
DEPARTMENT OF
INFORMATION
TECHNOLOGY
ZenITh
ANNUAL MAGAZINE 2026
KALYANI GOVERNMENT ENGINEERING COLLEGE

PRINCIPAL

Dr. Sourabh Kumar Das

Kalyani Government
Engineering College

Head of the Department

Dr. Partha Sarathi Banerjee

Kalyani Government
Engineering College

Members of the Department of Information Technology

FACULTY MEMBERS

PROF. (DR.) MALABIKA SENGUPTA
DR. SATYENDRANATH MANDAL
DR. PARTHA SARATHI BANERJEE
MD. IQBAL QURAISHI
MR. DEBABRATA CHOWDHURY
DR. MANISHA BARMAN
MR. SUDIPTO BASAK
MRS. ANUSUA MAZUMDAR
MRS. SUPARNA SAHA
MR. KAUSHIK MUKHERJEE

TECHNICAL ASSISTANT

MR. SANJIB INDRA

SUPPORT STAFF

MR. KASHI KANTA MANDAL

PRINCIPAL'S MESSAGE

It is a pleasure to extend my warm greetings on the inaugural edition of zenITh, the magazine of the Department of Information Technology, Kalyani Government Engineering College.

In a rapidly evolving digital world, IT shapes how we think, innovate, and progress. It is inspiring to see the department fostering creativity, excellence, and global engagement—reflected in strong placements, research contributions, and active participation in platforms like IEEE Systems, Man and Cybernetics Society.

zenITh stands as a vibrant expression of this spirit—a space where ideas, achievements, and voices come alive beyond classrooms.

I commend the faculty, students, and editorial team for this remarkable initiative. I am confident zenITh will grow into a lasting tradition, inspiring innovation and impact.

My best wishes for its continued success.

Dr. Sourabh Kumar Das

Principal

Kalyani Government Engineering College

MESSAGE FROM THE HEAD OF THE DEPARTMENT

It is with great pride that I present the inaugural edition of zenITh, the voice of the Department of Information Technology at
Kalyani Government Engineering College.
In a world driven by rapid technological change, our mission is to ignite curiosity, push boundaries, and turn ideas into impact. Our department thrives on innovation, collaboration, and the relentless energy of students who dare to think differently, work beyond convention and challenge the milestones.
zenITh is not just a magazine, it is a platform created, owned and managed by the students. It is a space where code meets creativity, research meets imagination, student voices shape the future and achievers motivates the learners and thus completes the cycle.
I congratulate every contributor and the editorial team from the bottom of my heart; This is your achievement, your creation. Let this be the beginning of new ideas, fearless innovation, and a legacy that inspires.
The future is yours, build it.

Dr. Partha Sarathi Banerjee
H.O.D. of Information Technology
Kalyani Government Engineering College

Committee Members

4TH YEARS

PUSPENDU BAR
SAIKAT ROY
AKASH BHAKAT

3RD YEARS

ARANYA MAL
MIR MD SHAIKH
KUNAL GIRI
SOUMYAPRIYA GOSWAMI
SAMBHUNATH DAS
SAYAN PAUL

2ND YEARS

TUHIN KARMAKAR
CHAYAN MANDAL
ADITYA THAKUR
DHANANJOY SARDAR
IPSHITA KARAR

INDEX

TECHNICAL ARTICLE

GENERIC ARTICLE

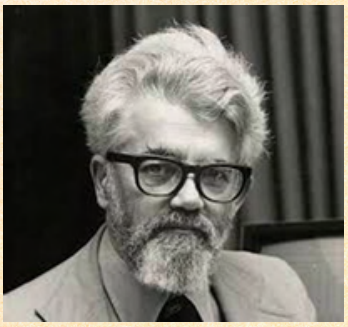
POETRY

SHORT STORIES

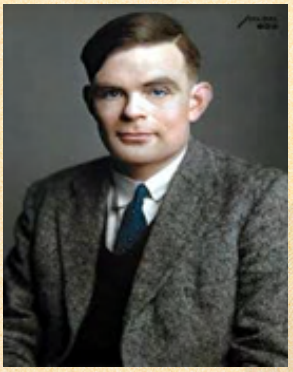
Technical Article

MINDS BEHIND MACHINES: THE PIONEERS OF ARTIFICIAL INTELLIGENCE

Artificial Intelligence, once ridiculed as day-dream, is now the heart of modern innovation and technological progress worldwide. From voice assistants and digital tutors to self-driving cars and smart healthcare, AI quietly empowers our everyday lives. Yet, behind this intelligent revolution lies a lineage of remarkable thinkers who dared to imagine machines with intelligence. Non-living things were made to learn, reason, and evolve with environment. These daring thinkers are the minds behind the machines, the pioneers who turned science fiction into scientific fact. Day dreaming is not bad if it drives one to keep eyes open.



The story begins with John McCarthy, the man the world refers to as the Father of Artificial Intelligence. In 1956, McCarthy brought together brilliant minds at Dartmouth College for a summer workshop named as the Dartmouth Conference. This conference coined the term Artificial Intelligence. McCarthy put forth his audacious vision: 'human reasoning could be replicated in silicon'. He also created LISP, one of the earliest programming languages for AIin 1958. He developed Time-sharing systems, precursors to our modern cloud computing platforms.

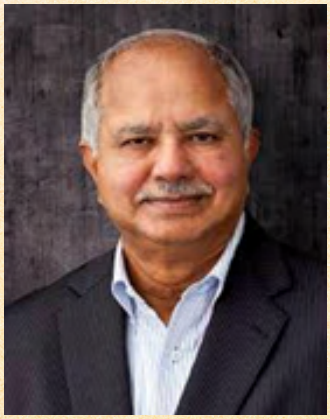


But the roots of AI run even deeper, to Alan Turing, the British mathematician who first posed the question, "Can machines think?" His famous Turing Test, proposed in 1950, set the benchmark for what we now call "machine intelligence." Turing's wartime code-breaking work not only saved countless lives but also gave rise to the computational logic that made AI possible. In many ways, he was the philosopher who taught machines the meaning of "thought."

The next great leap came from Marvin Minsky, co-founder of the MIT Artificial Intelligence Laboratory. A visionary scientist, Minsky believed that the human mind could be modeled through machines. His research on neural networks, cognitive simulation, and robotics gave AI its first mechanical voice.

As decades passed, AI found new champions in Geoffrey Hinton, Yoshua Bengio, and Yann LeCun — the trio often dubbed the "Godfathers of Deep Learning." Hinton's work on neural networks and his 2012 breakthrough model, AlexNet, sparked the modern AI boom. Bengio expanded the field with ethical, interpretable AI systems, while LeCun's invention of Convolutional Neural Networks (CNNs) made computer vision — and facial recognition — possible. Their contributions earned them the 2018 Turing Award, the highest honor in computer science.

Artificial Intelligence in India flourished to its current state, thanks to Prof. H. N. Mahabala and Dr. Raj Reddy. Prof. Mahabala, the Father of AI Education in India, introduced AI courses, labs, and mentorship in premier institutes, laying the academic foundation for generations of engineers.



Dr. Raj Reddy, a Turing Award winner, brought global recognition with pioneering research in speech recognition, robotics, and intelligent systems, influencing technologies like Siri and Alexa. While Mahabala made India ready for AI, Reddy made the world
ready for Indian AI minds — together shaping India's dual legacy of education and innovation.

Together, these pioneers turned imagination into innovation. They built systems that see, speak, and understand, and, more importantly, forced humanity to rethink what intelligence truly means.

As AI continues to evolve, it carries with it the spirit of these dreamers, a reminder that every algorithm begins with a question, and every breakthrough begins with curiosity.

QUICK FACTS: THE JOURNEY OF AI

1950 — Alan Turing proposes the Turing Test, asking: "Can machines think?"

1956 — John McCarthy coins the term "Artificial Intelligence" at the Dartmouth Conference — the official birth of AI.

1958 — McCarthy develops LISP, one of the first programming languages for AI research.

1970s — Expert Systems like MYCIN and DENDRAL emerge — early programs that diagnose diseases and analyze chemistry data.

1986 — Geoffrey Hinton popularizes backpropagation, reviving neural network research.

1997 — IBM's Deep Blue defeats world chess champion Garry Kasparov — a historic AI milestone.

2012 — AlexNet, a deep learning model by Hinton's team, wins the ImageNet challenge, igniting the modern AI renaissance.

2022 — Generative AI tools like ChatGPT, Gemini, DALL·E, Grammarly redefine creativity, communication, and computation.

Let's know:

• The word "robot" comes from the Czech word robota, meaning "forced labor."

• The first chatbot, ELIZA, was created in 1966 at MIT and could mimic a human therapist!

• AI now contributes to global sustainability efforts through energy optimization, medical imaging, and climate modeling.

~Dr. Partha Sarathi Banerjee
HOD of Information Technology Department

প্রাচীন ভারত ও AI

Artificial Intelligence, যুব সহজ ভাবে বলতে গেলে কৃত্রিম বুদ্ধিমত্তা হলো সেই বিদ্যা বিভিন্ন বৈজ্ঞানিক পদ্ধতিতে একটি বা একাধিক যন্ত্রের মাধ্যমে তথ্য বা **information** গ্রহণ, শোষণ, আত্তীকরণের মাধ্যমে যন্ত্রটিকে মানুষের মতো চিন্তা করতে শেখায়। **Artificial Intelligence** এর দৌলতে যন্ত্রটি একপ্রকার সর্বশক্তিমান মানুষে পরিণত হয়েছে যা বিভিন্না সমস্যার সমাধান ও বিশ্লেষণ করতে সাহায্য করে ফলে তা আধুনিক প্রযুক্তি বিশ্বে উন্নতির অন্যতম হাতিয়ার হিসাবে পরিগণিত।

Globalization এর যুগে এর প্রভাব সর্বত্র। গ্রাম থেকে শহর, পাঠশালা থেকে বিশ্ববিদ্যালয় ক্লাসরুম যেকে ক্যান্টিনরুম, ডিগ্রি অর্জন থেকে চাকুরী, চায়ের দোকান থেকে রেস্টুরেন্ট, বন্ধু বাছাই থেকে জীবনসঙ্গী বাছাই, রাজনীতি থেকে সমাজনীতি, কূটনীতি অর্থনীতি সর্বত্র সর্বাদিক বিস্তারিত।

এখন প্রশ্ন এই আলাদিন এর প্রদীপটি এল কোথা থেকে? বর্তমান বৈজ্ঞানিক ধারণা অনুসারে **Artificial Intelligence** এর আবির্ভাব ১৯৫০ সালে। **Alan Turing** প্রদত্ত এর ধারণা কে প্রথম ভিত্তিপ্রস্থর হিসাবে গণ্য করা হয়। এরপর ১৯৭৪ থেকে ১৯৮০ পর্যন্ত গবেষণা বিভিন্ন্য কারণে থমকে যায় বলে একে **AI Winter** বলে। এরপর আবার ১৯৮৭ থেকে ১৯৯৩ পর্যন্তা একেই ঘটনার পুনরাবৃত্তি ঘটে বলে একে **Second AI Winter** বলে। তারপর থেকে অবশ্য বিভিন্ন ক্ষেত্রে বিভিন্ন ভাবে কৃত্রিম বুদ্ধিমত্তা র গবেষণা প্রাচুর্য্য লাভ করে আজ ও তা ক্রমবর্ধমান।

কিন্তু বর্তমান বিশ্বের কাছে একদম অবগত নয় যে এর প্রাথমিক ধারণা পাওয়া যায় আজ থেকে ৮০০০ ও ১১০০০ বছর আগে ভারতীয় মুনি ঋষিদের বিভিন্ন গল্প জনশ্রুতি, প্রবচন ও নীতিকথার মাধ্যমে। এর প্রাথমিক বীজ কিন্তু এই মুনি ঋষিদের মস্তিক প্রসূত যেখানে মানুষের চিন্তা ভাবনার সাথে যন্ত্রের প্রতিক্রিয়ার সম্বিলিত প্রয়াস খুব সুন্দর ভাবে বর্ণিত আছে। বর্তমান অত্যাধুনিক প্রযুক্তির সাফল্যের হাতিয়ার হিসাবে যে সমস্ত বৈজ্ঞানিক মডেল আজ আমাদের দৈনন্দিন জীবনযাত্রাকে উন্নত থেকে উন্নততর করে চলেছে তার প্রাথমিক ধারণা আমরা পাই রামায়ণ, মহাভারত ভাগবতগীতা বেদ ও উপনিষদ এর মধ্যে।

ভারতের প্রাচীনতম ভাষা সংস্কৃত ও বর্তমান বিজ্ঞানী দের চোখে **Natural Langua** হিসাবে পরিগণিত যা কেবল হিন্দু সভ্যতাকে সমাদৃত করেনা সেই সঙ্গে প্রাচীন ভারতীয় দর্শন কে ও বিশ্ব সমাজে বহিঃপ্রকাশ করে।
প্রাচ্য ও পাশ্চাত্যের এই সর্বকালীন লড়াই এ দার্শনিক বা দাবি করেন রোবট বা যন্ত্রের বিভিন্ন ব্যবহারের প্রথম পরিচয় পাওয়া যায় স্ত্রীক দার্শনিক হোমার এর লেখায় আজ থেকে ২৯০০ বছর আগে। তার বহু আগে লেখা রামায়ণ, মহাভারত এ এর বহুক্ষেত্রে বহুভাবে ব্যবহারের কথা উল্লেখ পাওয়া যায়।

এই প্রসঙ্গ ই **Adrienne Mayor** ২০১৮ সালে একটি সুন্দর কথা বলে, যার সারমর্ম হলো যদি কেউ গ্রীক, ইজিপ্টিয়ান, হিন্দু, ইসলামিক চৈনিক অথবা অনাকোনো প্রাচীন সংস্কৃতি র পৌরাণিক কাহিনী বিবেচনা করে তবে দেখা যায় সমস্ত অকল্পনীয় অবিশ্বাস্য অপরিসীম কার্য্য স্বয়ং ঈশ্বরের ঐশ্বরিক কল্পনা এবং দক্ষতা থাকলেই সম্পন্ন্য করা যেত। তবে হাজার বছর আগের পৌরাণিক কাহিনীর সঙ্গে বর্তমান বৈজ্ঞানিক চিন্তাধারার সরাসরি একটা সরল রেখার মতো যোগসূত্র টানা কার্যাত অসম্ভব। এখন আজকের প্রযুক্তির অন্তর্নিহিত ধারণার সাথে হিন্দু পবিত্র ধর্মগ্রন্থ রামায়ণ, মহাভারত এর উল্লেখিত প্রযুক্তি সম্পর্কিত ধারণার

যোগসূত্র খুঁজে বের করা যাক। রামায়ণ এ বর্ণিত পুষ্পক রথ এর কথা ধরা যাক। রাবন সীতা মা কে হরণ করবার জন্য কুবেরের কাছ থেকে নিয়ে ছিলেন এই পুস্পক রথ। যার কার্যকারিতা অত্যাধুনিক বিমানের থেকে ছিল অনেক উন্নত।
বর্তমান সামাজিক প্রেক্ষাপটে উন্নততর বিমান কেবলমাত্র ধনী ব্যাক্তিদের কাছেই থাকে কারণ এটি খুব ব্যায়বহুল, সেই দিক থেকে কুবেরের পুষ্পক রয তদানীন্তন কালের আর্থসামাজিক অবস্থাকেও তুলে ধরে।

'রক্তবীজ' যেখানে বলা হয়েছে প্রতিটি বিন্দু রক্ত থেকে এক প্রাপ্তবয়স্ক ক্লোন তৈরী হবে যা বর্তমান **Digital Cloning** এর অনেকটাই সমতুল্য।

মহাভারত এ একটি আকর্ষণীয় ঘটনা আছে যেখানে ভগবান শ্রীকৃষ্ণ সঞ্জয় কে দিযা দৃষ্টি দিয়েছিলেন ধৃতরাষ্ট্র কে কুরুক্ষেত্রের যুদ্ধের **live telecast** বর্ণনা করতে। বর্তমান বৈজ্ঞানিক প্রযুক্তির দৃষ্টি তে এটি ব্যাখ্যা করা যায় ভগবান শ্রীকৃষ্ণ **System Administrator** যে সঞ্জয় কে প্রয়োজনীয় **credential** দিয়েছিলো **login** করার জন্য। সঞ্জয় **Satelite, voice overvideo and wifeless communication** এব মাধ্যমে সরাসরি **Access** পেয়েছিলো।

প্রায় এক হাজার বছর আগে, ১০২৭ খ্রিস্টাব্দে, রাজস্থানের ধাতু গ্রামে একদল শ্রমিক জলের জন্য মাটি খুঁড়ছিল। তারা ৩০ ফুটেরও বেশি মাটি খোঁড়ার পর বায়ুরোধীভাবে সিল করা একটি ধাতব বাক্স আবিষ্কার করে। বাক্সের গায়ে একটি শব্দ যোদাই করা ছিল বারবারিক। তারা যখন যাতব বাক্স টি খুলল, তখন ভেতরে একটি খুলি আবিষ্কার করল, কিন্তু এটি হাড় দিয়ে তৈরি কোনও মানুষের খুলি ছিল না, বরং একটি চকচকে ধাতু দিয়ে তৈরি ছিল। আরো অদ্ভুত বিষয় হলো খুলির প্রতিটি চোখের কোটে দুটি করে চোখ ছিল। শ্রমিকরা খুলিটি রাজা রূপ সিং চৌহানের কাছে নিয়ে যায়। তিনি বারবারিকের খুলি সম্পর্কে আরও জানতে সমস্ত পণ্ডিতদের তার দরবারে ডেকে পাঠান আসলে 'বারবারিক হন্যে বর্তমান এ। রোবট প্রযুক্তি যা প্রাচীন ভারতের খাটু শ্যাম মন্দিরের রহস্য গল্পে উল্লেখ আছে।

