# Feature prioritization

# **What is Feature prioritization**

Feature prioritization is a critical task in product management, where you decide which features or enhancements should be developed and released first based on various factors like value, effort, impact, and alignment with business goals. Effective prioritization ensures that the product development team focuses on the most important features that deliver the highest value to users and the business.

# **Why Feature prioritization**

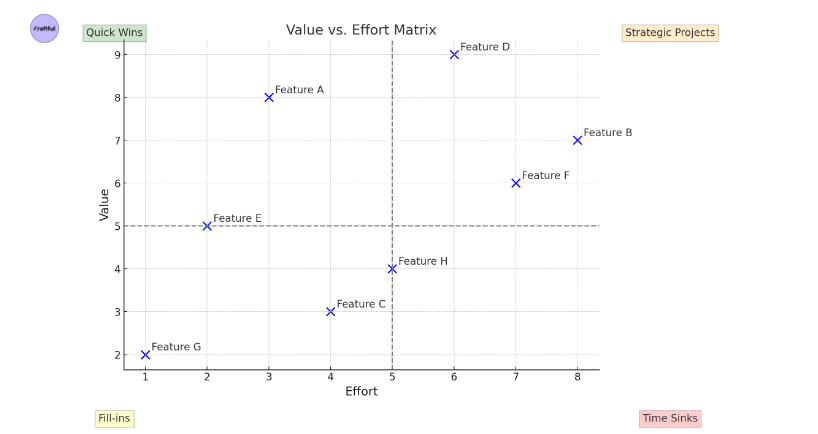
* **Resource Management:** Development resources (time, budget, people) are often limited, so prioritizing features ensures optimal use of these resources.
* **Strategic Alignment:** Helps ensure that the development work aligns with the company’s strategic goals.
* **User Satisfaction:** By prioritizing features that address the most pressing user needs, you improve user satisfaction and retention.
* **Market Relevance:** Keeps your product competitive by focusing on features that differentiate your product in the market.

# **Prioritization frameworks**

Feature prioritization frameworks help teams make informed decisions on which features to build, improve, or drop. Here’s a detailed look at some of the most popular frameworks:

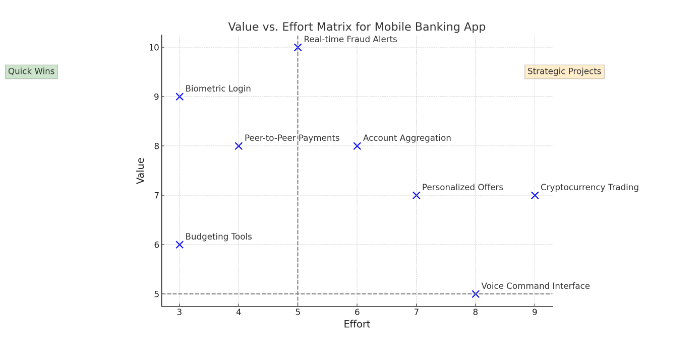
### **1. Value vs. Effort Matrix**

* **Overview:** This matrix helps you prioritize features by comparing their expected value (benefit) against the effort (resources) required to implement them.
* **Steps**:
  1. **List Features:** Write down all potential features.
  2. **Estimate Value:** For each feature, estimate the potential value it will bring to users or the business. Value can be assessed by how well the feature aligns with business goals, user needs, or competitive advantage.
  3. **Estimate Effort:** Estimate the effort in terms of time, resources, and complexity to build each feature.
  4. **Plot on the Matrix:** Place each feature on a 2x2 matrix where one axis is Value (Low to High) and the other is Effort (Low to High).
  5. Prioritize:
     + **High Value, Low Effort(Quick wins):** These are quick wins and should be prioritized.
     + **High Value, High Effort(strategic):** Consider these as strategic projects; they may take time but have significant impact.
     + **Low Value, Low Effort(fill ins):** These can be done if resources allow, but they’re not critical,indicating it can be worked on when there’s time available
     + **Low Value, High Effort(time sinks):** These are often not worth pursuing unless necessary for strategic reasons.



