This Week ...

- Finish up your initial product backlog
  - Get feedback on your backlog from your sponsor
  - Refine based on feedback
- Create initial UI sketches (w/ feedback)
- Draft of Section 1 and 2 of project plan
  - Plus revisions of previous
- Draft of major features checklist
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

A “**Spike**” is a special story for gathering information
- E.g., to develop a proof of concept, compare two approaches, do research, etc.
- As a story, brings the need for the activity to the attention of the customer (visibility)
- Is short in duration (like a normal user story)
User Stories ...

Stretch goals (not “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”

Backlog (review) ...

The “product backlog”

**Prioritized** list of the work to do (“backlog items”)

- Stories (features), spikes, bugs, non-functional requirements, …, meaningful to the customer

A “**living document**” throughout the project

- High priority items (generally) done first, and are smaller and more detailed than low priority items

Reviewed and updated regularly (every sprint) … sometimes called “grooming”
Carrying out sprints ...

The “sprint backlog”

- Items selected from the product backlog to finish in the sprint (iteration)

- Backlog items typically selected during a sprint “planning meeting” w/ customer

- Based on sprint goal, priorities, and time estimates (want to complete in one sprint)
User Story Tasks ....

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

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

Exercise ...

Give it a try …

- Individually, take a story and define associated tasks
- Discuss and refine as a group

⇒ Tasks for design, coding, & testing!
  - Since a sprint includes all of this
Task Boards (Project “Dashboards”)

Example ...

An Alternative Layout
Using the Task Board

Update the task board daily (at least)

- move finished tasks or stories to “done”
- “check out” new tasks or stories
- move “next” stories to swim lanes (main part of board)
- update burn down chart (more later)

For your team

- since not working full time …
- may not make sense to update daily
- but still a good idea to keep track of progress mid-week

How it works -- Day 1
How it works -- Day 2

How it works -- After a few more days
The Task Board

Conceptually, task board is **how to think about iterations**

- e.g., contrast this with *Gantt charts*
  - big picture of progress
  - focused on entire project
- whereas for *task boards*
  - day-to-day detailed view of progress
  - focused on one sprint (e.g., one week)

In general, task boards meant to be **flexible/adaptable**

- structure them to what works best for your team
- okay to add, remove, change layout

Examples of Task Boards “in the Wild”
Tracking Progress with Burn Down Charts

A burn down chart shows where you are in the iteration
- plot a **new point** on the graph **each work day**
- y-axis is the estimated effort (e.g., total dev days)
- x-axis is the number of actual dev days left

**Burn Down Charts**

- No finished tasks, but still on track!
- Uh oh! We’re not making progress

We finished more tasks than planned!
Unplanned Tasks Happen

Expect new (unexpected) tasks to pop up

- sometimes even entire stories

New tasks/stories usually throw off the schedule

- they add work left to your burn down chart
- may be able to “catch up” (e.g., if small tasks)
- may require reprioritization (talk with customer)
Finishing up the Sprint

Need to allow time to **check in** with customer ("**review**")
- demo the new features, show design changes, …
- get feedback to improve product
- discuss goals / stories for next iteration

Do a sprint "**retrospective**"
- what went well?
- what could have gone better?
- what changes for next time (process, tools, …)?
- what was your velocity?

Context Diagrams ...
**Context Diagram**

**Context Diagrams** describes the project “**boundaries**”
- what is inside (in scope) vs **outside** (out of scope)
- meant to be high-level (coarse grain)
- subsystems, environment, interactions
Context Diagram Examples

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, ...
Design in Software Engineering

Throughout a software project, we do lots of “design”:

- We (help) design the MVP
- We (help) design the feature set (e.g., stories)
- We (help) design the User Interface / Interaction (UI)
- We design the data model, architecture, and software
- We design tests
Design in Software Engineering

Design is a process involving various steps, including ...

- Idea generation (“ideation”)
- Idea testing, prototyping, and refinement
- Idea selection and communication

- Keys to keep in mind when designing ...
  - Go for quantity (“quantity breeds quality”)
  - Withhold judgement ... analyze & critique ideas later
  - Combine and improve

User Stories and Prototyping

User stories help us design & capture system features ...

- Help with communication (e.g., w/ customer)
- Help with project scope (estimate project size)
- Help in carrying out sprints

But hard to see how system should work with stories alone

- Can make it hard to know if set of stories is complete

Design “prototyping” is a tool to help extend/refine stories ...

**Basic Idea**: use lightweight materials to mockup system
Where prototyping is part of the design process
UI Design: Sketches vs Wireframes

Two flavors of UI design for our projects

1). To better understand the MVP
   - Broad brush strokes … picture is worth 1,000 words
   - Focus on content & flow, not layout & details

2). Detailed design of the UI to build the system
   - Details of each “page”, how laid out
   - Colors, format, widgets, messages, ...
   - Professional “look & feel”

Now in your project, we’re focusing on 1 -- the MVP

UI Prototypes (“Mockups”)

We’ll talk today about two types of mockups

- **UI sketches** … freehand drawing of UI “screens”
- **UI storyboards** … flow diagram of how screens “connect”

A UI Sketch is a high-level drawing of a part of the UI

- Each sketch is usually a single “screen” (or “page”)
- What a user sees on their screen in some state of the app
- E.g.: one page in a web site or one screen on a mobile app
UI Sketches

In a UI Sketch start simple then iteratively add detail ...

1) At first
   ○ Focus on basic elements needed (button, text, etc)
   ○ Don’t worry too much about placement
   ○ Develop alternatives (if unsure which pages are needed)

2) Create more detailed sketches
   ○ Add in missing elements
   ○ Consider different ways to organize elements
   ○ Consider different basic widgets

3) Create alternatives, get feedback, refine until you converge

UI Sketches

Some tips for creating UI Sketches ...

- Use “throw away” materials (don’t get attached)
- Consider whiteboard first (easy to change & chuck)
- Move to paper & pencil (also easy to change & chuck)
- Goal is to converge to a basic design (not a pretty picture)
- Once you converge, move to pen & paper (optional)
UI Storyboards

UI Storyboards connect the different screens together

- Describe the “flow” between screens/pages (like a “map”)
- Which screen is the “landing” page (home screen)
- What screens can be accessed from which other screens

Good Storyboards emphasize common navigation paths

- The “typical” paths through the screens users will take
- E.g.: Search for items ⇒ View item ⇒ Select item ⇒ Add to cart ⇒ Checkout

Tips for UI Storyboards

- Use throw-away materials (whiteboard, paper/pencil)
- Focus on common paths through the app
  - E.g., you may have a high-level storyboard
  - Plus specific flows through the app
  - Each path should have a specific point/goal (for the user)
- You want to make the typical paths “easy” ...
UI Storyboard Examples
From UI Sketches to Functional Prototypes

1). Start with sketches and storyboards
   ● Basic info on each screen & flow between screens

2). Create refined sketches and storyboards
   ● Add more details and organization

3). Create detailed diagrams for look and feel (“wireframes”)
   ● Take longer to develop

4). Create functional prototype mockups
   ● “Dummy” web forms, stubbed out UI screens, etc.

In 1-3, generate many alternatives!
In each step, get feedback, refine, & repeat until converge!
Some general UI Design Tips …

- Use “standard” UI components & design (where possible)
  - Follow the “law of least surprise”

- Look at (and borrow from) similar and popular UIs

- Get lots of feedback and often
  - Have users try out your sketches on paper!

- Optimize for frequent cases

- Aim for self explanatory
  - Try out on your peers, friends, etc.