Detailed Tech Stack Overview

Development Environment & Coding Tools

- **Visual Studio Code**
  My most frequently used editor for web, Python, and even small AI/ML projects. I rely heavily on extensions like Prettier (for formatting), GitLens (for version control insights), Python (for debugging), and Live Server (for frontend preview).
  Experience: Advanced — I use it daily for both academic and professional work.
  Advantages: Lightweight, customizable, integrates well with Git, supports a wide range of languages.
  Limitations: Can become resource-intensive with too many plugins, and sometimes lags with very large projects.

- **PyCharm**
  My go-to IDE for larger Python/Django applications where advanced debugging and environment management are critical. I use its database integration and Django templates often in backend projects.
  Experience: Intermediate.
  Advantages: Superb Python support, intelligent refactoring, excellent for large codebases.
  Limitations: Heavy on memory usage, slower startup compared to VS Code.

- **Kiro**
  A distraction-free, minimal editor that I experiment with during focus sessions. I mostly use it for writing scripts or when I want a clean workspace with fewer distractions.
  Advantages: Simple, lightweight, fast.
  Limitations: Limited integrations and plugins compared to mainstream IDEs.

- **Cursor**
  An AI-powered IDE I've explored for integrating agentic AI into coding. It helps in generating boilerplate code and explaining code snippets while I prototype.
  Advantages: Strong AI integration, speeds up prototyping.
  Limitations: Relatively new, ecosystem still developing.

- **Git & GitHub**
  Central to all my project workflows. I use Git for version control (branching, merging, rebasing) and GitHub for collaboration, pull requests, issue tracking, and CI/CD pipelines.
  Experience: Strong — used across almost all projects.
  Advantages: Industry standard, powerful collaboration features.

Limitations: Learning curve for advanced Git commands (rebasing, resolving conflicts).

- **Postman**
  Essential for API testing and automation. I build collections, define environments (dev/prod), and sometimes integrate Postman scripts into CI/CD for regression testing.
  Advantages: Feature-rich, powerful automation.
  Limitations: Heavy for quick, one-off requests.

- **Hoppscotch API**
  A browser-based API testing tool I use when I want lightweight, instant checks. Ideal for fast debugging during development.
  Advantages: Open-source, fast, requires no installation.
  Limitations: Limited compared to Postman (no complex automation or scripting).

Web Frameworks

- **Django**
  My main backend framework for building scalable and secure web apps. I leverage its ORM, authentication, and admin interface heavily in academic portals and personal projects.
  Experience: Proficient.
  Advantages: Batteries-included (ORM, forms, auth, admin), rapid development.
  Limitations: Less flexible for microservices, monolithic in nature.

- **React.js**
  Used extensively for building responsive and interactive frontends — dashboards, portals, and course apps. I frequently integrate it with APIs (Django/Flask backends).
  Experience: Intermediate.
  Advantages: Reusable components, strong community, great ecosystem.
  Limitations: Steep learning curve (hooks, state management with Redux/Context).

AI/ML Frameworks

- **TensorFlow**
  I use it mainly for structured deep learning tasks like image classification or NLP with pre-built models.
  Experience: Beginner to Intermediate.
  Advantages: Highly optimized, production-ready, TPU support.
  Limitations: Syntax can be verbose, harder to debug than PyTorch.

- **PyTorch**
  My preferred deep learning library for research and prototyping. I use it for computer vision (CNNs) and NLP (transformers). Its dynamic graph feature helps me debug models easily.
  Experience: Intermediate.
  Advantages: Flexible, Pythonic, large research community.
  Limitations: Deployment not as seamless as TensorFlow in some cases.

- **scikit-learn**
  My go-to for traditional ML tasks like regression, classification, clustering, and preprocessing (train/test split, pipelines).
  Experience: Beginner to Intermediate.
  Advantages: Easy to use, integrates with NumPy/Pandas well.
  Limitations: Not suitable for large deep learning models.

Agentic AI Tools

- **LangChain**
  I experiment with building conversational agents, RAG (retrieval-augmented generation) pipelines, and chaining multiple LLM calls.
  Experience: Beginner.
  Advantages: Modular, growing ecosystem, integrates well with vector databases.
  Limitations: Complex to design advanced workflows.

- **Auto-GPT**
  Explored for autonomous workflows like auto-document drafting and basic research.
  Advantages: Visionary approach to agentic AI.
  Limitations: Prone to unpredictability, high resource consumption.

LLM Platforms

- **OpenAI API**
  I integrate GPT models into chatbots, dashboards, and summarization features. Used frequently for project demos and personal productivity.
  Experience: Strong.
  Advantages: Cutting-edge performance.
  Limitations: Expensive at scale, rate-limited.

- **Hugging Face Transformers**
  I use pretrained models and occasionally fine-tune for NLP tasks like text classification.

Advantages: Large community, wide model hub, open-source.
Limitations: Requires GPU/TPU resources for training.

AI Tools You Frequently Use

- **ChatGPT**: Daily assistant for coding, brainstorming, and drafting. Helps me iterate faster.
  Limitation: May produce hallucinations.
- **Copilot**: Integrated into my IDE, suggests code snippets and autocompletes functions.
  Limitation: Suggestions sometimes irrelevant.
- **Google Colab**: Cloud notebooks I use for ML experimentation with free GPU.
  Limitation: Session timeout, limited compute.
- **Gemini**: Explored for multimodal AI (text + image).
  Limitation: Restricted access.
- **Deepseek**: Used in research assistance and note-taking.
  Limitation: Niche ecosystem.
- **Mistral.ai**: Experimenting with lightweight open-source LLMs.
  Limitation: Smaller community support.

Cloud Platforms

- **Google Cloud Platform (GCP)**
  I mainly use it for deploying apps, storing datasets, and experimenting with AI APIs (Vision API, AutoML).
  Experience: Intermediate.
  Advantages: Tight AI/ML integration, reliable infrastructure.
  Limitations: Pricing can be complex and costly for students.

Databases

- **MySQL**
  Used in Django and Flask projects for relational data management. Strong understanding of schema design, joins, and indexing.
  Experience: Strong.
  Advantages: Widely supported, reliable.
  Limitations: Not as strong in handling large analytical workloads.

- **PostgreSQL**
  Preferred when projects require complex queries, JSONB support, or scalability. I use it for modern web apps with Django/Flask integration.
  Experience: Intermediate.
  Advantages: Feature-rich, excellent indexing and performance.
  Limitations: Slightly steeper learning curve.