Example Features:

* **Feature A** is a quick win, offering high value with low effort.
* **Feature D** is a strategic project, with both high value and high effort.
* **Feature G** is a fill-in, providing low value with minimal effort.
* **Feature B** is a time sink, requiring high effort but offering less value.



This matrix helps prioritize which features to tackle first, ensuring that resources are used efficiently to maximize impact. ​​

* **Pros:** Simple and visual; easy to understand.
* **Cons:** Subjective estimates; doesn’t consider user impact in a nuanced way as its subjective estimation ,biased ,Bias Toward Measurable Value ,Overemphasis on Quick Wins

### **2.RICE Scoring**

* **Overview:** RICE stands for Reach, Impact, Confidence, and Effort. This framework provides a quantitative way to prioritize features based on these four criteria.
* **Steps:**
  1. **Reach:** Estimate how many people or users the feature will affect. This could be measured by the number of users, sessions, transactions, etc.

**Example:** If a feature will be used by 1,000 users per month, the Reach score is 1,000.

**Consideration:** Reach helps determine the scope of the feature's impact.

* 1. **Impact:** Estimate how much the feature will contribute to a specific goal, usually on a scale (e.g., 3 = high, 2 = medium, 1 = low)

**Definition:** How much the feature will affect an individual user or customer, typically measured on a scale (e.g., 1 to 5, where 5 is a massive impact and 1 is minimal impact).

**Example:** A feature that significantly improves the user experience might have an Impact score of 4.

**Consideration:** Impact assesses the magnitude of the benefit for each user.

* 1. **Confidence:** Rate your confidence in the accuracy of your Reach and Impact estimates, usually on a percentage scale

**Definition:** How confident the team is in the estimates of Reach and Impact. This is also scored on a percentage scale (e.g., 100% for very confident, 80% for somewhat confident).

**Example:** If the team has limited data, they might assign a Confidence score of 70%.

**Consideration:** Confidence adjusts the score to account for uncertainty, ensuring that projects with shaky assumptions are deprioritized.

* 1. **Effort:** Estimate the amount of time and resources needed to implement the feature. This is often measured in person-months or weeks.

**Definition:** How much time or resources are required to complete the feature, typically measured in person-months.

**Example:** If a feature takes 2 months of developer time, the Effort score is 2.

**Consideration:** Lower Effort scores contribute to higher prioritization.

* 1. **Calculate RICE Score:**

Use the formula: (Reach \* Impact \* Confidence) / Effort.

* 1. **Rank Features:** Higher RICE scores indicate higher priority

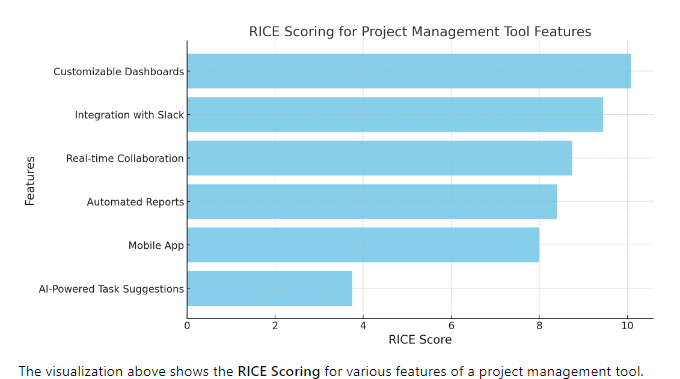
Example of RICE Scoring:

Imagine a team evaluating three different features:

