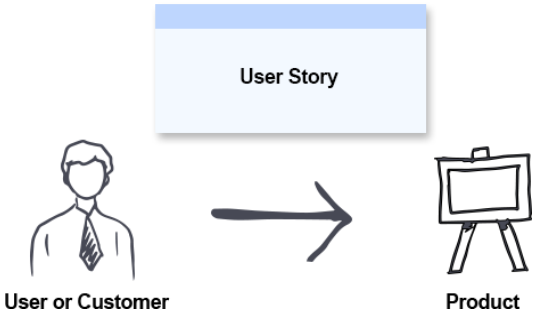
User Stories [week-7 theory]

What are user stories?



- A user story is the smallest unit of work in an agile framework. It's an end goal, not a feature, expressed from the software user's perspective. A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer.
- In software development and product management, a user story is an informal, natural language description of one or more features of a software system. A user story is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. A user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement.
- User stories are often recorded on index cards, on post-it notes, or in project management software. Depending on the project, user stories may be written by various stakeholders such as clients, users, managers or development team members.
- "User stories are part of an agile approach that helps shift the focus from writing about requirements to talking about them. All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality"

Why user story?

- A user story helps to create a simplified description of a requirement.
- The purpose of a user story is to write down how a project will deliver value back to the end user. It is then the development team's job to take care of how to develop the code that will satisfy the requirements of the user story.

User stories serve a number of key benefits:

1. **Stories keep the focus on the user.** A to-do list keeps the team focused on tasks that need to be checked off, but a collection of stories keeps the team focused on solving problems for real users.
2. **Stories enable collaboration.** With the end goal defined, the team can work together to decide how best to serve the user and meet that goal.
3. **Stories drive creative solutions.** Stories encourage the team to think critically and creatively about how to best solve for an end goal.
4. **Stories create momentum**. With each passing story, the development team enjoys a small challenge and a small win, driving momentum.

Basic concepts of user story

- A user story is a lightweight method for quickly capturing the "who", "what" and "why" of a product requirement.
- In simple terms, user stories are stated ideas of requirements that express what users need.
- User stories are brief, with each element often containing fewer than 10 or 15 words each.
- User stories are "to-do" lists that help you determine the steps along the project's path.
- They help ensure that your process, as well as the resulting product, will meet your requirements.

Characteristics of user story

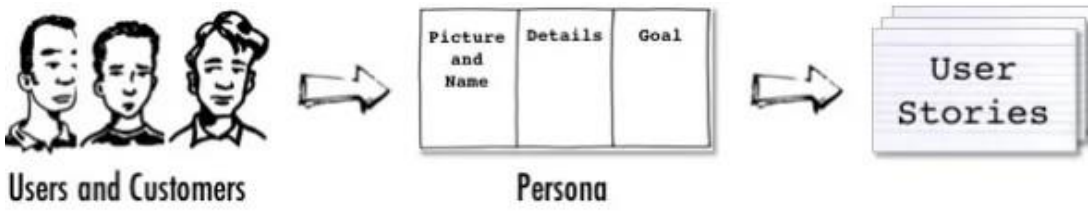
1) Be complete enough to demonstrate user value.
2) Be user-centric.
3) Start with an epic.
4) Be short, simple, and clear.
5) Contain supporting files and documentation if necessary.
6) Be comprehensive enough to demonstrate value, but simple enough to develop in a single iteration.
7) Be written based on the input of all stakeholders.
8) Be flexible and negotiable without impacting other stories or features.
9) Be easy to test.
10) Include acceptance criteria (conditions of satisfaction) for testers.

How to write/create user stories? Steps

1. **Users Come First**
   - As its name suggests, a user story describes how a customer or user employs the product; it is told from the user's perspective. What's more, user stories are particularly helpful to capture a specific functionality, such as, searching for a product or making a booking.