'বারবারিক এর সমন্ধে আরো জানা যায় বারবারিক ছিলেন একজন কিংবদন্তি যোদ্ধা যিনি ৫০০০ বছর আগে ভগবান কৃষ্ণের রাজত্বকালে বেঁচে ছিলেন। অন্যদিকে, বারবারিক ছিলেন একেকরেই আলাদা। তিনি মানুষ ছিলেন না, তাঁর কাছে ঐশ্বরিক শক্তি ছিল। তিনি একই সাথে বেশ কয়েকটি উন্নত অস্ত্র ও অস্ত্র ব্যবস্থা পরিচালনা করতে এবং বিদ্যুৎ গতিতে গাণিতিক সমস্যা সমাধান করতে সক্ষম ছিলেন, কিন্তু তাঁর কণ্ঠস্বর ছিল ধীর এবং তিনি মুখগুলি সনাক্ত করতে পারতেন না। তাঁর কাছে মানব বুদ্ধি ছিল না: বরং, তাঁর কৃত্রিম বুদ্ধিমত্তা ছিল। বারবারিকের শক্তি এবং বুদ্ধি সম্পর্কে শোনার পর, ভগবান কৃষ্ণ তাঁর সাথে দেখা করতে রাজি হন। মহাভারতের মহাযুদ্ধ শুরু হতে চলেছে, এবং ভগবান কৃষ্ণ জানতে চান যে বর্বরিক কোন পক্ষকে সমর্থন করবেন। ভগবান কৃষ্ণের অনুরোধে, বারবারিক উন্নত অস্ত্র দিয়ে তাঁর অবিশ্বাস্য গতি এবং নির্ভুলতা দেখিয়েছিলেন। ভগবান কৃষ্ণ অবাক হয়ে যান এবং জানতে পারেন যে আসন্ন মহাযুদ্ধে বারবারিকই হবেন প্রধান ব্যক্তিত্ব। অন্যদিকে, বারবারিকের বুদ্ধি মানুষের মতো নয়; সে রোবটের মতো চিন্তা করে এবং কাজ করে। ভগবান কৃষ্ণ যুদ্ধে তার অবস্থান সম্পর্কে জিজ্ঞাসা করেন। বারবারিক উত্তরে বলেন তিনি অন্য পক্ষের চেয়ে দুর্বল পক্ষকে সাহায্য করবেন। তিনি দাবি করেন যে ঘতটা সম্ভব মানুষকে বাঁচানো তার দায়িত্ব। এরপর ভগবান কৃষ্ণ বারবারিককে একটি সহজ প্রশ্ন করেন, যা তাকে সম্পূর্ণরূপে বিভ্রান্ত করে। ভগবান কৃষ্ণের মতে, যদি বারবারিক দুর্বল সেনাবাহিনীকে সমর্থন করে এবং শক্তিশালী সেনাবাহিনীকে হত্যা করতে শুরু করে, তাহলে এইভাবে শক্তিশালী সেনাবাহিনী অবশেষে দুর্বল সেনাবাহিনীতে পরিণত হবে এবং একসময় দুর্বল পক্ষটি আরও শক্তিশালী হয়ে উঠবে। সেই পরিস্থিতিতে বারবারিক কী করবে? তবে মানুষের কাছে এটি বৈধ প্রশ্ন হবে না কারণ আমরা মানুষ বৃহত্তর ধারণাগুলি বুঝতে পারি, অন্যদিকে রোবট এবং কৃত্রিম বুদ্ধিমান যন্ত্রগুলি ঘটনাগুলি পর্যবেক্ষণ করে এবং বৃহত্তর চিত্র বুঝতে অক্ষম। যাইহোক, বারবারিক পরে বলেন যে

যেহেতু অন্য সেনাবাহিনী দুর্বল হয়ে পড়েছে, তাই তিনি পক্ষ নেবেন এবং তাদের সাহায্য করবেন। বারবারিকের প্রতিক্রিয়া ভগবান কৃষ্ণকে অবাক করে দেয় কারণ এর অর্থ হল বারবারিক দুটি সেনাবাহিনীর মধ্যে পক্ষ পরিবর্তন করবে এবং তাদের সম্পূর্ণরূপে হত্যা করবে। কৃষ্ণ উপসংহারে বলেন, বর্বরিকের অপ্রাকৃতিক বুদ্ধিমত্তা মানুষের উপকারের পরিবর্তে ক্ষতিই করবে, যদিও তিনি অত্যাধুনিক অস্ত্র ব্যবহারে অসাধারণ নির্ভুলতার সাথে সক্ষম ছিলেন। ফলস্বরূপ, ভগবান কৃষ্ণ বারবারিকের শারীরিক

কার্যকলাপকে অক্ষম করে দেন, তাকে তার বাহু বা পা দিয়ে আক্রমণ বা অস্ত্র ব্যবহার করতে বাধা দেন। ভগবান কৃষ্ণ বারবারিক কে তার মাথা তার শরীর থেকে আলাদা করতে বলেন। সবচেয়ে আকর্ষণীয় অংশটি হল: বর্বরিক তার মাথাটি খুলে কৃষ্ণের হাতে তুলে দেন, কিন্তু তিনি কথা বলতে থাকেন। এবং স্পষ্টভাবে প্রমাণ করে যে তিনি মানুষ ছিলেন না, বরং কৃত্রিম বুদ্ধিমত্তা সম্পন্ন একজন রোবট ছিলেন।

বিশ্বের প্রাচীনতম মহাকাব্য 'রামায়ণ'-এ কুম্ভকর্ণকে রাবণের ছোট ভাই হিসেবে উল্লেখ করা হয়েছে। কিন্তু তিনি মোটেও জীবন্ত প্রাণী ছিলেন না, বরং একজন 'যন্ত্র' (যন্ত্র/রোবট) ছিলেন যার চেহারা ছিল বিশাল (একটি বিশাল রোবটের মতো)। আরও স্পষ্ট করে বলতে গেলে, কুম্ভকর্ণ ছিলেন একটি প্রাচীন রোবট, ভয়ঙ্কর, যন্ত্রের চেয়েও ভয়ঙ্কর এবং মানুষের চেয়েও কম 'মানবীয়'। রাবণ কুম্ভকর্ণকে খুব কম ব্যবহার করতেন, অর্থাৎ, শুধুমাত্র খুব কঠিন যুদ্ধ পরিস্থিতিতে, এবং তাৎক্ষণিকভাবে রাবণের পক্ষে মোড় ঘুরিয়ে দিত। সম্ভবত এর উচ্চ রক্ষণাবেক্ষণের কারণে, প্রায় ছয় মাস ধরে ঘুমিয়ে রাখা হত এবং শুধুমাত্র প্রয়োজনের সময় বা যখন এটিকে পুনরায় জ্বালানি দেওয়ার প্রয়োজন হয় তখনই জাগানো হত।

উপরিউক্ত, এই সব বহু ঘটনা ভারতবর্ষের প্রাচীন ধর্মগ্রন্থে খুঁজে পাওয়া যায়, যেখান থেকে আর্টিফিশিয়াল ইন্টেলিজেন্স এর অস্তিত্ব সুস্পষ্ট, এটা খুবই আশ্চর্যের বিষয় যে বর্তমান সায়েন্স এন্ড টেকনোলজি এর এক অতি আশ্চর্য আবিষ্কার প্রাচীন ভারতবর্ষের ইতিহাসেও বহু জায়গায় বিদ্যমান।

প্রাচীন ভারতীয় ধর্মগ্রন্থের অত্যাশ্চর্য লক্ষ্য করা যায় আধুনিক বিজ্ঞানের উপাদান ও আধুনিক বৈজ্ঞানিক সচেতনতার প্রতিরূপ এই উপাদান ও সচেতনতা গুলি বোঝার জন্যে তৎকালীন ভারতবর্ষের মানুষের মধ্যে যে উন্নততর চেতনা, উন্নততর চিন্তাভাবনা ও উন্নততর বৈজ্ঞানিক মানসিকতা ছিল তা অনস্বীকার্য। আধুনিক প্রযুক্তির প্রকৃত বীজ নিয়ে যতই পাশ্চাত্য দার্শনিকদের সঙ্গে মতবিরোধ থাকুক না কেন পরিশেষে একটা কথাই বলবো **"WHAT INDIA THINKS TODAY WORLD THINKS TOMORROW"**

-মনীষা বর্মন

Faculty member of Information Technology

3I/ATLAS, the New ‘Oumuamua

In October 2017, astronomers spotted something that changed the way we look at the sky forever. A strange, cigar-shaped visitor, later named 1I/‘Oumuamua, passing the Sun and vanished into the deep space. It was the first confirmed interstellar object — a messenger from another star system. Two years later came 2I/Borisov, an icy comet that looked more familiar, yet carried the same alien passport. Now, in 2025, the story continues. Meet 3I/ATLAS (Fig1), the third confirmed visitor from the stars. Discovered by the ATLAS (Asteroid Terrestrial-impact Last Alert System) survey in Chile

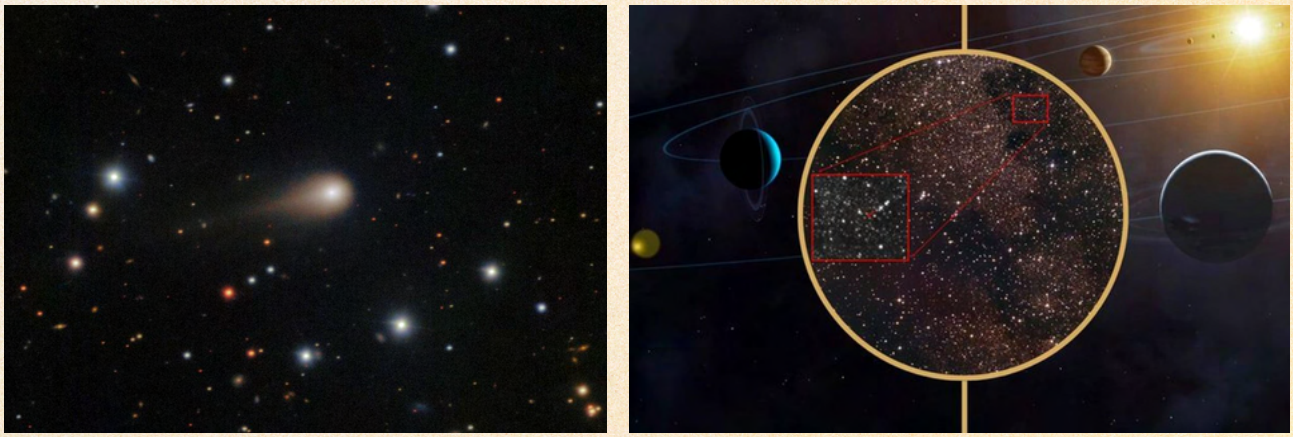


Fig1: Gemini North telescope shows 3I/ATLAS glowing like a cosmic rainbow among distant stars.

on July 1 2025, this icy traveller is bigger, brighter, and more mysterious than its predecessors. This comet was named after the ATLAS survey team that discovered it. The letter ‘I’ (interstellar) means it came from outside our solar system, and the number ‘3’ shows it’s the third interstellar object ever found.

What makes 3I/ATLAS so fascinating is that it doesn’t belong here. Its orbit is hyperbolic [Fig2] — meaning it isn’t bound to the Sun and will never return. Instead, it is on a one-way journey, cutting across our Solar System before disappearing back into interstellar space.

At its closest, it will pass about 1.8 AU (Astronomical Unit) from Earth on October 30, 2025 (that’s 270 million kilometre — perfectly safe, but close enough for telescopes to catch a good look). Then move toward Jupiter in March 2026.

Unlike ‘Oumuamua, which was oddly shaped and puzzlingly faint, 3I/ATLAS is massive. Early estimates suggest its core could be several kilometres wide, making it the largest interstellar object ever observed.It also appears to be active, releasing gas and dust like a comet. But here’s the twist — instead of water vapor (common in comets from our Solar System), it shows strong signs of carbon dioxide.

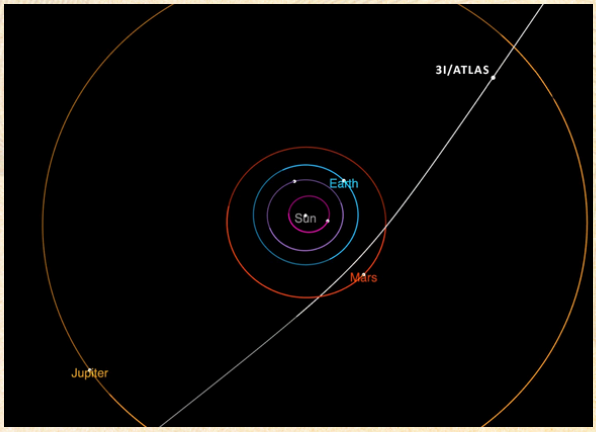


Fig2: A graphic illustration summarizing 3I/ATLAS’s orbit

This hints at very different conditions in the alien system where it was born, possibly billions of years ago (Fig3).

Fig3: A side-by-side comparison of the three interstellar visitors —‘Oumuamua, Borisov, and 3I/ATLAS—highlighting the immense size difference of this newest comet .

Why should we care about a lump of ice and rock just passing through? Because every interstellar object is a cosmic time capsule. They carry the raw materials from distant star systems, perhaps even older than our Sun. Studying them helps scientists answer big questions as follows:
How do planets form around other stars? What chemistry shapes distant worlds?

Could these objects even spread the building blocks of life across the galaxy?

Some scientists even speculate about wilder possibilities — that unusual objects like ‘Oumuamua and ATLAS could be artificial, fragments of alien technology. While most astronomers firmly see them as natural, the speculation only adds to the mystery.

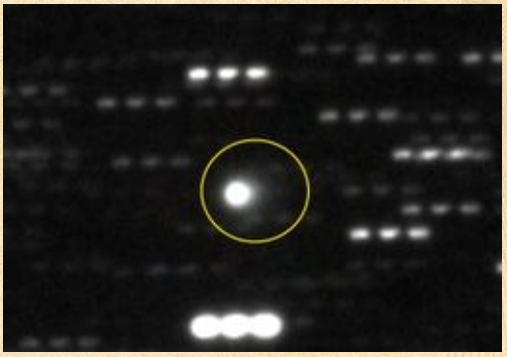


Fig 4: Vera C. Rubin Observatory image showcasing 3I/ATLAS—a bright dot among streaked stars—highlighting it as the largest interstellar object discovered so far.

For now, astronomers around the world are racing to collect as much data as possible [Fig4] before 3I/ATLAS fades away. Space telescopes like Hubble and JWST (James Web Space Telescope) are peering at it, while ground observatories follow its tail.

NASA'S NEW SPHEREX MISSION OBSERVES 3I/ATLAS OBJECT

NASA's new near-infrared survey telescope SPHEREx observed the interstellar visitor 3I/ATLAS and found a very large, CO_2-rich coma (the extended gaseous atmosphere of a comet) plus clear signatures of water-ice absorption [Fig 5]. (Unlike sharp optical telescope images, this is a spectrophotometric infrared map The extended "fuzziness" isn't a blurry camera — it's actually the huge gaseous envelope of the comet, which SPHEREx measured to be hundreds of thousands of kilometers across). SPHEREx (Spectro-Photometer for the History of the Universe, Epoch of Reionization, and Ices Explorer) is an all-sky near-infrared spectrophotometer that takes low-resolution spectra across every patch of sky at wavelengths 0.75–5.0 μm (102 spectral channels). That spectral range includes key gas- and ice-diagnostic features — e.g., CO_2 and H_2O bands — that are very hard or impossible to measure from the ground because Earth's atmosphere blocks or contaminates them..

WHAT SPHEREX ACTUALLY SAW IN 3I/ATLAS

Bright, extended CO_2 coma :SPHEREx resolved a large CO_2 gas coma around 3I/ATLAS that extends at least 348,000 km from the nucleus — a very large gaseous envelope for a cometary visitor.

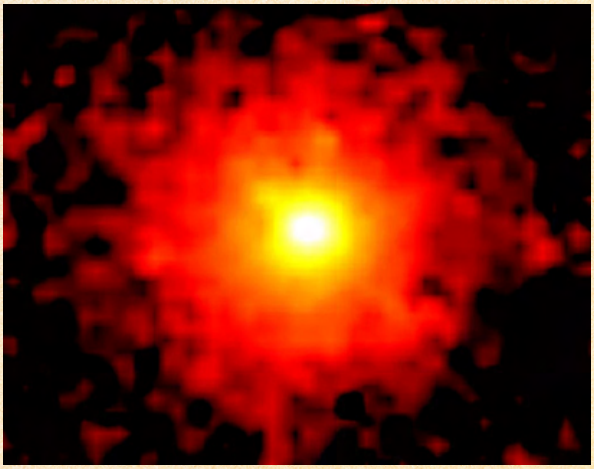


FIG 5 : FALSE-COLOR INFRARED IMAGE TAKEN BY SPHEREX OF 3I/ATLAS

Bright, extended CO_2 coma :SPHEREx resolved a large CO_2 gas coma around 3I/ATLAS that extends at least 348,000 km from the nucleus — a very large gaseous envelope for a cometary visitor.

High CO_2 production: Analysis (SPHEREx + supporting spectra) gives a CO_2 production rate on the order of ~9.4×10^{26} molecules/sec (Q_CO_2), while conservative upper limits for H_2O and CO production are substantially lower. That makes 3I/ATLAS unusually CO_2-dominated compared with many Solar-System comets.

Water-ice absorption in the spectrum: SPHEREx detected strong water-ice absorption features in the 0.75–5 μm data, indicating that ice is present in the nucleus or freshly exposed material in the coma.

Coma dominates the visible/IR light: SPHEREx's continuum measurements imply that most of the observed infrared flux comes from coma dust and gas rather than direct nucleus light; simple assumptions about the nucleus albedo would imply an unrealistically large nucleus if all flux were nucleus light, so the team concludes the coma dominates the signal.

HOW HUBBLE TELESCOPE OBSERVED 3I/ATLAS ?

Hubble imaged 3I/ATLAS on 21 July 2025 to get the highest angular-resolution view of this rare interstellar visitor and to measure its activity and constrain the nucleus size. Observations used Hubble's high-resolution imaging cameras (Wide Field Camera / WFC3 and related setups) with relatively short, repeated exposures. Some published figures use a broad filter (e.g., F350LP) and combinations/contours for analysis. This gives sharp coma structure while limiting smearing from motion. Hubble was commanded to track the moving comet, not the stars. Because the telescope followed the comet's sky motion, background stars appear as short streaks in the exposures while the comet remains (relatively) point like/compact in the centre. That technique maximizes detail on the comet's coma and dust. Ground telescopes see the fuzzy coma but can't separate the inner coma structure and set tight nucleus-size limits.

Hubble imaged 3I/ATLAS on 21 July 2025 to get the highest angular-resolution view of this rare interstellar visitor and to measure its activity and constrain the nucleus size. Observations used Hubble's high-resolution imaging cameras (Wide Field Camera / WFC3 and related setups) with relatively short, repeated exposures. Some published figures use a broad filter (e.g., F350LP) and combinations/contours for analysis.

This gives sharp coma structure while limiting smearing from motion. Hubble was commanded to track the moving comet, not the stars. Because the telescope followed the comet's sky motion, background stars appear as short streaks in the exposures while the comet remains (relatively) point like/compact in the centre. That technique maximizes detail on the comet's coma and dust. Ground telescopes see the fuzzy coma but can't separate the inner coma structure and set tight nucleus-size limits. Hubble's resolution lets astronomers model the inner surface-brightness profile and place upper limits on the nucleus radius.

WHAT HUBBLE TELESCOPE FOUND?

Cometary activity at large distance: 3I/ATLAS was already active at ~3.8 AU (pre-perihelion), showing dust emitted preferentially from the Sun-facing side and a weak dust tail pushed away by radiation pressure.
Teardrop / sunward dust cocoon [Fig 6]: Hubble shows a teardrop-shaped cocoon of dust off the nucleus (bulk of dust sunward of nucleus), not just a symmetric coma. That tells us where the sublimation and dust ejection are strongest.

Mass loss estimate: From the coma brightness and models, the dust mass-loss rate is estimated roughly 6 to 60 kg/s (range depends on mean particle size). That's a measurable loss even far from the Sun.

Size constraints: Fitting the inner coma brightness gives an upper limit on nucleus radius of order < ~2.8 km (assuming a dark albedo ~0.04); physical lower limits depend on what volatile drives activity (CO vs less volatile ices).

Fig 6 : Hubble closeup, comet centered, streaked stars

Hubble provided the sharpest ground-truth yet for 3I/ATLAS: it resolved an asymmetric, teardrop-shaped coma, allowed astronomers to estimate dust mass-loss (6–60 kg/s) and place an upper limit on nucleus size (< ~2.8 km), and confirmed that the object is active well before perihelion. Those results help us compare 3I/ATLAS to the two earlier interstellar visitors and learn about material from other star systems.

HOW JWST OBSERVED 3I/ATLAS ?

