acm - CEG
Student Chapter

OCT-NOV 2025
ISSUE #4

Stack'D

The Monthly Newsletter

What's inside?

JIT COMPILATION

How runtimes optimize and accelerate apps on the fly

MORE MEGAPIXELS ≠ BETTER PHOTOS

Why sensor size, lens quality, and processing matter more than megapixel count

WINDOWS SHORTCUTS

Boost productivity with shortcuts you weren't aware of earlier

INTERN INSIGHTS

A summer at Appian taught coding, teamwork, and the value of seizing every opportunity.



AND MORE...

A QUICK REWIND

SEPTEMBER



Masterclass

The second part of our Masterclass series was conducted by Rajeshwaran P, who shared insightful guidance on preparing for GATE, pursuing master's programs, exploring research opportunities, and learning about SPARKS and related initiatives. Participants shared their thoughts as well, making the session interactive and highly impactful.

Meme-O-Mania (3rd week)

A fun tech-meme fest where participants shared witty and relatable tech memes on their stories. The event brought out everyone's humour and creativity, making it a light and enjoyable break from the usual competitive events.

OCTOBER

Offline Meetup – II

We kicked off the month with our second offline meetup, bringing together our OBs, SCs, and JCs under the same umbrella. The session covered discussions on our club's upcoming activities, mainly Prodigy '25. We wrapped up the day with a fun inter-domain reel that added a playful spark to the meetup.

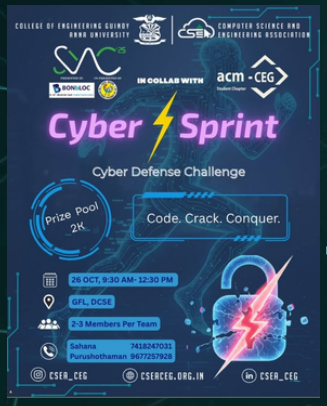


Codex

Our flagship coding series returned with its brand new 3rd edition. The first contest of this season featured Sherlock Holmes–themed problems, drawing participation from multiple colleges. From clues to clever logic, participants cracked the logic and coded their way through an exciting, mystery-styled challenge.

CyberSprint

A brand-new event introduced this year, CyberSprint challenged participants on concepts across information security, cybersecurity, and code-breaking. Held as part of SYNC, the Computer Science Department's Intra-college Symposium, the event pushed students to think fast, decode quicker, and stay sharp under pressure.



Bid & Build

Conducted as part of SAAS's Agni '25, Bid & Build combined quizzing, strategic bidding, and hands-on building. The event challenged participants to think creatively, collaborate effectively, and innovate under pressure, perfectly blending knowledge, strategy, and teamwork.



Prodigy'25

This year, Prodigy made a strong comeback with enthusiastic participation from students across various schools. Events like Synapse, Ideathon, Family Feud, and other interactive competitions kept the energy high throughout the day. We also hosted a Gen-AI workshop, giving students hands-on exposure to the rapidly evolving world of artificial intelligence.

NOVEMBER

Code Relay

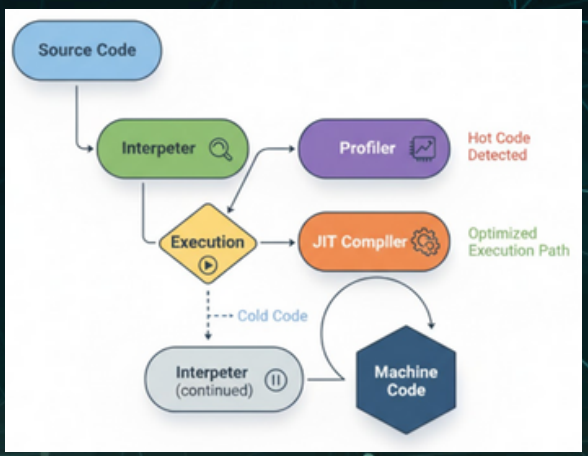
As part of the i++, IT Department's Intra-college Symposium, Code Relay highlighted the best of teamwork and problem-solving. Teams worked in perfect sync to navigate through coding challenges, showcasing both speed and coordination.