   - The picture illustrates the relationship between the user, the story, and the product functionality, symbolised by the circle. If you don't know who the users and customers are and why they would want to use the product, then you should not write any user stories. Carry out the necessary user research first, for example, by observing and interviewing users.

2. **Use Personas to Discover the Right Stories**
   - A great technique to capture your insights about the users and customers is working with personas. Personas are fictional characters that are based on first-hand knowledge of the target group.



- They usually consist of a name and a picture; relevant characteristics, behaviours, and attitudes; and a goal.
- The goal is the benefit the persona wants to achieve, or the problem the character wants to see solved by using the product. But there is more to it: The persona goals help you discover the right stories.

3. Create Stories Collaboratively

- User stories are intended as a lightweight technique that allows you to move fast. They are not a specification, but a collaboration tool.

- Stories should never be handed off to a development team. Instead, they should be embedded in a conversation: The product owner and the team should discuss the stories together. This allows you to capture only the minimum amount of information, reduce overhead, and accelerate delivery.

4. Keep your Stories Simple and Concise

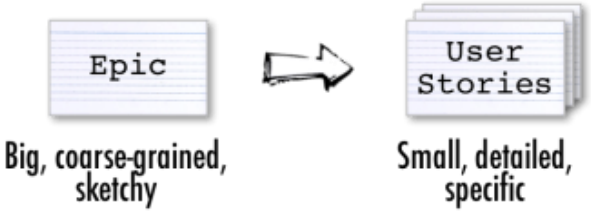
- Write your stories so that they are easy to understand. Keep them simple and concise. Avoid confusing and ambiguous terms, and use active voice. Focus on what's important, and leave out the rest.

5. Start with Epics

- An epic is a big, sketchy, coarse-grained story. It is typically broken into several user stories over time - leveraging the user feedback on early prototypes and product increments.
- You can think of it as a headline and a placeholder for more detailed stories. Starting with epics allows you to sketch the product functionality without committing to the details.
- This is particularly helpful for describing new products and features: It allows you to capture the rough scope, and it buys you time to learn more about how to best address the needs of the users. It also reduces the time and effort required to integrate new insights.

6. Refine the Stories until They are Ready

- Break your epics into smaller, detailed stories until they are ready: clear, feasible, and testable.



- All development team members should have a shared understanding of the story's meaning; the story should not be too big and comfortably fit into a sprint; and there has to be an effective way to determine if the story is done.

7. Add Acceptance Criteria

- As you break epics into smaller stories, remember to add acceptance criteria.
- Acceptance criteria complement the narrative: They allow you to describe the conditions that have to be fulfilled so that the story is done.
- The criteria enrich the story, they make it testable, and they ensure that the story can be demoed or released to the users and other stakeholders.

8. Use (Paper) Cards

- User stories emerged in Extreme Programming (XP), and the early XP literature talks about story cards rather than user stories.
- There is a simple reason: User stories used to be captured on paper cards.
- This approach provides three benefits:
  First, paper cards are cheap and easy to use.
  Second, they facilitate collaboration: Everyone can take a card and jot down an idea.
  Third, cards can be easily grouped on the table or wall to check for consistency and completeness and to visualise dependencies. If using paper cards is not an option for you, then choose a tool that allows you to create virtual cards, as Trello does, for ex.

9. Keep your Stories Visible and Accessible

- Stories want to communicate information. Therefore, don't hide them on a network drive, the corporate intranet jungle, or a licensed tool. Make them visible, for instance, by putting them up on the wall.

- This fosters collaboration, creates transparency, and makes it obvious when you add too many stories too quickly, as you will start running out of wall space. A handy tool to discover, visualise, and manage your stories is my product canvas shown above.

Remember: User stories are not about documenting requirements. They want to enable you to move fast and develop software as quickly as possible - not to impose any overhead.