1. **Feature A: Improved Onboarding Flow**
   * **Reach:** 2,000 users/month
   * **Impact:** 3 (medium)
   * **Confidence:** 90%
   * **Effort:** 2 months
   * **RICE Score:** 2000×3×0.90/2= 2,700
2. **Feature B: New Reporting Dashboard**
   * **Reach:** 500 users/month
   * **Impact:** 5 (high)
   * **Confidence:** 80%
   * **Effort:** 3 months
   * **RICE Score:** 500×5×0.83=667/3=667
3. **Feature C: Social Media Integration**
   * **Reach:** 3,000 users/month
   * **Impact:** 2 (low)
   * **Confidence:** 100%
   * **Effort:** 1.5 months
   * **RICE Score:** 3000×2×1.01.5=4,000\1.5=4,000

Prioritization Outcome:

* **Feature C** has the highest RICE score (4,000), suggesting it should be prioritized first.
* **Feature A** comes next with a score of 2,700, followed by **Feature B** with a score of 667.



* **Pros:** Provides a clear, data-driven decision; balances value and effort,accounts for uncertainty
* **Cons:** Requires accurate data and estimation; can be time-consuming
  1. **Subjectivity in Scoring:** Estimates for Reach, Impact, and Effort can still be subjective and may vary between team members.
  2. **Complexity:** The scoring process can be time-consuming, especially if the data is hard to gather.
  3. **Overemphasis on Quantifiable Factors:** Qualitative factors like strategic alignment or brand perception might be underrepresented.

RICE scoring provides a structured approach to feature prioritization, ensuring that the team focuses on the most valuable and feasible projects.

### **3.MoSCoW Method**

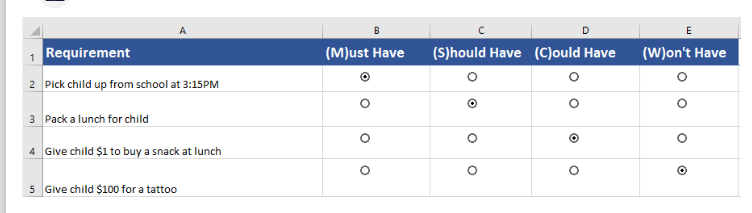
This method is particularly useful in managing scope and ensuring that the most critical elements of a project are completed first.

**Breakdown of the MoSCoW Framework:**

1. **Must Have**:
   * **Definition**: These are non-negotiable requirements that are critical to the success of the project. Without these, the project would fail or be deemed unsuccessful.
   * **Examples**: Core features that the product cannot function without, regulatory or legal requirements, essential functionality that must be delivered within a fixed timeframe.
   * **Considerations**: If any "Must have" item is not delivered, the project could be delayed or even canceled. It’s important to keep these to a minimum to ensure feasibility.
2. **Should Have**:
   * **Definition**: These are important but not vital requirements. They add significant value and are expected to be included if possible, but the project can still function without them.
   * **Examples**: Features that enhance user experience, add-ons that provide competitive advantage, or improvements that could boost performance.
   * **Considerations**: Should-have items are usually planned into the project but are the first to be rescheduled or scoped out if time, resources, or budget constraints arise.
3. **Could Have**:
   * **Definition**: These are desirable but not necessary requirements. They are often considered as enhancements or nice-to-haves that can be implemented if resources and time allow after the Must have and Should have items are completed.
   * **Examples**: Cosmetic changes, minor feature enhancements, or future-proofing additions that don't impact the current operation.
   * **Considerations**: These items often form a backlog for future phases or iterations and are the least likely to be included in the initial project scope.
4. **Won’t Have (or Would like to have but not this time)**:
   * **Definition**: These are items that have been agreed upon as the least critical or outside the current project scope. They are not planned for delivery in the current phase but may be revisited in the future.
   * **Examples**: Features that are out of scope for the current release, ideas that are interesting but not aligned with the current project goals.
   * **Considerations**: Clearly identifying these items helps manage stakeholder expectations and keeps the project focused on delivering value within the agreed constraints.

