CS169 Week 5 Section

BDD, User Stories, and Cucumber Intro

Administrivia

- HW 4 due 10/5 at 11:59pm. Peer reviews will be due the following Friday (10/11) at 11:59pm. Make sure to start early! bCourses can be buggy.
- Project list will be released **tonight**!
    - I recommend you meet with your groups by **Thursday night** to rank the projects
- Project ranking form will be released **Thursday night**, is due **Friday 11:59 PM**
- **Iteration 0** starts next week! Stay tuned.
- Midterm 1 on 10/8, from 7-9pm.

Behavior Driven Design (BDD)

- Working closely with stakeholders
- User stories capture app behavior (document user requirements)
  - Written as a couple sentences on 3x5 index cards (or online tracker like Pivotal)
  - All stakeholders brainstorm and prioritize features
- Tests *behavior*, not implementation
  - Even if implementation changes, behavior stays the same
- One way to test is Cucumber/Capybara

User Stories

- ≅ acceptance tests
- Not code — supposed to be normal english
- User tests (Connextra Format)

  Feature name

  As a [kind of stakeholder],
  So that [I can achieve some goal],
  I want to [do some task].
- Connextra formatted user stories in customer meetings

SMART Stories

- User stories should be SMART
  - Specific
  - Measurable
  - Achievable
  - Relevant
  - Timeboxed
- Ex: “resource providers can add resources to the database”
  vs. “as a resource provider, so that my resource can be added to the approval queue for site admins to look at, I want to make a POST request to the Innovation Resources API with the minimum requirements for a resource to be added”

User stories should be SMART

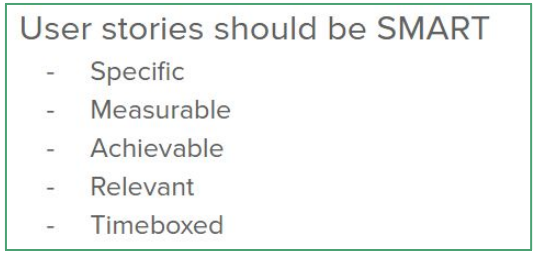
- Specific
- Measurable
- Achievable
- Relevant
- Timeboxed

User story example

Discuss whether the following user story is SMART:

- “The UI should be intuitive.”

User story example



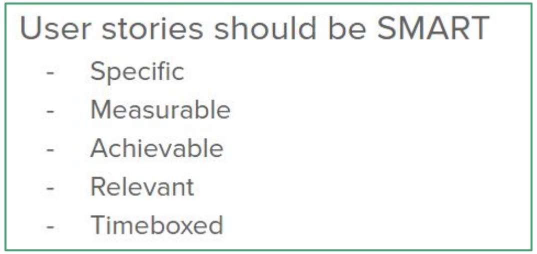
User stories should be SMART

- Specific
- Measurable
- Achievable
- Relevant
- Timeboxed

Discuss whether the following user story is SMART:

- “The UI should be intuitive.”
- “The login UI should be so intuitive that 80% of customers can log-in within twenty seconds.”

User story example


User stories should be SMART

- Specific
- Measurable
- Achievable
- Relevant
- Timeboxed

Discuss whether the following user story is SMART:

- “The UI should be intuitive.”
- “The login UI should be so intuitive that 80% of customers can log-in within twenty seconds.”
- “As an e-commerce website user, so that I can go to cart and checkout my purchase seamlessly, the pre-purchase login UI should be so intuitive that 80% of customers can log-in within twenty seconds.”

Cucumber

- BDD to acceptance tests ⇒ Cucumber and Capybara
- Cucumber is a framework for writing user scenarios and turning them into acceptance tests.
  - Used alongside Capybara, the framework used to simulate the user's browser as they navigate through your webapp
- Cucumber scenario:
  - Setup preconditions: **Given ...**
  - Action to test: **When ...**
  - Check postconditions: **Then ...**

Cucumber

- Ex. of turning user stories into acceptance tests:
  - **Story:** “As an e-commerce website user, so that I can aggregate all items I want to buy in one place to look at later, I can add items of interest to my cart.”
  - **Test:**
    - **Given** I am on the detail page for the novel “Ready Player One,”
    - **When** I click the “Add to Cart” button,
    - **Then** the novel “Ready Player One” will be in my cart.
  - **Step definitions**:
    - <regex matcher> do navigate_to_page(“/novels/id/1247234/detail”) end
    - <regex matcher> do click_on_button(:name => “Add to Cart”) end
    - <regex matcher> do assert user.cart.include? Novel.where(:title => “Ready Player One”).first end

Test your app with

Capybara

Tired of clicking around in your browser trying to make sure your applications work as expected? Capybara is a library written in the Ruby programming language which makes it easy to simulate how a user interacts with your application.

Capybara can talk with many different drivers which execute your tests through the same clean and simple interface. You can seamlessly choose between Selenium, Webkit or pure Ruby drivers.

Tackle the asynchronous web with Capybara's powerful synchronization features. Capybara automatically waits for your content to appear on the page, you never have to issue any manual sleeps.