3C's in user stories

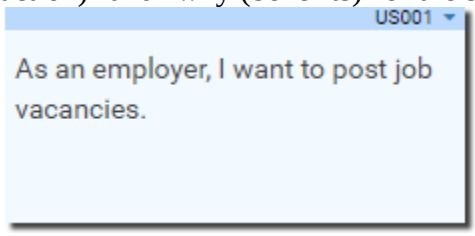
A User Story has three primary components, each of which begin with the letter 'C': Card, Conversation, and Confirmation to describe the three elements of a user story.

1. **Card**
   - Card represents 2-3 sentences used to describe the intent of the story that can be considered as an invitation to conversation.
   - The card serves as a memorable token, which summarizes intent and represents a more detailed requirement, whose details remain to be determined.
   - You don't have to have all of the Product Backlog Items written out perfectly "up front", before you bring them to the team.
   - It acknowledges that the customer and the team will be discovering the underlying business/system needed as they are working on it.
   - This discovery occurs through conversation and collaboration around user stories. The Card is usually following the format similar to the one below:

As a (role) of the product, I can (do action) so that I can obtain (some benefits / value)

Note:

The written text, the invitation to a conversation, must address the "who (role)", "what (action)" and "why (benefits)" of the story.



2. **Conversation**
   - Conversation represents a discussion between the target users, team, product owner, and other stakeholders, which is necessary to determine the more detailed behaviour required to implement the intent.
   - In other words, the card also represents a "promise for a conversation" about the intent.
   - The collaborative conversation facilitated by the Product Owner which involves all stakeholders and the team.
   - The conversation is where the real value of the story lies and the written Card should be adjusted to reflect the current shared understanding of this conversation.
   - This conversation is mostly verbal but most often supported by documentation and ideally automated tests of various sorts (e.g., Acceptance Tests).

3. **Confirmation**
   - Confirmation represents the Acceptance Test, which is how the customer or product owner will confirm that the story has been implemented to their satisfaction.

- In other words, Confirmation represents the conditions of satisfaction that will be applied to determine whether or not the story fulfils the intent as well as the more detailed requirements.
- The Product Owner must confirm that the story is complete before it can be considered "done"
- The team and the Product Owner check the "doneness" of each story in light of the Team's current definition of "done"
- Specific acceptance criteria that is different from the current definition of "done" can be established for individual stories, but the current criteria must be well understood and agreed to by the Team. All associated acceptance tests should be in a passing state.

Life cycle of user story

In a broad sense, there are six main states for each user story throughout a software project:

1. **Pending**
   Through the communication between user and project team, user stories are found. At this state, the user stories have nothing more than a short description of user's need. There is no detailed discussion of requirements, no system logic and no screen design yet. In fact, the only purpose of user story, for now, is just for reminding all parties for a future discussion of user's request written in this user story (card). It is possible that the user story will be discarded in the future.

2. **To-do**
   Through a discussion between different stakeholders, the user stories to be addressed in the next few weeks are decided, and are put into a time-box called a sprint. Such user stories are said to be in the to-do state. No detailed discussion has yet been carried out in this state.

3. **Discussing**
   When a user story is in the Discussing state, the end user will communicate to the development team in confirming the requirements as well as to define the acceptance criteria. Development team will write down the requirements or any decisions as conversation notes. UX specialist may create wireframes or storyboards to let user preview the proposed features in visual mock-ups, and to feel it. This process is known as user experience design (UX design).

4. **Developing**
   After the requirements are clarified, the development team will design and implement the features to fulfil user's requests.

5. **Confirming**
   Upon the development team has implemented a user story, the user story will be confirmed by the end user. He/she will be given access to the testing environment or a semi-complete software product (sometimes known as an alpha version) for confirming the feature. Confirmation will be performed based on the confirmation items written when detailing the user story. Until the confirmation is done, the user story is said to be in the Confirming state.

6. Finished

Finally, the feature is confirmed to be done, the user story is considered in the Finished state. Typically, this is the end of the user story. If user has a new requirement, either it is about a new feature, or it is an enhancement of the finished user story, the team would create a new user story for the next iteration.