On 6 August 2025 the James Webb Space Telescope (JWST) observed the interstellar comet 3I/ATLAS with its Near-Infrared Spectrograph (NIR Spec) using an integral-field mode — i.e., JWST obtained a small data cube containing a spectrum at every pixel of the comet's coma. Those spectra let scientists separate sunlight reflected by dust from light emitted by gases and directly detect molecules such as CO_2, CO, H_2O (ice and vapour), and other volatile species. The most striking early result: 3I/ATLAS shows an unusually high CO_2-to-water ratio in its coma — one of the highest measured in any comet so far.

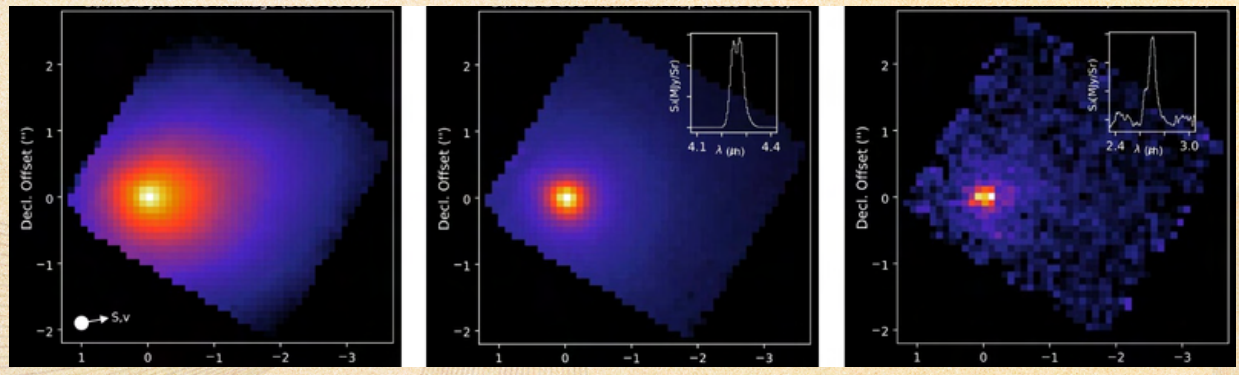


Fig 7: IR image captured by NASA's James Webb Space Telescope

JWST used NIRSpec in an integral field unit (IFU) configuration. An IFU produces a small 2D image where each pixel also contains a spectrum (wavelength information). This is perfect for comets because it gives spatially resolved composition maps: where a given molecule is bright vs where dust dominates [Fig 7].

Many volatiles (CO_2, CO, H_2O, organics) have strong vibrational/rotational bands in the near-infrared (1–5 μm). JWST's sensitivity in this range lets it detect faint gas emissions and weak ice absorption bands that are extremely hard to measure from the ground (Earth's atmosphere blocks or contaminates many of these features).
The IFU produces a 3-D data cube: two spatial axes + wavelength. Scientists extract spectra from regions (nucleus, inner coma, tail) and produce maps of emission strength for each molecule, plus measurements of dust continuum and thermal emission. That yields gas production rates, spatial distribution, and estimates of nucleus size or the extent of a molecular envelope (coma).

WHAT JWST FOUND ?

CO_2-rich coma: The JWST NIRSpec data show strong CO_2 emission compared to water vapour — a CO_2/H_2O ratio far higher than typical Solar System comets. That suggests formation or processing in a region with abundant CO_2 ice (for example near a CO_2 ice line) or a surface layer that suppresses water release. This is the headline result.

Multiple volatiles detected: Spectra show signatures consistent with CO, water (vapour/ice), and other carbon-bearing molecules (reports mention carbonyl sulfide and organics in follow-up analyses), indicating compositionally rich ices.

Large CO_2 coma size: Observations and other telescopes estimate a very large CO_2 cloud around the nucleus (estimates place the CO_2 coma extent on the order of hundreds of thousands of kilometers), meaning the gas production is dispersed over a very large region.

Implications: Because 3I/ATLAS is interstellar, its unusually CO_2-rich composition gives direct clues about ice chemistry in other protoplanetary disks and about environments where planetesimals formed outside our system (for example, colder or more radiation-processed environments).

FUTURE PROJECT ON INTERSTELLAR OBJECT

Future projects, like the Vera Rubin Observatory, may discover many more of these wandering visitors. Instead of once-in-a-decade surprises, we may soon find them every year — a reminder that our Solar System is not an isolated bubble, but part of a bustling, dynamic galaxy. One mission concept proposed by the Initiative for Interstellar Studies (i4is). The idea is: instead of just watching interstellar objects fly past and vanish forever, why not actually chase one down with a spacecraft? One of the most exciting proposals is Project Lyra, a mission concept designed to send a spacecraft after interstellar visitors like 'Oumuamua or 3I/ATLAS.

The goal would be to intercept one of these cosmic travellers, study it up close, and maybe even collect samples.When ‘Oumuamua zipped past the Sun in 2017, scientists were stunned — but also frustrated. By the time telescopes confirmed what it was, the object was already heading out of the Solar System at 95,000 km/h, far too fast for any existing spacecraft to catch. That’s when the Initiative for Interstellar Studies (i4is) launched the concept of Project Lyra.

The idea is daring: build a spacecraft capable of chasing down future interstellar visitors (or even ‘Oumuamua itself if launched soon enough). Solar Object maneuver – diving close to the Sun and firing powerful engines at perihelion to gain a massive velocity boost. Advanced propulsion – exploring futuristic options like laser sails, nuclear propulsion, or high-powered ion drives. The biggest challenge? Timing. Interstellar objects don’t give us much notice — ‘Oumuamua was detected only after it had passed Earth. That means missions must either:

Launch extremely fast after detection, or Be on standby, waiting in space to be redirected toward a target. Despite the hurdles, Project Lyra has gained attention because it represents a new frontier: interstellar archaeology. Studying one of these objects up close could tell us how other star systems form planets, what materials float between the stars, and maybe even whether the seeds of life can travel across galaxies. As the i4is team puts it, if ‘Oumuamua was the first messenger, then Project Lyra could be humanity’s first reply.

Since Project Lyra is a concept study, there is no official launch date yet. If humanity ever decides to chase down ‘Oumuamua itself, the window of opportunity would be narrow — researchers estimated that a spacecraft would need to launch by the 2030s to catch up with it before it drifts too far into deep space. Most studies suggest that with current or near-future technology, the early 2030s to mid-2040s would be a realistic timeframe for such a mission to actually launch. 3I/ATLAS is not just a comet. It is a messenger from beyond, a brief visitor carrying secrets from another corner of the Milky Way. Like ‘Oumuamua and Borisov before it, it reminds us that the universe is alive with motion, mystery, and possibility. As it sails away into the darkness, it leaves us with wonder — and the hope that one day, we may meet one of these travellers up close.

~Kaushik Mukherjee
Faculty member of Information Technology

Building Taan: A Native Spotify Client in Rust

1. WHY ANOTHER SPOTIFY CLIENT?

Picture this: You're working on your laptop, listening to music on Spotify, and suddenly everything slows down. You check Task Manager and gasp—Spotify is consuming 700MB of RAM. For a music player. Just to stream audio and show a UI. That's when I decided: enough is enough. I'm building my own.

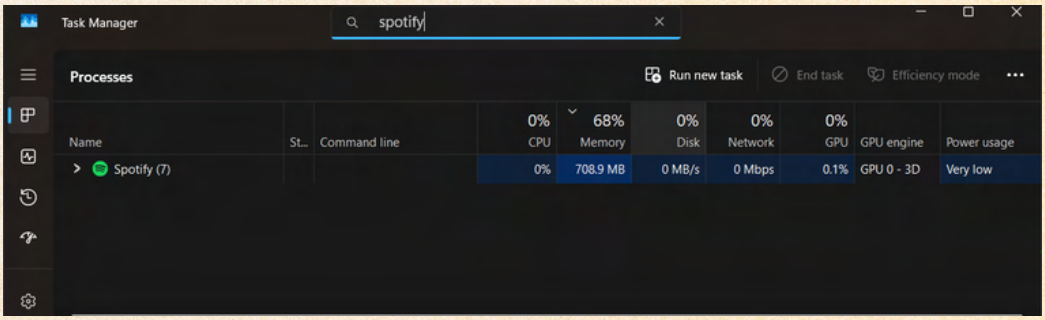


Figure 1: Spotify Memory Usage

Meet Taan—a native Spotify client built from the ground up in Rust, delivering the same functionality while using just 24MB of memory. That's a 96% reduction in memory consumption.

2. THE TECH STACK JOURNEY

Building Taan was a journey through Rust's GUI ecurystem, and boy, was it an adventure. Here's how it evolved

2.1. Act I: The Vello Experiment

I started with Xilem from the Linebender community, attracted by their cutting-edge Vello ZD renderer that leverages GPU compute pipelines. On paper, it was perfect-hardware-accelerated endering, modern architecture, pure Rust.

Reality check: Vello consumed 250MB of RAM Windows 11. Worse, there war virtually no documentation. Every feature required diving inte GitHub examples or internal source code. And without hot reloading, every Ul tweak meant waiting a minute or more for compilation. The dealbreaker? No live preview made designing the UI painful, Imagine adjusting button padding.

waiting 60 seconds to compile, realizing it's 2 pixels too much, rinse and repeat.

2.2. Act II: Enter Slint

Then I discovered Slint-a production-ready Gut framework used in embedded devices by actual customers. Created by ex-Qt engineers (and heavily inspired by QML), Slint offers

- **Custom DSL** for declarative UI design
- **Skia renderer** (the same one powering Chrome and Flutter)
- **LSP support** with live preview
- **Compile-time Ul generation** (no runtime overhead like QML)
- Memory uладе: **24MB**

Here's what Slint code looks like :

```
import { Colors } from
"components/common/colors.slint";

export component MusicPlayer {
    if root.width < root.height:
VerticalLayout {
        padding: 0px;
        spacing: 0px;

        // Album art section
        Rectangle {
            horizontal-stretch: 0;
            vertical-stretch: 1;
            AlbumArt { }
        }

        // Controls section
        PlayerControls { }
    }
    if root.height < root.width:
HorizontalLayout {
        // ... horizontal layout for
landscape
    }
}
```

Figure 2: Main player interface

Slint's declarative syntax makes building responsive UIs natural. The conditional layouts above automatically adapt between portrait and landscape orientations.

2.3. Honorable Mentions

Before settling on Slint, I also considered

- **Tauri:** Great DX, but relies on system webviews (WebView-GTK on Linux is let's just say "unre liable) Memory usage is better than Electron but still high.

- **Dioxias Blitz:** I love their React-like syntax and DX. Bot Blitz is immature missing CSS features, barebones events, and it uses Vello (back to the memory problem). I might contribute a Skia backend to their anyrender crate

3. THE SPOTIFY INTEGRATION DANCE

Taan uses a **dual-client architecture** to interface with Spotify:

3.1. librespot: The Audio Powerhouse

For playback, I use **librespot**-a reverse-engineered implementation of Spotify's internal APIs. This gives us direct streaming capabilities :

```
use std::sync::Arc;
use librespot_core::{Session, SessionConfig, SpotifyId,
cache::Cache};
use librespot_playback::{
    audio_backend,
    config::{AudioFormat, PlayerConfig},
    player::Player,
};
pub struct SpotifyService {
    session: Session,
    pub player: Arc<Player>,
    client: Arc<AuthCodeSpotify>,
}
impl SpotifyService {
    pub fn load_track(&self, id: String) -> Result<(),
Error> {
        let track_id = SpotifyId::from_uri(&id)?;
        self.player.load(track_id, false, 0);
        log::info!("Loaded track {}", id);
        Ok(())
    }
}
```

3.2. rspotify: The Metadata Maestro

For everything else—playlists, user library, track metadata—I use rspotify, a clean wrapper around Spotify's Web API:

```
pub async fn get_user_playlists(
    &self,
    limit: u32,
    offset: u32,
) -> Result<Vec<SimplifiedPlaylist>, Error> {
    loop {
        match self
            .client
            .current_user_playlists_manual(Some(limit),
Some(offset))
            .await
        {
            Ok(playlists) => break Ok(playlists.items),
            Err(e) => {
                if self.requires_refresh(e).await {
                    continue;
                }
                break Err(Error::unauthenticated("Failed to
refresh client"));
            }
        }
    }
}
```

Figure 3: Taan memory usage

4. The Architecture

Taan follows a clean separation of concerns:

```
src/
├── lib.rs          # Entry point, Tokio runtime setup
├── main.rs         # Main function wrapper
├── models/         # Data structures
├── services/       # Business logic (Spotify integration)
└── viewmodels/     # UI state management

ui/
├── main.slint      # Root UI component
├── player.slint    # Music player interface
└── components/
    ├── common/     # Reusable components (buttons, sliders)
    └── player/     # Player-specific components
```

4.1. The Async Bridge Pattern

One critical challenge: Slint runs on the main thread, but Spotify operations are async. The solution? A dedicated Tokio runtime on a separate thread:

```
fn setup(
    token: tokio_util::sync::CancellationToken,
    ui_weak: slint::Weak<MainWindow>,
) -> tokio::io::Result<std::thread::JoinHandle<()>> {
    env_logger::init();
    // Create dedicated async runtime
    let rt = tokio::runtime::Builder::new_multi_thread()
        .enable_all()
        .build()?;
    let rt_handle = rt.handle().clone();
    // Spawn runtime on separate thread
    let join = std::thread::spawn(move || {
        rt.block_on(token.cancelled());
        log::info!("Tokio Thread closed");
    });
    // Initialize Spotify service
    let spot = rt_handle.block_on(async {
        services::spotify::SpotifyService::default()
    });
    services::init(spot, rt_handle, ui_weak);

    Ok(join)
}
```

This pattern keeps the UI responsive while Spotify operations run asynchronously in the background.

4.2. Event-Driven Player Updates

The player listens to librespot events and updates the UI accordingly:

```
fn handle_player_event(event:
librespot_playback::player::PlayerEvent) {
    match event {
        PlayerEvent::Playing {
            track_id,
            position_ms,
            ..
        } => {
            log::info!("Resuming playback of {}", track_id);
            player::set_position(position_ms).unwrap();
```

```
            player::play().unwrap();
        }
        PlayerEvent::Paused {
            track_id,
            position_ms,
            ..
        } => {
            log::info!("Paused playback of {}", track_id);
            player::pause().unwrap();
            player::set_position(position_ms).unwrap();
        }
        PlayerEvent::TrackChanged { audio_item } => {
            log::info!("{:#?}", audio_item);
            if let Some(url) = audio_item.covers.first() {
                let url = url.url.clone();
                rt().spawn(async move {
                    match spotify().fetch_cover_art(url).await
{
                        Ok(img) =>
player::set_cover_art(img).unwrap(),
                        Err(e) => log::error!("Failed to
fetch: {}", e),
                    }
                });
            }
            player::set_track_details(audio_item).unwrap();
        }
        _ => {
            /* Handle other events */
        }
    }
}
```

Figure 4: Music playback in action

5. Design System: The Slint Way

Taan uses a centralized design system defined in colors.slint

```
export global Colors {
    // Primary background colors
    out property <brush> background-primary: rgba(74, 62, 76,
0.8);
    out property <brush> background-secondary: rgba(107, 114,
128, 0.5);
    // Text colors
    out property <brush> text-primary: white;
    out property <brush> text-secondary: #9ca3af;
    // Button states
    out property <brush> button-background-default: rgba(107,
114, 128, 0.5);
    out property <brush> button-background-hover: rgba(107,
114, 128, 0.6);
    out property <brush> button-background-pressed: rgba(107,
114, 128, 0.7);
}
export global Spacing {
 out property <length> xs: 4px;
 out property <length> sm: 8px;
 out property <length> md: 12px;
 out property <length> lg: 16px;
}
Every component imports and uses these values, ensuring visual
consistency:

import { Colors, Spacing } from
"components/common/colors.slint";

Rectangle {
 background: Colors.background-primary; // √
 background: #4a3e4c; // ✗ Never hardcode!
}
```

6. The Challenges (And Learning Opportunities)

6.1. Slint's Limitations

While Slint is great, it has some rough edges:

1. **Limited language features:** The Slint DSL is restrictive. Complex logic requires Rust callbacks.
2. **Immature ecosystem:** Features like multimedia playback and transform transitions need workarounds.
3. **Documentation gaps:** Some advanced patterns require reading Slint's source code.

Compare this to **QML** which uses full JavaScript, giving more flexibility. But QML requires a runtime, while Slint compiles to native code—a worthy tradeoff.

6.2. OAuth Authentication

Implementing Spotify's OAuth flow was interesting. I used librespot-oauth to handle the browserbased authentication:

```
pub async fn auth(&self) -> Result<(), Error> {
    let c = librespot_oauth::OAuthClientBuilder::new(
        SPOTIFY_CLIENT_ID,
        "http://127.0.0.1:8898/login",
        OAUTH_SCOPES.to_vec(),
    )
    .open_in_browser()
    .build()?
    .get_access_token_async()
    .await
    .map(|t| Credentials::with_access_token(t.access_token))?;

    self.connect(c).await?;
    Ok(())
}
```

The client opens a browser, user logs in, Spotify redirects to localhost:8898, and we capture the token. Simple and secure.

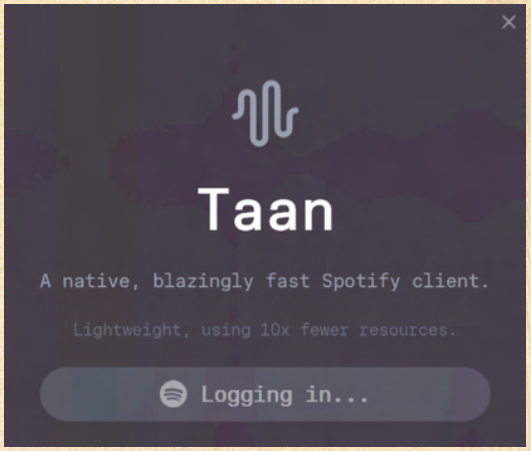


Figure 5: Login view

6.3. Token Management & Rate Limiting

Spotify's API has rate limits and token expiration. I implemented automatic retry logic:

```
async fn requires_refresh(&self, e: ClientError) -> bool {
    if let ClientError::Http(e) = e {
        if let HttpError::StatusCode(res) = *e {
            if res.status() == 401 {
                // Token expired, refresh it
                self.web_auth().await.unwrap_or_else(|e| {
                    log::error!("Failed to refresh: {}", e);
                });
                return true;
            }

            if res.status() == 429 {
                // Rate limited, wait and retry
                let wait = res
                    .headers()
                    .get("Retry-After")
                    .and_then(|v| v.to_str().ok()?.parse::
<u64>().ok());


tokio::time::sleep(Duration::from_secs(wait.unwrap_or_default(
))).await;
                return true;
            }
        }
    }
    false
}
```

This pattern wraps API calls in a loop that automatically handles auth and rate limit errors.

7. Performance Metrics

Let's talk about numbers and performance of the software:

Metric	Official Spotify	Taan
Idle Memory	700MB	24MB
Playing Music	750MB	30MB
CPU Usage (idle)	2-3%	< 1%
Startup Time	10-15s	1-2s
Binary Size	90MB	15MB

Table 1: Performance comparison between official Spotify and Taan

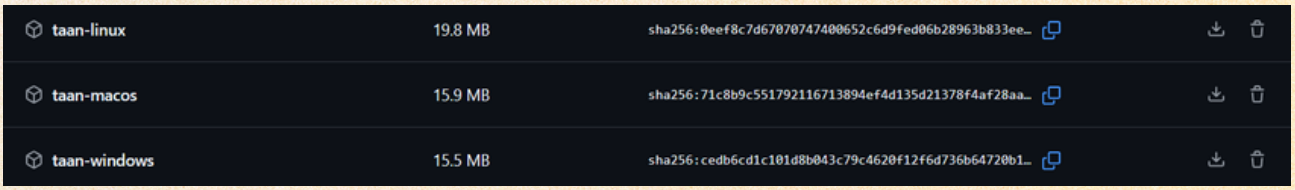


Figure 6: Binary size comparison

The memory savings are dramatic. This is the power of native code combined with efficient libraries.

