

DCE TECH CHRONICLE

— IDEAS | INNOVATION | TECHNOLOGY —

CSE EDITION

Volume V, Issue I
(July - December 2024)



CONTENTS

01



HOD Message

02



Editor's Message

03



Vision & Mission

04



PEOs, POs, PSOs

05



Student's Articles

06



Department
Achievements



HOD's MESSAGE

Dr. Ashima Mehta
(HOD CSE)



“Technology does not simply change the world; it empowers us to reimagine what is possible.”

It is a matter of great pride to introduce this technical magazine, a reflection of the ideas, imagination, and innovative spirit of our Computer Science & Engineering community. In an era where Artificial Intelligence, Data Science, Cybersecurity, Cloud Computing, and emerging technologies are redefining the way we live and work, the ability to learn, question, create, and innovate has become more important than ever.

This magazine is more than a collection of technical articles—it is a glimpse into the curiosity and potential of our students. Every idea shared, project explored, and insight presented represents a step towards becoming not just technology users, but technology creators and problem-solvers.

As a department, our endeavour is to provide an environment where students can experiment with ideas, embrace challenges, learn from failures, and transform their knowledge into meaningful solutions. I encourage our students to remain curious, keep exploring beyond the classroom, and have the courage to turn their ideas into reality.

My appreciation goes to the faculty members, students, and editorial team whose dedication has brought this edition to life. I hope these pages spark new thoughts, inspire innovation, and encourage every reader to ask one important question: “What can I create that has never been created before?”

Wishing the CSE community a future filled with curiosity, creativity, and limitless possibilities.

Dr. Ashima Mehta
Head, Department of Computer Science & Engineering

Editor's Message

Editor in-Chief



Prof. Renu
(Faculty; CSE Department)

It is my pleasure to present this edition of our Technical Magazine, celebrating the ideas, creativity, and technical achievements of our students and faculty. May this edition inspire every reader to explore new ideas, embrace innovation, and continue the journey of learning and discovery.

STUDENT EDITORS



Divyansh Sethi

CO Editor

Roll No : 25047



Vaibhav Khandelwal

Editor (Design)

Roll No : 25187



Saurabh S Parmar

Editor (Text)

Roll No : 25162

Small Idea, Big Impact



Vision

Steering the future of computer science through innovative advancements, fostering ethical values and principles through technical education.



Mission

- Directing future innovations in computer science through revolutionary progress.
- Instilling a foundation of ethical values and principles in every technologist.
- Offering a comprehensive technical education to equip individuals for a meaningful and influential future.

Program Educational Objectives (PEO)

- Apply the technical competence in Computer Science and Engineering for solving problems in the real world.
- Carry out research and develop solutions on problems of social applications.
- Work in a corporate environment, demonstrating team skills, work morals, flexibility and lifelong learning.

Program Specific Outcome (PSO)

- Exhibit design and programming skills to develop and mechanize business solutions using revolutionary technologies.
- Learn strong theoretical foundation leading to brilliance and enthusiasm towards research, to provide well-designed solutions to complicated problems.
- Work effectively with diverse Engineering fields as a team to design, build and develop system applications.

Programme Outcome (Pos)

- **PO1:** Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
- **PO2:** Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
- **PO3:** Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
- **PO4:** Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
- **PO5:** Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
- **PO6:** The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
- **PO7:** Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
- **PO8:** Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
- **PO9:** Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences
- **PO10:** Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
- **PO11:** Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.(WK8)

