

Defining Project Requirements



Register for a League

in list [Product Backlog](#)

Description [Edit](#)

Story

As a Player I want to register for a league because I want to play soccer.

SWEN-261

**Introduction to Software
Engineering**

Department of Software Engineering
Rochester Institute of Technology



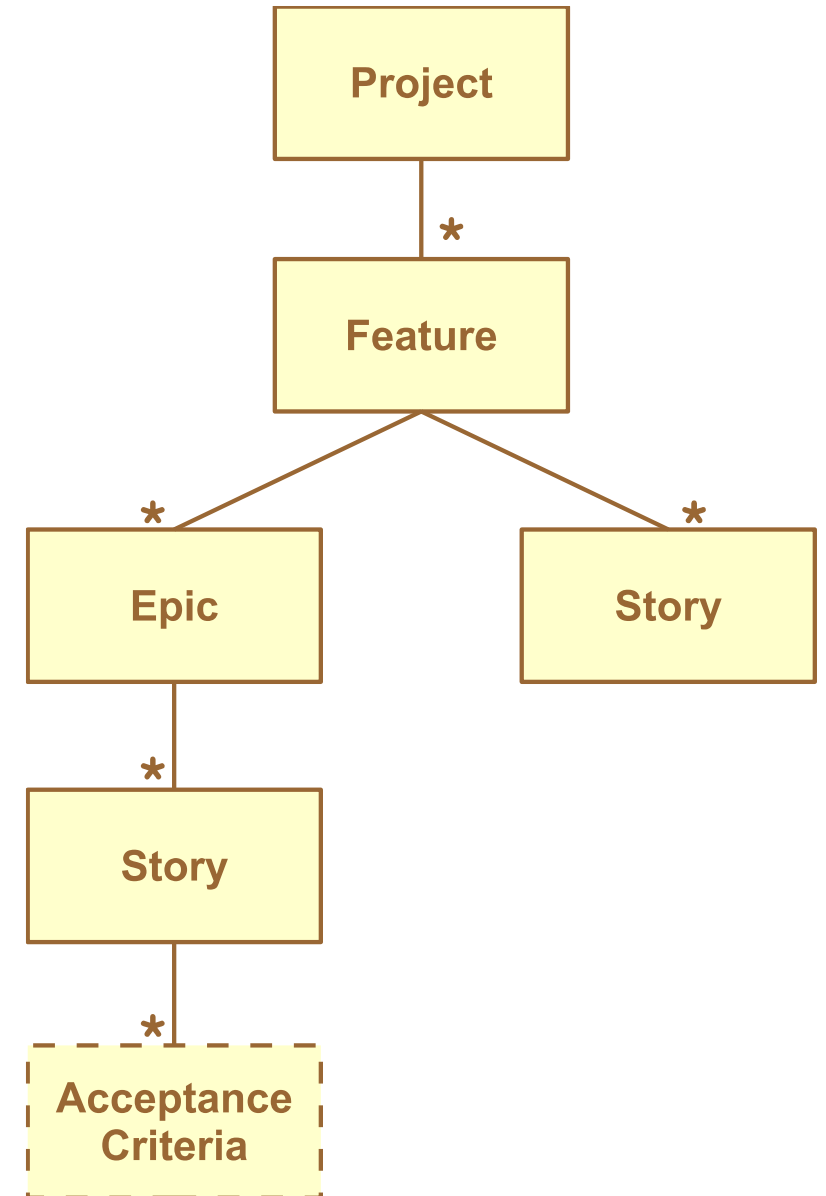
Software Engineering
Rochester Institute
of Technology

There are functional and non-functional requirements.

- Functional requirements define the behavior of the application that achieve user goals.
- Non-functional requirements define:
 - *Systemic qualities such as response time, reliability and availability*
 - *Systemic constraints such as security or accessibility*
 - *Evolutionary qualities such as testability and extensibility*

Functional requirements are in a hierarchy.

- Most applications have many large-scale *features*.
- These are decomposed into *epics* and *user stories*.
- Epics are further decomposed into additional stories.
- As analysis progresses, stories will be further defined with *acceptance criteria*.



Features are large scale application behaviors.

- Soccer league management application

The League Director manages a soccer league three seasons a year. Players need to register for a league. The Director selects a group of captains to form teams and the Director moderates the drafting of teams by captains. The Director must also schedule the games of the league across the whole season up to 15 weeks. The Director must also be able to record game results and keep track of team rankings.