TECH DIGEST

JIT COMPILATION: THE SMART ENGINE BEHIND MODERN RUNTIMES

When we talk about why programs run fast, we often think of better processors or efficient code. But another important part of that story is how the code is executed. Traditional programming languages either interpret code as it runs or compile everything to machine code in advance. Between these two approaches lies Just-in-Time (JIT) compilation. JIT begins by interpreting the code so the program can start quickly. As execution continues, it identifies parts of the program that run repeatedly. These "hot spots" are then compiled into machine code while the program is running, allowing those sections to execute much faster.

How JIT works?



When an application launches, execution starts in a basic interpreter.

While the program runs, the profiler, a component inside the JIT runtime, observes how the program behaves at runtime. It handles the bookkeeping work by monitoring code blocks, counting how often they run, and collecting invocation data. It tracks information such as which methods are called most often, which branches are taken more frequently, object allocation patterns, exception frequency, and how many times loops execute.

Based on this data, the JIT compiler optimizes the code blocks that cross a certain threshold and translates them into machine code. Over time, as the program continues to run, its behaviour may change as functions are used differently or execution patterns shift. JIT takes these changes into account and re-optimises code when needed, or de-optimises it if previous assumptions no longer hold. This is known as Adaptive Optimisation.

Where JIT Is Used?

- JIT isn't limited to a single language. It plays a major role across modern software ecosystems.
- The Java Virtual Machine (JVM) uses it for large-scale enterprise software.

- JavaScript engines like Chrome's V8 rely on it to keep web applications fast and responsive.
- PyPy uses JIT to speed up Python execution.
- The Android Runtime (ART) uses it to improve mobile app performance without draining battery.
- The .NET runtime uses JIT to make C# applications both portable and efficient.

Trade-offs of JIT Compilation

While JIT compilation significantly improves performance, it comes with a few trade-offs.

- Slower Startup (Warm-up Time): At program launch, the JIT compiler doesn't yet know which parts of the code are most important. It must spend time profiling before it can optimise anything, which makes the startup slower compared to pure interpreted execution.
- More Memory Usage: JIT needs to keep several things in memory, including the original bytecode, profiling data, and the optimised machine code it generates. This results in higher memory consumption than static compilation.
- Not Ideal for Short Programs: If a program runs for only a few seconds, JIT has no time to collect meaningful profiling data. The warm-up cost and additional memory usage occur, but the performance benefits are never realised.

Conclusion

JIT compilation has become a crucial technique for modern programming languages to balance flexibility with speed. By focusing on the code paths that matter most, it allows applications to run faster without sacrificing the advantages of interpretation. This approach is a major reason why languages like JavaScript, Java, Python (via PyPy), and C# perform so well in real-world scenarios. For long-running systems, JIT is often the reason performance improves over time rather than degrades. It is one of the quiet, powerful optimisations that shapes the modern computing experience and keeps our applications fast and responsive.

BUST THE MYTH

MYTH: HIGHER MEGAPIXELS = BETTER CAMERA

Many people assume that a 108-megapixel camera automatically takes better photos than a 12-megapixel camera. It sounds logical but more megapixels should mean more detail, right? In reality, megapixels alone don't define image quality.

The megapixel count only tells you how many tiny pixels make up an image. It's like saying a painting with more dots automatically looks better which isn't true if the paint quality, lighting, or artist's skill isn't good.

What actually matters more are these three key factors:

1. **Sensor Size**
   - The image sensor captures light and converts it into data. A larger sensor can gather more light per pixel, leading to better low-light performance, higher dynamic range (detail in bright and dark areas), less noise (grainy texture).
   - Example: A 12MP camera with a large sensor (like on the Google Pixel 5 or Samsung Galaxy S21) often outperforms a 48MP or 108MP camera with a small sensor (found in budget phones like the Redmi Note 10 Pro or Infinix Zero Ultra).
2. **Lens Quality**
   - A high-megapixel sensor is useless if the lens that captures light is low quality. Good lenses provide sharper focus, better colour accuracy, minimal distortion and glare.
   - That's why professional cameras with 24MP sensors and premium lenses produce stunning shots far superior to many high-MP phone cameras.
3. **Image Processing**
   - After capturing light, your phone's image processor (ISP) and software algorithms handle colour balance, sharpness, and noise reduction.
   - Companies like Apple and Google use powerful AI-driven image processing to enhance detail, tone, and contrast, often making 12MP images look more natural and vibrant than 108MP ones.

What More Megapixels Are Actually Good For?