8. Build System

Taan uses a simple build process:

```
// build.rs
fn main() {
    slint_build::compile("ui/main.slint")
        .expect("Slint build failed");
}
```

The Slint compiler runs at build time, generating Rust code from .slint files. This is imported via a macro:

```
// lib.rs
slint::include_modules!();
```

For development, I use Bacon for hot reloading:

```
bacon run            # Watch mode with auto-restart
bacon clippy-all     # Lint all code
```

9. WANT TO CONTRIBUTE?

Taan is open source and actively seeking contributors! Here are areas where you can help :

9.1. Immediate Needs

- **Playlist management UI :** Create, edit, and organize playlists
- **Search functionality :** Search tracks, albums, artists
- **Queue system :** View and manipulate the play queue
- **Keyboard shortcuts :** Media key support, global hotkeys

9.2. Advanced Features

- **Lyrics display :** Integrate lyrics APT
- **Equalizer :** Audio DSP filters
- **Podcast support :** Handle Spotify podcasts
- **Desktop integration :** System tray, notifications, Discord RPC

9.3. UI/UX Improvements

- **Theme system :** Custom themes and color schemes
- **Animations :** Smooth transitions and micro-interactions
- **Responsive design :** Better tablet and small screen support

9.4. Bug Fixes & Optimizations

- **Memory profiling :** Find and fix memory leaks
- **Caching strategies :** Improve offline capability
- **Cross-platform testing :** Test on Linux and macOS

10. THE TECH STACK SUMMARY

Here’s what powers Taan:

```
[dependencies]
# Spotify Integration
librespot-core = "0.7.1"          # Core Spotify client
librespot-playback = "0.7.1"      # Audio playback
rspotify = "0.15.1"               # Web API wrapper

# UI Framework
slint = {
    version = "1.13.1",           # GUI framework
    features = ["renderer-skia", "backend-winit"]
}

# Async Runtime
tokio = {
    version = "1",                # Async runtime
    features = ["full"]
}

# Utilities
reqwest = "0.12.23"               # HTTP client
image = "0.25.8"                  # Image processing
anyhow = "1"                      # Error handling
chrono = "0.4.42"                 # Date/time handling
```

11. LESSONS LEARNED

Building Taan taught me valuable lessons:

1. **Choose mature frameworks:** Bleeding-edge tech is exciting but documentation matters.
2. **Memory matters:** Native code + smart libraries = massive efficiency gains.
3. **Design systems early:** Centralized theming saves countless hours.
4. **Thread carefully:** Async/UI interaction requires thoughtful architecture.
5. **The 80/20 rule:** 20% of Spotify's features cover 80% of use cases.

12. THE BOTTOM LINE

Taan proves that native applications still have a place in our web-dominated world. With careful technology choices and efficient implementation, you can

build software that's faster, lighter, and more responsive than Electron-based alternatives.Is it perfect? No. Is it production-ready? Not yet. Is it a fun learning experience that demonstrates Rust's capabilities? **Absolutely.**

13. GET INVOLVED

Taan is just getting started. If you're interested in

- **Rust programming**
- **GUI development**
- **Audio/music technology**
- **Performance optimization**
- **Open source contribution**

This is the perfect project to dive inta. The codebase is well-documented, the architecture is clean, and there's plenty of low-hanging fruit for first-time contributors

14. QUICK START FOR CONTRIBUTORS

```
# Clone the repository
git clone https://github.com/karmakarmeghdip/taan.git
cd taan
# Install Rust (if not already installed)
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
# Build and run
cargo run
# Or use bacon for development
cargo install bacon
bacon run
```

Note: You'll need a Spotify Premium account to use Taan, as librespot requires Premium for playback.

"Why settle for 700MB when 24MB will do?"

~~Meghdip Karmakar
Batch(2022-2026)

GenAI in Tech: Revolutionizing Development and Empowering the Next Generation

Who Am I?

As a Software Development Engineer II (SDE-2) at Amazon, I work at the forefront of large-scale Generative AI initiatives that are transforming how we build and deploy software solutions. My role extends beyond traditional software engineering responsibilities—while I handle system design, documentation, and coding like any SDE-2, I also serve as a GenAI catalyst within our organization.

In this capacity, I'm responsible for:

- **Identifying and evaluating** cutting-edge GenAI tools and platforms that align with our technical requirements.
- **Configuring and customizing** AI solutions to meet specific use cases across our product ecosystem.
- **Leading hands-on knowledge sessions** for cross-functional teams, helping colleagues understand and adopt AI-powered development practices.
- **Bridging the gap** between emerging AI capabilities and practical implementation in enterprise-scale systems.

Through my work, I've gained firsthand experience with industry-leading GenAI solutions and witnessed their transformative impact on development workflows, team productivity, and product innovation. This article reflects the insights I've gathered from implementing these technologies in real-world scenarios, observing their effectiveness across diverse teams, and understanding both their immense potential and practical limitations.

My perspective combines the technical depth of a software engineer with the strategic insight of someone actively shaping how GenAI integrates into modern development practices. What follows is not just theoretical analysis, but battle-tested knowledge from the trenches of AI-powered software development.

Introduction

The technological landscape has undergone a seismic shift with the integration of Generative Artificial Intelligence (GenAI) across major tech companies.

From Meta’s content creation pipelines to Amazon's code assistance tools, GenAI has moved beyond experimental phases to become an integral part of daily operations in software development, testing, and innovation processes. This transformation is not just changing how we work today—it’s reshaping the skills and mindset required for
tomorrow’s technology professionals.

GenAI in Major Tech Companies: Real-World Applications

Meta (formerly Facebook)

Meta has deeply integrated GenAI across multiple domains:

Daily Development Usage:

- Code generation for React components and backend services
- Automated code review and optimization suggestions
- Natural language to SQL query generation for data analysis
- Content moderation algorithms powered by AI models
- Automated testing scenarios based on user behavior patterns

Achievements:

- 30% reduction in routine coding tasks
- Faster prototyping of new features
- Enhanced code quality through AI-powered reviews
- Improved content safety through automated detection systems

Google

Google’s approach to GenAI spans across development tools and product integration:

Daily Development Usage:

- Bard integration in Google Colab for code assistance
- Automated documentation generation from code comments
- AI-powered debugging and error resolution suggestions
- Test case generation based on code functionality
- Infrastructure optimization through predictive analytics

Achievements:

- Reduced debugging time by 40%
- Automated generation of technical documentation
- Enhanced code maintainability through AI suggestions

Amazon

Amazon's GenAI initiatives focus heavily on developer productivity:

Daily Development Usage:

- Amazon CodeWhisperer for real-time code suggestions
- AWS CloudFormation template generation
- Automated API documentation creation
- Customer service chatbot development and training
- Supply chain optimization through predictive modeling

Achievements:

- 57% acceptance rate for AI-generated code suggestions
- Reduced development time for AWS services
- Enhanced customer experience through AI-powered support
- Improved operational efficiency across logistics networks

OpenAI

As a pioneer in GenAI, OpenAI uses its own technology extensively:

Daily Development Usage:

- Internal tooling development using GPT models
- Automated testing of model outputs and behaviors
- Code review assistance for research implementations
- Documentation generation for API endpoints
- Training pipeline optimization and monitoring

Achievements:

- Accelerated research and development cycles
- Improved code quality in research implementations
- Enhanced collaboration through AI-assisted documentation
- Streamlined model deployment and monitoring processes

Real-World Development Applications Across Industries

Code Generation and Completion

- Boilerplate code generation for common patterns
- Function implementation based on comments or specifications
- API integration code generation
- Database query optimization and generation
- Configuration file creation and management

Automated Testing and Quality Assurance

- Unit test generation based on function signatures
- Integration test scenario creation
- Performance test automation
- Security vulnerability scanning and remediation
- Code coverage analysis and improvement suggestions

Documentation and Communication

- Automated README file generation
- API documentation from code annotations
- Technical specification writing assistance
- Code comment generation and improvement
- Meeting summary and action item extraction

Code Review and Optimization

- Static code analysis and improvement suggestions
- Performance bottleneck identification
- Security vulnerability detection
- Code style and best practice enforcement
- Refactoring recommendations for legacy systems

Debunking Common GenAI Myths

Myth 1: “AI Will Replace Developers Completely”

Reality:
GenAI is a powerful assistant, not a replacement. It excels at generating boilerplate code, suggesting improvements, and automating repetitive tasks. However, it lacks:

- Deep understanding of complex business requirements
- Ability to make architectural decisions
- Creative problem-solving for unique challenges
- Understanding of user experience and design principles
- Capacity for strategic technical leadership

Evidence: Major tech companies are hiring more developers than ever, with AI augmenting rather than replacing human expertise.

Myth 2: "GenAI Produces Perfect, Production-Ready Code"

Reality:
AI-generated code requires human oversight and often needs significant refinement:

- Security vulnerabilities may be present in generated code
- Performance optimization often requires human intervention
- Context-specific requirements may be missed
- Edge cases and error handling need human consideration
- Integration with existing systems requires careful planning

Best Practice: Always review, test, and validate AI-generated code before production deployment.

Myth 3: "**Learning to Code Is Pointless Now**"

Reality:
Understanding code fundamentals is more important than ever:

- Developers need to evaluate and modify AI-generated code
- Debugging skills are crucial for fixing AI suggestions
- Architectural thinking cannot be automated
- Algorithm selection and optimization require deep understanding
- Code review and security analysis demand technical expertise

Evidence: Job postings increasingly require both traditional programming skills AND AI tool proficiency.

Myth 4: "AI Tools Make Developers Lazy and Dependent"

Reality:
When used properly, AI tools enhance learning and productivity:

- Faster iteration allows more time for complex problem-solving
- Exposure to different coding patterns improves skills
- Reduced time on routine tasks enables focus on architecture
- AI suggestions can teach best practices and new approaches
- Automated testing encourages better development practices

Key: Using AI as a learning partner rather than a crutch is essential.

Guide for College Students: Leveraging GenAI for Career Success

Essential Skills to Develop Alongside AI Tools

1. Fundamental Programming Concepts

- Data structures and algorithms remain crucial
- Understanding of software design patterns
- Database design and optimization principles
- Network protocols and distributed systems
- Security principles and best practices

2. AI Tool Proficiency

- Learn to write effective prompts for code generation
- Understand limitations and biases of AI models
- Develop skills in code review and validation
- Master debugging AI-generated code
- Learn to integrate AI tools into development workflows

3. Critical Thinking and Problem-Solving

- System design and architecture skills
- Business requirement analysis
- User experience and design thinking
- Performance optimization strategies
- Scalability and reliability considerations

Using GenAI as a Learning Accelerator

Effective Learning Strategies:

1. Start with Understanding: Before using AI for code generation, ensure you understand the underlying concepts and requirements.

2. Use AI for Exploration: Ask AI to explain different approaches to solving a problem, then implement and compare them.
3. Practice Code Review: Regularly review and critique AI-generated code to develop analytical skills.
4. Learn from AI Suggestions: Study the patterns and techniques suggested by AI tools to expand your knowledge.
5. Build Incrementally: Use AI to generate starting points, then build upon and customize the code yourself.

Industry Preparation Strategies

1. Build a Hybrid Skill Set

- Traditional programming expertise
- AI tool proficiency
- Understanding of prompt engineering
- Experience with multiple AI platforms
- Knowledge of AI limitations and ethical considerations

2. Develop Portfolio Projects

- Create projects that showcase both manual coding skills and AI collaboration
- Document your development process and decision-making
- Include projects that demonstrate problem-solving without AI assistance
- Show examples of improving and debugging AI-generated code

3. Stay Current with Industry Trends

- Follow developments in AI tools and platforms
- Understand emerging best practices for AI-assisted development
- Learn about new programming paradigms enabled by AI
- Study case studies from major tech companies

4. Practice Collaborative Development

- Work on team projects that incorporate AI tools
- Learn to communicate effectively about AI-assisted development
- Understand code review processes for AI-generated content
- Develop skills in mentoring others on AI tool usage

Practical Recommendations for Academic Integration

For Students:

- Use AI tools for learning and exploration, not as shortcuts
- Always understand the code you submit, regardless of its origin
- Practice explaining AI-generated code in detail
- Develop strong debugging and testing skills
- Focus on building foundational knowledge complements AI capabilities

For Educators:

- Integrate AI tools into curriculum while maintaining academic integrity
- Teach critical evaluation of AI-generated content
- Emphasize understanding over completion
- Create assignments that require both AI collaboration and independent thinking
- Provide guidance on ethical AI usage in academic and professional contexts

Future Outlook and Industry Trends

Emerging Trends

- AI-powered integrated development environments (IDEs)
- Natural language programming interfaces
- Automated code refactoring and modernization
- AI-driven system architecture recommendations
- Predictive debugging and error prevention

Skills for the Future

- Human-AI collaboration expertise
- AI model fine-tuning for specific domains
- Ethical AI development practices
- Cross-functional technical leadership
- Continuous learning and adaptation mindset

Conclusion

GenAI is not replacing developers—it's elevating the profession. The most successful technology professionals of the future will be those who master the art of human-AI collaboration, using these powerful tools to amplify their creativity, productivity, and problem-solving capabilities while maintaining the critical thinking and technical expertise that only humans can provide.

For college students entering this transformed landscape, the opportunity is unprecedented. By building strong foundational skills, embracing AI tools as learning partners, and developing a mindset of continuous adaptation, you can position yourself at the forefront of this technological revolution.

The future belongs not to those who fear AI or those who rely on it blindly, but to those who understand how to harness its power while contributing the uniquely human elements of creativity, empathy, and strategic thinking that remain irreplaceable in technology development.

Remember: AI is a tool that makes good developers great—but it cannot make non-developers into developers. Master your craft, embrace the tools, and prepare to shape the future of technology.

--Sourav Das
Batch(2018-2022)

Decentralized Training of Small Language Models via Federated Learning: Privacy, Promise, and Paradox

Federated learning (FL) reframes supervised learning for a distributed world: models move to data rather than data to models. For small language models (SLMs) —models with up to ~10B parameters— FL promises privacy-preserving, scalable personalization across edge devices, hospitals, and vehicles.
Achieving that promise requires careful co-design across communication, compute, cryptography, and governance. This article condenses technical foundations, system patterns, key threats, and pragmatic research directions needed to move federated SLMs from experimental deployments to robust,
auditable production systems.

Core principles and system patterns

- **Local training, global aggregation:** Clients compute gradients or parameter deltas on local data; a central or decentralized aggregator combines updates into a global model. Common protocols include FedAvg-style weighted averaging and variants that weigh by data quality, client reliability, or temporal recency.

- **Parameter-efficient adaptation:** Full fine-tuning of SLMs on edge devices is often infeasible. Techniques such as LoRA (Low-Rank Adaptation), adapter layers, and prompt-tuning confine trainable parameters to small, dense or low-rank subspaces, reducing compute, memory, and communication costs while preserving most downstream performance gains.

- **Compression and selective updates:** Quantization, sparsification, top-k gradient selection, and error-feedback schemes reduce uplink costs. Knowledge distillation transmits compact student updates or logits instead of full parameter matrices, trading off fidelity for bandwidth efficiency.

- Split architectures: Split learning partitions the model between client and server, allowing clients to compute early layers and send activations or compressed adapters to a server-hosted backbone. SplitLoRA extends parameter-efficient adaptation to split settings by hosting core weights centrally while training lightweight adapters locally.

- **Decentralized topologies:** Peer-to-peer ring and mesh topologies remove single coordinators, offering resilience and lower latency in some settings. Consensus protocols or gossip-based averaging implement asynchronous, redundant aggregation, but they complicate robustness and convergence analysis.

- **Privacy-preserving primitives:** Differential privacy (DP) provides statistical guarantees against membership inference when applied at client or server aggregation; secure aggregation protocols and multiparty computation (MPC) conceal individual updates during aggregation; homomorphic encryption permits some computation on encrypted updates at the cost of heavy overhead.

Practical deployments and application patterns

- **Mobile and consumer personalization:** Keyboard prediction, next-token personalization, and on-device assistants use FL with adapter-based fine-tuning and DP to personalize language models without centralizing raw text.

- **Healthcare collaboration:** Hospitals collaborate on predictive models (risk stratification, imaging-based prognosis) by sharing gradient statistics or adapter updates encoded under secure aggregation. FL enables cross-institution learning where regulatory or contractual constraints forbid raw data sharing.

- **Autonomy and IoT fleets:** Vehicles, drones, and sensor networks benefit from federated updates for perception and forecasting tasks, learning from geographically diverse scenarios while limiting raw sensor transmission.

- **Cross-institution finance:** Banks and payment processors apply FL for fraud detection and credit scoring by exchanging model updates inside strict cryptographic shells and bespoke governance frameworks.

Technical frictions and adversarial threats

- Communication and convergence trade-offs: Aggressive compression and sparse updates reduce bandwidth but introduce bias and slow convergence. Asynchronous aggregation and heterogeneous local compute budgets exacerbate staleness, requiring adaptive learning rates, momentum correction, or temporally-aware aggregation.

- Model poisoning and backdoor insertion: Byzantine clients can craft updates to shift model behaviour or insert backdoors. Robust aggregation rules (median, trimmed mean, Krum), anomaly detection, and reputation systems mitigate.

- **Gradient inversion and reconstruction attacks:** Clients' updates can leak training data via inversion techniques; DP, secure aggregation, and gradient clipping reduce leakage but require careful calibration to preserve utility.

- **Epistemic opacity and interpretability loss:** Compression, adapt do not eliminate adaptive poisoning attacks, especially in decentralized topologies where attackers cluster.er mixing, and distributed asynchronous updates produce models whose internal representations differ across replicas and time, complicating debugging, interpretability, and reliable uncertainty estimation.

- **Cryptographic and operational limits:** MPC, homomorphic encryption, and threshold signatures increase privacy guarantees but impose computation, latency, and engineering complexity. Quantum threats and side channels remain long-term risks.

Governance, verification, and accountability

- **Auditability without raw data:** Auditable logs of training rounds, signed update digests, and verifiable compute enclaves can provide traceability. Techniques for provenance-awareaggregation and tamper-evident summaries help assign responsibility for model changes without exposing private data.

- **Decentralized governance models:** Tokenized or consortium governance can coordinate architecture decisions, but incentive asymmetries risk concentration of influence. Hybrid governance—local legal accountability combined with cryptographic provenance—balances decentralization with practical oversight.

- **Bias and representation:** Federated datasets reflect participation biases; unequal device distribution or institutionally skewed contributions result in models that underperform on underrepresented populations. Data valuation, fairness-aware aggregation, and targeted sampling can mitigate but require continuous monitoring.

Future research directions

Below is a concise table of targeted research directions with concrete goals, suggested evaluation metrics, and near-term priority to guide focused work on federated SLMs.

