Topics
- Quiz 3
- User stories (wrap up)
- Context Diagrams

Homework
- HW4 due
- HW5 out
More on User Stories

User stories can be at varying levels of detail

- "Epics" are large stories
  - Covers a large amount of functionality
  - Requires multiple sprints to complete
- "Themes" are smaller than epics but larger than stories
  - An epic usually is broken into a set of themes
  - Each theme consists of a set of stories

So Epics ⇒ Themes ⇒ Stories

What is the "right size" for a story?

- Provides concrete value
- Describes (smallest) action a user wants to do
- Done in one sprint
When to decompose stories into smaller ones:

- Look for “x, y, and z” (3 stories) or “u or v” (2 stories)
- If the story is vague ... i.e., unclear how to start working on it
- If the story is just too big for a single sprint
- If the story has “red flags” ...
  - Cannot easily, or with certainty, determine amount of work involved
  - Unclear what the corresponding features would be
  - Has complex, inconsistent, or unrelated acceptance criteria
  - Still has big questions surrounding it
"As a student I can purchase a parking permit so that I can drive to/from school."

Acceptance Criteria:

- Must be a registered/enrolled as a student
- Must not have a hold
- Must provide a valid car license plate
- Must pay all parking tickets before being issued a pass
- Must pick up parking pass from parking office after purchase
- Must be able to buy a semester-only or year-long pass

Q: How could we decompose the parking permit story?

- ..., I can pay my parking tickets so I can purchase a parking pass
- ..., I can register my car information so I can purchase a parking pass
- ..., I can receive a confirmation after purchasing a parking pass so that I can pick up the pass from the parking office
- ..., I can purchase a semester-long parking pass so I can drive to/from school
- ..., I can purchase a year-long parking pass so I can drive to/from school

Exercise 4:

- Try decomposing a story into smaller ones
A “spike” is a special story for gathering information

- To develop a proof of concept, compare two approaches, do research, etc.
- Brings the need for the activity to the attention of the customer (visibility)
- Is short in duration (like a normal user story)
- Note: not typically in the user story format

Stretch goals (non “requirements”)

- What to do next if requirements are finished (e.g., after MVP)
- Good to identify and prioritize (helps define MVP)
- Also for avoiding “feature creep”

A “task” captures work that has to be done for a story

- Meaningful to a developer / team member
- Not that meaningful to customer (in developers words)
- Each story is broken into a collection of tasks
- Can usually fit on a “sticky” note
Some examples of (generic types of) tasks ...

- Design table and add to DB
- Create a new model and controller class
- Implement new views
- Write unit and integration tests
- Write automated UI tests
- System test new UI extensions

Exercise 5:

- Take a story and decompose it into tasks
- Be sure to include tasks for design and testing as well!
### Product Backlog (review)

**Ordered (Prioritized) list of the work to do** ("Backlog items")

- Stories (functions), spikes, bugs, non-functional requirements

A **“living document”** (updated) throughout the project

- Higher priority items (generally) done first, and are refined/smaller and more detailed than low priority items
- Reviewed and updated regularly (every sprint)
- sometimes called **“grooming”**
Context Diagrams

Context Diagrams describe the system “boundaries”

- what is inside ("in scope") vs outside ("out of scope")
- things out of scope that you interface with
- meant to be high-level (coarse grain)
- subsystems, environments, interactions

Context Diagram Examples ...

![Context Diagram Example Image]
Good context diagrams highlight items outside the boundary

- interactions & dependencies to other systems/subsystems
- usually important or non-obvious systems/subsystems
- sometimes to people (e.g., for approvals, etc.)

Focus on what exists that your project will use versus build

- All about what you won't implement (but need)
- DBMS, Authentication systems, GPS/Map Displays, ...