My Tech Stack 🪄

1. Development Environment & Coding Tools

- **Cursor IDE** – I use it for both work and personal projects to enhance productivity with AI-assisted workflows.
    - ✅ ***Strengths***: AI-assisted coding boosted productivity and reduced boilerplate.
    - ⚠️ ***Limitations***: Heavier compared to VS Code, slower startup.

- **VS Code** – My go-to for practicing competitive programming or working through Leetcode problems as I intentionally disable AI features to avoid distractions and focus on learning.
    - ✅ ***Strengths***: Lightweight, large extension ecosystem.
    - ⚠️ ***Limitations***: Lacks advanced AI features without extra plugins.

- **Kaggle Notebook** – Used for ML experiments in **20-20** (actor classifier) and **Pre-Owned** (car price prediction).
    - ✅ ***Strengths***: Free GPU, community datasets.
    - ⚠️ ***Limitations***: Limited long-term storage and collaboration.

- **Bash Shell** – For scripting and automation.
    - ✅ ***Strengths***: Powerful for scripting and automation.
    - ⚠️ ***Limitations***: Steeper learning curve for collaborators unfamiliar with Linux commands.

2. Web Frameworks

- **FastAPI** – My primary choice lately for building backend APIs as it offers out-of-the-box **Swagger UI** and jumpstarts **prototype development.**
    - I recently contributed features to **Promote** (a pre-event and live-promotion toolkit at promote.makemypass.com) using FastAPI.

    - ✅ ***Strengths***: Excellent support for async, intuitive auto-generated documentation.

    - ⚠️ ***Limitations***: FastAPI heavily leverages async/await for performance. While beneficial, this can introduce a steeper learning curve for developers unfamiliar with asynchronous programming concepts.

- **Flask** – Used in prototypes like **Pre-Owned**.
    - ✅ ***Strengths***: Lightweight and beginner-friendly.
    - ⚠️ ***Limitations***: Less efficient for async workloads compared to FastAPI.

- **Frappe** – Explored for ERP-style integrations at work for **The Apprentice Project** (TAP), building an internal CRM to streamline operations
    - ✅ ***Strengths***: Built-in admin and user management.
    - ⚠️ ***Limitation***: Steep learning curve, heavy setup.

- **Streamlit** – Used for rapidly turning ML pipelines, such as **20-20**, into shareable web demos.
    - ✅ ***Strengths***: Very quick to turn ML scripts into shareable apps
    - ⚠️ ***Limitations***: Limited customization for production-level apps.

3. AI/ML Frameworks

- **TensorFlow** – Applied in deep learning tasks, such as CNN-based actor classifiers in *20-20.*

    ○ ✅ ***Strengths***:Highly scalable, production-grade deep learning with hardware acceleration support and deployment tools like TensorFlow Serving.

    ○ ⚠️ ***Limitations***: Steeper learning curve and verbose low-level API compared to higher-level frameworks.

- **Scikit-learn** – Ideal for quick classical ML modeling and regression baselines in *Pre-Owned*.

    ○ ✅ ***Strengths***: Beginner-friendly with simple, consistent API, rich documentation, built-in model evaluation, and great for prototyping.

    ○ ⚠️ ***Limitations***: Not suitable for deep learning tasks, struggles with scalability to massive datasets or real-time streaming, and customization can be limited.

- **Pandas + NumPy** – For data analysis and preprocessing.
    - ✅ ***Strengths***: Powerful data manipulation, tabular operations, and labeled data handling.
    - ⚠️ ***Limitations***: Can be memory-inefficient (Pandas may use 5–10x memory of raw data), doesn't optimize query plans or support multi-core parallelism by default.

4. Agentic AI Tools

- **LangChain** – To build AI agents and RAG systems.
    - ✅ ***Strengths:*** Makes it easier to chain together steps for AI workflows.
    - ⚠️ ***Limitations:*** Changes often, so updates can break older code.
- **ChromaDB, PineconeDB** – For vector storage and retrieval.
    - ✅ ***Strengths:*** ChromaDB is lightweight and simple, PineconeDB is scalable and cloud-based.
    - ⚠️ ***Limitations:*** ChromaDB struggles with very large data, while Pinecone can be costly and locks you into their service.

5. LLM Platforms

- **OpenAI (GPT models)** – For text generation, coding help, and chatbots.
    - ✅ ***Strengths***: Strong performance in text generation and reasoning.
    - ⚠️ ***Limitations***: API costs add up with heavy usage.
- **Hugging Face** – For hosting and experimenting with open source models.
    - ✅ ***Strengths***: Large open-source model hub.
    - ⚠️ ***Limitations***: Some models underperform compared to OpenAI.

6. Cloud Platforms

- **AWS** – For deployment and storage (Eg: EC2 instance for hosting a Frappe application at **TAP**)

- **Vercel** – Used for deploying **SQL Ease frontend**.
  - ✅ ***Strengths***: One-click deployments, CI/CD out of the box.

  - ⚠️ ***Limitations***: Less backend control.

7. Databases

- **PostgreSQL** – Used in the **teacher registration project** at TAP.

  - ✅ ***Strengths***: Strong relational integrity and advanced queries.

  - ⚠️ ***Limitation***: More setup effort than MySQL.

- **MySQL** – Used in building **Aaronchettan** bot for Mulearn community

  - ✅ ***Strengths***: Simpler to set up.

  - ⚠️ ***Limitations***: Lacks advanced modern features.

8. AI Tools I Frequently Use

- **Jupyter Notebook** – For interactive testing, data visualizations, and rapid experimentation.

- **ChatGPT & Claude** – To assist with research, ideation, and debugging

- **Cursor IDE AI Features** – For coding assistance.

9. Additional Web Development Technologies

- **HTML, CSS, JavaScript** – Core building blocks for web development.

  - ✅ ***Strengths***: They are used everywhere, supported by all browsers, and are the foundation for any web page.

  - ⚠️ ***Limitations***: Writing them can get repetitive, and you need to learn modern techniques like making websites mobile-friendly and easy to use.

- **ReactJS** – JavaScript library for building UI components.

  - ✅ ***Strengths***: Makes websites faster by updating only what changes, lets you reuse components, has a huge community, and works well for search engines.

  - ⚠️ ***Limitations***: JSX adds complexity to the learning curve; documentation and ecosystem evolve fast, requiring constant updates; being only a view layer means you must integrate other libraries for full app functionality.