

Information & Communication Architecture — Lab 1 — User Stories & Acceptance Criteria

Contacts

Course Instructor

Dr. Nourhan Hamdi

nourhan.hamdi@giu-uni.de
M2.312

Teaching Assistant

Mr. Nour Gaser

nour.aly@giu-uni.de
M2.419

Teaching Assistant

Ms. Menna Khalid

TBD
M1.###

Teaching Assistant

Ms. Shorouk Salah

TBD
M1.###

Teaching Assistant

Mr. Mahmoud

TBD
M2.506

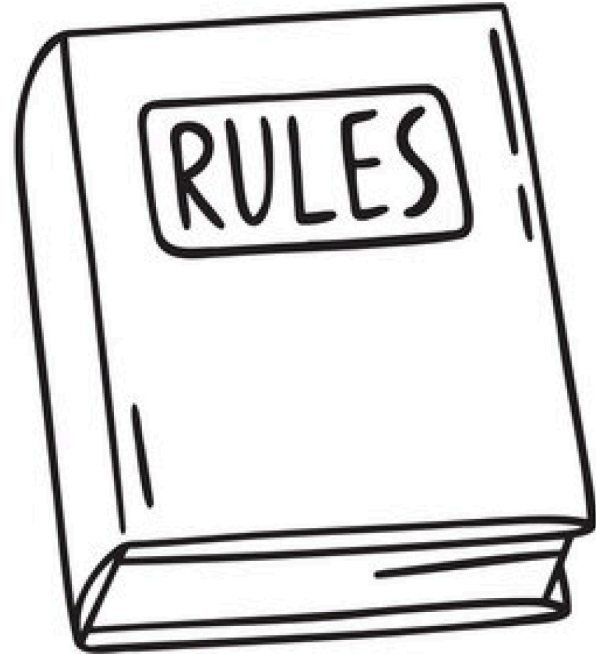
Teaching Assistant

Mr. Younes

younes.zied@giu-uni.de
M2.506

Rules and code of conduct

- **Punctuality:** a grace period of 15 minutes is permitted solely for attendance.
- **Mobile phones:** prohibited during the session unless used for note-taking. Games are not allowed.
- **Focus:** no side conversations during the session. Active participation is expected.
- **Communication:** all emails must be written in formal business English.





Lab agenda

Duration Topic

10 min	Course & lab introduction
25 min	What is a user story? (WHO / WHAT / WHY)
20 min	User story checklist & User Story vs. Task
25 min	Acceptance criteria & Definition of Done
30 min	In-class activity (Talabat case study)
10 min	Wrap-up & questions



01 — User stories

Capturing software requirements from an end-user perspective



What is a user story?

A **user story** is a lightweight tool in Agile software development used to capture a description of a software feature from the viewpoint of the person who desires it.

- It describes **who** the user is, **what** they want to achieve, and **why**.
- It shifts the focus from writing technical specifications to having meaningful conversations about value.
- It acts as a simplified, user-centered requirement.

The standard user story template follows this equation:

As a ⟨type of user⟩, I want ⟨some feature⟩ so that ⟨benefit / reason⟩

Simple example:

As an online shopper, I want to add an item to my cart so that I can purchase it when I finish browsing.

The 3 Ws of a user story

Every strong user story answers three fundamental questions:

- **WHO** — As a <type of user>
Who are we building this for? Defines the specific persona or role.
- **WHAT** — I want <goal or objective>
What is the user's intent? What functionality or capability is needed?
- **WHY** — So that <value / benefit>
Why does the user want this? What tangible value does it deliver?

Concrete breakdown

```
1  WHO:
2  As an online banking customer,
3
4  WHAT:
5  I want to view a rolling transaction history,
6
7  WHY:
8  So that I know my remaining balance immediately
9  after each payment is processed.
```

The **WHY** is the most critical clause: features without clear user value should not be built.



User story examples

Notice how each story identifies a distinct role and clear customer value:

1. Internet banking customer:

As an internet banking customer, I want to see a rolling balance for my everyday accounts so that I know the balance of my account after each transaction is applied.

2. System administrator:

As an administrator, I want to create other administrator accounts so that I can delegate administrative responsibilities safely.

3. Digital marketer:

As a marketer, I want to configure automated onboarding email campaigns so that I can keep trial evaluators engaged.

User story quality checklist

When drafting user stories with your team, evaluate each story against these rules:

- **Keep them short and simple:** A story is an invitation to a conversation, not an exhaustive specification.
- **Write from the user's perspective:** Avoid technical jargon (e.g. *"As a user, I want an SQL query..."* is an anti-pattern).
- **Make the value unmistakable:** Always state the underlying rationale in the `so that` clause.
- **One discrete capability:** If a story contains the word *"and"*, it usually bundles multiple features. Split it into two separate stories.
- **Written collaboratively:** Stories are refined as a cross-functional team (developers, designers, product owners), not written in silos.

