** (clicks per open). Exploratory Data Analysis revealed that **short emails** outperformed long ones (**11.59% vs 9.12% open rate** and **2.39% vs 1.85% click rate**), while **personalized emails** significantly outperformed generic ones (**12.78% vs 7.93% open rate** and **2.73% vs 1.51% click rate**). Timing also played a key role, with the best engagement between **9 AM – 12 PM** and lowest between **8 PM – 11 PM**. **Midweek days (Tuesday–Thursday)** yielded the highest click rates, while Fridays and weekends lagged. Country-wise, **UK (12.02% open, 2.47% click)** and **US (11.90% open, 2.44% click)** users were the most engaged, while France and Spain showed weaker response. Purchase history strongly influenced engagement, with users having **10+ past purchases achieving click rates above**
4.66%, and highly loyal customers (14–22 purchases) reaching **9%+ CTR**, in some cases up to **100%**. The dataset included email metadata, open and click logs, and user attributes, enabling extensive feature engineering such as encoding content type, timeof-day categorization, weekend/work-hour flags, interaction features, country CTR averages, and purchase history bins. Multiple models were trained, including Logistic Regression, Decision Trees, Random Forests, Gradient Boosting, KNN, XGBoost, and SVM. Initial modeling before feature engineering saw **Logistic Regression** perform best but with a **-1.68% CTR improvement**, indicating it performed worse than the baseline. After feature engineering, **SVC** achieved **97.73% accuracy** but only **-0.39% CTR improvement**, suggesting possible overfitting. Grid Search tuning of the SVC yielded a **+2.89% CTR improvement**, demonstrating the potential for targeted campaign enhancements. Although tuning was limited due to computational constraints, results suggest that further optimization could yield even greater performance gains. Ultimately, a positive CTR lift confirms the model's ability to inform **targeted email campaigns** that can increase engagement beyond baseline performance. This is a data science project
10. **SignSync** is an AI-powered learning application designed to bridge the communication gap for the Deaf and Mute community by converting American Sign Language (ASL) gestures into human-readable text and vice versa. It uses advanced gesture detection techniques with OpenCV and MediaPipe, combined with custom-trained machine learning models, to identify and process hand gestures in real time. These gestures are then refined into natural, grammatically correct human language through Large Language Model (LLM) integration. The application also supports transforming ASLbased inputs into conversational text, with scalability to include more gestures and dialects over time. Built using Python, TensorFlow,

and FastAPI, SignSync offers both model training notebooks and a real-time API, along with an HTML-based interface for user interaction. The project's structured dataset focuses on ASL recognition, with potential expansion for more comprehensive coverage. Future plans include enhancing gesture recognition accuracy, improving processing speed, upgrading the UI, integrating text-to-sign video generation, and enabling direct communication between users through a dedicated website. Overall, SignSync strives to create an inclusive, accessible communication platform for the Deaf and Mute community. This project also uses open cv

11. **HealthCare-Hub** is a **Streamlit-based web application** that integrates multiple **machine learning models** to provide predictive solutions for various healthcare-related conditions. It is designed to support healthcare professionals and individuals by offering data-driven insights and assisting in early detection, diagnosis, and decision-making. The platform combines image-based detection models, health metric–based prediction tools, and even a mental health chatbot into a single interface.

The application includes the following features:

Bone Fracture Detection – Utilizes image-based ML to detect fractures in X-ray images.
Repository: Bone Fracture Detection
Model File: Download Bone Fracture Model
Brain Tumor Detection – Detects brain tumors from medical images.
Repository: Brain Tumor Detection
Model File: Download Brain Tumor Detection Model
Asthma Prediction – Predicts the probability of asthma based on patient data.
Repository: Asthma Prediction
Breast Cancer Prediction – Classifies tumors as benign or malignant using health data.
Repository: Breast Cancer Prediction
Calories Burnt Prediction – Estimates calories burnt from physical activity data.
Repository: Calories Burnt Prediction
Diabetes Prediction – Predicts diabetes likelihood based on health parameters. Repository: Diabetes Prediction

Heart Disease Prediction – Calculates the risk of heart disease from patient health metrics.

Repository: Heart Disease Prediction

Medical Insurance Prediction – Estimates medical insurance costs using personal and health data.

Repository: Medical Insurance Prediction
Mental Health Support Chatbot – An AI-driven chatbot offering guidance and support for mental health concerns.
Repository: Mental Health Support Chatbot

12. The **Car Price Prediction** project is a complete end-to-end machine learning pipeline designed to predict car prices. It covers all stages of the ML workflow, from **data preprocessing and feature engineering** to **model training, evaluation, and deployment**. The entire process is automated using **ZenML**, ensuring reproducibility, scalability, and easy tracking of experiments.
Key features include **data ingestion** (loading and structuring raw data), **data cleaning**

(handling missing values, outliers, and encoding features), **feature engineering** (creating and transforming features for better model performance), **data splitting** (train-test division), **model building** (training multiple algorithms like Linear Regression, Random Forest, and XGBoost), and **model evaluation** (comparing models using RMSE, MAE, and R^2 metrics). The pipeline also has components for **prediction service loading** and **real-time predictions**.

ZenML plays a crucial role by orchestrating the pipeline steps in the correct order and providing a dashboard to visualize runs, logs, and metrics. The setup involves installing dependencies, initializing ZenML, running the pipeline via Python scripts, and accessing the dashboard.

Before pipeline automation, **Exploratory Data Analysis (EDA)** is conducted, covering basic inspection, missing values, univariate, bivariate, and multivariate analyses. Technologies used include **Python**, **Pandas**, **NumPy**, **Scikit-Learn**, **Matplotlib**, **Seaborn**, and **ZenML**.

The repository is well-structured, with directories for **data**, **analysis scripts**, **source code for ML steps**, **ZenML step definitions**, and **pipeline scripts** for training and deployment. The organization promotes maintainability and scalability. In conclusion, this project is a strong demonstration of how modular design and MLOps tools like ZenML can make machine learning workflows reproducible, efficient, and production-ready. This is a data science project

13. For more projects please visit https://github.com/21lakshh