Sprints 3 and 4

• Sprint 2 ends and Sprint 3 starts, this Thursday

• Break down stories into tasks & components associated with design
  – Prioritize stories
  – Assign timings to stories/use cases AND store/use-case tasks
  – Specify acceptance/test that can be used to verify a story is finished

• Sprint: Prototype tasks (primary implementation before demo)
  – Prioritize tasks
  – Assign timings to tasks
  – Specify what test(s) are to be used as evidence of task completion/acceptance (use case post condition OR user story acc test)
  – Each member/developer chooses task, implements, and tests task
  – Another member does code review/test and accepts the pull request
    • Test is the one specified above (Acceptance)
  – When store/case is complete, some member performs story test/acceptance
Team Activities Today in CS189A

- Scrum for sprint 2, finish PRDv1, prep for Sprint 3
- sync/asynch interaction diagrams & UI/Agent interaction diagrams
- Team meetings with Chandra: stories, individual tasks, burndown
- TA meetings
- **Thursday:**
  - TA meetings
  - Sprint 2 retrospective
  - Sprint 3 planning: new stories, trello (stories+tasks w/timings)
    - Burndown, assign tasks to members (2+ per)
    - Check that all stories and tasks have acceptance tests
    - 1-2 Git commits per task (add to PRDv2)
    - 3 sync/asynch interaction diagrams + 3 UI/Agent interaction diagrams
PRDv2: Your Living Requirements Document: A Shared Google Doc

- Authors, Team, Project Title
- Intro: problem, innovation, science, core technical advance (3+ pages)
  - Define project specifics, team goals/objectives, background, and assumptions
- System architecture overview
  - High level diagram (1 page)
  - User interaction and design (1+ pages) – ie detailed design
- Requirements (functional and non-functional)
  - User stories or use cases (links) → 20+ for PRDv2 prioritized w/acc. tests
  - Prototyping code, tests, metrics (10+ user stories): github commits/issues
- System models (1+ pages)
  - Contexts, interactions, structural, behavioral (UML)
  - Use cases, sequencing, event response, system state, classes/objects
- Appendices - Technologies employed
Your Project Design: PRDv2

• Architecture (hardware/software)
  – Evolve your overview picture from PRDv1 to provide significantly more detail and any updates or changes

• Detailed design
  – UML diagrams of primary data structures that comprise the system architecture connected via their associations (if any)
    • Ensure that each "class" is balanced in terms of cohesion & coupling
    • Annotate with pre/post conditions when appropriate
  – Sequence diagrams
    • synchronous and asynchronous for key interactions between classes
      – At least 3 different interactions
    • User interactions with the system
      – At least 3 different interactions
      – Can be a human user or a machine user (API) interaction
        » Event response, updated application state
      – If you have a user interface: Provide mockups for primary UIs
PRDv2 User Stories / Use Cases

• Revise spec to add detail to the functional specification to match your design

• Add user stories and break up the stories you have into finer grained stories
  – Provide UML, sequence diagrams, dataflow diagrams
  – Goal: a CS senior should be able to take your doc and implement the project

• For each fine-grained story, provide a description and acceptance test
  – Provide time estimates (1 person-hours) for each story implementation
    • Ensure you can finish the implementation in the time you have (this/next quarter)
  – Prioritize tasks to have a complete prototype by the end of this quarter
    • Focus on the externally facing interfaces, mock out what you cannot get to
  – Write unit tests to implement tasks for mandatory tasks
    • Document these tasks (autogen the documentation/usage)
  – Add trello/pivotal task links (titles must match) to PRDv2 for each story

• Prototype designed mandatory tasks; add github commit ID/link to PRDv2
  – Github must have unit tests, documentation (for anything without unit tests), and prototyping implementations for each story in Sprint

• If you have a user interface
  – Provide mockups that are tied to the functionality described in 1+ components
Use Case (Example)

Use case: Update Benefits


Precondition: Employee has logged on to the system and selected “update benefits” option

Flow of Events:

Basic Path:
1. System retrieves employee account from Employee Account Database
2. System asks employee to select medical plan type; uses Update Medical Plan
3. System asks employee to select dental plan type; uses Update Dental Plan
...

Alternative Paths:
If health plan is not available in the Employee’s area the employee is informed and asked to select another plan (exceptional cases that must be handled)
Employee selects cancel, logs out, or leaves page at any point prior to confirming the update (an end-early path)

Postcondition: Employee account plan type has been updated in the Employee Account Database or nothing has changed (end-early paths)

Note that code tests can be written for pre/post conditions
User Stories

- **As a [role], I can [feature] so that [reason]**
  - Use index cards and a sharpie
- **Make it testable with acceptance criteria or demo plan**
  - Acc. Test: Title: **Given [context], when [event], then [outcome]**
  - Should be easily coded (commit tests as part of pull request)
- **Connect the dots**
  - Lay the stories out, determine which ones are dependent on others, prioritize them in order to provide a working system/product each sprint

From: http://codesqueeze.com/the-easy-way-to-writing-good-user-stories/