**User Stories — Lecture Notes & How-To**

**1) What is a User Story?**

* **Definition**  
  A user story is a **short narrative told from the user’s perspective** (who / what / why) describing how the system should help them achieve an outcome. It lives on a **small card** and is used throughout planning and development (prioritization, estimation, scheduling, iteration, acceptance).
* **Relation to “User Tasks” & Engineering Tasks**
  + **Story** = the **business-facing unit of work** used for commitments and acceptance.
  + **User tasks** = concrete actions implied by the story (e.g., *create project*, *upload file*, *rename folder*).
  + **Engineering tasks** = developers’ technical breakdown to implement the story (code, tests, integration, etc.).
* **Compared with Use Cases**  
  Use cases aim to cover many flows (success + alternates).  
  Stories are **small, concrete scenarios** that can be **finished and accepted within an iteration**; alternates/failures are captured in **acceptance criteria/tests** as they are discovered.

**2) “Capital-S Story” — The Four Tests**

From Kent Beck’s guidance

1. **Testable** — you can write automated or scripted **acceptance tests**.
2. **Represents Progress** — the customer team accepts it as **visible forward motion** toward a larger goal (some technical or compliance items can count if customers agree they’re progress).
3. **Bite-sized** — completable **within one iteration**.
4. **Estimable** — the team can give a credible effort/time estimate.

Shorter iterations & smaller teams ⇒ **smaller stories**. Oversized items become **themes/epics** and should be split.

**3) Organizing the Story Set (lightweight but actionable)**

Sort stories into piles by:

* **Priority** (high / medium / low)
* **Risk** (high / medium / low)
* **Time** (this release vs. later; then into **specific iterations**)

This isn’t a rigid hierarchy—just pragmatic “binning” for planning.

**4) How to Write a Card (Template + Examples)**

**4.1 Minimal Template**

* **Title** — one line capability
* **Narrative (Who / What / Why)**
  + *As a* **[role]**,
  + *I want to* **[do X]**,
  + *so that* **[business outcome / why]**.
* **Acceptance Criteria** — success & key failure/alternate paths; later become acceptance tests
* **Estimate** — story points / ideal days
* **Notes** — constraints, dependencies, risks

Avoid **HOW** (implementation). Put details into acceptance criteria/tests or into engineering tasks.

**4.2 Example A — From Narrative to Tasks**

**Original narrative (summarized from the provided content)**

A project manager needs a web place where the whole team can access drawings, models, findings, minutes, and schedules to stay in sync and avoid scattered data.

**Story card**

* **Title**: Create a shared project workspace
* **As a** project manager
* **I want** to create an online project space and centralize artifacts
* **So that** the team stays in sync and access issues go away
* **Acceptance criteria** (sample)
  1. I can **create a project** and see a default **folder structure**.
  2. I can **upload files** (drawings/models/minutes/schedules).
  3. Team members can access via a **single link** honoring permissions.
  4. **Helpful errors** for name conflicts/invalid files/permission issues.
  5. **Audit log** captures create/upload/rename/move/delete.

**User tasks derived**

* Create project; create folders; upload files; rename/move/delete items.

Developers then split into **engineering tasks** (API, storage, ACLs, UI, acceptance tests, etc.) and estimate.

**4.3 Example B — Estimable & Testable Story**

* **Title**: Choose shipping option and instantly update total
* **As a** shopper on the **order review page**
* **I want** to select a **shipping option**
* **So that** the **order total** immediately reflects shipping cost
* **Acceptance criteria**
  1. A shipping option **must be selected**.
  2. Total **updates immediately** after selection.
  3. Options include **admin-set flat rate** or **live rate** from one of two external services based on **total weight & destination**.
  4. If an external service fails/times out, **fallback** and **clear error** are provided.
  5. Source of rate and parameters are **logged for reconciliation**.

This formulation supports planning, estimation, testing, and prompts questions (which external fields, timeouts, retries, etc.).

**5) From Story to Plan & Acceptance (classroom workflow)**

1. **Capture stories** with clear **who/what/why** through conversation with customers.
2. **Apply the four tests**: testable, progress, bite-sized, estimable.
3. **Split / merge**:
   * If a story estimates **> ~3 weeks**, **split** (it’s not understood enough).
   * For estimation convenience, temporarily **batch tiny stories**; implement individually later.
4. **Prioritize & risk-sort** with the customer; assign to **release/iteration**.
5. **Make it executable**:
   * Break into **engineering tasks** (≤ 1 day each is easier to manage).
   * Write **acceptance tests** owned/approved by the customer.
   * Combine with developers’ **unit tests**—these are the evidence of “done”.

**6) Pitfalls & Pro Tips**

* **Missing the “why”** → Always include **So that …** on the card; it improves prioritization and invites better solutions.
* **System/implementation phrasing** (“Require entry of userID”) → Rewrite as **user action** (“Enter user ID to sign in”).
* **Stories too big** → Split into **iteration-sized**, independently testable slices; use **theme/epic** for the umbrella.
* **Ignoring failures/alternates** → Capture them in **acceptance criteria/tests**.
* **Over-specifying HOW** → Keep the story about **intent & observable outcomes**; move HOW to engineering tasks.
* **Thinking only “business value” counts** → If the **customer team recognizes it as progress** and it’s **testable** (e.g., audit logging foundation for compliance), it can be a story.