Self-Healing Software Systems

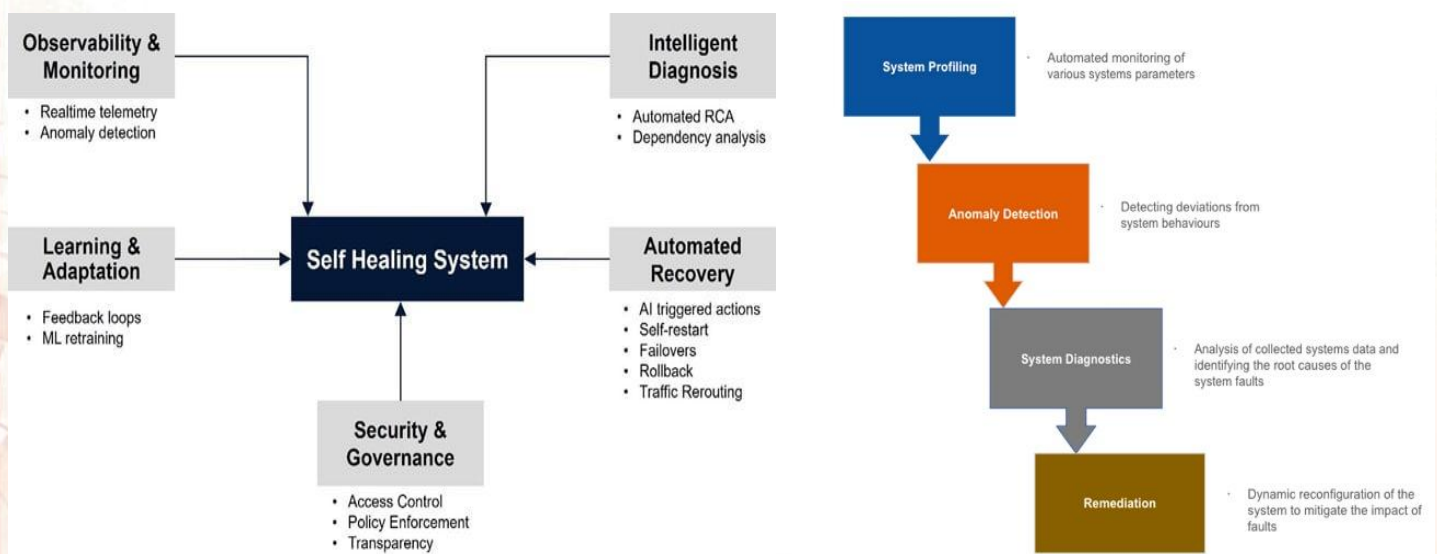


Pulkit

Roll. No. : 25127(3rd Year)

The Future of Autonomous and Resilient Computing

Modern software systems have become increasingly complex. Cloud platforms, mobile applications, financial systems, healthcare technologies, and industrial infrastructure often operate continuously, making even a few minutes of downtime costly. Traditionally, when software encounters an error, human engineers must identify the problem, diagnose its cause, and deploy a solution. **Self-healing software systems aim to change this model by enabling software to detect, diagnose, and recover from certain failures automatically.**



Student's Articles

What Are Self-Healing Software Systems?

A self-healing software system is designed to **identify abnormal behavior or failures and take corrective action without requiring immediate human intervention**. Inspired by the human body's ability to detect and repair damage, these systems continuously monitor their own health.

A typical self-healing system follows a cycle:

Monitor → Detect → Diagnose → Recover → Learn

For example, if a server becomes unresponsive, the system may automatically detect the failure, determine that the server is unhealthy, redirect traffic to another server, restart the failed service, and continue monitoring the system.

How Does Self-Healing Work?

Self-healing systems generally combine **automation, artificial intelligence, machine learning, monitoring tools, and intelligent decision-making**.

1. Continuous Monitoring

The system continuously observes application performance, network traffic, server health, memory usage, response times, and error rates.

2. Anomaly Detection

Machine-learning models or predefined rules identify unusual behavior. For example, a sudden increase in failed requests may indicate that a service is experiencing a problem.

3. Automated Diagnosis

After detecting an anomaly, the system attempts to determine its cause. It may analyze logs, system metrics, recent deployments, and dependencies.

Student's Articles

4. Automatic Recovery

Once the problem is identified, predefined or AI-assisted actions can be triggered. These may include restarting a service, replacing a failed container, rolling back a software release, reallocating resources, or rerouting network traffic.

5. Learning and Improvement

Advanced systems can analyze previous incidents and improve future responses. Over time, this can make recovery mechanisms more effective.

Real-World Applications

Self-healing technologies are particularly valuable in environments where continuous availability is important.

Cloud Computing: Cloud platforms can automatically replace unhealthy virtual machines or containers and redistribute workloads.

Cybersecurity: Security systems can detect suspicious activities and automatically isolate compromised devices or services.

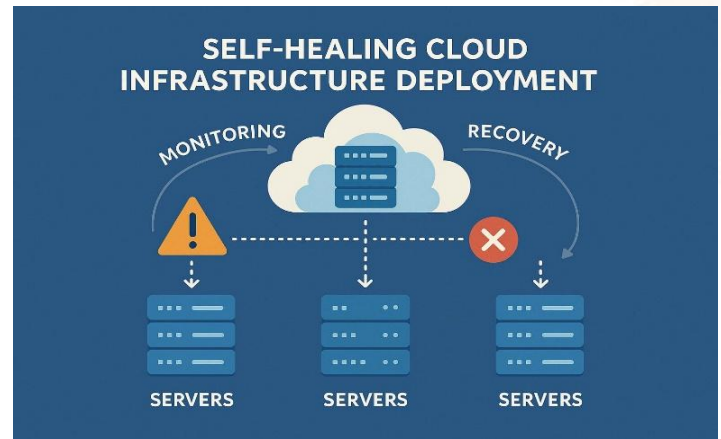
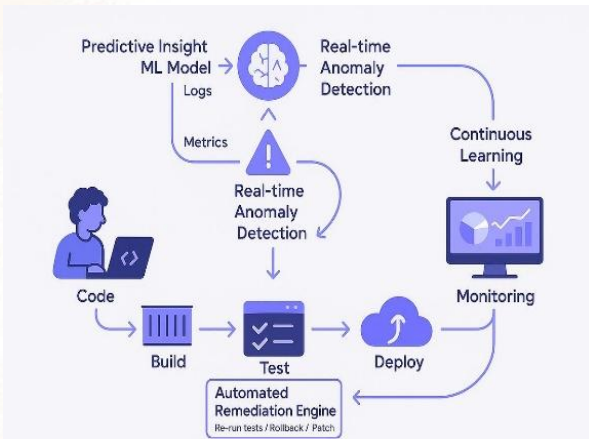
Telecommunications: Networks can identify failures and reroute traffic to maintain connectivity.