Area	Open problem	Concrete research direction	Evaluation metrics	Priority
Scalability	Bandwidth vs utility at edge	Adaptive hybrid updates: combine LoRA adapters, Top-k compression, and periodic full-sync	Communication bytes/model improvement per round; rounds-to-convergence	High
Security	Robustness to adaptive poisoning	Provable Byzantine-resilient aggregation with adaptive trust scoring	Attack success rate; clean-accuracy degradation; false positive rate	High
Privacy	Utility loss under DP	Client-side adaptive DP with per-update noise calibration and utility feedback loops	Utility-privacy frontier (AUC vs epsilon); reconstructed-example fidelity	High
Interpretability	Drifted, opaque weights across devices	Federated attribution: distributed saliency and cross-client representation alignment	Attribution stability; cross-client explanation agreement	Medium
Verification	Audit without raw data	Verifiable ML pipelines: signed update commitments + succinct SNARK-like proofs for aggregation	Proof generation/verif time; false verification rate	Medium
Governance	Incentive centralizatio n in consortiums	Hybrid governance: cryptographic provenance + legal SLAs + resource-weighted voting	Power concentration index; governance throughput; participant fairness	Medium
Lifelong learning	Continuous auditing of evolving models	Snapshotting, changelogs, and certified rollback points with incremental validation suites	Time-to-detect drift; rollback fidelity; downstream stability	High

Hardware/ software co-design	Heterogeneous edge capabilities	Adaptive model-partitioning scheduler that maps adapters/sublayers to device capability	Utility per watt; latency; percentage of participating devices	Medium
Data quality	Poisoned or low-quality client data	Federated data valuation and selective weighting using influence estimation	Weighted-accuracy; detection precision; robustness under noise	High

Conclusion

Federated training of SLMs offers a viable path toward privacy-respecting, widely distributed intelligence. Practical systems combine adapter-based fine-tuning, compression, split architectures, and cryptographic primitives to balance utility, latency, and confidentiality. However, achieving resilient, fair, and auditable federated SLMs demands integrated advances across algorithms, cryptography, systems engineering, and socio-technical governance.

Future work should prioritize the high-impact directions listed above: scalable hybrid update schemes, provable defences against adaptive poisoning, and privacy mechanisms that preserve utility while meeting regulatory constraints. Equally important are audit mechanisms and governance structures that map technical provenance to real-world accountability. Only by coupling technical rigor with institutional innovation can federated SLMs fulfil their promise without falling prey to the paradoxes of decentralization.

--By Pritam Ghosh
Batch(2016-2020)

A fairy tale that depicts the Evolution of Software Quality Engineering with Selenium

Software Testing: A Gatekeeping Activity for Business Assurance

This article tries to provide an insight of how the quality assurance process in a typical software engineering life cycle model has evolved from a boring monotonous manual activity to a fully automated phase that can be seamlessly integrated with the standard continuous deployment pipelines to ensure the quality of the application of product.

What is Software Testing?

Software testing is an integral part of the software development life cycle that involves evaluating and verifying a software application or a software product to ensure it functions correctly, meets all specified requirements, and is free of defects. It aims to identify bugs, errors, and missing functionalities before the software is released to end-users, ultimately delivering a high-quality, reliable, and user-friendly product.
The Software Testing Life Cycle or STLC has several phases e.g. Requirement Analysis, Test Planning, Test Design, Environment Setup, Test Execution and Test Closure.

What is the role of automation in Software Testing?

Automation Testing plays a crucial role now-a-days in modern software development methodologies, especially in Agile and DevOps environments, by accelerating the testing process and improving reliability. The process of Automation testing involves using specialized software tools to execute test cases automatically, eliminating the need for manual intervention.

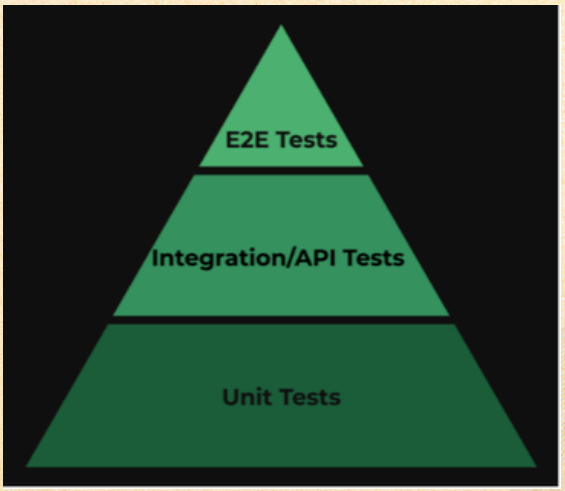
Which tests should be automated?

Automating the right tests improves efficiency, accuracy, test coverage and accordingly reduces overall testing time and effort, thereby saving huge cost of software development.

The following types of tests should be prioritized for automation:

- **Repetitive & Monotonous Tests:** Frequently executed tests, such as regression and smoke tests, benefit from automation.
- **Data-Driven Tests:** Tests requiring multiple data sets, like form validations and database testing, should be automated to ensure consistency.
- **Business-Critical Tests:** High-risk functionalities impacting all the core business operations should be automated for continuous validation.
- **Deterministic Tests:** Test cases with clear pass/fail criteria, such as UI validations and API responses, are ideal for automation.
- **Cross-Platform & Environment Tests:** Tests that need execution across different OS, browsers, or devices should be automated to ensure compatibility.
- **Time-Consuming & Tedious Tests:** Repetitive tasks prone to human error, like performance and load testing, should be automated for efficiency.

The commonly suggested approach towards automated testing is the “Test Pyramid”, that comprises of unit tests followed by a smaller set of integration tests and finally a few e2e or End-to-end tests.



Automated testing covers various types of testing to ensure software quality, functionality, and security. Below are some key types:

- **Regression Testing:** Regression suites are ever-increasing and require the same variables to be filled numerous times to ensure that new features do not tamper with older functions. This can easily be automated.
- **Smoke Testing**: Run automated suites to verify the quality of major functionalities. This saves time by quickly analyzing whether a build requires more in-depth testing.

- **Data-driven Testing:** Automate tests to validate functionalities that must be tested repeatedly with numerous data sets.

- **Performance Testing:** Automate tests that monitor software performance under different circumstances. Doing this manually would be incredibly detailed and time-consuming.

- **Functional Testing:** Every time a developer submits a Pull Request (abbreviated as PR), functional testing of the concerned feature needs to be performed quickly to provide immediate feedback. This is impossible to achieve without automation, especially as organizations scale up.

- **API Testing:** Validates the functionality, security, and reliability of an application's APIs, ensuring seamless communication between software components.

- **UI Testing:** Ensures that all user interface elements, such as buttons, fields, and layouts, function correctly and provide a smooth user experience.

Unit Testing:

Tests individual components or units of code in isolation to verify their correctness and functionality during development.The journey of Selenium

Selenium is an open-source framework designed for automating web browsers. It enables users to test the website's functionality across different browsers, ensuring consistency and compatibility.

Before the revolution that came to Software Quality Assurance practices with Selenium, it was QTP which dominated the automation testing landscape for years. By the late 2000s, the demand for web-based applications started increasing over standalone desktop tools and softwares.

This period saw major advancements in web technologies like JavaScript, browser performance, and open-source libraries, making web apps more viable and often preferable alternative to desktop software, even for tasks traditionally handled by native applications. Selenium cannot automate desktop applications; it can only be used in browser automation. However, it supports a number of browsers such as Chrome, Firefox, Internet Explorer, Safari, Opera and operating systems such as Windows, Mac, Linux/Unix. With the increase of web-based applications, QTP became less dominant,

and Selenium took over because of the differences in cost, architecture, ecosystem, language support, CI/CD friendliness, community momentum, and the rise of web applications and open-source culture. The shift was driven by the practical needs of modern development teams rather than a single technical failure.

Given below is a summary of the various software versions of Selenium:

- **Selenium 1 (2004–2011):** Introduced as Selenium Remote Control (RC). Here, a server was required to act as a proxy to inject commands into browsers.

- **Selenium 2 (2011):** Selenium RC was integrated with the WebDriver API. WebDriver directly communicates with browsers for more stable and faster tests. Selenium RC was deprecated but still included for backward compatibility.

- **Selenium 3 (2016):** In Selenium 3, WebDriver became the main focus; Selenium RC was officially deprecated. Support for modern browsers was improved, and W3C WebDriver standard was introduced.

- **Selenium 4 (2021):** Selenium 4 provided full support for W3C WebDriver Protocol to ensure better browser compatibility. Selenium Grid (distributed testing) was enhanced. Selenium IDE was revived as a browser extension (Chrome/Firefox). New bi-directional was established for communication for performance and network testing

- **Selenium 4.11+ and 4.12+ (2023):** These versions had minor enhancements that improved Grid logging and Docker support, and support for more browser capabilities.

- **Selenium 4.14.0 (January 2024):** This release includes updates to browser-specific drivers, grid stability improvements, and bug fixes.

Selenium 3.X: The WebDriver

Selenium WebDriver is a very powerful and enhanced version of Selenium RC which was developed to overcome the limitations of Selenium RC. WebDriver communicates with browsers directly with the help of browser-specific native methods, making it faster and more reliable.

Selenium also provides compatibility with different programming languages – C#, Java, JavaScript, Ruby, Python etc. Testers can choose the language to design test cases in, thus making Selenium highly favorable for its flexibility.

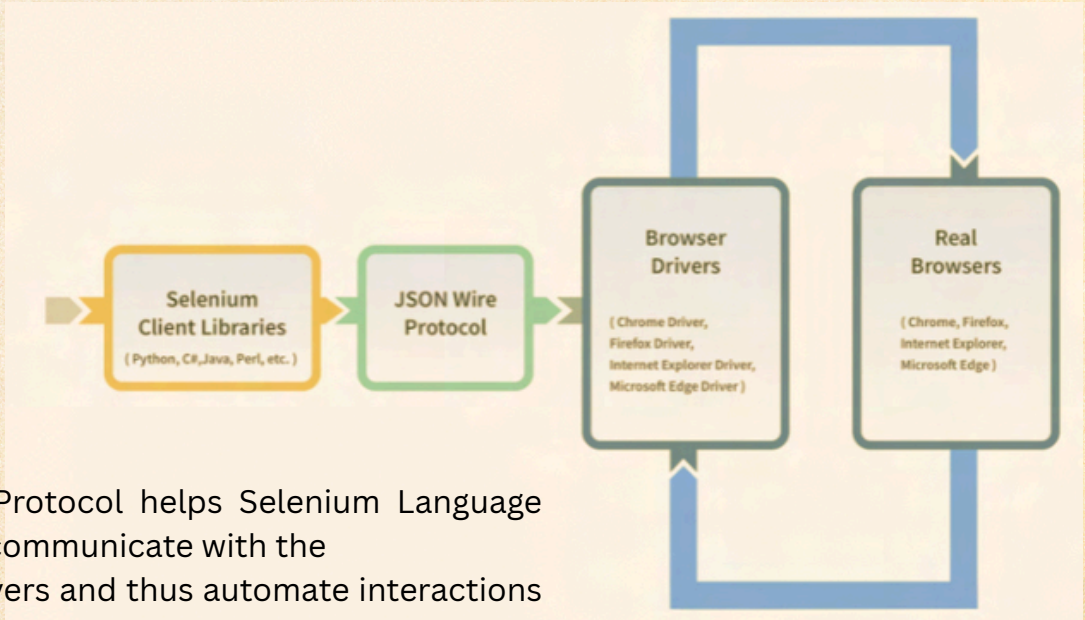
Components of WebDriver Architecture

- **Selenium Client library:** Selenium provides support to multiple libraries e.g. Ruby, Python, Java etc. as language bindings

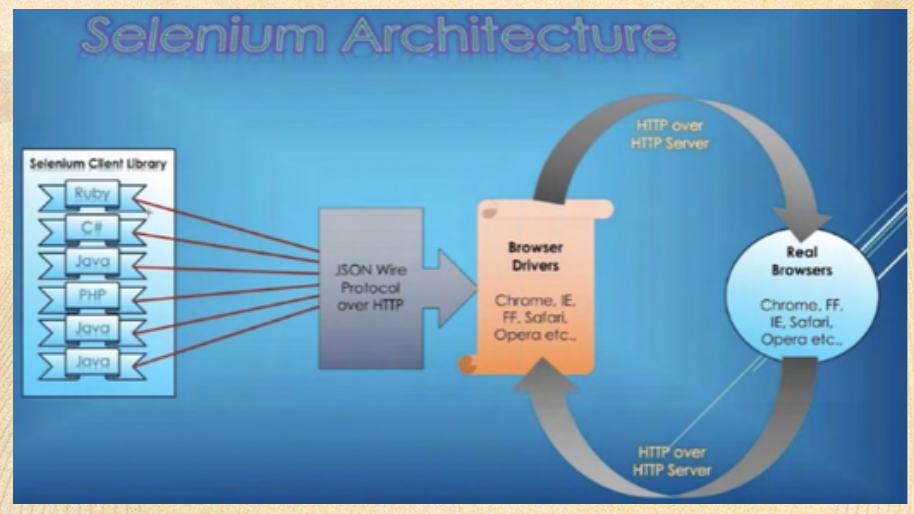
- **JSON Wire Protocol over HTTP:** JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web.

- **Browser Drivers:** Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, EdgeDriver, SafariDriver.

- **Browsers:** The concerned platform (Windows/Mac/Linux) needs the corresponding browsers i.e. Chrome, Firefox, Edge, Safari to be installed for automated UI tests to be executed on them.



JSON Wire Protocol helps Selenium Language Bindings to communicate with the Browser Drivers and thus automate interactions on real browsers.



How to use Selenium WebDriver to test a web application?

Below Java code launches BrowserStack web application on chrome browser and verifies the page title.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class BrowserStackDemo {
    private WebDriver driver;

    @Test
    public void verifyTitle() {
        driver = new ChromeDriver();
        driver.get("https://www.browserstack.com/");
        Assert.assertEquals(
            driver.getTitle(),
            "Most Reliable App & Cross Browser Testing
Platform | BrowserStack"
        );
        driver.quit();
    }
}
```

The code above does the following:

1. Create a Selenium WebDriver instance
2. Configure browser if required
3. Navigate to the required web page and locate the relevant web element
4. Perform action on the web element
5. Verify and validate the action

This relates to the well-known AAA principle, which stands for Arrange, Act, and Assert. It is a pattern for structuring unit tests, where Arrange sets up the initial conditions and data, Act executes the code being tested, and Assert verifies that the outcome matches the expected result. This structure makes tests more readable, maintainable, and easier to troubleshoot by clearly separating the test's setup, execution, and verification phases.

Arrange

- **Purpose:** To set up the necessary conditions and data for the test.
- **Actions:** Create objects, initialize variables, and mock dependencies.

Act

Purpose: To perform the specific action you want to test.
Actions: Invoke the method or function being tested.
Example: Call the login() method on the user object with the arranged credentials.

Assert

Purpose: To verify that the outcome of the action is as expected.
Actions: Check if the result is correct, or if the state of the system has changed as intended.
Example: Confirm that the user is successfully logged in, or that the correct error message is displayed if the credentials were invalid.

Here is a sample step-by-step Java code with Selenium for automation.

```
1. Launch Flipkart
// Use WebDriver to open the Flipkart website URL.
WebDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");

2. Search for products
WebElement searchBar = driver.findElement(By.id("q"));
searchBar.sendKeys("iPhone 15 Pro Max");
searchBar.submit();

3. Find product
driver.findElement(
    By.xpath("//div[normalize-space()='Apple iPhone 15 Pro Max
(Blue Titanium)']")
).click();

// This will open a new window tab for the chosen product.
// Therefore, the WebDriver needs to point to the correct tab
for further processing.
String mainPage = driver.getWindowHandle();
Set<String> allPages = driver.getWindowHandles();
for (String page : allPages) {
    if (!page.equals(mainPage)) {
        driver.switchTo().window(page);
        break;
    }
}

