1. What is Tech Debt? List examples of tech debt that Parts Unlimited has and what was the cause.

Technical Debt refers to the additional cost and effort incurred when taking shortcuts in software development to achieve faster delivery. It occurs when teams sacrifice code quality, proper architecture, or best practices, creating work that must be addressed later to maintain or improve system stability, performance, or scalability. Essentially, it's like "borrowing time" now but paying interest in the future through increased maintenance and slower development.
At Parts Unlimited, the legacy database systems are slow, rigid, and difficult to scale. This technical debt arose because the company prioritized short-term delivery over investing in modernizing its data infrastructure. Many parts of the system are manually tested, causing frequent errors and production failures. The root cause was a culture of rushing releases without building automated test suites. Deployments require many manual steps, are error-prone, and take a long time, slowing down delivery. This occurred because the organization had no standardized CI/CD pipelines and focused on immediate feature delivery rather than process improvement. The codebase is tightly coupled and difficult to modify without breaking other systems. This happened because developers often implemented quick fixes and shortcuts rather than refactoring for long-term maintainability. Most of these debts were caused by short-term thinking, where management prioritized immediate business goals over sustainable system design. A lack of automation, standardization, and cross-team communication also contributed, making it harder to address problems efficiently and increasing future maintenance costs.

2. What are 'the 3 ways' and how does it relate to DevOps behaviors and processes?

The Three Ways in *The Unicorn Project* describe foundational principles that guide DevOps behaviors and processes. It explains how work flows through an organization, how feedback is managed, and how a culture of learning is built. **The first way, Flow,** focuses on optimizing the entire value stream from development to operations, ensuring work moves quickly and smoothly without delays or bottlenecks. It encourages automation, continuous integration (CI), and continuous delivery (CD), and reduces dependencies between teams. The DevOps team use this principle to deliver features faster, reduce rework, and improve overall system efficiency. **The second way, Feedback,** emphasizes creating fast and constant feedback loops so that problems are detected early and corrected quickly. It encourages monitoring, automated testing, real-time alerts, and retrospectives. The DevOps team adopt continuous feedback to ensure quality, prevent small issues from becoming major failures, and support learning from mistakes. **The third way, Continual Learning and Experimentation,** focuses on a culture of experimentation, risk-taking, and knowledge sharing, learning from failures, and continuously improving. It encourages blameless reviews, innovation, and mentoring across teams. The DevOps team builds a learning-oriented culture, drives innovation, and ensures processes evolve to meet new challenges.

3. What are the '4 types of work' and how does it relate to DevOps behaviours and processes?

The four types of work are a way to categorize tasks within a technology organization, helping teams prioritize and manage their efforts effectively. They include:

- **Business Projects:** Work that delivers new features or products directly to customers. This type of work drives business value and innovation.
- **Internal IT Projects:** Work that improves internal systems, tools, or infrastructure, such as upgrading databases or implementing better deployment pipelines.
- **Changes / Unplanned Work:** Tasks that arise unexpectedly, like urgent bug fixes, incidents, or system outages that require immediate attention.
- **Technical Debt:** Work that addresses shortcuts or compromises made earlier in development, such as refactoring code, fixing fragile systems, or updating outdated components.

Understanding these four types enables DevOps teams to balance priorities and minimize friction in software delivery. For example, addressing technical debt prevents frequent firefighting of unplanned work, which aligns with **The First and Second Ways** by optimizing the value stream and creating fast feedback loops.
Managing **internal IT projects** ensures automation and infrastructure improvements, supporting smoother deployment and higher system reliability. Recognizing **business projects** keeps teams focused on delivering customer value, while planning for **changes/unplanned work** allows the organization to respond quickly without disrupting flow. By categorizing work this way, DevOps behaviors such as automation, continuous improvement, and collaboration become more structured and effective, helping teams maintain focus, reduce delays, and improve system quality.

4. What are the '5 ideals' and how does it relate to DevOps behaviours and processes?

The five ideals in *The Unicorn Project* describe the goals of a high-performing DevOps organization, guiding how teams should work and interact. They are:

- **Locality and Simplicity:** Teams should be able to work independently, with minimal dependencies on others. Systems and processes should be simple to understand and maintain.
- **Focus, Flow, and Joy:** Work should allow uninterrupted focus, smooth progress, and a sense of satisfaction for team members. Reducing interruptions and unnecessary complexity improves productivity and morale.
- **Improvement of Daily Work:** Teams should continually refine processes, address problems, and eliminate inefficiencies in daily operations. This fosters a culture of learning and optimization.
- **Psychological Safety:** Team members should feel secure in speaking up, sharing ideas, and reporting mistakes without fear of blame. This encourages collaboration, transparency, and innovation.
- **Customer Focus:** Every decision and action should ultimately serve the customer's needs, ensuring that work delivers real value.

The Five Ideals directly shape DevOps behaviours by encouraging collaboration, continuous improvement, and customer-centred thinking. **Locality and Simplicity** align with DevOps practices, such as microservices and independent deployment pipelines, allowing teams to release features quickly without blocking others. **Focus, Flow, and Joy** promote uninterrupted work and reduce waste, which DevOps addresses through automation, CI/CD, and streamlined processes. **Improvement of Daily Work** supports continuous feedback and iterative learning, which are core to DevOps culture. **Psychological Safety** fosters open communication and rapid problem-solving, enabling faster response to failures and innovation. **Customer Focus** ensures that DevOps practices prioritize value delivery over internal bureaucracy or technical showmanship.

5. **In the current workflow, give 3 examples of Parts Unlimited work process that is preventing the 5 ideals.**

At Parts Unlimited, deployments are highly manual and tightly coupled between multiple teams. Developers cannot independently release features because they must coordinate with operations for each step of the process. This violates the **Locality and Simplicity** ideal, as work is blocked by dependencies and complexity increases, making it difficult to move quickly or simplify processes. Due to legacy systems, a lack of automated tests, and unreliable processes, developers frequently interrupt their planned work to address urgent issues. This prevents the ideal of **Focus, Flow, and Joy**, as developers are pulled away from meaningful work, experience stress, and cannot maintain smooth, uninterrupted progress. Many problems at Parts Unlimited are only discovered after they have caused outages or delays, due to inadequate monitoring and feedback. This prevents the ideal of **Improvement of Daily Work**, as teams cannot learn from small issues in real time or proactively refine their processes. Without timely feedback, continuous improvement is stalled, and technical debt continues to accumulate.