Benefits of Using MoSCoW:

* **Clarity and Focus**: It helps teams and stakeholders clearly understand and agree on what’s most important, preventing scope creep.
* **Resource Management**: By prioritizing tasks, teams can allocate resources more efficiently, ensuring that critical elements are completed on time.
* **Flexibility**: MoSCoW allows for flexibility in managing changes. If unexpected issues arise, the project can adapt by shifting the focus to Must-have items and postponing less critical work.
* **Stakeholder Alignment**: It fosters communication and alignment among stakeholders, ensuring that everyone is on the same page regarding priorities.



**Cons**

1. Over-Categorization of Requirements:

* **Challenge**: Teams may struggle to avoid placing too many requirements into the "Must Have" category. This can lead to overloading the project scope with high-priority tasks, making it difficult to deliver within time and budget constraints.
* **Impact**: This dilutes the effectiveness of the prioritization, as everything cannot be a top priority​

2. Subjectivity in Prioritization:

* **Challenge**: The process of determining what falls into each category can be highly subjective, depending on the stakeholders' perspectives and interests.
* **Impact**: Disagreements among team members or stakeholders can arise, leading to conflicts or the need for repeated negotiations to agree on priorities​

3. Neglect of Non-Critical Features:

* **Challenge**: Features categorized as "Should Have," "Could Have," and especially "Won’t Have" might be neglected or deprioritized, even if they are valuable to the product or user experience.
* **Impact**: This can result in a product that meets its basic functional requirements but lacks the enhancements that make it competitive or appealing to users​

4. Potential for Misalignment with Stakeholders:

* **Challenge**: If stakeholders are not fully involved or aligned with the prioritization process, there may be misalignment between their expectations and the project's outcomes.
* **Impact**: This can lead to dissatisfaction, especially if stakeholders feel their needs are not being met or if they do not agree with the prioritization outcomes​

5. Lack of Flexibility:

* **Challenge**: MoSCoW is often seen as a rigid framework, which can be limiting in dynamic projects where requirements or priorities frequently change.
* **Impact**: The inflexibility can hinder the team’s ability to adapt to new information or changing project dynamics, which is especially problematic in agile environments​

6. Over-Reliance on Initial Prioritization:

* **Challenge**: Once the initial MoSCoW prioritization is set, there may be resistance to revisiting and re-prioritizing tasks as the project progresses.
* **Impact**: This can lead to a lack of responsiveness to new challenges or opportunities that arise during the project lifecycle​

7. Risk of Neglecting Lower Priority Items:

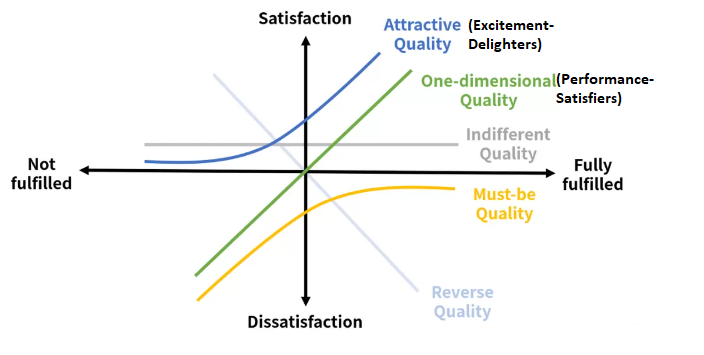
* **Challenge**: Items in the "Could Have" and "Won’t Have" categories may be overlooked or forgotten entirely.
* **Impact**: This can result in missed opportunities for delivering additional value or addressing user needs that, while not critical, are still important​ ([Creately](https://creately.com/guides/moscow-method/)).

Despite these drawbacks, the MoSCoW method remains a widely used tool for managing project priorities

### **4.Kano Model**

Developed by Professor Noriaki Kano in the 1980s -This model is particularly useful for understanding and prioritizing customer satisfaction and delight, helping teams to differentiate between essential features and those that can significantly enhance user satisfaction.