User story map

- A user story map can help us to arrange user stories into a manageable model for plan, understand and organize the functionality of the system systematically.
- By manipulating the structure of the map, we can identify holes and omissions in your backlog and interrelating the user stories in a meaning structure; helping plan holistic releases effectively that deliver value to users and business with each release.
- User story map allows you to add a second dimension to your backlog.

Here are a few reasons you should consider using this technique:

- It allows you to see the big picture in your backlog.
- It gives you a better tool for making decisions about grooming and prioritizing your backlog.
- It promotes silent brainstorming and a collaborative approach to generating your user stories.
- It encourages an iterative development approach where your early deliveries validate your architecture and solution.
- It is a great visual alternative to traditional project plans.
- It is a useful model for discussing and managing scope.
- Allows you to visualize dimensional planning and real options for your project/product.

Estimation: User story point: basics

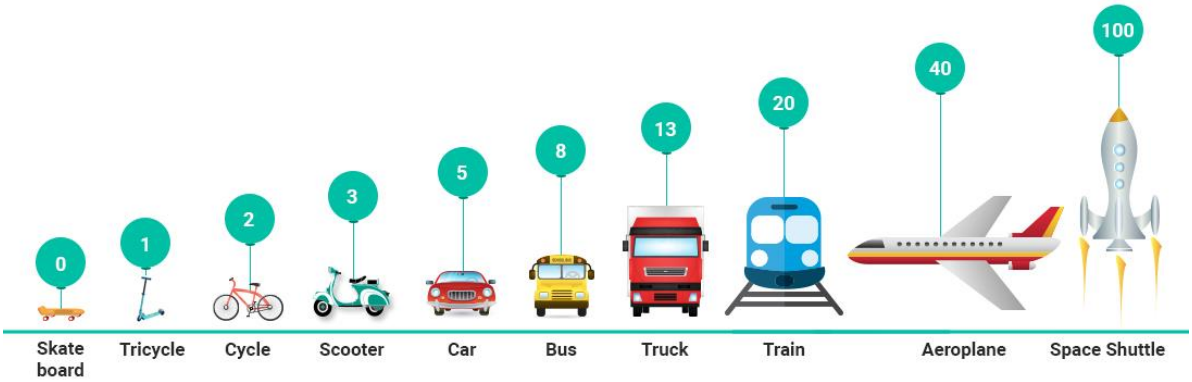
- Story points represent the relative sizing of the user story. It is a unit of estimation used by Agile teams to estimate User Stories.
- When the product owner wants some features to be developed, he/she desires to know how soon the team can complete the features and how many resources it will take to complete the work.
- From the developer's perspective, it's next to impossible to predict the exact time in which he/she can complete the work.
- The person can, however, give a rough estimate in terms of how much time it might take to complete the work.
- Note that instead of "will" the developer chose to use "might" because he/she is not absolutely "sure" about the time factor but "feels" it might take that much time. This is user story estimation in a nutshell.

You don't give an exact number explaining how complex the story is and how long it'll take to develop - you give a rough "estimate".

- We are good at comparing size, so estimating a story using Fibonacci series sequence (0, 1, 2, 3, 5, 8, 13, 20, 40, and 100) gives more clarity of its complexity and relative

sizing in terms of development. It is helpful to have a set of stories nearby to make a comparison and recommendation to set priority.

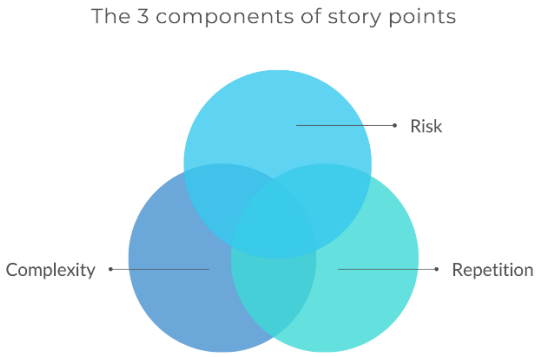
Here are the examples of relative sizing and its estimation points to develop the following vehicles:



Components of story point estimation

Story point estimation includes three main components:

1. **Risk:** The risk of a particular project or item includes vague demands, dependence on a third party, or changes mid-task.
2. **Complexity:** This component is determined by how difficult the feature is to develop.
3. **Repetition:** This component is determined by how familiar the team member is with the feature and how monotonous certain tasks are within development.