Defining requirements involves a continual discussion with the customer.

- Eliciting requirements through:
 - *Observation*
 - *Interviewing users and subject matter/domain experts*
 - *Brainstorming*
- There are many approaches for specifying requirements
 - *Agile methods write user stories*
- This documentation forms a contract between the Product Owner and the Development Team.
- The Product Owner will be involved with the definition of the user stories to ensure that the overall system will satisfy the business needs.

User stories express goals of users.

- There are many ways to [write a user story](#).
- In class, you will use this notation:
"As a <ROLE>, I want <GOAL> so that <BENEFIT>."
Example:
As a Player, I want to register for a league so that I can play soccer.
- The *ROLE* identifies the type of user to achieve the goal.
- The *GOAL* is what the system will do for the user.
- The *BENEFIT* is the value that the system provides the user.

More examples of user stories...

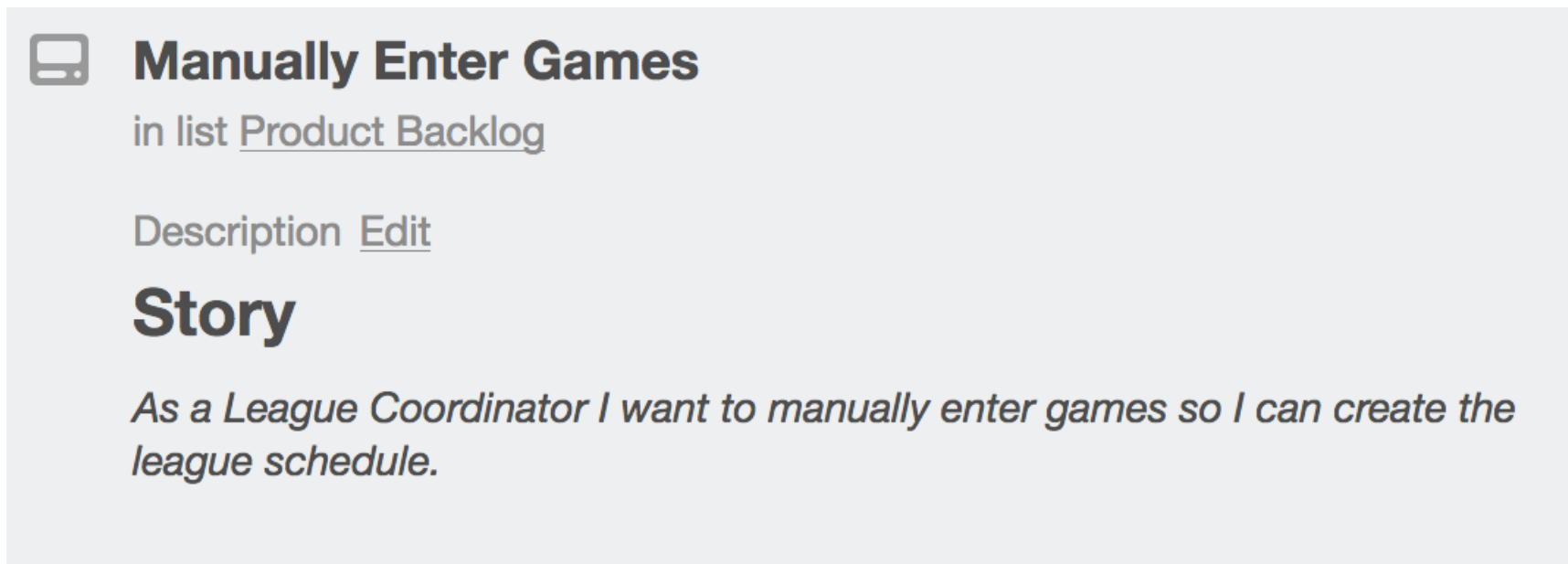
- From the Soccer League application:
 - *As a captain I want to enter the scores of my games so my team's rank can be determined.*
 - *As a player I want to see my teams rank so I can gloat to my competitors.*
 - *As the director I want to create a tournament schedule so teams know when and who to play.*
- From a Social Media application:
 - *As a member I want to check-in that I'm having dinner with friends so my mom sees me with my friends.*
 - *As an ad-bot I want to access user usage statistics so that I can create personalized ads.*

You will want to avoid the characteristics of poorly written user stories.

