Prototyping and prioritization

The journey to a new feature

Exciting!

The journey to a new feature

Exciting!

- You: a product-focused engineer at a very small company

The journey to a new feature

Exciting!

- You: a product-focused engineer at a very small company
- Your company: “secrets management for teams”

The journey to a new feature

Exciting!

- You: a product-focused engineer at a very small company
- Your company: “secrets management for teams”
- The feature: “social login”

The journey to a new feature

Exciting!

- You: a product-focused engineer at a very small company
- Your company: “secrets management for teams”
- The feature: “social login”

What work does adding this new feature imply?

How much do we want to build?

How much do we want to build?

The smallest amount that gets us signal?

How do we choose what to build?

How do we choose what to build?

- Three good first things to build
- Three bad first things to build

Exercise: Dronuts

Exercise: what do the “good things” have in common?

Exercise: what do the “good things” have in common?

Examples from last year

Exercise: what do the “good things” have in common?

Lower your cost of change

- If you can easily change, you don’t have to get it right the first time.
- Small modifications are easier than 100% correct guesses on the first attempt.
- Even if you are correct, the world around you might change

Examples from last year

Exercise: what do the “good things” have in common?

Lower your cost of change

- If you can easily change, you don’t have to get it right the first time.
- Small modifications are easier than 100% correct guesses on the first attempt.
- Even if you are correct, the world around you might change

Ignore details early on

Examples from last year

Exercise: what do the “good things” have in common?

Lower your cost of change

- If you can easily change, you don’t have to get it right the first time.
- Small modifications are easier than 100% correct guesses on the first attempt.
- Even if you are correct, the world around you might change

Ignore details early on

Don’t worry about tomorrow

- It’s not a problem till it is a problem
- Don’t waste time on problems you don’t have
- Worry about getting today right, worry about tomorrow tomorrow
- Don’t worry about scale before you have users

Examples from last year

Exercise: what do the “good things” have in common?

Lower your cost of change

- If you can easily change, you don’t have to get it right the first time.
- Small modifications are easier than 100% correct guesses on the first attempt.
- Even if you are correct, the world around you might change

Half, not Half-baked

- Only build what is essential
- Start with half the features you think you need
- What you do implement, implement well

Ignore details early on

Don’t worry about tomorrow

- It’s not a problem till it is a problem
- Don’t waste time on problems you don’t have
- Worry about getting today right, worry about tomorrow tomorrow
- Don’t worry about scale before you have users

Examples from last year

Exercise: what do the “good things” have in common?

Lower your cost of change

- If you can easily change, you don’t have to get it right the first time.
- Small modifications are easier than 100% correct guesses on the first attempt.
- Even if you are correct, the world around you might change

Half, not Half-baked

- Only build what is essential
- Start with half the features you think you need
- What you do implement, implement well

Ignore details early on

Design your interface first

- This is how users will interface with your application
- All features in the back-end should support the interface somehow
- As soon as you have your interface you can start “hallway usability testing”

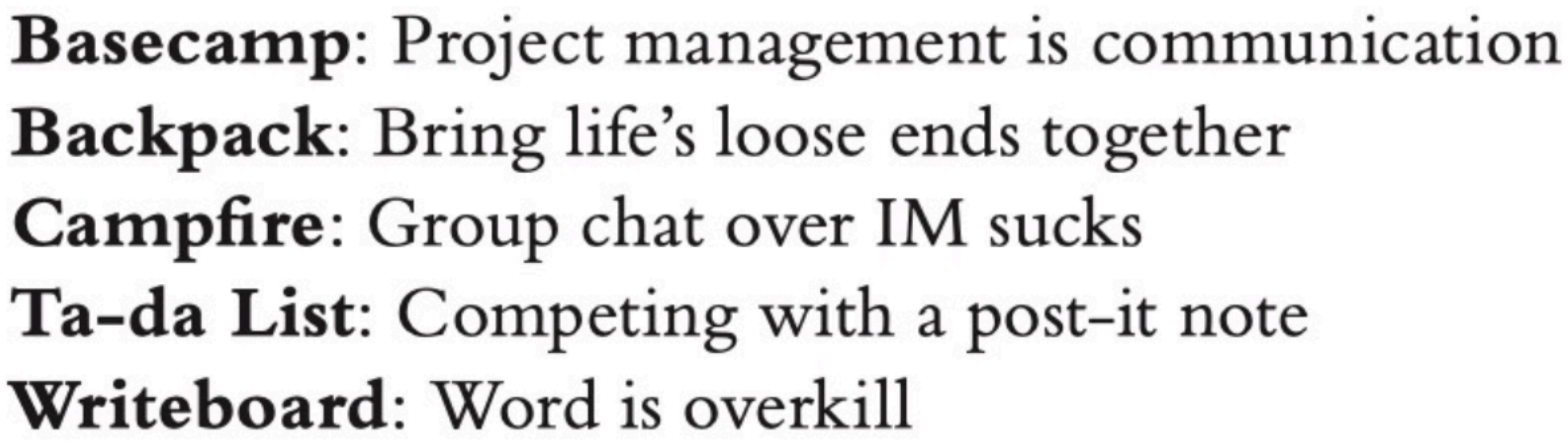
Don’t worry about tomorrow

- It’s not a problem till it is a problem
- Don’t waste time on problems you don’t have
- Worry about getting today right, worry about tomorrow tomorrow
- Don’t worry about scale before you have users

What is the Big Idea?

