Briefly describe the fundamental differences between project-based and product-based software engineering.

Project-based software: A project is unique in that it is not a routine operation, but a specific set of operations designed to carry out a singular goal. project-based companies create a solution based on many products and sell it as a packaged solution to a problem. It is a way of executing a plan. The developer follows a service-oriented strategy. It carries out a software development process from requirement collection to testing and maintenance. It is carried out in compliance with implementation methods specified period for achieving the software package meant for use. Project-based software has a specific goal with a specific start date and end date. It is carried out in compliance with implementation achieving the software package meant for use. It is an iron triangle framework (up front, time, and budget) and very resistant to change when compared to product-based it is comparatively costly. It is a short term where teams will stay for the duration of the time.

For example: In the advertising company will take on a number of projects at once and plan how to make an advertisement for an item and once it is executed, it will be delivered to the customer. If a customer dislikes it, the ad will be changed depending on customer preferences.

Product-based software: Product-based companies make a specific product and try to market it as a solution. A product is a good, service, platform, application, system, etc., that is created, generally for sale, to meet customer and business needs. It is focused on tangible deliverables and outcomes and plans the project from the perceptive of the customer also it is focused on market demand or user expectation and shifting focus whenever the need arises. It also focused on a sustainable environment. Companies will decide when to change their product, which usually has a long life, and it is easier to pivot to other priorities if any new opportunity arises. It is focused on the delivery of value. There is no fixed end date, and it will be continued till discovery and development are needed. It is a long-term where teams stay together with slight changes.

Examples: Microsoft Office, in which the teams will keep updating MS Office as needed and new opportunities arise.

What are three important differences between software products and software product lines?

software product lines refer to software engineering methods, tools, and techniques for creating a collection of similar software systems from a shared set of software assets using a **common means of production**. A **software product** is software that has been developed and maintained for the benefit of a user base and often to satisfy a need in the market.

The software product line is used to deliver a product in a short span whereas the software product consumes time to deliver comparatively.

Software products represent high-end work that is done by vendors. The end-user should primarily interact with the software itself to solve a problem or derive value whereas the software product line will change the source code.

Why do software product managers have to be generalists with various skills, rather than technical specialists?

A project manager is a professional who organizes, plans, and executes projects while working within restraints like budgets and schedules. Project managers lead entire teams, define project goals, communicate with stakeholders, and see a project through to its closure. Whether running a marketing campaign, constructing a building, developing a computer system, or launching a new product, the project manager is responsible for the success or failure of the project. To be a generalist he needs to know the business, technical skills, and customer needs.

For **businesses** to be successful, they must ensure that the software they use meets both the business goals and objectives of the software product company and its customers. This is ultimately the area where the success or failure of our products will be decided. **Tech – To work** effectively with teams of developers, project managers have to have an understanding of the technology involved so we can speak a common language when we work together. When the team develops new product propositions, the Project manager needs to know what we can do with the technology we have and what its limitations are. **Customer needs** – This is our awareness of the customer and being able to take a value proposition and work with our colleagues in UX/UI to create a pleasing user experience for our customers

Why should you implement a prototype before you start developing a new software product?

An idea is the starting point for creating a new product. To determine a product's value, designers develop a prototype or working model before turning an idea into a finished product. Understanding the several types of prototypes is essential to decide which is the best for presenting an idea or testing a new design feature. The prototyping process is the foundation for generating innovative ideas by putting them into physical form. It helps designers turn a concept into a functioning item and it uses basic sketches and rough materials to help innovators decide what they need to improve and fix in their design.

Example: scale building models used in architectural design. An architect takes their understanding of a client’s wants and drafts blueprints to match, but the blueprints might not be enough. So, the architect builds a scale model of the building. In this case, it allows the client to see the plans for the building (or product) and provide feedback about what they like or don’t like. it’s easier and cheaper to make changes based on the early scale model prototype than when the finishing touches are being made to the building.

When creating a software product, prototyping is the ideal way to test, evaluate, and validate your idea with users. It lets you confirm that you are building the RIGHT product and features before you code anything. **prototyping reduces project risk.** After months of costly developer time and budget, you don't want to find out that the features don't meet real user needs. Instead, starting with a prototype lets you get user feedback on what key features to build. As some developers like to say, “If a picture is worth a thousand words, then a prototype is worth a thousand meetings. When building prototypes, they don’t need to include ALL the features you may need. Instead, focus only on the core features needed to solve your problem to ensure a quick and valuable feedback loop.