Technologies to Consider + Ask Mentors

• Work on **tutorials** if new to you
• To support workflow (*required in red*)
  – Trello or PivotalTracker, Podio, Jira
  – Github
  – Issue tracking (github, Jira)
• Fast prototyping
  – firebase, react/react-native, angular vs bootstrap
• Continuous builds
  – Jenkins, travis
• Wireframes
  – gomockingbird (mockingbird), balsamiq
• Useful components/technologies
  – Oauth
• Mobile app platforms
• IDEs, programming languages

• Server and cloud:
  • System configuration: Ansible, Puppet, Chef, Saltstack/Saltcloud
  • Containers: Docker/kubernetes
  • Virtual servers/object store: AWS, Google, Azure
  • Use **free tier** & student credits
  • Platforms: Google App Engine, Heroku
  • Mobile Backends: Backendless, Google Endpoints, AWS Lambda
  • Services: MongoLab, Instacluster, Amazon RDS, Firebase
  • Hadoop/ElasticMapReduce, Spark
  • APIs: Twitter, Facebook, Google technologies (maps/earth/drive)
  • Multicloud/Java: Apache Jclouds
  • **Local Linux server/DB: email**
  • **instructor/TA**
Getting Started

• Setup public repository (GitHub: github.com or github.ucsb.edu)
  – Identify workflow: https://www.atlassian.com/git/workflows
    • Suggested: feature branch, gitflow
    • Git branching basics

• Setup an issue tracker – GitHub, Jira, other…

• UML design tools:
  – Draw your UML diagrams online (no SW installation necessary!): http://yuml.me/
  – http://www.visual-paradigm.com/solution/freeumldesigntool

• Understand/learn about writing user stories
  – http://www.mountaingoatsoftware.com/agile/user-stories,
    http://www.romanpichler.com/blog/user-stories/writing-good-user-stories/,
    http://codesqueeze.com/the-easy-way-to-writing-good-user-stories/,
    https://help.rallydev.com/writing-great-user-story
Getting Started

• Investigate language, frameworks, install necessary software
  – Debuggers, IDE, tools, **TDD**
    • Web site testing: **Selenium, Watir** Chrome DevTools, javascript debugging tools
      – Ruby: [http://guides.rubyonrails.org/testing.html](http://guides.rubyonrails.org/testing.html)

• Investigate mock up tools: flexmock, wireframing/mock ups

• Code coverage/metrics
  – Ruby: [https://www.ruby-toolbox.com/categories/code_metrics](https://www.ruby-toolbox.com/categories/code_metrics)
  – Python: [https://coverage.readthedocs.io/en/v4.5.x/](https://coverage.readthedocs.io/en/v4.5.x/)
Getting Started: Recommendations

- Set up agile process for sprint plan
  - Story writing & tasks ([Trello](https://trello.com)),
    - Alternatively: [PivotalTracker](https://www.pivotaltracker.com/) is free for public projects
    - Project board with stories (and perhaps story breakdown via tasks)
  - Trello board per sprint (Backlog, on deck, in progress, done)
    - Each card is a task with a duration/timing and 2+team members (one for implementation, one for testing/followup/review)
      - Copy tasks from story board
    - Burndown chart per sprint ([google drive worksheet](https://docs.google.com/spreadsheet/ccc?key=0AhrKfDpTBt9GZTBzc0ZzRkFrc1ZaU0F6Zy9aWllQQWU))
- Set up continuous build process: [Travis CI](https://travis-ci.org) or [Jenkins](https://jenkins.io)
- Setup container system ([Docker](https://www.docker.com)) on everyone’s laptop
- $50 reimbursable for tools/clouds/services you really need. Speak to mentor/Chandra if you need more (before you spend!)
  - Turn receipts into CS financial office, tell them to contact instructor for approval and to charge the CS Capstone grant
Use Cases and User Stories

- **Use cases:**
  - [http://alistair.cockburn.us/get/2465](http://alistair.cockburn.us/get/2465)
  - [http://www.gatherspace.com/static/use_case_example.html](http://www.gatherspace.com/static/use_case_example.html)

- **User stories:**
  - [http://www.mountaingoatsoftware.com/agile/user-stories](http://www.mountaingoatsoftware.com/agile/user-stories)
Capstone Project Requirements (1/2)

- Use of agile development process with per-sprint task tracking (recommended: Trello or PivotalTracker)
- Daily scrums recorded by scribe in shared Google Doc
  - Class/discussion days: last 15mins of class
  - Shared with Instructor, Mentor, TA, and team
- Weekly meetings (virtual is ok) with mentor
- Weekly meeting with TA
- Class/discussion attendance and participation in team activities
  - Bring laptop to class
- Vision statement turned in by deadline (& approved by mentor)
- Draft 1 and 2 of requirements specification turned in by deadlines
  - Evolve as you design and prototype; approved by mentor
- Working prototype for base functionality demonstrated in the last week of the quarter
Capstone Project Requirements (2/2)

- Use of a code repository (recommended: GitHub)
  - Ongoing contributions by all members throughout
    - Using a clear workflow
  - Can include preparation of requirements documents
- Use of an issue tracker (recommended: github)
- Documented code
- Automated unit tests and integration and/or functional tests
  - Code defensively!
- Use of user stories and/or use cases for requirements and design
- Use of UML for system requirements modeling and design
- Wireframes for user interfaces if any
- Complete 4 2-week sprints, record retrospectives for each
On-going Process

- Evolving (aka “living”) requirements document
  - Identify/learn (and teach each other) the technologies required
  - Write user stories in particular; update the requirements as you go:
    - Prioritize stories and mark **mandatory, important, or desirable**
    - Assign time estimates to stories; improve your estimation ability over time
    - Specify **acceptance test** for each story – should be in code

- Concurrently as part of Sprint
  - Break down stories into tasks (begin design/prototyping process)
    - Prioritize tasks
    - Assign timings to tasks
    - Specify what (code) test(s) are to be used as evidence of task completion/acceptance
    - 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 a Story is complete, some member performs story test/acceptance