Healthcare: Critical software infrastructure can automatically respond to system failures and maintain availability of essential services.

Autonomous Vehicles: Vehicle software can detect faults in sensors or computing components and switch to backup mechanisms where possible.

Industrial Systems: Smart factories can monitor machines and software infrastructure to detect failures before they cause major disruptions.

Student's Articles



Advantages

Self-healing software systems offer several important benefits:

Reduced downtime: Problems can be addressed immediately.

Faster recovery: Automated responses are often faster than manual intervention.

Improved reliability: Systems can continue operating despite certain failures.

Reduced operational workload: Engineers can focus on complex problems rather than repetitive recovery tasks.

Scalability: Automated recovery becomes increasingly valuable as infrastructure grows.

Predictive maintenance: Potential failures may be identified before they occur.

Challenges and Risks

Despite their advantages, self-healing systems are not completely autonomous or risk-free.

An incorrect diagnosis can lead to an incorrect recovery action. For example, automatically restarting a service may temporarily hide a deeper problem rather than solve it. An AI system could also make unexpected decisions when encountering conditions that were not represented in its training data.

Student's Articles

Other challenges include:

Ensuring safe automated decisions

Avoiding cascading failures

Protecting monitoring and automation systems from cyberattacks

Maintaining human oversight

Explaining why an automated action was taken

Testing recovery mechanisms before deployment

Therefore, **self-healing should not simply mean removing humans from the process**. A better approach is to allow software to handle routine failures automatically while keeping humans involved in critical or uncertain decisions.

The Future of Self-Healing Software

The future may bring software systems capable of detecting problems, predicting failures, automatically applying tested fixes, and learning from previous incidents. Combined with **AI agents, cloud computing, observability, cybersecurity, and automated DevOps**, self-healing could become an important component of next-generation digital infrastructure.

The ultimate goal is not software that never fails—such systems are unrealistic. Instead, the goal is software that can **recognize failure quickly, limit its impact, recover safely, and continuously improve its resilience**.

Conclusion

Self-healing software represents a major shift in software engineering—from systems that depend heavily on humans for recovery to systems capable of managing many operational problems themselves. As organizations increasingly depend on always-available digital services, autonomous recovery and resilience will become increasingly important.

Student's Articles

Cloud or Edge? Deciding Where Computation Should Happen



Anishka
Roll. No. : 24014(Final Year)

Imagine a self-driving car approaching a pedestrian. It has only a fraction of a second to decide whether to apply the brakes. Should the car send the camera feed to a remote cloud server, wait for it to process the information, and then receive a response? Or should the decision happen inside the vehicle itself?

For a system like this, the answer is clear: **the computation needs to happen as close to the car as possible.**

This simple example introduces an important question in modern computing: **Where should computation happen?** For years, cloud computing has been the preferred solution for storing data and running applications. However, the rapid growth of IoT, artificial intelligence, autonomous systems, and real-time applications has created a strong need for another approach—**edge computing.**

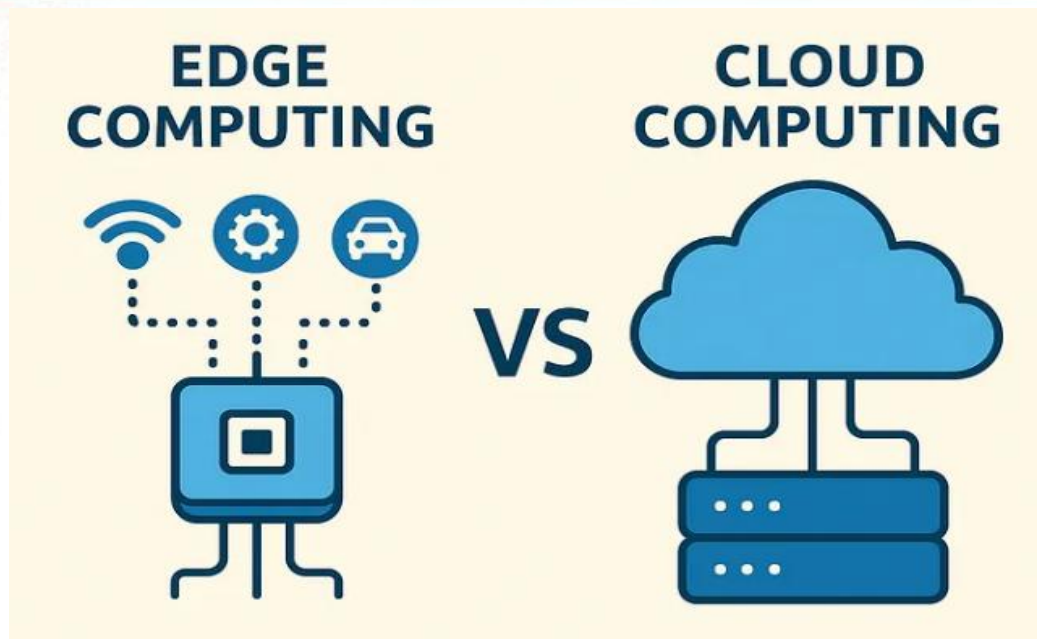
The future of computing may not be about choosing cloud *or* edge, but about deciding which tasks should be performed where.