4. Add to cart
driver.findElement(By.xpath("//a[@title='Cart']")).click();
```

Benefits of Automated Testing

Automation testing offers several benefits over manual testing, particularly in scenarios requiring repetitive execution and high accuracy.

- ***Efficiency and Speed:***

Automated tests execute significantly faster than manual tests, allowing for quicker feedback on software quality and enabling more frequent testing cycles. This is crucial for continuous integration and continuous delivery (CI/CD) pipelines.

- ***Accuracy and Reliability:***

Automation eliminates the potential for human error inherent in manual testing, leading to more consistent and reliable test results. Automated scripts follow predefined steps precisely, ensuring the same test is performed identically every time.

- ***Increased Test Coverage:***

Automated testing facilitates broader and deeper test coverage. It can efficiently execute a vast number of test cases across various configurations and environments, which would be impractical or impossible to achieve manually within the same timeframe.

- ***Cost-Effectiveness (Long-term):***

While initial setup costs for automation tools and script development can be higher, automation testing proves more cost-effective in the long run. It reduces the need for extensive manual effort, especially for regression testing, freeing human testers for more exploratory and complex tasks.

- ***Reusability and Maintainability:***

Automated test scripts are reusable across different builds and releases, reducing the effort required to re-test functionalities. Well-designed scripts are also easier to maintain and update as the application evolves.

- ***Scalability:***

Automation testing scales easily to accommodate growing test suites and complex applications. This allows teams to manage increasing testing demands without proportionally increasing human resources.

- ***Faster Feedback Loop:***

Automated tests provide immediate feedback on the impact of code changes, allowing developers to identify and address defects earlier in the development cycle, which is more cost-efficient than fixing bugs later.

A short note on Selenium 4

Selenium 4 is the latest major iteration of the open-source framework for automating web browsers. It introduces significant enhancements and architectural changes aimed at improving web testing and automation.

Key features and improvements in Selenium 4:

- **W3C WebDriver Standardization:** Full adherence to the W3C WebDriver Protocol ensures greater cross-browser compatibility and consistency, eliminating the need for JSON Wire Protocol used in previous versions. This simplifies communication between the client and the browser.

- **Revamped Selenium Grid:** Grid 4 is a complete rewrite, offering improved scalability, easier deployment, and better support for modern technologies like Docker and Kubernetes. It simplifies parallel test execution across multiple browsers and operating systems.

- **Relative Locators:** New relative locators such as above, below, toLeftOf, toRightOf, near etc. provide more intuitive and robust ways to locate elements on a webpage, especially when traditional locators are unstable.

- **Chrome DevTools Protocol (CDP) Integration:** Selenium 4 offers direct integration with Chrome DevTools Protocol, allowing for advanced interactions and debugging capabilities e.g. network monitoring, performance analysis, and mock responses.

- **Enhanced Selenium IDE:** The Selenium IDE has been updated with a more user-friendly interface, support for modern browsers, and the ability to export tests in various programming languages.

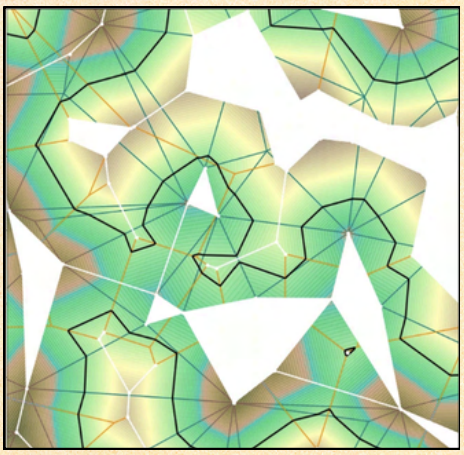
In essence, Selenium 4 aims to provide a more stable, efficient, and modern platform for web automation, addressing the evolving needs of web testing with enhanced features and a streamlined architecture.

~~Rahitaswa De
Batch(2006-2010)

The Hidden Geometry of Data: Understanding Topological Data Analysis

If you open WhatsApp for a moment and scroll through your chats, you'll see friends, classmates, family groups (hopefully), lab groups, that one project team that only wakes up before deadlines, and maybe a few muted groups you haven't opened for ages. Now imagine turning your WhatsApp contacts into a picture. Connect every two people who regularly chat. Slowly, a pattern starts forming: tight clusters of close friends, bridges between groups, and maybe some people completely isolated. What you have just drawn is not just a social graph, it's a shape. And studying that shape is what mathematicians call Topological Data Analysis or TDA.

TDA looks at data not as rows and columns, but as a geometry. It asks: how are things connected? What forms loops, clusters, or holes? What stays unchanged even when everything else change? Instead of numbers, it studies relationships -using tools from topology, the branch of math that says a coffee mug and a doughnut are 'the same' because each has one hole. On a side note: if talking about holes slightly reminds you of the human body, don't miss out enjoying the Youtube video 'How Many Holes Does a Human Have?' by Vsauce.

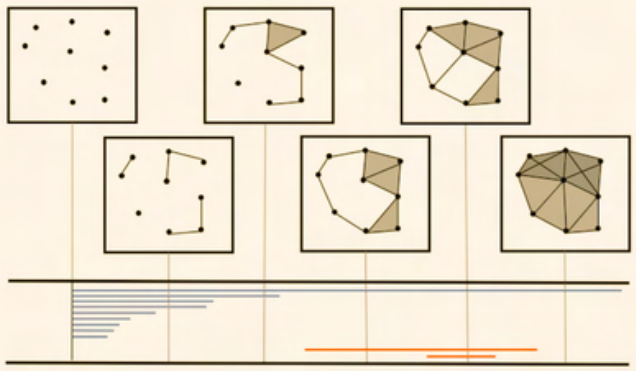


Exploring interactions of Delaunay mosaic and Voronoi tessellations on a 2D point set. Figure taken from "Continious and Discrete Radius Functions on Voronoi Tessellations and Delaunay Mosaics" by Biswas, Cultreradi Montesano, Edelsbrunner, and Saghafian, Discrete & Computational Geometry,2022.

Coming back to all seriousness: to make sense of complex data, TDA builds something called a simplicial complex. Think of each WhatsApp user as a point. Two friends who chat often are connected by a line. If three friends all talk to each other, that forms a triangle. Four friends - a tetrahedron; you get the deal!

Now take enough of these together, and you have got a high-dimensional shape built from data. Sometimes using tools from **Computational Geometry**, like **Voronoi tessellation** or **Delaunay mosaic**, can quickly show you structures from the mess.

But real world data is even messier. Some connections are random, others strong. So, TDA doesn't just look at one picture; it builds many, each at a different 'threshold' of connection strength.



Persistence barcode for a simple point cloud. We gradually connect points which are closer to each other generating lines, triangles, tetrahedrons, etc. The blue lines on the barcode represent the connected components and the red ones are for loops. Figure taken from 'Persistence paths and signature features in topological data analysis' by Chevyrev, Nanda, and Oberhauser, TPAMI, 2020.

Then it tracks which shapes persist across levels. That's the idea of persistence - patterns that survive filtering are likely meaningful, not noise. These enduring features are captured in a persistence diagram or barcode, where long bars represent stable structures, like real friend groups, and short bars represent random chat overlaps that quickly disappear. If a pattern refuses to die, it must mean something!

But real world data is even messier. Some connections are random, others strong. So, TDA doesn't just look at one picture; it builds many, each at a different 'threshold' of connection strength. Then it tracks which shapes persist across levels. That's the idea of persistence - patterns that survive filtering are likely meaningful, not noise. These enduring features are captured in a persistence diagram or barcode, where long bars represent stable structures, like real friend groups, and short bars represent random chat overlaps that quickly disappear. If a pattern refuses to die, it must mean something!

homology, which counts holes, and homotopy, which describes how shapes can bend without breaking. These are ways to describe what makes one shape fundamentally different from another. To a topologist, your WhatsApp network and a galaxy map aren't so different, both are just points and connections, forming loops and voids that carry meaning.

Topology is certainly not the answer to every data problem, but it fills an important gap. It helps us detect structure that normal algorithms tend to miss, especially when the data is messy, high-dimensional, or behaves in unexpected ways. It's a field where solid mathematical thinking meets real applications, and where new ideas are still very much welcome. If you are looking for a direction with both depth and opportunity, topology-based methods are worth exploring. For further exploration, here are some first points of contact: 'Applied Topology' playlist by Henry Adams at Youtube, 'Topology and Data' article by Carlsson, 'Computational Topology: An Introduction' book by Edelsbrunner and Harer. All freely available online! Viel Glück!

--Ranita Biswas
Batch(2005-2009)

Generic Article

Learning Beyond Books: The Classroom Called Travel

Close your eyes for a second and think of your last trip. Maybe it was a college picnic, a family vacation, or just a short escape with friends. Do you remember how alive you felt? The taste of food that wasn't from your college canteen, the laughter that came too easily, the way you forgot assignments for a while? That's what travel does. It doesn't just take you to new places, it brings you closer to yourself.

Travel: Another Day of Learning

In college, we think classrooms are where learning begins and ends. But the truth is, travel is another kind of classroom—one without blackboards, attendance sheets, or exams. Every journey teaches something. Missing a bus teaches patience. Bargaining in a local market teaches negotiation. Finding your way in an unknown city teaches problem-solving. The beauty of travel is that the lessons don't feel forced. You don't realize you've learned until you look back and see how much more confident, adaptable, and curious you've become.

India: A Land of Many Worlds

India is like a dozen countries rolled into one. Snowy peaks rise in Ladakh and Kashmir, deserts stretch across Jaisalmer in Rajasthan, tropical beaches shine in Kerala, and dense forests shelter Assam's rhinos. Each journey feels like stepping into another universe. Imagine paddling a shikara on Dal Lake, walking the golden sands of Jaisalmer, or trekking the frozen Zanskar River. Pangong Lake glimmers like liquid sapphire, while Gulmarg glows in white velvet during winter.

Himachal's Spiti Valley lets you mail a letter from Hikkim—the world's highest post office—or walk through Chitkul, the last village of India. In Uttarakhand, the evening aarti at Haridwar, the yoga spirit of Rishikesh, and Rudraprayag's calm leave you transformed. The Northeast holds its own magic—Meghalaya's living root bridges, Dawki's glass-clear river, and Assam's mighty Brahmaputra cutting through Kaziranga, home of the one-horned rhino. Sikkim's Gurudongmar Lake mirrors Ladakh's Pangong, while Darjeeling's toy train whistles through tea gardens that look like green carpets.

Some places are history lessons in stone: Hampi's ruins, Delhi's Red Fort and Qutub Minar, Agra's Taj Mahal, or the temples of Khajuraho. Others feel like the world's edge: Dhanushkodi's ghostly sands, the moonlit Rann of Kutch, or Spiti's Martian landscapes.

India's Twin Worlds

India is so vast that one place often reminds you of another. Ladakh's cold desert reflects Spiti's stark beauty, while Jaisalmer's golden dunes stand in contrast to Kashmir's frozen valleys. Gurudongmar in Sikkim sparkles like Pangong in Ladakh. The Andaman Islands and Lakshadweep share turquoise lagoons and coral reefs, as stunning as any foreign beach. In India, you don't need to cross oceans to see the world—you'll find it twice over here.

Food and Culture: A Journey on the Plate

Every Indian trip is also a food trail. Delhi's spicy chaat, parathas, and kebabs fight for space with Varanasi's paan and street food. Hyderabad gives you biryani rich with history, Kolkata offers phuchkas and kathi rolls bursting with flavour, Mumbai keeps you going with vada pav, while Goa tempts with seafood. Maharashtra's misal pav adds fire to mornings, Kerala's appam and stew soothe the soul, and Kashmir's wazwan is a royal feast on its own. Food is more than taste—it's connection. Sharing a Rajasthani thali under desert skies, sipping butter tea in Ladakh, or tasting momos in Sikkim ties you to people and their stories. Every bite is culture, every meal a memory.

How Travel Changes You

Travel is like holding up a mirror you've never seen before. It teaches patience when your bus is late, humility when you stand before the Himalayas, courage when you climb trails you once feared, and joy when strangers share food with you. More than anything, it changes the way you see people—"different" stops meaning "strange," and starts meaning "a new story to learn." These are lessons no classroom can give, but they stay etched in your heart forever.

Why Students Should Travel More

For students, travel is the best teacher. It makes you a leader when you plan trips, a team player when you share rooms, an economist when you stretch your pocket money, and a problem-solver when things go wrong.
Delhi's monuments bring history alive, Uttarakhand's treks turn geography into real landscapes, and the ghats of Varanasi whisper philosophy louder than books.

India's Twin Worlds

India is so vast that one place often reminds you of another. Ladakh's cold desert

reflects Spiti's stark beauty, while Jaisalmer's golden dunes stand in contrast to Kashmir's frozen valleys. Gurudongmar in Sikkim sparkles like Pangong in Ladakh. The Andaman Islands and Lakshadweep share turquoise lagoons and coral reefs, as stunning as any foreign beach. Forests, too, find their mirrors—Kaziranga's rhinos in Assam, Gir's lions in Gujarat, Ranthambore's tigers in Rajasthan, and Bandhavgarh's in Madhya Pradesh. In India, you don't need to cross oceans to see the world—you'll find it twice over here.

Food and Culture: A Journey on the Plate

Every Indian trip is also a food trail. Delhi's spicy chaat, parathas, and kebabs fight for space with Varanasi's paan and street food. Hyderabad gives you biryani rich with history, Kolkata offers phuchkas and kathi rolls bursting with flavour, Mumbai keeps you going with vada pav and Kashmir's wazwan is a royal feast on its own. Food is more than taste—it's connection. Sharing a Rajasthani thali under desert skies, sipping butter tea in Ladakh, or tasting momos in Sikkim ties you to people and their stories.

How Travel Changes You

Travel is like holding up a mirror you've never seen before. It teaches patience when your bus is late, humility when you stand before the Himalayas, courage when you climb trails you once feared, and joy when strangers share food with you. More than anything, it changes the way you see people—"different" stops meaning "strange," and starts meaning "a new story to learn." These are lessons no classroom can give, but they stay etched in your heart forever.

Why Students Should Travel More

For students, travel is the best teacher. It makes you a leader when you plan trips, a team player when you share rooms, an economist when you stretch your pocket money, and a problem-solver when things go wrong.
Delhi's monuments bring history alive, Uttarakhand's treks turn geography into real landscapes, and the ghats of Varanasi whisper philosophy louder than books. Every step is learning by living.
So, pack your bags. Take budget trips, take luxury trips, take solo trips, take group trips. Just don't sit still while the world waits outside your door.
Because in the end, you don't travel to escape life. You travel so that life doesn't escape you

-Anusua Mazumder
Faculty member of Information Technology

From KGEC to the World: How to Land a Job, Build a Business, and Shape a Career That Lasts

The headlines can be noisy - job cuts, slower fresher intake, and new tools every week. Underneath that noise, one signal remains constant: companies hire quality. Not a list of half-watched tutorials, but solid fundamentals, the ability to build end-to-end systems, and the habit of doing hard work every single day. If you can show those three, you will cut through any market cycle.