These categories help product managers and teams prioritize features by understanding how each feature influences customer happiness.



Kano Model Categories

1. **Must-Be (Basic expectations )**:
   * **Definition**: These are fundamental features that customers expect. Their absence leads to dissatisfaction, but their presence doesn’t significantly increase satisfaction because they are assumed to be a given.
   * **Examples**: A mobile phone must have basic calling functionality, or a website must load when accessed.
   * **Impact**: These are "hygiene factors" that do not necessarily delight the customer but are necessary to avoid dissatisfaction.
2. **One-Dimensional (Performance Needs)**:
   * **Definition**: These features have a direct relationship with customer satisfaction. The better you execute on these features, the more satisfied customers will be.
   * **Examples**: Faster load times on a website, better battery life on a smartphone.
   * **Impact**: Performance needs provide a competitive advantage and are often the focus for continuous improvement.
3. **Attractive (Excitement Needs)**:
   * **Definition**: These are features that customers do not expect but are delighted when they experience them. They can create significant positive feelings.
   * **Examples**: A free add-on service, surprise discounts, or innovative design features.
   * **Impact**: Attractive features can set a product apart from competitors and increase customer loyalty.
4. **Indifferent**:
   * **Definition**: Features that customers are indifferent about; they neither increase nor decrease satisfaction.
   * **Examples**: Features that are nice to have but do not influence the customer’s purchase decision.
   * **Impact**: These features should be minimized or avoided as they do not add significant value.
5. **Reverse**:
   * **Definition**: Features that some customers may like, but others may dislike. These can create satisfaction for one group and dissatisfaction for another.
   * **Examples**: A highly customizable interface might be appreciated by power users but confusing for average users.
   * **Impact**: These features require careful consideration and often need to be tailored to specific customer segments.



Applying the Kano Model

1. **Customer Surveys**: Use surveys to ask customers how they would feel if a feature was present or absent. This can help you categorize features into the Kano categories.
2. **Prioritization**: Once features are categorized, focus on ensuring all Must-Be features are included, optimizing One-Dimensional features, and strategically adding Attractive features to surprise and delight customers.
3. **Review and Iterate**: As customer expectations evolve, regularly review and update the feature categories to ensure alignment with current market demands.

Pros and Cons of the Kano Model

* **Pros**:
  + Helps in prioritizing features based on actual customer satisfaction.
  + Encourages a focus on innovation and customer delight.
  + Provides a framework for balancing basic needs with exciting features.
* **Cons**:
  + Requires detailed customer feedback, which can be time-consuming to gather.
  + May be challenging to categorize features accurately without clear customer input.
  + The model's qualitative nature can lead to subjective interpretations of results.

The Kano Model is a valuable tool for product managers who want to prioritize features based on customer satisfaction and competitive differentiation. It complements other frameworks like MoSCoW by adding a focus on the emotional impact of features on users.

### **5.Eisenhower Matrix -Urgent important matrix**

U.S. President Dwight D. Eisenhower :This framework is particularly useful for time management and prioritizing tasks based on their urgency and importance,sorting out less urgent and important tasks which you should either delegate or not spend much time on.



