Chapter 7 - Behavior-Driven Development

Agile - follows BDD

* User stories to elicit functional requirements
* Low fidelity (Lo-Fi) user interfaces and story boards to elicit UI requirements
* Points to turn user stories into cost estimates
* Velocity to measure and estimate schedule
* Tools
  + Cucumber to transform user stories into acceptance tests
  + Pivotal tractor to track project progress, calculate velocity and estimate time to milestones

Plan and Document Lifecycle

* Requirements elicitation via interviewing, scenarios and use cases
* Requirements documentation via Software Requirements Specification
* Requirements fulfillment using requirements traceability
* Cost estimates using COCOMO
* Scheduling and monitoring progress using PERT charts
* Change management using version control systems
* Risk analysis

Both life cycles illustrate the difference between

* functional and non-functional requirements
  + non-functional requirements include performance, dependbility and security
* explicit versus implicit requirements

TimeBoxed

* Stop story when exceed time budget
* Give up or divide into smaller stories or reschedule what is left undone
* To avoid underestimating length of project

Low-Fi UI Sketches

* After you’ve come up with your app’s user stories, it’s time to start thinking about how you want the UI to look (UI very important when you’re building an actual product).
* Before you start writing out the HTML pages, you have to have a layout/design in mind. In the first couple stages of prototyping a UI, you create what are called Lo-Fi UI sketches (part of your milestone 1) – they’re basically what they sound like, UI sketches that show how you want your pages to be laid out, and how you want the user to be able to interact with your page.
* Don’t spend a ton of time making these look fancy – they are generally drawn on paper because its really easy and quick to change (you can even stick extras like post-its to represent popup menus) etc. There’s also some cool wireframing software out there that lets you drag and drop elements to create these UI sketches . It’s up to you guys what you want to use, as long as its really simple, easy to make changes and put notes on.

User Stories to Sketches

* It’s really important to match your user stories to your UI-sketches to determine what features you need in your layout and how you want the user to be able to interact with your application.
* For example, if you were creating an application like Facebook and you had a user story that said:
  + as a user I want to be able to view all of my friends’ top news when I log in or
  + as a user I want to be able to view my friends’ upcoming birthdays right when I log in,
  + you would have to take those into account when laying out the different parts of the page that users are taken to right after they log in.
* Ideally, you should have sketches that incorporate all of the user stories, so you would be making sketches of each page that the user encounters. It might sound like a lot/tedious but you’re going to have to go through this process anyway when you’re actually coding up the HTML pages

Sketches to HTML

* Tedious to do sketches and storyboards,
* but easier than producing HTML! And…
  + Less intimidating to nontechnical stakeholders
  + More likely to suggest changes to UI if not code behind it
  + More likely to focus on *interaction*rather than colors, fonts, …
* CSS (Cascading Style Sheets) can make it look nice later

Storyboards

* Need to show how UI changes based on user actions
* HCI => “storyboards”
* Like scenes in a movie, but not linear

Cucumber and Capybara

* [Cucumber](https://cucumber.io) turns customer-understandable user stories into acceptance tests and integration tests