- Cropping or zooming without losing quality
- Large-format printing (billboards, posters)
- Professional photography where ultra-high detail is required

acm -CEG
Student Chapter

CRAFTING SOLU
SPE



LUMOS AND NOX: LIGHTING UP RURAL INDIA WITH TECH MAGIC

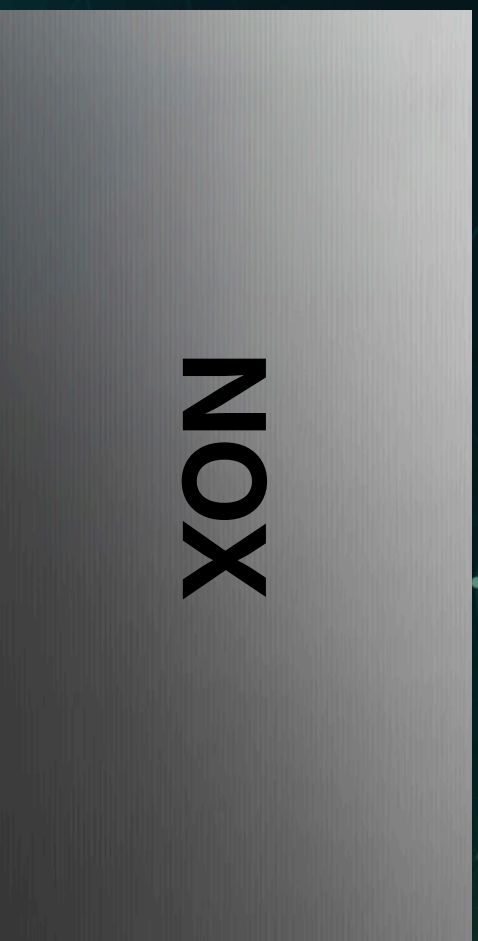
In this month's episode, the spells Lumos and Nox inspire a journey from darkness to light not through wands, but through technology. Even today, rural India faces power shortages, poor digital connectivity, and limited awareness of government schemes. What if Lumos could mean light through innovation, and Nox could mean control over misinformation and inefficiency?

Project Lumos envisions AI, IoT, and Blockchain powered smart solar microgrids for rural communities. IoT sensors track solar panel output, battery health, and energy usage in real time. Machine learning models forecast demand patterns and detect faults before they happen, ensuring uninterrupted electricity. Every watt generated and consumed is securely recorded on a blockchain

LUMOS

UTIONS OUT OF
ELLS

ledger, bringing transparency to energy distribution and preventing misuse of rural electrification funds. Villagers could trade excess power through energy tokens, promoting community-driven sustainability.
Meanwhile, Project Nox focuses on dispelling informational darkness. AI-driven offline learning pods—low-cost Raspberry Pi servers—could store educational videos, healthcare awareness material, and skill-based tutorials in regional languages. Natural Language Processing (NLP) tools can filter fake news and ensure the spread of authentic information through local networks and community WhatsApp groups.



Steps Students Can Take

1. Build Your Own "Lumos" Prototype

Start small — you can create an IoT-based solar monitoring system using Arduino or Raspberry Pi. Connect sensors to measure voltage, current, and sunlight intensity, then display real-time data on a simple dashboard. This hands-on project helps you understand how smart grids work and how renewable energy can be monitored efficiently.

2. Train an AI Model for Prediction

Use publicly available datasets to train machine learning models that predict energy demand, detect faulty panels, or even identify misinformation in regional languages. This step helps you explore the power of AI for social good, blending data and purpose.

3. Simulate Energy Transparency with Blockchain

Experiment by designing a simple blockchain ledger to record energy generation and distribution. You can simulate how every watt produced is logged securely, ensuring transparent and tamper-proof energy transactions. This introduces you to decentralized systems that promote trust and accountability.



4. Take Tech Beyond the Classroom

Collaborate with local NGOs, rural schools, or student innovation clubs to deploy small-scale offline learning pods or AI-powered awareness tools. Use low-cost devices like Raspberry Pi to bring digital content and verified information to areas with limited internet access.