**The Four Quadrants of the Eisenhower Matrix**

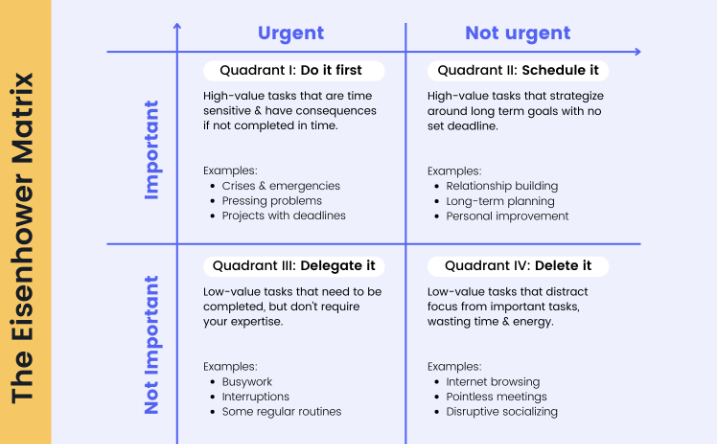
1. **Quadrant 1: Urgent and Important (Do First)**:
   * **Definition**: Tasks that are both urgent and important. These require immediate attention and are crucial to achieving your goals.
   * **Examples**: Crises, deadlines, pressing problems.
   * **Action**: Do these tasks as soon as possible.
2. **Quadrant 2: Not Urgent but Important (Schedule)**:
   * **Definition**: Important tasks that are not urgent. These contribute to long-term goals and development.
   * **Examples**: Planning, strategy, learning new skills, relationship-building.
   * **Action**: Schedule these tasks for a later time, as they are critical for long-term success.
3. **Quadrant 3: Urgent but Not Important (Delegate)**:
   * **Definition**: Tasks that are urgent but not important. They often distract from your main goals.
   * **Examples**: Interruptions, some emails or calls, requests from others.
   * **Action**: Delegate these tasks to others if possible, or minimize the time spent on them.
4. **Quadrant 4: Not Urgent and Not Important (Eliminate)**:
   * **Definition**: Tasks that are neither urgent nor important. These are often time-wasters.
   * **Examples**: Busywork, trivial activities, excessive internet browsing.
   * **Action**: Eliminate or reduce these tasks, as they do not contribute to your success.

**Benefits of the Eisenhower Matrix**

* **Clarifies Priorities**: Helps you see which tasks are truly important versus those that feel urgent but aren’t crucial.
* **Improves Time Management**: Encourages focusing on tasks that matter most and reduces time spent on non-essential activities.
* **Reduces Stress**: By clearly organizing tasks, you can avoid the overwhelm that comes from handling too many things at once.

**Example**

Imagine you’re a Product Manager:

****

**Pros**:

1. **Clarity and Focus**:
   * The Eisenhower Matrix provides a clear structure for decision-making by categorizing tasks into four quadrants: Important/Urgent, Important/Not Urgent, Not Important/Urgent, and Not Important/Not Urgent. This helps in identifying what truly needs immediate attention and what can be deferred or delegated.
2. **Improves Productivity**:
   * By focusing on important tasks, especially those that are not urgent (Quadrant II), users can work on strategic goals and prevent crises from occurring, ultimately leading to increased productivity and efficiency.
3. **Reduces Stress**:
   * Prioritizing tasks based on urgency and importance can reduce the overwhelming feeling that comes with having too many tasks. By tackling high-priority tasks first, you can alleviate stress and maintain better control over your workload.
4. **Encourages Delegation**:
   * The matrix promotes effective delegation by helping users identify tasks that are urgent but not important (Quadrant III). Delegating these tasks allows individuals to focus on more critical activities.
5. **Enhances Long-term Planning**:
   * By regularly using the Eisenhower Matrix, users are encouraged to focus on long-term goals and strategic planning, rather than just reacting to immediate issues.

**Cons**:

1. **Oversimplification**:
   * The matrix categorizes tasks into only four quadrants, which might oversimplify complex tasks or situations. Some tasks might not fit neatly into one category, making the decision process more challenging.
2. **Subjectivity**:
   * The determination of what is "important" or "urgent" can be subjective and may vary from person to person. This subjectivity can lead to inconsistent prioritization, especially in team settings.
3. **Neglect of Quadrant II**:
   * While the matrix emphasizes the importance of Quadrant II (Important/Not Urgent), users may still gravitate towards urgent tasks, neglecting important but non-urgent tasks that are crucial for long-term success.
4. **Does Not Consider Task Dependencies**:
   * The matrix does not account for tasks that are dependent on others or require sequential completion, which can be a limitation in more complex project management scenarios.
5. **Time-Consuming**:
   * Regularly categorizing tasks into the matrix can become time-consuming, particularly if you have a large number of tasks or if your priorities frequently change.