Understanding Cloud Computing

Cloud computing allows applications to use computing resources located in remote data centers. Instead of relying entirely on the processing power and storage available on a local device, data can be sent to powerful cloud servers for processing.

Consider an online photo-storage application. A smartphone captures an image, but the cloud can provide large-scale storage,

Student's Articles



synchronization, image processing, and AI-based features. The user does not need a highly powerful computer because much of the computation happens remotely

Cloud computing provides three major advantages:

scalability, computing power, and flexibility. Cloud data centers can support millions of users, store enormous amounts of data, and allocate resources according to demand.

However, cloud computing has an important limitation: **the data has to travel.**

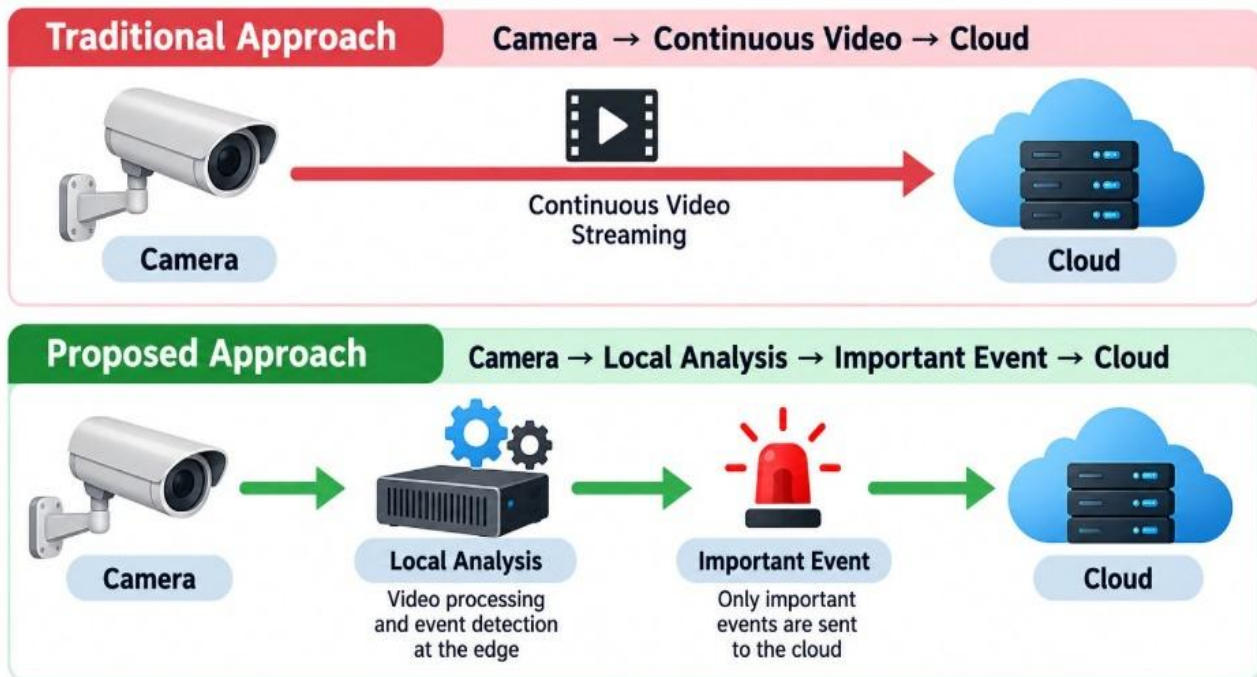
When information moves from a device to a remote server and back, communication introduces latency. For applications such as email or document storage, this delay is usually insignificant. But for autonomous vehicles, industrial robots, augmented reality, or emergency monitoring systems, even a small delay can affect performance.

Bringing Computation to the Edge

Edge computing takes a different approach. Instead of sending all data to a distant cloud server, some processing is performed **near the location where the data is generated.**

Student's Articles

Consider a smart surveillance camera installed on a college campus. A traditional system could continuously send video footage to the cloud for analysis. An edge-based system can analyze the video locally and transmit only important information.



For example, the camera may identify an unusual activity locally and send only an alert to the central system. This reduces network traffic and can produce a much faster response.

Why Latency Matters

One of the biggest advantages of edge computing is reduced **latency**. Imagine an industrial robot working on a production line. If its sensors detect an obstacle, the robot may need to stop immediately. Sending the sensor information to a remote server and waiting for a response may introduce unnecessary delay. An edge computer located within the factory can process the information almost immediately.

Student's Articles

This makes edge computing particularly useful for:

Autonomous vehicles

Smart surveillance

Industrial automation

Smart healthcare

Augmented and virtual reality

Smart cities

IoT applications

Real-time gaming

The basic principle is simple:

When a decision needs to happen quickly, computation should move closer to the device.