acm - CEG
Student Chapter

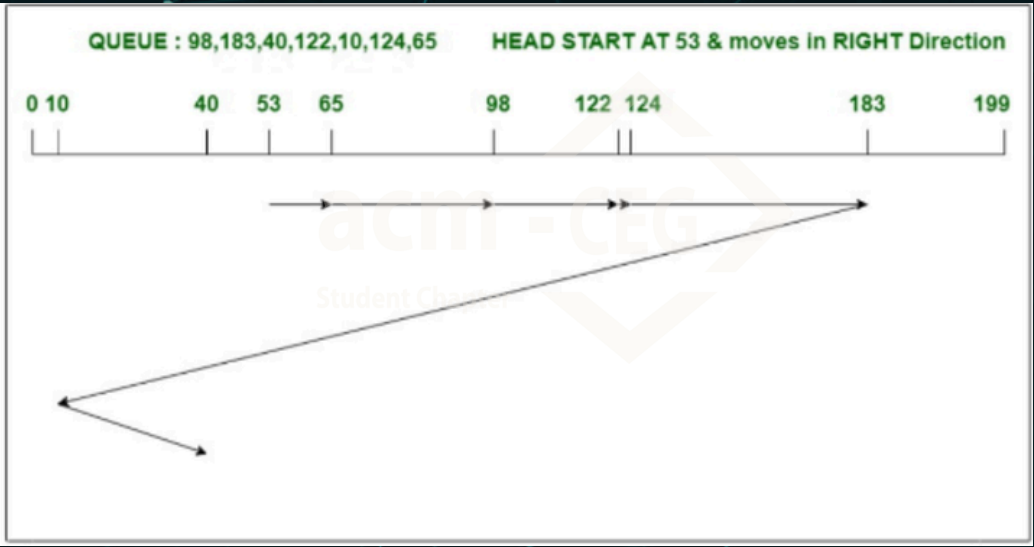
Each of these steps is a spell in itself — simple yet powerful. Through Lumos and Nox, you can use your technical skills to light up homes, minds, and futures — proving that the real magic lies in innovation fueled by empathy.

ALGO OF THE MONTH

Ever stood waiting for a lift and wondered, "How does this thing decide which floor to visit first when multiple people press the button at the same time?"

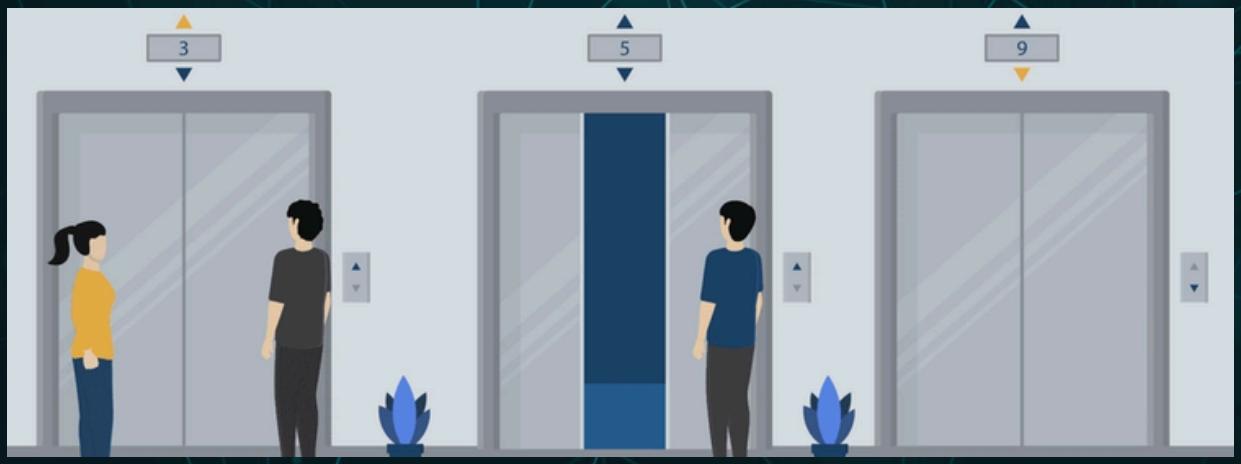
Surprisingly, that question brings us to the world of Scheduling. It is the very concept that decides which process your CPU runs, or how a hard disk reads data efficiently. The SCAN algorithm, a scheduling classic that actually appeared in elevators long before computers borrowed it.

How does SCAN work?



Picture the lift sitting idle on a floor. If only one person calls it, it simply goes there, picks them up, and goes to the requested floor.
But when multiple floors request the lift simultaneously, things get interesting, and that's where SCAN steps in:

- The elevator starts moving in one direction (say, upward).
- It stops at every floor in that direction where someone has requested the lift.
- Once it reaches the highest requested floor, it reverses and begins serving requests in the opposite direction.