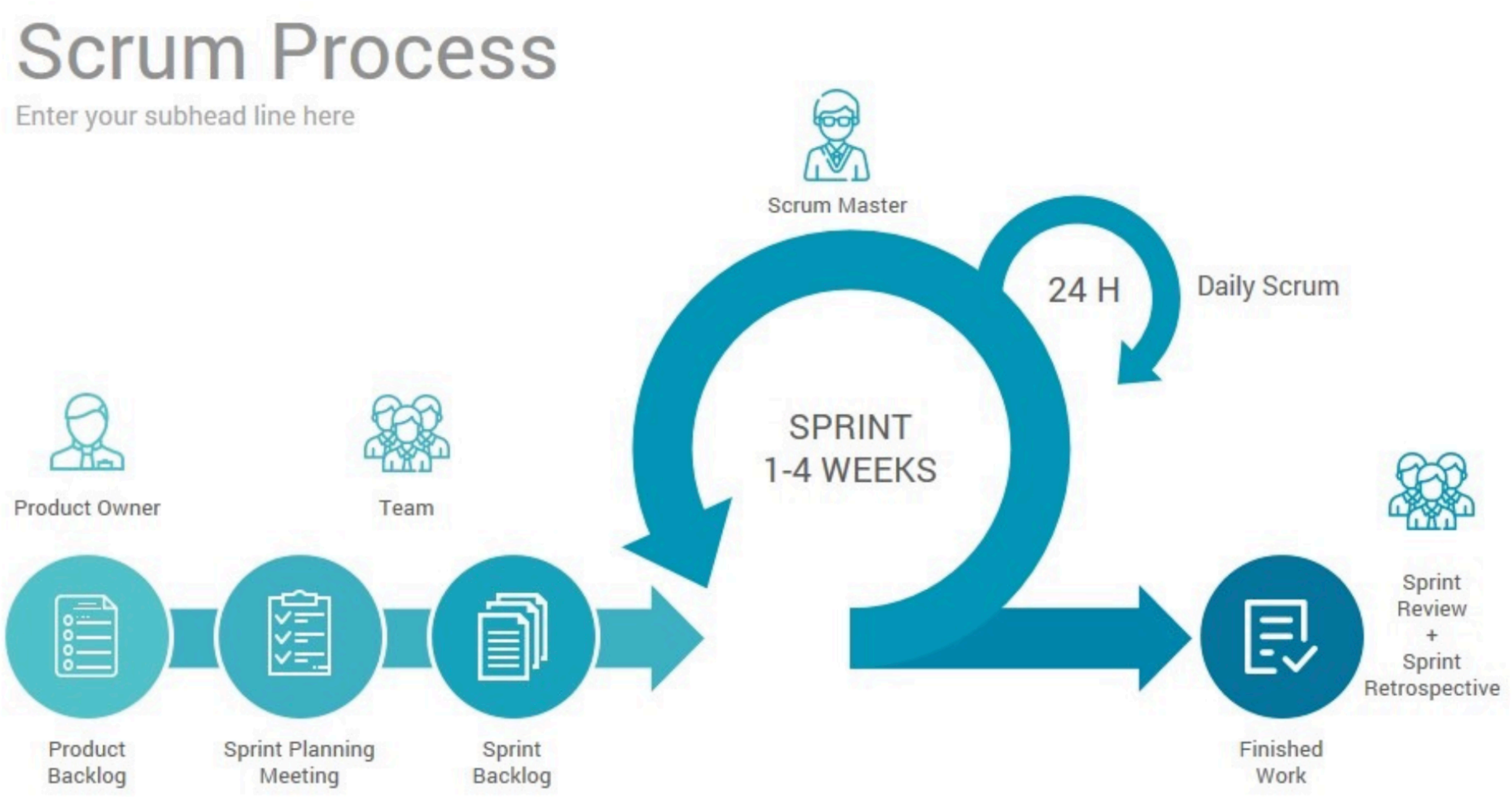
- You should have a single sentence description of what you are building
- What is your product's reason for existing?
- What makes your product different than others?