Cloud Computing vs Edge Computing

Cloud and edge computing have different strengths which can be categorised as follows:

Requirement	Cloud	Edge
Large-scale storage	Excellent	Limited
Computing power	Very high	Relatively limited
Response time	Can be higher	Usually lower
Real-time processing	Less suitable for some applications	Highly suitable
Scalability	Excellent	More limited
Network dependence	Higher	Lower for local tasks
Centralized analytics	Excellent	Limited
Local processing	Limited	Strong

Student's Articles

This comparison shows that neither technology is universally better. The appropriate choice depends on the requirements of the application.

Edge AI: Intelligence at the Source

The combination of artificial intelligence and edge computing has created an important concept known as **Edge AI**.

Traditionally, an AI application may send data to a centralized server where a machine-learning model processes it. With Edge AI, trained AI models can run directly on devices or nearby edge systems.

For example, a smartphone can perform image recognition, speech processing, or camera enhancement without sending every piece of raw information to a remote server. This can provide two major benefits: **faster response and reduced data transmission**. It can also support better privacy in some applications because sensitive information may be processed locally instead of being continuously transmitted to external infrastructure.

The Future: Cloud + Edge

It would be incorrect to assume that edge computing will replace the cloud. In reality, many modern systems are moving towards a **hybrid architecture** that combines both. Consider a smart manufacturing plant. Sensors continuously collect information about temperature, vibration, pressure, and machine performance.

An edge computer can immediately analyse the data and detect an abnormal condition. If necessary, it can trigger an alert or stop a machine without waiting for the cloud. At the represented as:

Student's Articles

same time, selected data can be sent to the cloud for long-term storage and analysis. The cloud can examine months or years of information to identify patterns and improve predictive-maintenance models.

The architecture can therefore be



Here, the edge provides speed, while the cloud provides scale and computational power.

In Conclusion, Cloud computing transformed modern software by making powerful computing resources available on demand. Edge computing is extending this idea by bringing computation closer to the devices and users that generate data. The question is therefore no longer simply **“Cloud or Edge?”**

The more important question is: **“Where should this particular computation happen?”**

The answer depends on latency, computing requirements, network availability, security, scalability, and cost.

The most intelligent systems will be those that know **what to compute, where to compute it, and when to move the computation between the edge and the cloud.**

Student's Articles

The Future of Software Engineering: Autonomous, Intelligent and Adaptive Systems



Anjali Sharma
Roll No.: 25016(3rd Year)

Software engineering is entering a new era in which software systems are no longer limited to following predefined instructions. Advances in **Artificial Intelligence, Machine Learning, automation, and cloud computing** are enabling the development of autonomous, intelligent, and adaptive systems that can learn from data, make decisions, detect problems, and respond to changing environments.

From Traditional Software to Intelligent Systems

Traditional software generally operates according to rules defined by developers. Any major change in requirements, user behaviour, or operating conditions often requires manual intervention. Modern intelligent systems, however, can analyse information and adjust their behaviour dynamically.

For example, an intelligent application can identify unusual system activity, predict potential failures, recommend solutions, and in some cases take corrective action automatically. This shift is transforming software from a static product into a **continuously evolving system**.

[Traditional Software → AI-Powered Intelligent Software]

Student's Articles

Autonomous Software Engineering

Autonomous software systems aim to perform development and operational tasks with minimal human intervention. AI-powered tools can assist developers in generating code, identifying bugs, creating test cases, reviewing code, and preparing documentation.

In the future, an intelligent system could analyse a requirement, suggest an architecture, generate code, test it, identify vulnerabilities, deploy the application, and continuously monitor its performance.

Developers would increasingly focus on **system design, decision-making, validation, creativity, and innovation** rather than repetitive programming tasks.

[AI-Assisted Software Development Lifecycle]

Adaptive Systems

One of the most important characteristics of future software will be its ability to adapt. Adaptive systems continuously observe their environment and modify their behaviour according to changing conditions.

Consider a cloud application experiencing a sudden increase in users. An adaptive system could automatically allocate additional computing resources, optimise performance, and release those resources when demand decreases.

Similarly, cybersecurity systems can analyse emerging attack patterns and dynamically strengthen their defence mechanisms.

Student's Articles

The Role of Artificial Intelligence

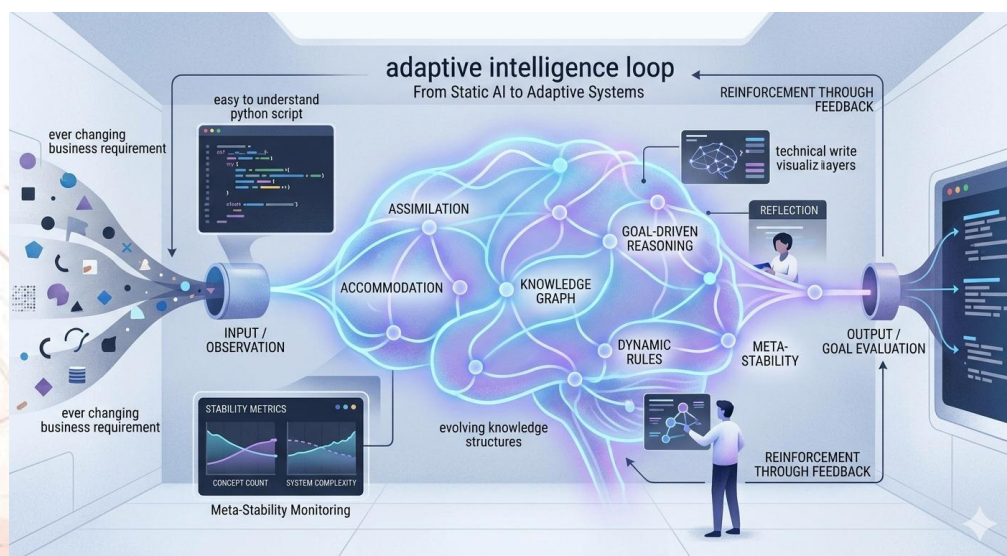
Artificial Intelligence will be a major driving force behind this transformation. AI can support software engineers throughout the development lifecycle—from requirements analysis and design to testing, deployment, and maintenance.