Though SCAN forms the backbone of elevator scheduling even now, the modern lifts have their own nuances. With sensors, predictive algorithms, load detection, and AI-driven control, they've evolved into far smarter and more adaptive.

So the next time you're waiting for a lift, hopefully you won't be frustrated, but instead you'll understand why it takes a little longer.

acm CEG
Student Chapter

TIP OF THE MONTH

Your Windows computer can be used far more effectively with these lesser-known shortcuts. Try them out and unlock a bit of hidden magic.

INTERESTING WINDOWS KEYBOARD SHORTCUTS

- C**trl + Shift + Esc** - Opens the Task Manager, letting you view all running processes along with CPU, GPU, RAM, Disk, and Network usage.
- **Windows + V** - Displays your Clipboard History, showing everything you recently copied.
- **Windows + . (Period)** - Opens the Emoji & Symbol Keyboard for quick emojis and special characters.
- **Windows + X** - Opens the Power User Menu with advanced tools like Disk Management, Computer Management, Device Manager, and more.
- **Windows + Pause/Break -** Instantly opens the System / About page in Settings.
- **Windows + Alt + R -** Starts a Screen Recording using the Xbox Game Bar.
- **Shift + Delete -** Permanently deletes a file or folder without sending it to the Recycle Bin.



CAREER CORNER



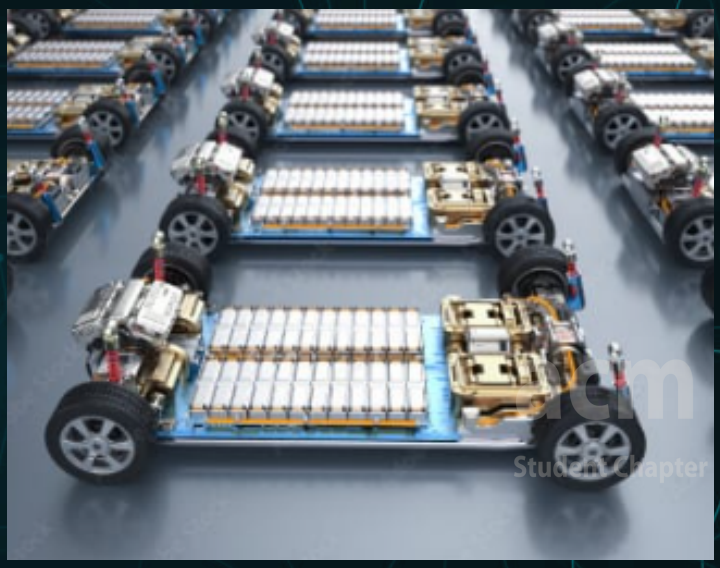
I am Madhubaalika, a final-year CSE student. I had the opportunity to intern as a Software Development Engineer (SDE) for 2 months at Appian. During my two-month internship at Appian, I gained valuable technical and professional exposure. The program began with orientation, where I received hands-on training in Git, set up my workspace, and learned about company policies and culture. I joined the Records Powered UI Squad, working with Java, React, and SAIL (Appian's proprietary language). My tasks ranged from fixing bugs in React to enhancing the UI of low-code components, which gave me first-hand experience in analysing feasibility before implementing solutions. Unlike academic projects, real-time development required me to think critically about every decision and collaborate closely with senior professionals, who mentored me throughout. I also participated in daily standups, Agile meetings, speaker sessions with global leaders, team outings, and even presented my work to all Appian employees.

The internship concluded with an intern hackathon, where I collaborated in a fast-paced environment, applying what I had learned. Overall, this journey equipped me with stronger technical expertise and taught me essential workplace skills such as communication, teamwork, and professional etiquette. To future interns stepping into the corporate world, I would say focus not only on technical growth but also on learning how to work in a team, ask questions without hesitation, and actively participate in all opportunities, from meetings to hackathons. These experiences will shape you as much as your coding tasks do.

READER'S AREA

"The future of EVs depends not just on power, but on mastering heat"

REVOLUTIONISING MOBILITY: ADVANCED THERMAL MANAGEMENT IN EV BATTERIES



Battery Thermal Management System

Introduction

Electric vehicles (EVs) are rapidly shaping the future of mobility, but one critical challenge often goes unnoticed- managing heat inside their batteries. The heart of every EV is the lithium-ion battery, which performs best only within a narrow temperature range of 20°C to 40°C.