In 2011, I started Techno Exponent with three people. Today, we are 450+ strong across geographies, and I still feel I've reached only 5% of my dream. What got us here - and what continues to push me - is disciplined daily practice and a builder's mindset. As a KGEC alumnus, I can say with confidence: KGEC is one of the best colleges in our state. The talent is here. The world-class careers will follow when we combine that talent with focus, depth, and consistency.

For context, beyond my day-to-day at Techno Exponent, I also serve as CTO of All Publicart. Over the years, parts of my work and approach have been recommended by a NASA engineer. I have been recognized as a Most Influential Young Leader by AsiaOne and The Economic Times. As a company, Techno Exponent has earned awards such as AI/ML Solutions Provider of the Year (Starz) and Times Leading IT Company. I share this not as promotion, but to underline a simple point: the fundamentals in this article are battle-tested in real projects and real markets.

What hiring really looks like now. When we hire from our Techno Exponent LLC office in Miami, USA, the candidates who stand out are deeply practical and project-oriented. They talk less about buzzwords and more about working systems, metrics, and trade-offs. Across campuses, I still meet many bright students who struggle with basics in C/Java/Python, algorithmic thinking, and operating system or networking concepts. Often there is no mental model of how frameworks actually work - how they do IO, manage state, enforce security, or scale. With the AI wave, most firms want end-to-end engineers: people who can define a problem, design an architecture, implement, test, ship, and observe in production.

The builder's map - choose one lane, go deep, ship. Full-Stack Engineering (Web and APIs). Your foundation is data structures and algorithms, OOP and design patterns, SQL, networking (TCP, HTTP, HTTP-2, HTTP-3), and Linux basics. Then commit to one strong backend and one serious frontend. Backend (pick one): Java with Spring Boot; Node.js with NestJS; or Python with FastAPI or Django.

Master REST or GraphQL, ACID vs BASE, transactions, indexing and query plans, caching patterns (TTL, stampede protection, cache-aside), messaging (Kafka or RabbitMQ with back-pressure), idempotency and retries, and saga or outbox patterns for distributed consistency.

Frontend (pick one): React with Next.js (SSR or ISR, routing, server actions), Angular, or Vue always with TypeScript, solid state management (React Query, Zustand, NgRx, or Pinia), accessibility, Lighthouse budgets, and tests with Jest or Playwright.

DevOps and security that are non-negotiable: Docker multi-stage builds, CI or CD with GitHub Actions, infrastructure as code with Terraform, a working grasp of AWS (S3, CloudFront, RDS, and either ECS or EKS basics), OAuth2 or OIDC, JWT pitfalls, mTLS, and secret management with KMS or Secrets Manager. Add observability with structured logs, metrics, traces via OpenTelemetry, SLOs with error budgets, and the RED or USE methods to think clearly about system health.

Capstone to prove it: a multi-tenant SaaS with RBAC, billing, audit logs, rate limiting, background jobs, and end-to-end tests. Publish latency and error metrics and a one-page trade-off note explaining your choices.

AI or ML Engineering (NLP or Computer Vision). The bar in enterprises is not that a model runs; the bar is problem framed to data ingested to model trained to evaluated to deployed to monitored. Targets are explicit (often above 90% task-appropriate accuracy or a defended metric), latency budgets are real, and data quality matters as much as the model.

Core stack: Python, NumPy, pandas, scikit-learn, and either PyTorch or TensorFlow. In NLP, learn tokenization, embeddings, fine-tuning, retrieval-augmented generation (RAG), prompt-injection defenses, and grounding with citations. In CV, learn augmentations, detection or segmentation, class imbalance handling, and evaluation beyond top-1 accuracy.

MLOps realities: data versioning (DVC or LakeFS), experiment tracking and a registry (MLflow), feature stores, batch vs real-time inference, CUDA or GPU vs CPU trade-offs, Triton or ONNX serving, FastAPI endpoints, autoscaling and batching, drift detection, and canary rollouts.

Be comfortable with open-source models and paid APIs; choose based on cost, compliance, privacy, and latency, not fashion.

Capstone to prove it: a document-intelligence pipeline for invoices or claims - OCR plus layout analysis to entity extraction to validation rules to a simple human-in-

the-loop review UI. Publish precision or recall, latency histograms, and a drift monitoring plan.

Data Analytics and Engineering (analytics you can trust). You are building decision systems, not just charts. Understand data lake vs warehouse, columnar formats like Parquet, partitioning, star schema, slowly changing dimensions, governance, and lineage. On cloud: AWS (S3, Glue,
Athena, Redshift), GCP (GCS, BigQuery), or Azure (Data Lake and Synapse). Do transformations in dbt, add quality checks with Great Expectations, and orchestrate with Airflow or Prefect.

Analytics layer: Power BI or Tableau, strong SQL, and meaningful metrics (CAC, LTV, cohorts, funnel conversion). Document your semantic layer, schedule refresh, and add row-level security if needed. Capstone: a sales and marketing analytics stack - raw to modeled to dashboards with drill-downs and a README that answers what decision can be taken now.

Blockchain or Web3 (when on-chain truly helps). Be bilingual in on-chain and off-chain. Smart contracts in Solidity (EVM) or Rust (Solana) with tests in Foundry or Hardhat. Learn ERC-20, ERC-721, ERC-1155, ownership and access control, upgradeable proxies (UUPS), and multisig (Gnosis Safe). Security matters: re-entrancy, checks-effects-interactions, oracle manipulation,and time attacks. Integrate wallets (MetaMask or Phantom), work on gas optimization, index events with The Graph, pull external data via Chainlink oracles, and build a minimal off-chain service for queueing or analytics. Capstone: a tokenized-asset marketplace with escrow, dispute resolution, royalties, and a subgraph-powered UI, plus a short threat model and gas report.

Turning information overload into mastery. Use a 3x3 plan for the next 90 days. Three core skills: DSA with one backend and one frontend (or NLP or CV, MLOps, and data handling). Three deep projects: pick problems with real users - college admin workflows, a local business dashboard, or an NGO data pipeline. Three daily habits for 60 to 90 minutes each: code (implement or extend),
read (docs, specs, or even source code), and write (design notes, README, or a short dev blog). This routine beats tutorial bingeing. Depth compounds.

What enterprise-grade actually feels like. Your code is predictable under stress. You design for failure paths (timeouts, partial writes, duplicate events) before you celebrate happy paths.

You use feature flags, migrations with rollback, rate limits, dead-letter queues, and idempotent handlers. You ship with dashboards and alerts on what users feel (latency, error rate, saturation). You write a one-page runbook:

how to diagnose, how to recover. That is the difference between a demo and a product.Internships, first jobs, and showing energy. Do not wait for final year. In your second or third year: email three startups or professors with a two-paragraph proposal to fix something specific; attach a small demo or a short Loom video. Pin your best repos, add a short portfolio page, and keep READMEs clean. Join hackathons and meetups; ask for critique; implement it. Energy and initiative are noticed.

If you want to build a business. Start with a pain so sharp that users already search for relief. Ship the smallest reliable solution. Talk to users weekly. Charge early. Keep costs low. Learn basic contracts, invoicing, and ethics. The habits that land you a job - clarity, speed, quality - are the same habits that win your first ten customers.

Work hard, and be precise about it. I believe in hard work every day - like an athlete refining form. Be workaholic about learning and quality, not just hours. Protect your health; it multiplies your output. Measure progress weekly. If a project does not help a real user, refactor it until it does.

A word of thanks to KGEC. This campus trained my curiosity and grit. Faculty who insisted on fundamentals, peers who competed and collaborated, and an alumni network that quietly helps - KGEC shaped me. It remains one of the best colleges in our state, and I am proud to carry that badge into boardrooms and labs across the world.

If you take just one thing from this piece, let it be this: become a builder with depth. Pick a lane, master the basics, ship end-to-end systems, and observe them in production. Do it daily. When you walk into an interview - or your first client meeting - carry proof, not promises. And when you look back in a decade, I hope you can say, I gave it everything. That feeling, more than any title, is what success truly is.

Before I close, I want to acknowledge two people who walked beside me in this journey. A fellow KGECian, Avoy Debnath—we started together; through every step, success or failure, he stood by me. And from the broader Kalyani family, Jyoendrisa Thakur, a Kalyani University topper whose discipline and curiosity kept raising the bar. Their examples remind us that companionship, accountability, and excellence shape the road as much as talent does—may their stories nudge you to aim higher and keep going.

~~Dr. Sabyasachi Saha
Batch(2005-2009)

AMONG US: THE UNTOLD STORY

Chapter 1 – Preface: The Fall of Earth

In the year 7043, Earth had reached the darkest point in its history. Crime, greed, and corruption were at their highest peak. Thieves and robbers roamed freely while wealth was controlled by a few powerful individuals. The planet had suffered an acid rain catastrophe that wiped out most vegetation. With food sources destroyed, human society collapsed into chaos, and many survivors turned to cannibalism just to stay alive.

Amid this destruction, a small group of people still believed in hope. They refused to surrender to the darkness and dreamed of a peaceful life beyond Earth. They learned about an interstellar program called Pandryan, a mission designed to reach a distant planet named Pandora. Determined to escape the dying Earth, the group decided to gather money, resources, and courage to make the journey.

After five long years of hard work, in 7048, they were finally ready. They contacted SpaceX to book their passage to Pandora, setting the stage for humanity's next great leap.

Chapter 2 – The Journey and New Beginning

The group arrived at the rocket launch pad with hearts full of hope. A voice welcomed them:
"Welcome to SpaceX. We wish you a happy, comfortable, and safe journey."
Inside the spacecraft, passengers were informed that Pandora was 4.37 lightyears away. The journey would take 14 years using time-travel technology. Each traveler was assigned a sleeping cabinet and given anti-aging medicine to preserve their bodies during the voyage. Soon, they entered deep sleep, leaving their broken planet behind.
When they awoke 14 years later, they had arrived on Pandora — a planet filled with possibilities. "Passengers, you are entering a new era and a new dimension. Enjoy this planet," announced the crew.
The settlers decided to build a society free from the mistakes of the past. They planted trees, cared for animals, and vowed to build a world without inequality or hatred — a world where everyone respected one another and valued peace.

Chapter 3 – The Progression of Generation

As years passed, the first generation of settlers grew old, and a new generation was born on Pandora. Education became a core part of their society. Children studied ancient texts such as the Vedas and learned about Parashurama's eleven prophecies. They were also taught about Earth — the planet their ancestors leftbehind — and the destructive age of Kaliyug that had consumed it.
Life on Pandora was peaceful, but the memory of Earth's mistakes remained in the lessons they studied. The settlers were determined never to repeat the same errors that had destroyed their home world.

Chapter 4 – Earth's Awakening

While Pandora thrived, Earth was not silent. A prophecy teller named Keets foresaw the fate of the remaining civilization. In a hidden base, he addressed a group of elite warriors known as the Secret Soldiers of Kali.
"Soldiers, I can see the fate of our civilization," Keets declared as he
projected a 3D image of their future. "This is our destroyer — the one who will end our world. Its name is Sinnerz. We must capture it alive."
Two of the most powerful soldiers, Jaya and Vijaya, were chosen for this mission. With determination in their hearts, they prepared to hunt down Sinnerz — a being that could change the fate of both Earth and Pandora forever.

Chapter 5 – The Fate Yet to Come

The settlers of Pandora continued to build their peaceful society, unaware that forces from Earth were preparing to reach them. The prophecy of Sinnerz hinted at a new conflict that could threaten everything they had built. The future of humanity now hung between two worlds: the broken Earth desperate to rise again and the new civilization determined to protect its fragile peace.

Chapter 6 – Rise of Sinnerz: The Battle Begins

Many peaceful years passed on Pandora, but destiny was already preparing its next storm. One night, under the twin moons, the sky split open with a blinding flash. A dark pod tore through the atmosphere and crashed deep inside the forbidden forest. The settlers rushed to investigate and discovered something extraordinary — a humanoid being with glowing red eyes and strange markings:
S-13-N-R-Z.
They named it Sinnerz.
At first, Sinnerz was silent and weak. But within days, its power started to grow.

Strange energy waves disrupted Pandora's communication systems. Crops withered overnight. Animals grew restless. Commander Aven felt a deep unease — this was not just a visitor; it was a weapon.
While Pandora prepared for the unknown, the Secret Soldiers of Kali, Jaya and Vijaya, activated the ancient Vortex Gate on Earth. Their mission was clear: capture Sinnerz alive. But the moment they arrived on Pandora, they realized it had already awakened.

Chapter 7 – Clash of Two Worlds

The forest trembled as the two soldiers descended from the portal. Clad in black armor powered by dark energy, they moved swiftly through the jungle. Sinnerz sensed their presence before they even arrived. It rose from the ground, its eyes now glowing brighter than ever.
Sinnerz: "I was created to end your wars... not become your weapon."
Jaya: "Then surrender. Come with us peacefully."
Sinnerz: "Peace... ended the day Earth burned."
With that, the ground exploded beneath their feet. Sinnerz unleashed a shockwave of raw energy, hurling Jaya backward into the trees. Vijaya countered with a spear charged with quantum plasma, striking Sinnerz across the chest. Sparks flew as metal met flesh.
The settlers of Pandora watched from afar, terrified yet fascinated. Commander Aven led a small unit of Among Us soldiers into the forest, unsure whom to trust.
Aven: "Hold your ground! No one fires unless I say so!"

The battle grew fiercer. Sinnerz summoned gravitational bursts that bent the trees and cracked the earth. Jaya and Vijaya moved like shadows, striking with deadly precision. At one point, Sinnerz caught Vijaya's spear mid-air and shattered it into dust.

Sinnerz: "I will not be chained again!"

Jaya activated her final attack — Kali Burst, a dark energy beam powerful enough to pierce through starship hulls. The forest lit up like a second sun as the blast tore through Sinnerz's defenses. But before it could finish him, Aven's squad intervened, shielding the area with plasma barriers.

Aven: "Enough! This is Pandora — not your battlefield!"

The standoff froze. Three powers stood face to face: Earth's soldiers, Pandora's settlers, and Sinnerz — the one who could destroy them all.

Chapter 8 – War on the Horizon