Generative AI can assist with code generation and documentation, while Machine Learning can help predict software failures and identify performance problems. AI-powered testing systems can explore multiple scenarios and discover defects that may be difficult to identify through conventional testing.

Challenges Ahead

Despite its potential, autonomous software engineering introduces important challenges. AI-generated code may contain errors, security vulnerabilities, or unintended behaviour. Systems that make autonomous decisions must therefore be carefully monitored and validated.

Other concerns include **data privacy, cybersecurity, transparency, accountability, bias, and intellectual property**. Human oversight will remain essential, particularly in applications where software decisions can have significant consequences.

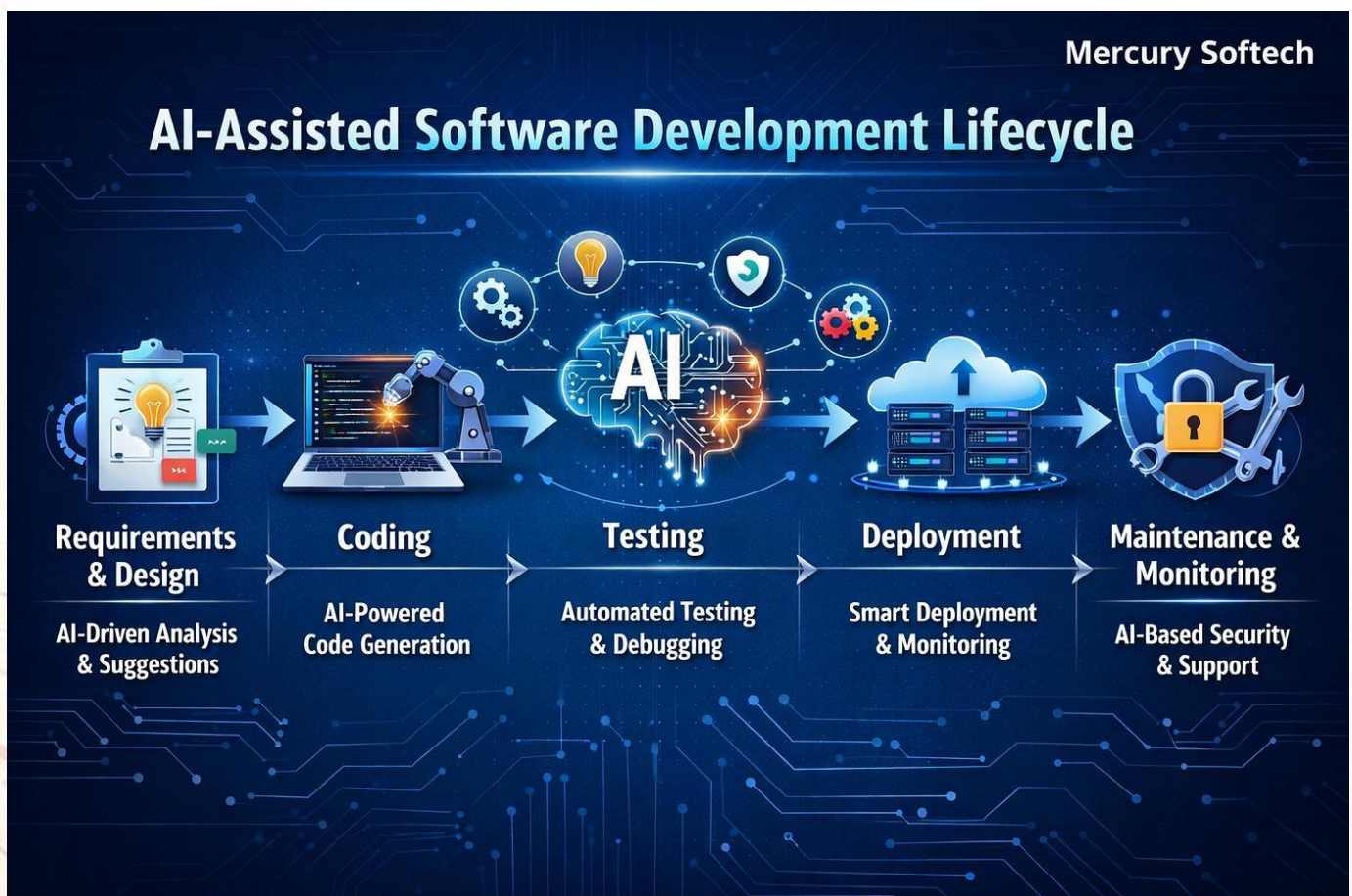


Student's Articles

The Changing Role of Software Engineers

The future does not necessarily mean the replacement of software engineers. Instead, their roles are likely to evolve. Engineers will increasingly work as **architects, problem-solvers, AI collaborators, security experts, and system supervisors**.