Steps involved in estimation

Step 1 - Use Fibonacci sequence numbers

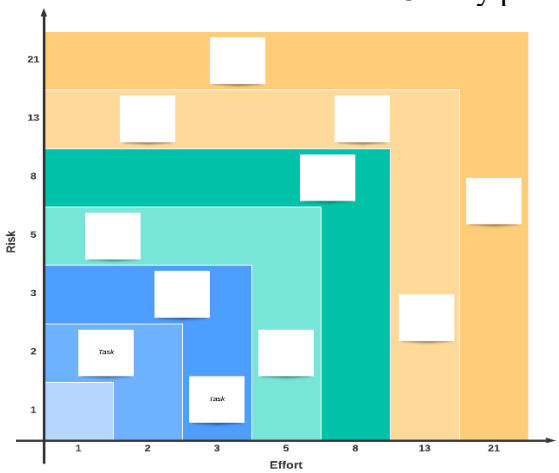
Why Fibonacci and not a linear scale?

- There are two types of scales used for creating estimation matrices: **the linear scale** and **Fibonacci sequence.**
- It's tempting to assign items with a linear scale, but those integers aren't differentiated enough to clearly define an estimate.

Ex: You've likely encountered this at the doctor's office with a pain scale. If 1 on the pain scale represents "totally fine" and 10 is a pain so severe it feels like you may be dying, what is 4? And, furthermore, how is 4 different from 5? And where does a kidney stone fit in on the scale if you've never experienced severe pain before?

- Fibonacci sequence numbers eliminate those minor jumps. As you might remember, the Fibonacci sequence is a series of numbers where each number is the sum of the two previous numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, etc.

- For Agile, the sequence is typically modified to 0.5, 1, 2, 3, 5, 8, 13, etc. Using these numbers, it's much easier to decide if an item is 3 story points or 5 story points.



Step 2 – Create a story point estimation matrix

After you've decided to use the Fibonacci sequence, it's time to determine a baseline for each story point. For instance:

1 = Add a new product to a dropdown menu
2 = Add order tracking for logged-in users
3 = Add a ratings system to the website
5 = Add a forum to the site
8 = Add GDPR and CCPA compliance across the site

- Your baseline is included in this matrix as 1, which sets the standard for what the least amount of risk, complexity, and repetition looks like in practice.
- This matrix is a way to more concretely measure effort; keep this in mind instead of defaulting to judging items based only on length of time.

Step 3 – Play planning poker to decide on story points

- Planning poker is a great way to have the team agree on the correct story point approximation for every item in the backlog.
- During the sprint planning meeting, each developer receives a set of cards depicting the Fibonacci sequence.
- A backlog item is brought to the table so that the team may ask questions and clarify features.
- When the discussion is closed, each developer and tester privately select the card that most accurately reflects their estimate.
- When all cards have been selected, the estimators reveal their cards at the same time. If a consensus is met, it's time to move on to the next backlog item. If the estimates vary, the leaders discuss until they arrived at a consensus.

It's useful to have a completed matrix on hand for the estimators to reference during planning poker, as it allows for greater consistency across tasks. Also, it's useful to set a maximum limit (13, for instance). If a task is estimated to be greater than that limit, it should be split into smaller items. Similarly, if a task is smaller than 1, it should be incorporated into another task.

At this point, within your sprint planning meeting, items in the product backlog can be prioritized and divided out amongst the team based on the team's workload capacity.