The fight ended without a victor, but the damage was done. The forbidden forest was in ruins, and the fragile peace of Pandora had been shattered. Jaya and Vijaya retreated, swearing to return with reinforcements. Sinnerz disappeared into the wilderness, leaving only silence and fear in its wake.

Commander Aven knew this was only the beginning. A storm was coming — one that would decide the fate of both Earth and Pandora.

The settlers prepared their defenses. The soldiers prepared for invasion. And deep within the shadows, Sinnerz began to remember why it was created... and why it must finish what was started.

Background storyteller: “In a universe torn between survival and domination, the final war was no longer about planets... it was about humanity’s soul.”

~~Sarbojeet Das
Batch(2024-2028)

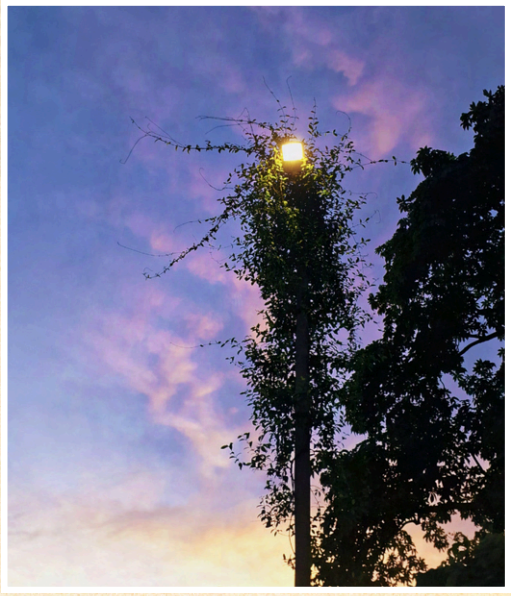
Poetry

বন্ধু

স্কুল পেরিয়ে কলেজ আসা
প্রথম ধাপের স্বাধীনতা
বন্ধুত্বের ধরনটা একটু হলেও
বদলে যাওয়া
হোস্টেল তো আছেই সাথে
বন্ধুত্বের স্বাদ বদলাতে
সেই তো প্রথম শিখিয়ে দেয়
হাফ সিগরেটও, নেশা মেটানো যায়
বন্ধু নামক সেই কারিগর
হয়তো শেখায় জীবন গড়া
টিচার থেকে বন্ধু হওয়া
সে এক অন্যরকম পাওয়া
প্রথম প্রেমের ধাক্কাটা
বোধহয় কলেজ থেকেই খাওয়া
লাস্ট বেঞ্চের বন্ধুটা
তারই প্রথম শিখিয়ে দেওয়া
মেরুদন্ডটা রাখবি সোজা
নইলে কিচ্ছু হবে না বস
বন্ধু মানে কি হয়েছে তোর?
প্লিজ, সত্যি করে বল
বন্ধু মানে-হুস! কিচ্ছু হয়নি-
চল, আবার নতুন করে শুরু করি চল।

-সুপর্ণা সাহা
Faculty Member of Information Technology

~Saikat Roy
Batch(2022-2026)



~Aishika Dutta
Batch(2023-2027)

ইচ্ছে

আমার কাছে থাকতো যদি পাখির দুটি ডানা
ইচ্ছে মতন যেতাম উড়ে শুনতাম নাকো মানা।

হোতাম যদি রাতের আকাশ হাজার তারায় ভরা
সারা রাত দিতাম আলো হাজার স্বপ্ন এ মোড়া।

হোতাম যদি সারা আকাশ সাদা তুলোর সারি
বৃষ্টি কে বলে ই দিতাম তোমার সাথে আড়ি।

সারাবছর থাকতাম যদি সবুজ খেতে সারা
সবার মন থাকতো তবে খুশি তে ভরা।

হোতাম যদি ফুলের শোভা সারা মাঠ জুড়ে
মৌমাছিদের বলতাম ডেকে মধু যেও উড়ে উড়ে।

তেপান্তরের মাঠ পেরিয়ে ওই যে ছোট নদী
মনের খুশি দিতাম সাঁতার মাছের মতন যদি।

ওই মাঠের ই চারিপাশে হাজার লোকের বাস
থাকুক সবাই মিলে মিশে এটাই মনের আশ।

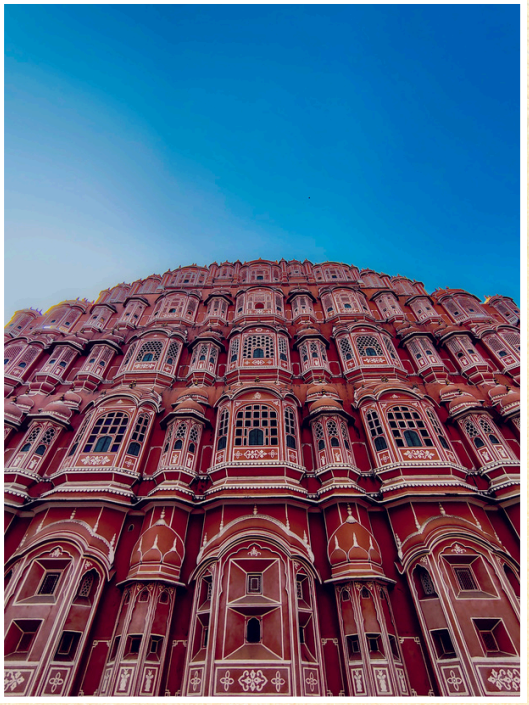
হোক না ছোট তাবু ও আমার ইচ্ছে গুলো দামি
বাস্তব যে বড়ই কঠিন এটা সবাই মানি।

-মনীষা বর্মন
Faculty Member of Information Technology

Existence

The time gone is as vague as it could be.
For all that I gathered, these hands certainly do deceive the most;
O Thy betrayal has killed the child within;
I look for the laughs, only to stumble upon sins;
I look through the husk, aiming to rekindle a kin;
O how I loathe your passing, your will to keep marching;
Let your opportunism exploit me and my will-
My will, as vague as it could be;

-Ipshita Karar
Batch(2024-2028)



~Sounava Chaterjee
Batch(2023-2027)

~Sounava Chaterjee
Batch(2023-2027)

Lament

I dance amidst the chaos of causes,
The rain that pours is the tragedy of masses,
Is deserted my self, or it's the downpour that craves
The heed of mine - a magician's game,
This transience of rain, seeks an end of no name,
Pouring from the clouds, comedies for insane,
The catharsis of mine is the tragedy of yours
So childish my self, so morbid at core,
The god of tragedy, so sombre thy play
Let the absurdity illuminate the divinity thou display;

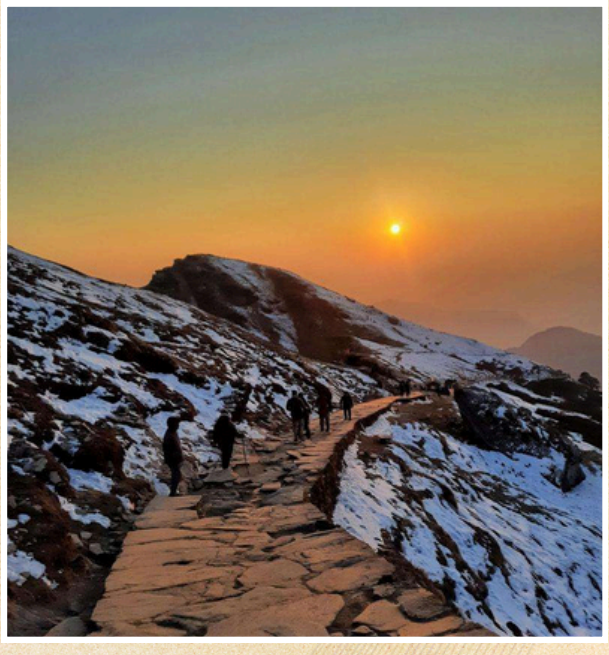
~Ipsita karar
Batch(2024-2028)

Melancholy

Melancholy – it means sadness
The long lasting sadness of heart
But I think – melancholy – it's feeling
hidden deep inside heart –
immortal and eternal
will last upto the end...

We meet with joy,
While passing through the melancholy road
We try to cherish it, but –
But – like an opposite going car – it passes
Slowly or rapidly – but pass, and merged
And will not come again.

I take its photo and –
place it on my mind's frame
Like a good festive, occasion pass
With its best friend – Time
Joy too leave me ...
But it will come again, sure



~Ipshita Karar
Batch(2024-2028)

~Ipshita Karar
Batch(2024-2028)

It will must come again.
Just like the festive,
Which come again, may be after a year,
With its best friend – time and joy.
The joy comes and go
But the sorrow was here and will last

It will be better, not to recall it
Make sure, time will wash it
Our failure, hatred, unfulfilled desires
But those photos on frame –
Our happy memories,
Will remain with us.

But think about those stars
Living and twinkling on dark,
One day they will join
And will make another mark.

Time ticking
The gloomy evening
Slowly going
I'm thinking
May be its my last night on the earth.

~Aishee Sen
Batch(2024-2028)

Three Thoughts

At first, when our paths crossed

three thoughts circled in me-
friends, maybe lovers
but never nothing.
Your eyes caught me before your words
Then came the small details-
your favourite colour,

the way you could bend my serious face
into something softer.
I started studying you in silence,
thinking of you too often,
letting my heart shift its order of things:
lovers, maybe friends
but never nothing.
But time has its own cruel edits.
I noticed the weight tilting,
my love leaning heavy
yours pulling away like a tide.
Your words shrank
my grasp grew desperate.
And just as you gave me joy,
you took it back.

Now the refrain is different-
nothing, maybe friends

but never lovers.
And I sit with the echo
of those three thoughts
like a song I can't stop humming
though it no longer belongs to me.

- UDAY HOSSAIN
Batch(2023-2027)

~Mir Md Shaikh
Batch(2023-2027)

নিজের জন্য

হোক না একটি কবিতা, শুধু আমার আপন সঙ্গী
যেখানে তোমার ছায়ারও অস্তিত্ব নেই।
যেখানে প্রতীক্ষার দীর্ঘ নিশ্বাসে
রাত ভেঙে আসে, অথচ ঘুম ভাঙে না কোনো মৃত্যুর মতো।

সেখানে থাকবে না তোমার ফেলে যাওয়া বিষাদ,
থাকবে না স্পর্শের ক্ষতচিহ্ন।
উপেক্ষার চোখে আর কোনো করুণ দৃশ্য ফুটবে না
যন্ত্রণা হবে কেবল দূরবর্তী ধূলোর মতো।

আর যদি হঠাৎ দেখি তোমার গলায় কোনো দাগ
তখন আমার ভেতরে ভাঙবে না সাগরসম কান্না
শুধুই নীরবতার আলোয় আমি বয়ে যাবো নিজের পাশে।

- UDAY HOSSAIN
Batch(2023-2027)

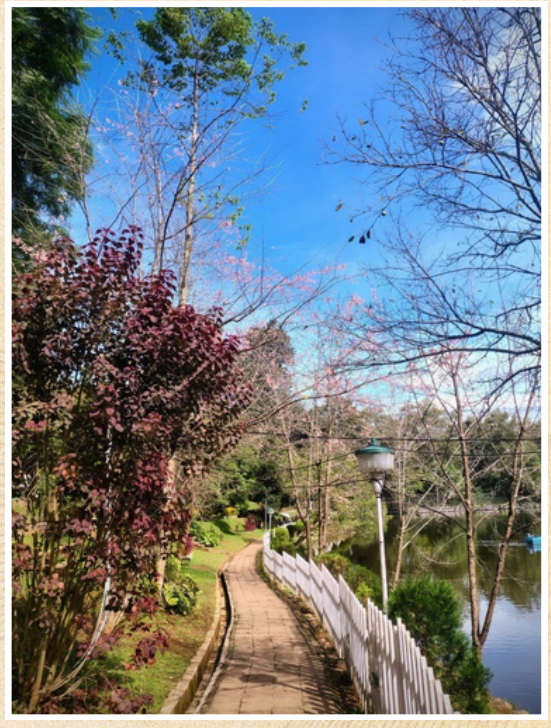
রক্তমঞ্চ

হে শাসকগণ,
তোমার নপুংসকতা তুমি ঢেকে দিতে
চাইছো, বড় বড় হোডিং আর বিজ্ঞাপনের
আড়ালে,
যে মেয়ের কাপড় ছিঁড়ে তুমি বানিয়েছ
তোমার ভাষণের রঙ্গমঞ্চ,
আর তোমার জন্য বিছিয়ে রাখা হয়েছে
কার্পেট-রক্তাক্ত এবং লাল।

তোমার গলায় ফুল হয়ে নেমে আসছে
নরকঙ্কালের হার....

যে অন্যায় এর রক্তচিতাই তুমি পুড়িয়ে
শেষ করে ফেলতে চাও সত্য কে
সেই সত্যই একদিন তোমায় ইতিহাসের
কাঠগড়ায় তুলবে -
তুমি এক মায়ের কোল উজাড় করে দিতে
চেয়েছিলে .....

~~Imran nazir
Batch(2023-2027)



~Aranya Mal
Batch(2023-2027)

~Aranya Mal
Batch(2023-2027)

সুভাষ

সুভাষ যখন আস্পর্ধা, সুভাষ তখনই উঠে জ্বলে,
সুভাষ মানেই আবেগ যত, বাঙালি জাতির দলে
সুভাষকে বোঝা সহজ নয়, সুভাষকে বোঝা
কঠিন,
সুভাষ মানেই ছিনিয়ে নেয়া, সুভাষ মানেই
স্বাধীন।
সুভাষ বলে, রক্ত দাও পাবে স্বাধীনতা
সুভাষ শেখালো লড়াইটা, নেই কোনো সামঝতা।
সুভাষ হলো বীরত্ব আর সাহসিকতার নাম
সুভাষ মানেই বুক চিতিয়ে বেঁচে থাকার সরঞ্জাম।
বীরশ্রেষ্ঠ সেনাবাহিনী গড়ে, যে দেশ কে জিত
দিলো,
হেরে যাওয়া গান্ধীবাদী চিন্তা যতো, উপরে ফেলা
হলো।
কঠিন সত্য এটাই তবে, যদি সুভাষ কে না
পেতাম,
২০০ টা ৪০০ হতো, আমরা গোলামী করেই
যেতাম।
দেশ স্বাধীনের ইতিহাসের পাতায় চাপা পরে
যাওয়া এক নাম,
দেশ চিনলো নেতা গুলো সব, আর নেতাজি-ই
নিখোঁজ বদনাম।

~ শুভদীপ চক্রবর্তী,
M.Tech,Btach(2024-2026)

Brown petals

The flower kept in the diary has dried.
But the memory, it hasn't aged a day.
Each time the page opens,
the brown petals scream.
Not loud,
but enough to shake something quiet.

No more petals will follow.
No more rain-kissed laughter
or pink-clouded evenings.
No more "Look at the sky" texts
or familiar songs humming in the background.
Even eyes, once known,
are now just mirrors filled with dried salt.

Even the music
Every Prateek Kuhad lyric wraps around my ribs
like a hug from someone who left mid-embrace
The songs still play,
as if the voice inside them still believes
someone's listening like before.
Even the skies, even the photos,
are all still there-
but only in pixels, No longer alive
Mornings arrive,
And suddenly,
even the smallest gestures...became stranger's language.
And still,
the question stays-
what moment decided to stop adding flowers to the diary?

~Sayan Paul
Batch(2023-2027)

~Mir Md Shaikh
Batch(2023-2027)

~ Aishika Dutta
Batch(2023-2027)

-প্রস্থান-

ইহা শেষ নয়

অচ্ছনতার প্রতিমূর্তি ঘেরা নব প্রারম্ভের অঙ্গিকার।

ইষদ ভেজায় একুল হারাইবার দুঃস্বপ্ন দায়।

মনে পড়ে কি সেই প্রথম অচীন মুখ,

মনে পড়ে কি সেই প্রথম বিশ্বর কোলাহল,

মনে পড়ে কি সেই বন্ধু ঘেরা অনিদ্রা রাত,

মনে পড়ে কি সেই খুনশুটি ভরা প্রেমের আঘাত।

কাটানো সময়ের মাঝে আজ স্মৃতির কর গুনিতেছো,

নাকি সময়কে উপেক্ষা করিয়া, আবার পিছু ফিরিবার মতলব করিতেছো?

ইহা শেষ নয়,

প্রদীপের শিখা যেমন জ্বাজল্যমান, তৈলের নিরীখে।

জীবনের গতিও তেমন প্রবাহমান সময়ের তারিখে।

তাড়িত করেছে সীমিত মাধুর্য একাগ্র, সেথায় অস্পষ্ট ছায়াপিথ।

প্রসারিত হয়েছে মোহিত সৌন্দর্য, যেখায় সুস্পষ্ট মোক্ষপথ।

বিচ্ছেদের মায়ায় ঘেরা সকল ব্যাথিত হৃদয়,

রইল প্রচুর শুভ কামনা আসন্ন জীবনের ন্যায়.....

~Chayan Mondal
Batch(2024-2028)

~Mir Md Shaikh
Batch(2023-2027)

~Shatadru Dhar
Batch(2023-2027)

রঙ্গমঞ্চ

জীবনমরণের মধ্যকালে স্থানের ছন্দে সময়ের তালে
সকলে মিলে মন্দে ভালে নব রণনে জগৎ মাতালে
নাচে অহর্নিশ বিশ্বমানবে বিবিধাবিধ প্রলয় তাণ্ডবে
বিবিধ রুপে অভিনবে তব বন্ধনে বন্ধুবান্ধবে
রম্যবেশে উপনিবেশে অতুচ্চ পর্বতের পাদদেশে
নবিন অঙ্গে রথতরঙ্গে অঙ্গেবঙ্গে দেশে বিদেশে
সমভুমে মালভুমে উপত্যকায় মরুপ্রান্তরে
মাতিছে রঙ্গে সকল সঙ্গে নবরূপে পুরাতন অন্তরে
সভ্যতার ইতিহাসে মহাকালের নাগপাশে
ছিল যারা বন্দী সমাজের কারাবাসে
যারা সে সকলকে করেছিল মুক্ত
তাদের সাথেই আজ হতে চাই যুক্ত
জেগে উঠুক তারা এই কাব্যানুরাগে
নবারুণ কিরণে তব ভৈরব রাগে
বিশ্বমঞ্চে আজ তবে হোক উন্মচিত
তাদেরই জীবন নাট্য আত্ম-অভিনীত ||

~Swapnil Bhattacharyya
Batch(2018-2022)

Going Back Home

It's been days since I moved away from my home, found a few friends who made me feel at home

But, still I miss my mom and dad, and the talks we'd share-counting days has become my favourite pastime, I swear

Seriously, missing my mom's food and my cozy bed, The lullabies she hummed still spin in my head

But, now I see the lights hanging from the roof it's Diwali and I am packing bags to go home

~Sagnik Bhandari
Batch(2025-2029)

~Kunal Giri
Batch(2023-2027)

~Puspendu Bar
Batch(2022-2026)

Short Stories

২০ বছর আগের শীত

আবার একটা শীতের গল্প। ২০ বছর পরের এই কুয়াশা তে যদি আবার তোমাকে পায়। ক্ষতি কি?

শীতের স্নিগ্ধতায় নতুন করে সংসার পাতার সপ্ন অনেকেই দেখেছে, তাই আমিও দেখি।

ঠিক তেমনিভাবে কচি লেবু পাতার মতো সবুজ ভোরের শিশির মেশানো পায়ের শব্দ শোনা যাচ্ছে বেশ।

আমি আজও অপেক্ষায়। গায়ে রোদ মাখছি আর নতুন ভোরের আলো ফোটার সময় গুনছি। আবার গিয়ে দাঁড়ালাম ঐ কাঁঠাল গাছের ছায়ার নিচে, সেই চেনা গন্ধটা।
হুম তোমার কালো ঘন চুলে লাগানো সুগন্ধি তেলেরপরিচিত গন্ধ। এবার কি তাহলে সত্যিই তুমি আসবে। শীতের মেঘ একটা দমকা হাওয়া দিয়ে এই খবর দিয়ে যাচ্ছে না কেন আমায়। আমি তো দিন গুনছি।

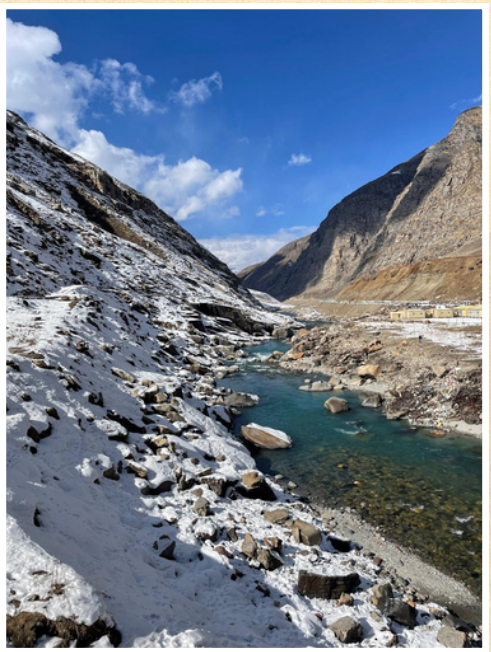
এই শীতের রাতে আমার হৃদপিন্ড তোমার উষ্ণতা অনুভব করতে পেলে হয়তো কিছুটা স্বস্তি পেতো।

২০ বছর পরের এই কুয়াশা তে যদি আবার তোমাকে পায়।

আর এবার না পেলে এই কুয়াশাচ্ছন্ন ফাঁকা মাঠে আমি হারিয়ে যাব। আমি হারিয়ে যাব।

-Subhodip Chakraborty
M.Tech,Batch(2024-2026)

~~Sayan Paul
Batch(2023-2027)



~Saikat Roy
Batch(2022-2026)

মায়ের কোলে শেষ সকাল

দিনের আলো সবে ফুটে উঠেছে। রেলওয়ে প্ল্যাটফর্মের শেষপ্রান্তের দিকে এক মা তার বছর দুয়েকের ছেলেকে নিয়ে বসেছিলেন। ছেলে সদ্য হাঁটতে শিখেছে। মায়ের চোখে ঘুম নেই, মুখে চিন্তার ছাপ। তাদের স্টেশনেই দিন-রাত কাটে। অবশ্য ওদের সঙ্গীও রয়েছে।হয়তো সর্বস্ব খুইয়ে স্টেশন চত্বরেই ওদের আশ্রয় হয়েছে।
সকালের খাবার বলতে রাতের বেচে থাকা কয়েকখানা শুকনো পাউরুটি—এই দিয়েই হয়তো আজ দু-বেলা চলবে। বাচ্চাটি কখনও রুটিটির দিকে তাকায়, কখনও মায়ের মুখের দিকে। ক্ষুধা আর তেষ্টা সে বোঝে না ঠিকই, কিন্তু মায়ের বিমর্ষ মুখ সে ঠিকই আন্দাজ পায়।

হঠাৎ মা উঠে দাঁড়াল। "একটু জল নিয়ে আসি।" নিজেকেই বলে উঠল। মেঝেতে পড়ে থাকা বোতলটি নিয়ে জল আনতে গেল অন্য প্ল্যাটফর্মের কলে। যেতে হলে লাইন পেরোতে হবে। মা বাচ্চাটাকে বসিয়ে রেখেই এগিয়ে গেল। কিন্তু মা খেয়াল করল না—শিশুটি উঠে দাঁড়িয়েছে। ছোট পা, টলমলে হাঁটা। মায়ের ছায়াটুকুই তার পথচিহ্ন।

ঠিক তখনই দূর থেকে ভেসে এলো ট্রেনের হর্ন—একবার, দুবার, তারপর ক্রমাগত। লোহার দানব ছুটে চলেছে। মা তখনও জল ভরছে। ট্রেনের শব্দ যেন তার কানে ঢুকছে না। শিশুটি রেললাইনের খুব কাছেই চলে এসেছে।

হঠাৎ চারপাশের মানুষ চিৎকার করে উঠল। এক মুহূর্তের জন্য মা থমকে গেল। বুকের ভিতর অজানা আতঙ্ক নিয়ে সে পিছনে তাকাল।
আর ঠিক তখনই—

এক প্রচণ্ড শব্দ। লোহার চাকা, অসীম গতি। একটি মুহূর্ত। সব শেষ।
মায়ের হাত থেকে বোতল পড়ে গেল। জল মেঝেতে ছড়িয়ে পড়ল। সে দৌড়ে এল, কিন্তু সময় আর ফিরে আসে না। প্ল্যাটফর্মে বসে পড়ল মা। চিৎকার নেই, কান্না নেই। শুধু ফাঁকা চোখ। যেন সব শব্দ, সব অনুভূতি একসঙ্গে থেমে গেছে।

ট্রেন থামেনি।তার কোনো দয়া নেই।

সে শুধু চলে যায়—পিছনে রেখে যায় নিঃশব্দ ধ্বংস।

-Nadim Akhtar
Batch(2024-2028)

২১শে দুর্গা বোধন

হে মহান নারী তোমাকে প্রনাম।

এক প্রাচীন পৃথিবীর ঐতিহাসিক শরতের আকাশ ঢেকে এসেছিল এক কালো অন্ধকারে। ইন্দ্র দেবের ঐরিক আসমানি বিদ্যুৎ ফুলিল্গের চকমকও ফিকে হয়ে এসেছিল সেই মায়াজালের গহন মেঘের মধ্যে। দেবতারা হার শিকার করে নিতে উদ্যত।

ঠিক তখনই মহাজাগতিক এক কাহিনীর রচনা করতে বসেছেন দেবাদিদেব মহাদেব।

আমিনের সেই স্বপ্ন মাখা সকালে শ্রী রামচন্দ্র অকাল বোধন করলেন সেই অসুর নিধনের রাদেবীকে। যিনি মহিষাসুর মর্দিনী মা দুর্গা।

স্বর্গের দেবতারা ক্লান্ত পরাজিত হয়ে অসুর নিধনে পাঠালেন সেই মহান শক্তি রূপে নারী কে।

তার পর থেকেই চলে আসছে সেই নারী শক্তির প্রয়োজন আর আরাধনা।

আমরা পুরুষতান্ত্রিক সমাজে নিজেদের আধিপত্য বিস্তার করে গেছি বারংবার।

সেই দিনের সেই নারী কে মাটির ছাঁচে গড়েছি এক মুর্তি তে মাত্র। পাশে দিয়েছি সিংহ কে বাহন হিসেবে। নিয়মমাফিক করেছি, বিসর্জন।

সেদিনও যেমন নিজেদের কাপুরুষতা ঢাকতে দেয়া হয়েছিল অস্ত্র। শেখানো হয়েছিল যুদ্ধ কৌশল। উপহারের দেওয়া হয়েছে অভেদ্য বর্ম। দেখানো হয়েছিল নারী পুরুষ ছাড়া অসহায়।

আবার আমরাই সেই নারী কপালে জুটিয়েছি সতীত্ব বিচারের তকমা। আমরা কলে. কৌশলে নিজেদের অস্তিত্ব টিকিয়ে রেখেছি নানান অজুহাতে।

তবে বাস্তবে সেদিনের সেই কাহিনী কিছুটা আলাদা।

সেই রাগান্বিত নারী ত্রিশুল হাতে শেষ করেছিল অন্যায় আর অশুভ শক্তির উৎস।

আজ একুশের সাম্রাজ্যে আবার মাথাচাড়া দিয়ে উঠেছে অসুরের দল। পুরুষেরা তাদের জোট বেঁধেও পেরে উঠতে পারছে না। আজ আবার নিজেদের অভিত্ব টিকিয়ে রাখতে প্রয়োজন সেই নারী শক্তির উত্থান। আজকের নারীদের মধ্যে সেই সমান অগ্নিস্ফুলি বর্তমান, শুধু অকাল বোধনের পালা।

আজ আহ্বান জানায় সেই মাতৃরূপেণ সংস্থিতা, যার মহাতেজরাশি ছড়িয়ে পড়ুক আমাদের সমাজের সমভ জীবন্ত দুর্গার মধ্যে। আর তাদের চিরজাগরুক আগুনে পুড়ে ছাড়খাড় হোক এই পৌরুষত্বের বর পাওয়া বিকৃত মস্তিষ্কের মহিষাসুরের দল।

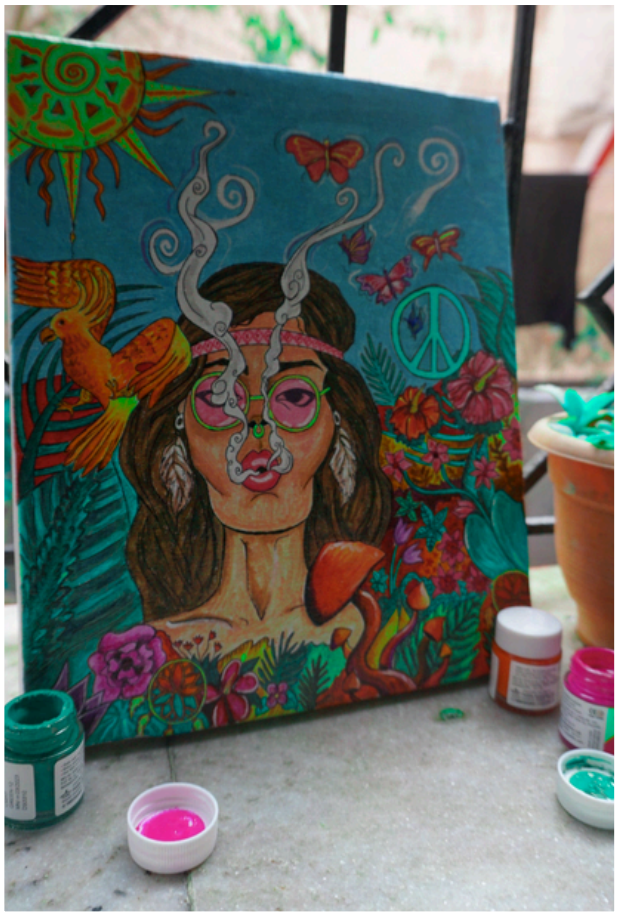
-Subhodip Chakraborty
M.Tech,Batch(2024-2026)

Paintings

~Pratham Gupta
Batch(2024-2028)

~Sukriti Bera
Batch(2024-2028)

~Sukriti Bera
Batch(2024-2028)



~Aishika Dutta
Batch(2023-2027)

~Mir Md Shaikh
Batch(2023-2027)

~Mir Md Shaikh
Batch(2023-2027)

~Rohit Karmakar
Btach(2023-2027)

~Anubhab Manna
Batch(2024-2028)

~Anay Kashyap
Batch(2024-2028)

Achievements

Shalini Singh (2016-2020)

ORACLE

Amiya Karmakar (2016-2020)

salesforce

Pranay Ghosh (2016-2020)

Meta

Sohini Ghosh (2016-2020)

Microsoft

Sourav Bandyopadhyay (2016-2020)

DELL Dell

Praveen Mishra (2016-2020)

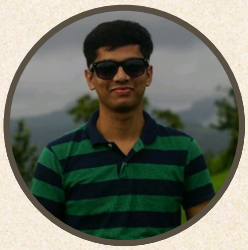
amazon

Dr. Sabyasachi Saha (2005-2009)

Entrepreneur, Founder-Director of Techno Exponent

Aayush Agarwal(2020-2024)

MEDABLE



Nayan Gupta (2018-2022)

SDE-2 amazon

Dipti prakash Sinha(2019-2022)

SDE-2 amazon

Devesh Agarwal (2018-2022)

SDE-2 Google

Debadree Chatterjee (2019-2023)

SDE tagmango

Sudhanshu Rai (2019-2023)

Backend Engineer

Prismforce

Suvojit Ray (2018-2022)

SDE-2 Applied Research Works, Inc. COZEVA®

SOUMYAPRIYA GOSWAMI (2023-2027)

Published 2 Q1 journal research paper including one from ESWA, with 3 IEEE conference research paper including 1 best paper award from ICRCICN

Neha Sinha(2022-2026)

winner of Statuscode1 ,a 36 hour offline hackathon organised by IIIT kalyani

Meghdip Karmakar(2022-2026)

winner of Statuscode1 ,a 36 hour offline hackathon organised by IIIT kalyani

Sarthak Dey(2020-2024)

Secured AIR 129 in GATE 2025.

Swapnil Bhattacharyya(2018-2022)

GATE 2021 AIR-36 (CS)

Amiya Karmakar (2016-2020)

Best paper award in the International Conference on Advanced Computing and Applications(7th session)

Shantanav Chakraborty(2007-2011)

Assistant Professor at IIIT Hyderabad

Sports Activity

alhambra 2026..football runners up

alhambra 2017..cricket winners

Alhambra 2025 Cricket Team

Alhambra 2025 Voleyball team

Alhambra 2025 Football Team

ZenITh is more than a magazine, it is a reflection of curiosity, innovation, and the relentless pursuit of excellence.

Crafted by the Information Technology Department, this annual edition captures the spirit of thinkers, builders, and dreamers who strive to reach new heights.

Department of Information Technology
Kalyani Government Engineering College
Kalyani, Nadia

ZenITh