Understanding AI, system architecture, cybersecurity, data management, and ethical software development will become increasingly important. The ability to evaluate AI-generated solutions and make informed technical decisions may become just as valuable as writing code manually.



Student's Articles



Conclusion

The future of software engineering is moving toward systems that are **autonomous, intelligent, and adaptive**. Such systems have the potential to make software development faster, more reliable, secure, and responsive to changing needs.

However, technology alone cannot determine the future. Responsible human oversight, ethical principles, strong security practices, and continuous learning will remain essential.

The software engineer of tomorrow will not simply write code—they will **design intelligent systems that can learn, adapt, and create value in an increasingly dynamic digital world**.

Student's Articles

The Hidden Life of an App: What Happens After You Click “Login”?



Deepak Mehra
Roll No.: 24035(Final Year)

We interact with applications every day, often without thinking about what happens behind the screen. You enter your email and password, click “**Login**,” and within seconds your dashboard appears. But behind that simple click, a complex chain of software, networks, databases, and security mechanisms starts working together.

1. Your Request Leaves the Device

The moment you click **Login**, the application collects the information you entered and prepares a request for the server. This request usually travels through the internet using protocols such as **HTTPS**.

HTTPS encrypts the communication between your device and the application's server, helping prevent attackers from easily reading sensitive information while it is being transmitted.

2. The Server Receives the Request

The request reaches the application's backend server. The server determines what the user is asking for and routes the request to the appropriate software component.

Student's Articles

In modern applications, this may involve an **API gateway**, **load balancer**, **authentication service**, or **microservice** before the request reaches the component responsible for login.

This entire process can happen in milliseconds.

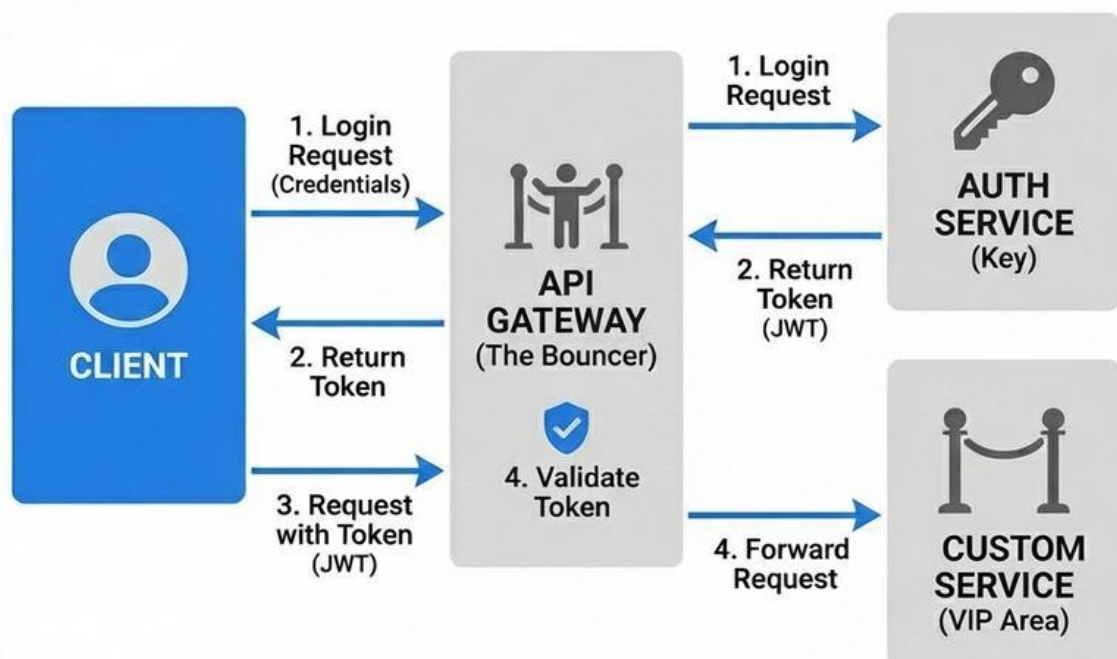
3. Your Identity Is Verified

The authentication service now needs to determine whether the credentials are valid.

Importantly, well-designed applications should not store passwords as plain text. Instead, passwords are normally processed using secure **hashing techniques** before being stored.

The server compares the submitted credentials with the securely stored information

Microservices Auth Flow



Student's Articles

4. The Database Gets Involved

Behind almost every login system is a database containing information about users.

