Planning and Executing Sprints Using the Scrum Framework

## Set the Vision

A project vision is a high-level statement that defines the purpose, goals, and desired outcomes of a project. It provides a clear direction for the project team and stakeholders and helps to align everyone around a shared understanding of what the project is trying to achieve. Here are some tips for writing an effective project vision:

1. **Start with the problem**: A project vision should start with a clear description of the problem that the project is trying to solve. This helps to focus the team on the needs of the user or customer and provides a clear rationale for why the project is necessary.
2. **Be concise**: A project vision should be short and to the point. It should be easy to remember and communicate and should capture the essence of the project in just a few sentences.
3. **Use plain language**: A project vision should be written in language that is easy to understand, even for non-technical stakeholders. Avoid using jargon or technical terms that might be confusing to those outside the project team.
4. **Be inspirational:** A project vision should inspire and motivate the team to work towards a common goal. It should convey a sense of purpose and the potential for significant impact.
5. **Define success**: A project vision should clearly define what success looks like for the project. This could include specific metrics, goals, or outcomes that the project is aiming to achieve.
6. **Align with strategy**: A project vision should be aligned with the broader strategic goals of the organization. It should support the organization's mission and values and contribute to the achievement of its long-term goals.
7. **Involve stakeholders**: A project vision should be developed in collaboration with key stakeholders, including the project team, customers, and other relevant parties. This helps to ensure that everyone has a shared understanding of the project's purpose and goals and increases buy-in and support for the project.

## Decompose the Vision

Decomposing a project vision into its functional parts involves breaking down the overall vision into smaller, more manageable components or features that can be worked on by the project team. Here are some steps to effectively decompose a project vision into its functional parts:

1. **Identify the high-level features**: Start by identifying the high-level features or components that make up the project vision. These should be the major functional areas that the project will address.

**For example**: Let's say the project vision is to build a mobile app for a food delivery service. The high-level features might include user registration and login, order placement and tracking, menu browsing and selection, payment processing, and delivery scheduling.

1. **Define the user stories**: Once you have identified the high-level features, break them down further into specific user stories. User stories describe a user's goal or need, and the functionality required to meet that need.

**For example:** For the order placement and tracking feature, some example user stories might include "As a customer, I want to be able to browse the restaurant menu and select items to add to my order", "As a customer, I want to be able to track the status of my order and receive notifications when it's on its way", and "As a customer, I want to be able to rate my delivery and provide feedback on my experience".

1. **Prioritize the user stories**: Prioritize the user stories based on their importance to the project vision and the potential impact they will have on the user or customer.
2. **Estimate the effort required**: Estimate the effort required to implement each user story, considering any dependencies or technical constraints.  
     
   To elaborate; here are some examples of dependencies and technical considerations that might need to be included in the product backlog:

* **Integration dependencies:** If the app needs to integrate with external systems, such as payment gateways or third-party APIs, the user stories related to those integrations might need to be prioritized and implemented before other user stories.
* **Backend dependencies**: If a user story requires changes to the backend database schema or APIs, any dependent user stories may need to be prioritized and implemented first to ensure that the backend components are properly updated.
* **Technical limitations**: If the project is subject to technical limitations, such as hardware or software requirements, those limitations should be taken into account when prioritizing and estimating user stories.
* **Architecture considerations**: If the project has specific architectural requirements, such as using microservices or serverless functions, those requirements should be considered when defining and estimating user stories.

1. **Create a product backlog:** Create a product backlog that includes all the user stories and their priorities. The product backlog should be regularly reviewed and updated throughout the project to reflect changing priorities and requirements.
2. **Break down the user stories into tasks**: Once the user stories have been defined, break them down further into tasks that can be completed by individual team members. This will help to ensure that each user story is broken down into manageable pieces that can be completed within a single sprint.  
     
   **For example:** For the "menu browsing and selection" user story, some example tasks might include   
     
   designing the user interface for the menu, implementing the backend logic to retrieve and display the menu items, and writing automated tests to verify the functionality.
3. **Assign tasks to team members**: Assign tasks to individual team members based on their skills and availability. This will help to ensure that each team member has a clear understanding of their responsibilities and can work effectively towards the project vision.

By following these steps, you can effectively decompose a project vision into its functional parts, creating a clear roadmap for the project team to follow and ensuring that the project stays on track towards achieving its goals.

## Writing User Stories

User stories are an important tool for communicating customer needs and requirements to a Scrum team. Effective user stories should be clear, concise, and focused on the user's needs. Here are some tips for writing effective user stories for a Scrum team:

1. **Start with the user**: User stories should always be written from the perspective of the user. They should focus on the user's needs, desires, and goals. Begin each story with a user persona, describing who the user is and what they want to achieve.
2. **Use the** "As a...I want...So that..." format: This is a common format for user stories that helps to keep them concise and focused on the user's needs. The "As a..." describes the user persona, the "I want..." describes what the user wants to achieve, and the "So that..." describes the value or benefit the user will get from achieving their goal.
3. **Make them specific and measurable**: User stories should be specific enough to guide the team's work and measurable so that progress can be tracked. Use concrete language and avoid ambiguity or vagueness.
4. **Keep them small and manageable**: User stories should be small enough to fit within a single sprint or iteration. This helps the team to focus on a clear and achievable goal, and also helps to reduce risk by allowing for frequent feedback and adjustments.
5. **Prioritize them**: Prioritizing user stories is important for ensuring that the team is working on the most valuable items first. Use a prioritization method such as MoSCoW (Must have, Should have, Could have, Won't have) to help the team understand which stories are most important.
6. **Involve the team**: User stories are not just the responsibility of the product owner. The entire Scrum team should be involved in writing and refining user stories, as this helps to ensure that everyone has a shared understanding of the user's needs and goals.
7. **Review and refine them regularly**: User stories should be reviewed and refined regularly, as new information becomes available, or priorities change. Use retrospectives and other feedback mechanisms to continuously improve the quality and relevance of user stories.

#### **References**

Linked in Learning: <https://www.linkedin.com/learning/scrum-the-basics/practicing-scrum-in-your-work-environment?autoplay=true&resume=false&u=2104084>

Atlassian: <https://www.atlassian.com/agile/scrum>