Week 2 Lab Exercise: Historical Perspectives

1. **Based on the content of the article "Brooks, F.P., *No silver bullet, Essence and accidents of software engineering*, IEEE Computer 20 (4), April 1987", and using the notes that you have made record what you expect to learn from a module entitled Fundamentals of Software Engineering and in particular how you might use these skills in the future, giving a brief justification for the points that you identified. As part of your report identify one emerging trend in software engineering. Note – you may have to do some additional searching online to determine an emerging trend. Your word limit here should be around 350-400 words.**

   In this article, Brooks main argument is that there are "no silver bullets" when it comes to software development, i.e., there is not one single development technology or management technique that we can apply to solve every single problem. Brooks goes further into this idea, by categorising the difficulties in software technology in two: *essence difficulties* – which are inherent to the problem domain –, and *accidents* – arising from inadequate tools and methods. He identifies four essential difficulties: complexity, conformity, changeability, and invisibility. Thus, I expect to learn strategies for managing these inherent challenges through proper documentation, modular design, version control, and visualisation techniques that make software development more manageable. In addition, I expect to learn multiple complementary approaches – requirements engineering, design methodologies, testing strategies, and project management techniques. This will help me avoid the trap of seeking magical solutions and instead focus on disciplined engineering practices.

   Brooks also advocates for "growing" software organically through incremental development, starting with a working system and progressively adding functionality. I expect to gain proficiency in Agile methodologies and iterative design approaches that embody this philosophy, enabling me to deliver value early and adapt to changing requirements throughout the development lifecycle. These skills will prove invaluable when leading development teams and managing stakeholder expectations. Understanding that complexity increases non-linearly with system size will help me provide realistic project estimates and invest in team capabilities rather than technological quick fixes.

   The most significant emerging trend is AI-enhanced low-code/no-code development platforms. By 2025, 75% of new applications are expected to utilise such platforms, with AI providing intelligent code generation and automated

testing capabilities.

This trend perfectly illustrates Brooks' insights – AI-powered platforms primarily attack accidental complexity by automating routine coding tasks and reducing implementation overhead. However, they cannot address essential complexity: understanding business requirements, making architectural decisions, and solving domain-specific problems still require human expertise and judgement. This validates Brooks' argument that the hardest part of software engineering remains the conceptual work of specification and design, not the mechanical labour of coding.

2. **Read through the "Rashid et al, *Software Engineering Ethics in a Digital World*, IEEE Computer, 42 (6) June 2009." and make notes. Concentrating on Facebook and Google Maps, briefly discuss in around 450 -500 words any potential ethical issues for the software engineer developing these applications. Include a brief discussion of what you think you need to learn about ethics in relation to software engineering.**

   Based on Rashid et al.'s exploration of software engineering ethics in a digital world, the concept of ethics-aware software engineering emerges as crucial, where ethical considerations are explicitly integrated throughout the software development lifecycle as part of risk assessment and acceptance criteria. This framework becomes particularly relevant when examining the ethical challenges faced by software engineers developing platforms like Facebook and Google Maps, where technical decisions have far-reaching societal implications.

   Facebook software engineers confront significant ethical dilemmas regarding data collection and user consent, as the platform harvests vast amounts of personal information including education, interests, work history, political and religious affiliations, often without users fully comprehending the implications. The Cambridge Analytica scandal exemplifies these challenges, where software systems permitted third-party applications to access not only consenting users' data but also their friends' information without consent, forcing engineers to navigate the tension between enabling platform functionality and preventing data misuse. Engineers developing Facebook's recommendation algorithms face additional ethical questions about their responsibility for content curation, as engagement-maximising algorithms designed to keep users active potentially create filter bubbles and contribute to political polarisation, raising questions about whether optimising for engagement aligns with broader social responsibilities. The 2014 emotional manipulation experiment, where Facebook altered users' news feeds to display more positive or negative content without

explicit consent, demonstrates how software engineering decisions can directly impact users' psychological wellbeing, whilst the platform's scale of over 2 billion users means seemingly minor software changes can have massive societal impacts, with features designed to connect people potentially weaponised for harassment, misinformation, or political manipulation.

Similarly, Google Maps engineers face complex ethical decisions about location data collection and usage, as the service requires precise location tracking to provide navigation and traffic updates whilst creating detailed profiles of users' movements, habits, and personal lives, forcing engineers to balance service functionality with privacy protection. The persistent location tracking that continues even when users believe they've disabled it raises fundamental questions about informed consent and transparency in software design, particularly regarding whether users truly understand what data is being collected and how it's used. Maps engineers must also protect vast amounts of sensitive geospatial data from potential misuse by governments, commercial entities, or malicious actors, since location data can reveal intimate details about individuals' lives, including medical appointments, political activities, and personal relationships. Furthermore, these engineers face ethical challenges regarding algorithmic bias and representation, as decisions about which areas receive detailed mapping, real-time traffic updates, or business listings can impact economic opportunities and social equity, whilst the Simon Weckert "99 phones" experiment demonstrated how easily mapping platforms can be manipulated, raising concerns about data validation and the potential for malicious actors to disrupt traffic patterns or emergency services.

To address these challenges effectively, I need to learn systematic frameworks for ethical decision-making that help evaluate the implications of software features during development, including identifying stakeholders, assessing potential harms, and weighing competing interests when designing systems. Understanding professional codes such as the IEEE Computer Society and ACM ethics guidelines, alongside emerging regulations like GDPR, will be crucial for making informed decisions about data handling, privacy protection, and user consent mechanisms. I must develop skills for anticipating broader societal implications of software systems, particularly how features designed for one purpose might be misused or have unintended consequences at scale, whilst learning to navigate situations where business requirements conflict with ethical considerations, including advocating for user protection within commercial organisations and communicating ethical concerns to non-technical stakeholders. Most importantly, I need to understand how to translate ethical principles into concrete technical decisions, such as implementing privacy-by-

design, creating transparent algorithms, and building systems that give users meaningful control over their data, ensuring that ethical considerations become an integral part of the software development process rather than an afterthought.