[1] Requirements in Waterfall Vs. Scrum

① Waterfall Approach

↳ Requirements are fixed and written before development Starts. All requirements are fully documented at the Beginning and Treated as final.

* Main Features:

① Requirements Defined upfront
↳ Everything is planned in detail before design or coding begins

② Minimal change Tolerance
↳ changes are avoided and only accepted through formal approval

③ Comprehensive Documentation
↳ long, detailed requirement documents act as a Contract between the Client and developer

④ Sequential process → each Stage Starts after the previous one ends
Requirements → Design → Implementation → Testing → Maintenance

② Scrum [Agile] Approach

↳ Scrum handles requirements flexibility and Continuously through Conversation and feedback.

* Main Features

① Progressive Refinement
↳ Requirements evolve gradually – high level ideas become detailed just before development

② Conversations Drive Requirements
↳ Instead of huge documents, Scrum uses ongoing discussions with Stakeholders.

③ Product Backlog evolution
↳ The backlog Keeps changing
↳ items are added, removed, reprioritized

④ Non functional Requirements
↳ Performance, usability, Security are added as Special Stories or acceptance Criteria.

[2] Progressive Refinement in Scrum

* The process called "Progressive refinement" or "Backlog Refinement"
↳ in Scrum requirements aren't written once and forgotten, they're refined over time.

* INVEST principle for Quality Series:

I → Independent: Story Can be developed Separately from others
N → Negotiable: Flexible and open to discussion
V → Valuable: Provides value to the Customer
E → Estimable: Clear enough to estimate time / effort
S → Small: Can be Completed in one Sprint
T → Testable: Has clear acceptance Criteria.

* Tools and Techniques:
① Story Mapping: Visualizing user activities to Plan features
② Regular Refinement Sessions: Keep the backlog clean and up to date
③ Collaboration: Product owner + Developers + Stakeholders
Discuss requirements frequently

* Backlog Refinement process

① Creating independent Stories:
↳ each Story Should Stand alone to allow flexibility in Scheduling.

② Just-in-Time Detailing:
↳ Only Refine Stories that will be implemented Soon. Keeping things relevant and flexible

③ Conversations for clarification:
↳ Continuous discussion replaces huge documents everyone understands what's needed.

④ Product Backlog evolution:
↳ The backlog becomes more detailed as time goes on.

* Estimation techniques:
① Story Points: Compare effort between Stories
② Planning Poker: Team members pick Cards with estimates and discuss differences
③ T-Shirt Sizing: Classify Stories as XS, S, M, L, XL

[3] Product Backlog Evolution

① Creation and Prioritization

* The Product Backlog is the fundation of all Scrum work
* lists everything needed for the product written as user Stories
* Steps:

① Backlog Creation:
↳ Collect requirements → user needs, features, improvements

② Prioritization Techniques:
↳ Use Methods → MoSCoW, Value based, Risk based prioritization.

③ Requirements Workshops:
↳ Meetings with Stakeholders and team to identify and write user Stories

④ Story writing format

② Continuous Backlog Grooming [Refinement]

* Backlog items are Constantly updated and Recorded
* The goal is to always have Sprint-ready Stories
* Track Progress with charts → Burndown, heat maps
To visualize changes and progress.

[4] MoSCoW Prioritization

↳ A Simple framework for deciding which requirements matter most

* M → Must have: [essential] Without these the product fails → login system
* S → Should have: [Important] Adds major value but Can wait if needed
* C → Could have: [Optional] Nice to have, improves user experience
* W → Won't have: [lowest Priority] Not included in the Current release.

* Benifits:
① Clear priorities → Focus on what matters
② Better Communication between team and Stakeholders
③ Faster decisions about what to include in each Sprint
④ Adjust Priorities easily between Sprints.

[5] User Stories in Scrum

↳ User Stories describe a Requirement from user's Prospective in Simple language

* Components of user Story:

① Card: The Short written Statement
② Conversation: The ongoing discussion that gives details
③ Confirmation: The acceptance Criteria that prove the Story is Complete

3C Frame work
→ Represent the Story
→ Clarify requirement
→ Test and verify the outcome

[7] Story Management Techniques

① Story Gathering Workshops:

Collaborative meeting to create Stories Together
Including → Product owner, Scrum Master, Developers, Stakeholders, users.

* Steps:
① Share Product Vision
② Identify users and Roles
③ Write and prioritize user Stories
④ Define acceptance Criteria

② Story Mapping

↳ Visual tool Showing the entire user journey

* Steps:
① Create Backbone
↳ Main user activities: login → Browse → logout
② Add user Tasks
↳ Details under each activity
③ Continuously update as new information appears.