If the battery overheats, it degrades more quickly, which can lead to failure. On the other hand, extremely cold conditions can cut the driving range by up to half, making the vehicle unreliable. Clearly, keeping the battery within the right temperature window is vital not only for efficiency but also for safety and durability.

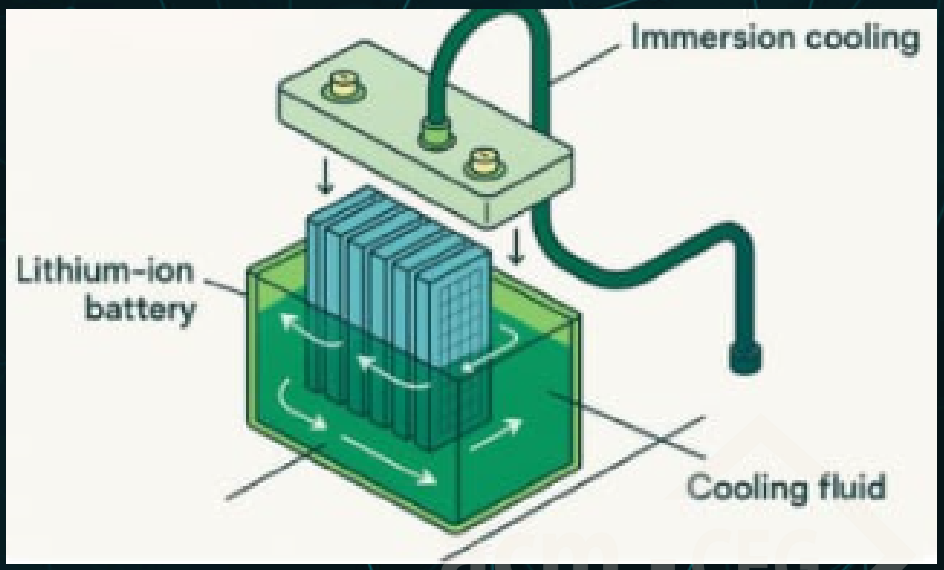
Conventional Cooling and Its Limits

Traditionally, EVs have used air cooling or indirect liquid cooling systems to control battery heat. While these methods are cost- effective, they face major drawbacks such as bulky designs, uneven cooling, hotspots, and extra energy consumption.

With the growing need for fast charging and higher energy-density packs, these conventional systems are no longer sufficient. This has pushed researchers and engineers to explore smarter solutions.

Smarter Thermal Solutions

Recent advancements include the use of phase change materials that absorb excess heat, heat pipes that quickly transfer heat away from hotspots, and



Immersion Cooling for Lithium Batteries

hybrid cooling systems th
by almost 90 percent.
However, the most promis
immersion cooling, wh
submerged in special
approach ensures faster
lighter design, and ev
charging.

Closing Thoughts

The race to improve EV b
chemistry or design it is a
heat.
Smarter thermal manage
electric vehicles safer,
reliable, accelerating the
green and sustainable futu

REFERENCES:

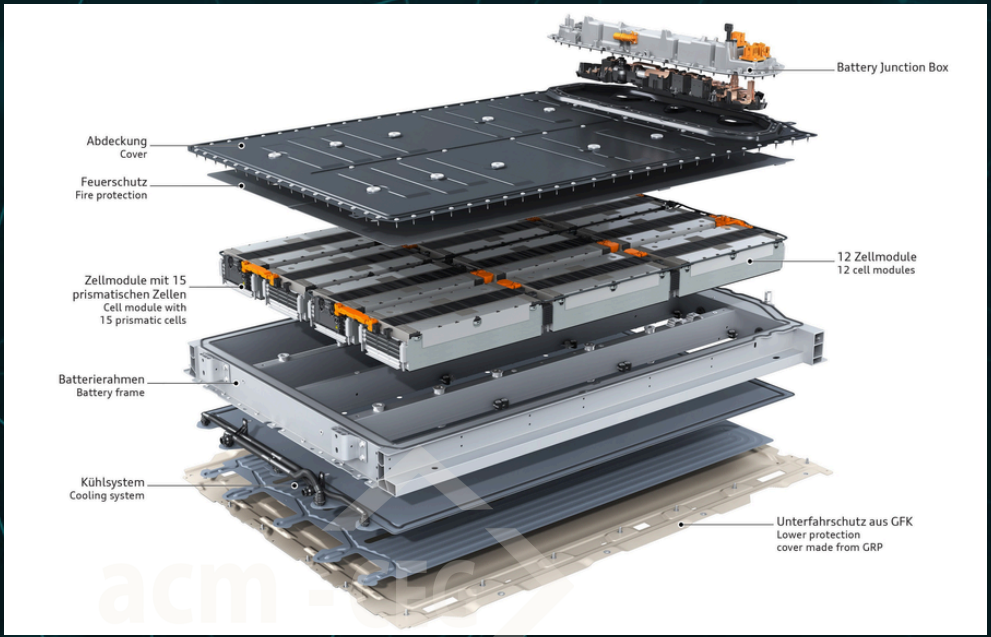
• Wahab, A., Najmi, A.-U.
N., Kemper, H., & Khayya
cooling innovations and
battery cooling for future e
• Weragoda, D. M., Tian,
K.-H. (2023) - A compre
pipe based battery therma