Example:



Basecamp: Project management is communication
Backpack: Bring life's loose ends together
Campfire: Group chat over IM sucks
Ta-da List: Competing with a post-it note
Writeboard: Word is overkill

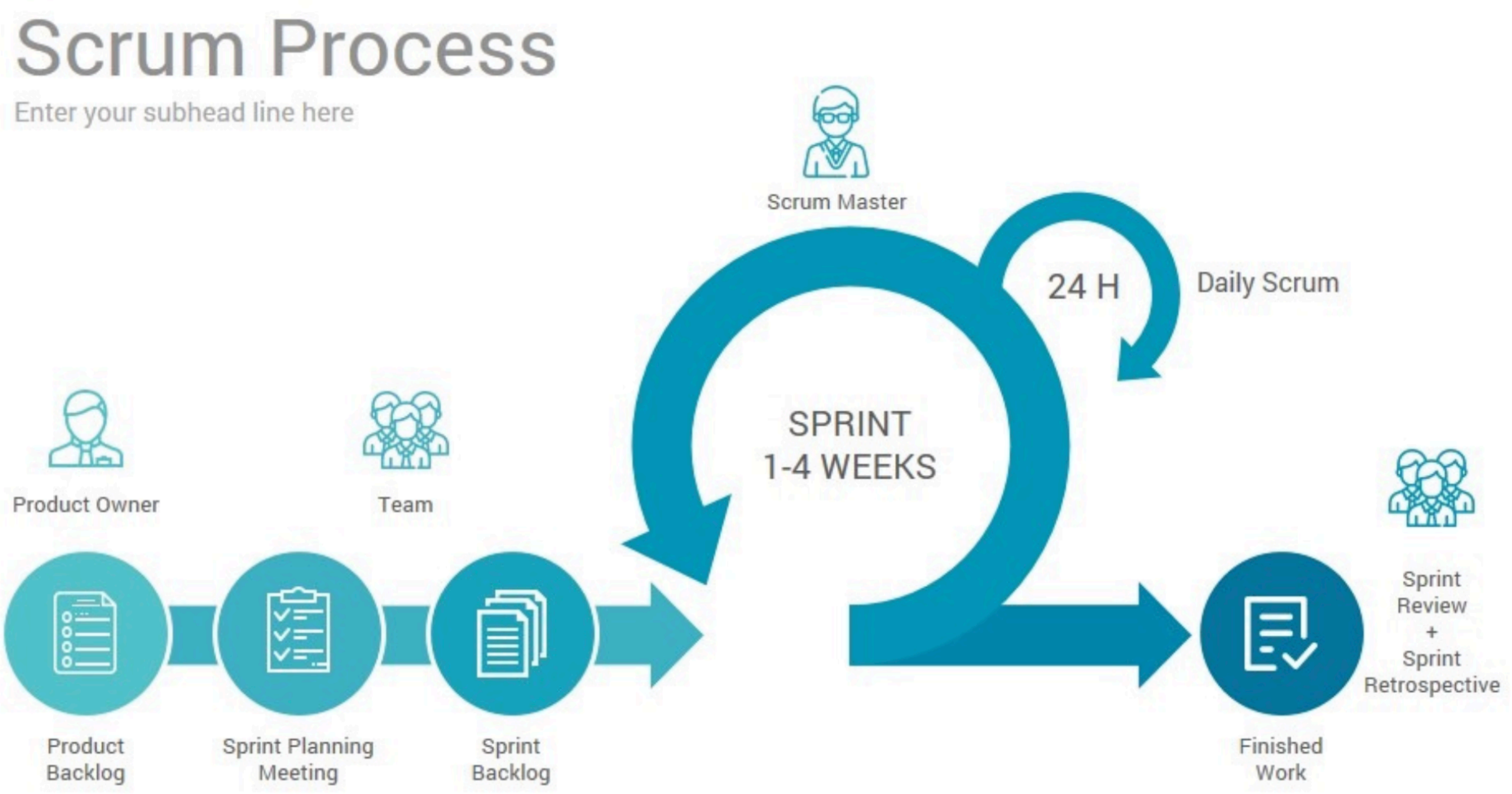
With all that in mind, let’s review



Scrum Process

Enter your subhead line here

Where could we be prototyping?



Activity

- Create a product backlog for a better online ticketing startup
- By yourself: think of two+ user stories, write them on paper (5 min)
- Share users stories with your neighbor, deciding on four user stories

Activity

- Create a product backlog for a better online ticketing startup
- By yourself: think of two+ user stories, write them on paper (5 min)
- Share users stories with your neighbor, deciding on four user stories

- Merge with the neighboring pair
- Prioritize the backlog
- Share highest priority item with the class

Where are we in the process at this point?

- Create a product backlog for a better online ticketing startup
- By yourself: think of two+ user stories, write them on paper (5 min)
- Share users stories with your neighbor, deciding on four user stories

- Merge with the neighboring pair
- Prioritize the backlog
- Share highest priority item with the class

Process for this class

- Each team must have a GitHub repository
- The project backlog will be in Issues.
- Each repository must have a project board.
  - The project board will have cards for (at least) all issues in the sprint backlog.
  - The project board will keep track of issues in progress, done, etc. (more fine-grained categories are ok)
- Each card will have a number of attributes.
  - Tags labeling features, bugs, t-shirt size, etc.
  - All in-progress cards must have at least one assigned team member.
- All checkins must be done by pull request, not pushed directly to main branch.
- All pull requests must be linked to a card to support traceability.