### **6.CoD - Cost of Delay**

**Cost of Delay (CoD)** helps to quantify the economic impact of delaying a project or feature. It helps prioritize work by focusing not just on the value of delivering something, but also on the cost of not delivering it in a timely manner.

Key Concepts

1. **Value:** This is the benefit or revenue that a feature or project is expected to generate.
2. **Time Sensitivity:** This represents how quickly the value of a feature decreases over time. Some features are more time-sensitive than others.
3. **Urgency:** This is the degree to which a delay impacts the overall value. Higher urgency means a higher cost of delay.

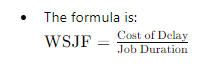
Why Cost of Delay Matters

* **Informed Prioritization:** It helps teams prioritize features or projects that are time-sensitive and would result in higher economic losses if delayed.
* **Resource Allocation:** It guides resource allocation towards work that will have the greatest economic impact if delivered sooner.
* **Focus on Value:** It shifts focus from simply delivering features to delivering them at the right time to maximize value.

The formula for calculating Cost of Delay is:



However, in many practical applications, teams use simpler approaches:

1. **Simple Numerical CoD:**
   * Assign a dollar value to the potential revenue or cost savings a feature would generate.
   * Estimate how much value is lost per time unit (e.g., per week or month) if the feature is delayed.
2. **Weighted Shortest Job First (WSJF):**
   * WSJF is a prioritization technique that uses CoD divided by the job duration (or effort).
   * 
   * Prioritize the work with the highest WSJF score, meaning it provides the most value in the shortest time.

Types of Cost of Delay

1. **Immediate Cost of Delay:** 
   * Losses that begin as soon as a feature is delayed, such as lost sales or penalties.
2. **Future Cost of Delay:**
   * Potential future revenue or benefits that are reduced or lost because a feature is delivered late.
3. **Accumulated Cost of Delay:**
   * Continuous and cumulative losses over time until the feature is delivered.

Visualizing Cost of Delay

* **CoD Curve:** This curve illustrates how the value of a feature diminishes over time. A steep curve indicates high urgency, meaning delays significantly decrease the potential value.
* **Cost of Delay Profiles:**
  + **Standard Profile:** Value decreases at a constant rate over time.
  + **Accelerating Profile:** Value decreases slowly at first but then accelerates rapidly.
  + **Fixed Deadline Profile:** Value remains stable until a certain deadline, after which it drops sharply.

### 

Here's a diagram illustrating the concept of Cost of Delay over time.

* The **blue curve** represents how the value of a feature decreases as time progresses.
* Initially, the feature holds high value, but as time passes (e.g., due to delays), the value diminishes, highlighting the increasing cost of delay.

This visual helps to understand that the longer a valuable feature is delayed, the more potential value is lost, emphasizing the importance of timely delivery. ​​

Example

Imagine you have two features:

* **Feature A**: Expected to generate $50,000/month once released.
* **Feature B**: Expected to generate $30,000/month, but its value drops by $10,000 every month it's delayed.

If Feature A takes 2 months to build, delaying it by 1 month results in a cost of delay of $50,000. For Feature B, delaying it by 1 month costs $10,000, but if delayed by 2 months, the cost is $20,000.

Using Cost of Delay

To effectively use CoD:

* **Estimate the economic impact:** Determine the value of delivering the feature or project on time.
* **Understand time sensitivity:** Evaluate how quickly the value decreases with delay.
* **Apply WSJF if necessary:** Use the WSJF method to combine CoD with job duration for more nuanced prioritization.

Conclusion

Cost of Delay is a powerful tool that helps organizations focus on delivering value at the right time by considering the economic impact of delays. It aligns the team's priorities with business objectives, ensuring that the most time-sensitive and valuable work is prioritized

### **7.Placeholder**