hat reduce energy losses

sing breakthrough is direct here battery cells are dielectric fluids. This r cooling, better safety, ven supports ultra-fast

batteries is not just about about mastering the art of

ement systems will make onger-lasting, and more world's journey toward a ure.



EV Battery Pack Exploded View

-H., Senobar, H., Amjady, m, H. (2025) - Immersion critical hurdles in Li-ion electric vehicles.
G., Burkitbayev, A., & Lo, ehensive review on heat al management systems.

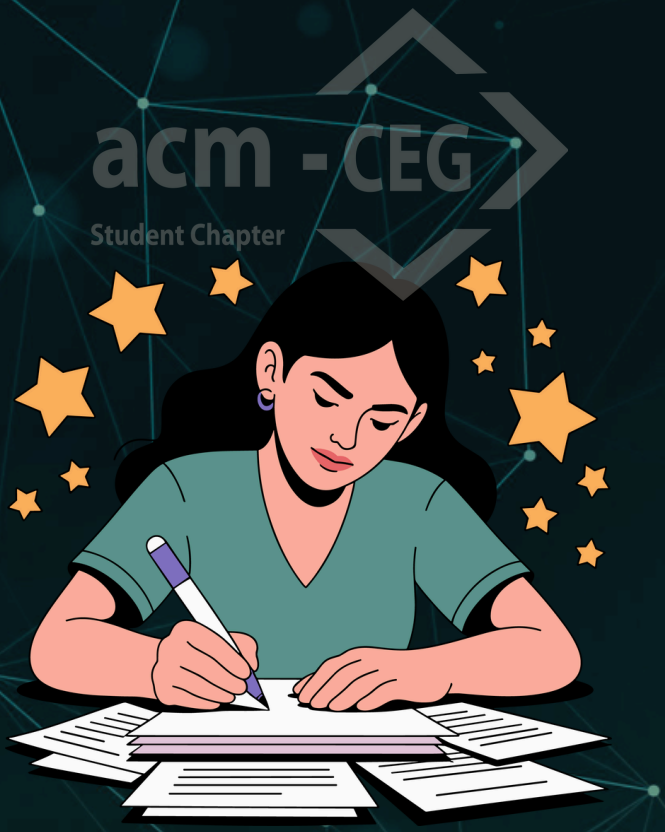
--Narendhiran V

(2nd year, Mechanical Engineering)

READER'S AREA

Have a fresh story to tell? An experience to share? Or maybe an unique idea that you just thought randomly?
Whether it's a project you're proud of, insights from a tech trend, or even a quirky take on student life - We'd love to hear from you! This space is for our community, by our community.

Send us your write-ups, articles, or even short reflections. Who knows? Your piece might just inspire, inform, or entertain hundreds of readers in the next issue of our newsletter.
Share your content with us (acm.ceg2025@gmail.com) and become a part of our growing conversation!



Stack'D

Issue #4 | Oct-Nov 2025

acm - CEG
Student Chapter

acmceg | ACM Student Chapter of CEG | auceg.acm.org

Credits

Content Contributors:
Abirami R, Madhubaalika M, Sanjay Kumaran, Roopa Varshini, Niranjan K

Editor:
Shree Vaishnavi, Niranjan K

Design & Layout:
Gabriella D

References:
https://hw.glich.co/p/just-in-time-jit-compilation-how-it-speeds-up-code