- Stories with no clearly defined user:
 - *"I want to identify soccer league captains."*
 - *Or worse just: "identify soccer league captains"*
- Stories with no clear benefit:
 - *What benefit does selecting captains provide?*
- Stories should not:
 - *Over constrain the solution*
 - *Dictate user interface details*

In Trello you create a new card for each User Story.

- Your team will use [Trello](#) to record your project's requirements.
- Every User Story will be captured in a Trello card, like this:



The image shows a screenshot of a Trello card. At the top left is a card icon. The title is "Manually Enter Games" in bold. Below the title is the text "in list [Product Backlog](#)". There is a "Description" label and an "Edit" link. The main content is the user story: "As a League Coordinator I want to manually enter games so I can create the league schedule." The card has a light gray background.


Story text is usually high-level, therefore you define acceptance criteria to provide refinement.

- *Acceptance criteria (AC)* are detailed statements about how the system should behave.
 - ***Use this format:***
GIVEN some condition WHEN some action occurs THEN system does something.
 - ***Example:***
Given that I have not yet signed in when I see the Home page then I must see a means to sign-in.
- A story typically has multiple AC
- The AC should drive and constrain the design and development.
- The AC are completed by the story's tester.

Writing User Stories for REST API and other services

- REST APIs and other services should also be driven by requirements.
- However, these requirements are different than the User or Product requirements we have discussed so far in that they may not come directly from a Product description.
- It is typical that a team will start on backend services like a REST API before the development on the UI begins.
- Therefore, the <role> for a REST API User Story is generally a developer or system integrator
 - *User Story*
 - ◆ AS A developer I WANT to submit a hero id to the API SO THAT I can retrieve the details of that hero
 - *Acceptance Criteria*
 - ◆ GIVEN I submit a hero id WHEN the hero exists for that id, THEN the system should return the hero object and a status code of 200
 - ◆ GIVEN I submit a hero id WHEN the hero does not exist for that id THEN the system should return a status code of 404


In Trello you add the Acceptance Criteria using a checklist.

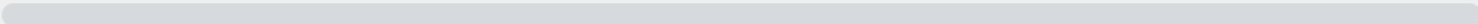
 **Player Sign-in**
in list [Sprint Backlog](#)

Description [Edit](#)

Story

As a Player I want to sign-in so that I can play a game of checkers.

 **Acceptance Criteria** [Delete...](#)

0% 

- Given that I have not yet signed in when I see the Home page then I must see a means to sign-in. (such as a link or button)*
- Given that I am not signed-in when I do click on the sign-in link then I expect to be taken to the Signin page, with a means to enter a player name.*
- Given that no one else is using my name when I enter my name in the sign-in form and click the **Sign-in** button then I expect system to reserve my name and navigate back to the Home page.*

Sometimes a story is too large and it must be broken into smaller stories that compose an Epic.

- A story must be achievable within a single sprint.
 - *For your project try to create stories that one student can code in a single week.*
- Break up a larger story into smaller stories.
 - *Each sub-story must provide value but it can be a small step towards the epic goal.*
 - *Identify dependencies between sub-stories to produce a priority ordering of these stories.*
- An epic may be spread out over multiple sprints.
- The *acceptance criteria* of an epic is the successful completion of its sub-stories.

In Trello you can represent an Epic with a Checklist of links to Story cards.

EPIC: Schedule Games
in list [Product Backlog](#)

Description [Edit](#)

Story

As a League Director I want to schedule the league games across the complete season so that I can run an efficient league.

Stories [Delete...](#)

0%

- [Manually Enter Games](#)
- [Validate Game Conflicts](#)
- [Adjust the Schedule](#)
- <https://trello.com/c/F8ukujjS/20-schedule-games-automation>

[Save](#) × ... [Convert to Card](#)

Stories

0%

- [Manually Enter Games](#)
- [Validate Game Conflicts](#)
- [Adjust the Schedule](#)
- [Schedule Games Automation](#)

Paste the Story card URL into the checklist item

Spikes help explore technology challenges.

- A Spike is:
A story or task aimed at answering a question or gathering information, rather than at producing shippable product.
(from ScrumAlliance.org)
- Spikes are used for:
 - *Learning new technologies or new techniques*
 - *Typically, just proof-of-concept coding; usually throw-away*
- Spikes are usually time-boxed to fit into a single sprint though there are occasionally epic spikes.

The Minimum Viable Product is all of the stories required to be in the first product release.

- MVP is defined by the Product Owner.
- It clearly identifies the set of stories that must be done before the product can be released.
- These stories must be prioritized to be worked on first.