The application may retrieve details such as:

User ID

Account status

Roles and permissions

Profile information

Security settings

Login history

The database does not simply “give everything” to the application. Properly designed systems use controlled queries and access permissions to limit what information can be retrieved.

Technologies such as **PostgreSQL, MySQL, MongoDB**, and others can be used depending on the application's architecture.

The database doesn't necessarily handle every part of the login process itself. Modern applications often divide responsibilities among multiple services.

5. “You’re Logged In”—But How Does the App Remember?

Once your identity has been verified, the system needs a way to recognise you during subsequent requests.

Student's Articles

This is where **sessions, cookies, and authentication tokens** come into play.

Instead of repeatedly sending your password, the application can use a secure session or token to indicate that you have already been authenticated.

This is why you can move from your profile to messages, settings, notifications, and other sections without entering your password again.

6. The Server Responds

After authentication succeeds, the server sends a response back to the application.

The response may contain information indicating that authentication was successful, along with the data required to establish a secure session.

The application receives the response and begins loading the information needed for your next screen.

7. The Home Screen Is Only the Beginning

Even after you see the home screen, the application may continue communicating with its backend.

It could request:

Your latest notifications

Profile information

Messages

Recommended content

Account details

Recent activity

This means that the home screen you see is often being assembled dynamically from information retrieved from different backend services.

Student's Articles

8. What If the Password Is Wrong?

If authentication fails, the server does not create a successful authenticated session.

Instead, it returns an appropriate response, and the application displays a message such as:

“Invalid username or password.”

Modern applications may also use additional security measures such as **rate limiting**, **CAPTCHA**, **temporary account restrictions**, and **suspicious-login detection** to make automated attacks more difficult.



9. Sometimes a Password Isn't Enough

Many applications now use **Multi-Factor Authentication (MFA)**.

After entering your password, you might be asked to provide:

A one-time password (OTP)

An authenticator-app code

A security key

A push notification

Another verification factor

Student's Articles

The Bigger Picture

A modern application is much more than the interface displayed on your smartphone.

Behind the screen may be an entire ecosystem of interconnected components:

User → App → API → Authentication Service → Server → Database → Other Services → Response

Each component has a specific responsibility, and together they create the seamless experience we take for granted.

The next time you tap “**Login**,” remember that you are not simply opening an application.

You are initiating a carefully coordinated process involving **software engineering, networking, cybersecurity, databases, cloud infrastructure, encryption, and authentication.**

And all of this may happen before you have even finished noticing the loading animation.

Conclusion

The most interesting part of an application is often the part you never see.

The buttons, colours, and screens are only the visible layer. Beneath them lies a hidden world of code and systems working continuously to make a simple tap feel effortless.

So, the next time you click “Login,” pause for a moment. Behind that tiny button is an entire digital journey.

Department's Achievements



NBA Accreditation

Dronacharya College of Engineering proudly celebrates the NBA accreditation of its CSE programme. This achievement reflects our commitment to academic excellence and quality education. It stands as a testament to the dedication of our students, faculty and staff.

Department's Achievements



GLC-HP Global Hackathon 2024

Dronacharya College of Engineering proudly celebrates the achievement of Aditya Punia (Roll No. 24204, 3rd Year - CSE (AI & ML)) and Sarthak Rawat (Roll No. 24269, 3rd Year - CSE (AI & ML)), winners of the GLC-HP Global Hackathon 2024. Their innovation and technical excellence brought great pride to the institution on an international stage.

Department's Achievements



Congratulations! Team SMART INSPECT

For Winning Software Edition of

SMART INDIA HACKATHON 2024



1 LAKH INR

Prize

Team Members

Raghav Sharma 25465(IoT); 3rd Year	Priyanshu 25464(IoT); 3rd Year
Vikrant 25193(CSE); 3rd Year	Tanya 25480(IoT); 3rd Year
Sapna Pokharal 26547(IoT); 2nd Year	Harsh Tiwari 27507(IoT); 1st Year

*Mentors: Dr. Megha Goel
Ms. Jyoti Yadav*







Smart India Hackathon 2024

Team Smart Inspect emerged as winners of the Software Edition of Smart India Hackathon 2024. They secured a ₹1 Lakh prize for their innovative and impactful solution. This achievement reflects their teamwork, creativity, and technical excellence.

Department's Achievements

DRONACHARYA
College of Engineering

Congratulations!
For winning

1st Prize

Ideathon Haryana
“Unlock Your Entrepreneurial Spirit”

Nikhil Singhal
Roll No:- 25106 | CSE

Nishtha Bhateja
Roll No:- 25109 | CSE

Rajat Bhardwaj
Roll No:- 25136 | CSE

- Won Cash Prize of Rs. 31,000 for the whole team and Received Certificate of Achievement

Ideathon Haryana

Dronacharya College of Engineering proudly congratulates Nikhil Singhal (Roll No. 25106, CSE), Nishtha Bhateja (Roll No. 25109, CSE) and Rajat Bhardwaj (Roll No. 25136, CSE) for securing 1st Prize at Ideathon Haryana - “Unlock Your Entrepreneurial Spirit”. The team won a cash prize of ₹31,000 along with a Certificate of Achievement.