User Story vs. Task

User Story (Outcome)

- Expresses a **business outcome** or user goal.
- Answers **WHAT** is needed and **WHY**.
- Written in **plain domain language**.
- Verified through **acceptance criteria**.
- Estimated in **story points**.

```
1  As a shopper,  
2  I want to pay using Apple Pay  
3  so that checkout is fast on mobile.
```

Engineering Task (Activity)

- Expresses a **technical step** to implement a story.
- Answers **HOW** it will be constructed.
- Written in **technical terms**.
- Done when the code is merged and verified.
- Estimated in **hours**.

```
1  1. Configure Apple Merchant ID.  
2  2. Install Stripe SDK for iOS.  
3  3. Implement PassKit payment sheet callback.
```



02 — Acceptance criteria

Defining conditions of satisfaction before development begins



What are acceptance criteria?

Acceptance criteria (AC) — also called *conditions of satisfaction* — provide the detailed boundary and scope of a user story. They define the exact rules and system behavior required for the story to be accepted as complete.

Core goals of acceptance criteria

1. **Clarify scope upfront:** Ensures the team knows exactly what needs to be built *before* writing any code.
2. **Establish a shared mental model:** Aligns product owners, developers, and QA engineers on expected behavior.
3. **Provide objective pass/fail boundaries:** Removes guesswork about whether a feature works correctly.

AC Example 1 — Strong password

User Story

As an online banking customer, I want a strong password, so that my account and financial data remain secure against unauthorized access.

Acceptance Criteria

- Password length must be at least **8 characters**.
- Must contain at least one character from each required group:
 - Lowercase letter (a - z)
 - Uppercase letter (A - Z)
 - Numeric digit (0 - 9)
 - Special character (! , @ , # , \$, % , ^ , & , *)
- If any rule is violated, display an inline error message identifying which rule failed.
- The submit button remains disabled until all password conditions are met.

AC Example 2 — Online conference registration

User Story

As a conference attendee, I want to register online, so that event sign-up is simple, fast, and paperless.

Acceptance Criteria

- User cannot submit the form with empty mandatory fields (Full Name , Email , Organization).
- Submitted registrant data is validated and safely persisted to the database.
- An automated confirmation email containing a digital QR ticket is sent within 60 seconds.
- Protection against spam submissions (CAPTCHA / rate limiting) is enabled and operational.

Acceptance criteria checklist

Comprehensive acceptance criteria should address both standard usage and edge cases:

- **Happy path flow:** The ideal end-to-end user path from initial action to success.
- **Negative scenarios & validation:** Invalid inputs, expired tokens, server timeouts, network dropouts.
- **Functional requirements:** Exact calculations, data transformations, database mutations.
- **User experience (UX):** Loading states, disabled buttons, confirmation modals, error alerts.
- **Performance & security boundaries:** Response time thresholds, rate limits, sensitive data masking.

What AC should NOT include

Do NOT put these in AC

- *"Code review was completed."*
- *"Unit tests and integration tests written."*
- *"Feature deployed to staging."*
- *"Non-blocker bug fixes resolved."*

These items are **not** specific to one user story — they are engineering standards that apply to **every single piece of work**.

AC vs. Definition of Done (DoD)

Acceptance Criteria (AC)	Definition of Done (DoD)
Specific to one story	Universal across all stories
Written by Product Owner / Team	Agreed upon by the Engineering Team
Verifies feature behavior	Verifies <i>software quality</i>
E.g. Password has 8+ characters	E.g. <i>PR approved by 2 peers, CI green</i>



03 — In-class activity

Hands-on requirements specification for a delivery application

Case study — Talabat restaurant browsing

Given User Story

As a hungry customer, I want to browse restaurants near me so that I can choose where to order food from.

Sample Acceptance Criteria

- The app displays a list of open restaurants located within a **5 km radius** of the user's current GPS location.
- Users can filter the restaurant list by **cuisine type** (e.g. Italian, Burgers, Asian).
- Users can sort the restaurant list by **estimated delivery time** or **customer rating**.

✓ DELIVERABLE

1. Write **3 different user stories** for Talabat (e.g. food search, customizing an order, checkout, or driver tracking).
2. Select **2 of your stories** and write at least **3 clear acceptance criteria** for each:
 - Include at least **one negative scenario** (e.g. empty cart, out of stock, out of delivery zone).
 - Verify that your criteria do not contain DoD items.



References & useful links

AGILE & USER STORIES

[Atlassian Agile Coach — User Stories](#) — template and best practices

[Mountain Goat Software \(Mike Cohn\)](#) — the definitive guide to user stories

Review these guides as you refine user stories for your semester project.

ACCEPTANCE CRITERIA & DOD

[Martin Fowler — GivenWhenThen](#) — structuring acceptance criteria

[Scrum Alliance — Definition of Done vs Acceptance Criteria](#) — quality gates vs feature rules

Practice writing crisp user stories and testable acceptance criteria!

