

# Milestone Five

## Primary Objectives:

1. Demonstrate how to hold a successful retrospective meeting
2. Prove architecture: continuous deployment to Azure
3. Team project inception concludes
4. Team project construction ready to begin

## Overall Requirements:

- Class Project Retrospective artifacts
- Proof of working continuous deployment to Azure (class project)
- Team Project Inception (Part 3)
  - Updates of documents, modeling, and other artifacts
  - Development guidelines/team rules/requirements/... "Contributing to Project" written in Markdown format and linked from project Git repo welcome page
- Team Project Construction
  - Backlog refinement/grooming
  - Backlog and user stories look really good
  - Select user stories to prove architecture early in iteration
  - Select user stories for each team member for full 2 week iteration

Grade sheets (ods)

**READY NOW for 2020**

## Tasks:

1. **Class Project Retrospective** — hold a retrospective meeting with your team. There needs to be a time set aside for every team member to contribute. Let each person say what went well and what didn't. Be positive, say what went well. When something didn't go well, still be positive. Treat it as a problem that needs to be identified and then solved.

Take short notes as you hold your meeting and then write up a short summary (that of course goes in your repository) of the following:

1. Conduct a Safety Check; what were the results?
2. What did you learn from the Sprint?
3. What isn't going well?
4. What can the team do better during the next Sprint? You must make a specific Action Plan.
5. Are there any items that need to be brought up with someone outside the team? (i.e. in this case the instructor)

2. **Class Project Continuous Deployment** — Set up continuous deployment to Azure from your GitHub repository for your class project. This should be configured to automatically build and deploy from your `master` branch to a Azure Web App (that you'll use for the rest of the term). (Note: we're not doing continuous deployment of our database. That is for now updated and deployed manually.)

The intent is to learn how to do this now; then for Sprint 1 of the Team Project you can simply switch over to your team project, using the same infrastructure. Note, it is also trivial to change which branch the deployment is based off of, so you should test this off your `dev` branch and then when it's working, switch back to `master`.


Be prepared to demonstrate live during your review meeting that continuous deployment is set up and working correctly. You'll be asked to make a trivial change to a display element in your application and merge the change into `master` during the meeting. The website should then update automatically with your change. Have this ready to go before the meeting to save time. It should only take 1 minute to do this. (The official maintainer can do this from a feature branch off master, directly merging into master with no pull request.)

3. **Team Project Inception (III)** — Now's the time to finish your inception activities. All items in the template markdown file should be completed. Expectations are that each team is prepared to demonstrate the highest quality work.

You should have all the features for your "base application" well defined. This means discussed, modeled (architecture, Use Case diagrams, UI wireframes, ...) and described with text and lists. Remember, you are defining the requirements. It should not be ambiguous or unknown what the client wants and how they want it to work. From these base features you should have defined at least 6-10 Epics each. From there you should have good, well defined user stories that will get you through the first two sprints. See below for how many this is. At this point in the milestone you have not yet estimated effort for the user stories so you really don't know how many you'll need to get through the first 2 sprints. So get quite a few ready in your backlog.

All Epics and user stories should be entered into your new Pivotal Tracker project and that project should be set up correctly (see below).

**4. Team Project Inception (III)** — now that you're about ready to begin construction, you should settle on some procedures that the team has agreed (amongst themselves) to follow. Basically, to be a part of the team, what do you need to do or know? Write all this up and include it in a very visible way on, or hyperlinked from, your Welcome page on GitHub. All these should be in Markdown format and available by clicking on links from your Welcome page. Here are some things I've thought of.

- A description of your project on your Welcome page. You've got it figured out, and have a Vision statement, so put it there for any visitors to see. Explain your project. If it helps, pretend this is an open source project that someone far away might want to contribute to. Explain it to them. Here's a very brief version I wrote up for a CS 490 class: WOU 3D Scanner. And here's a far better version from a past team: StarTech Do not simply copy these. You are deciding yourselves what you want to do, so talk about it and then write it up.
- Guidelines and Contributing code to the project. How to contribute. What rules to follow when writing code? Database? Should you always make the primary key just ID or use the style of ProductID ? What about id's in HTML elements? Class names? Model naming convention? Here's my too short example from the same class: Guidelines
- Clearly list all team members. Link to their personal GitHub pages. This will be a great project for potential employers to look at, so make it obvious you worked on it!
- What software construction processes or lifecycles are you using? Do you have a link to any team pages?
- Any Team Rules? Team song?(← required) Team cheer? Dance moves? Mascot? 
- List of tools used, including versions where relevant. Continuous development/deployment? Link to your deployed site?

**5. Team Project Construction** — hold a backlog refinement/grooming meeting and then also a planning meeting (probably do them at the same time). Now, right before your first real 2-week Sprint is the time to get your backlog into shape. Your goal is to have a nice looking, orderly backlog. It has to be ready for the first sprint with really good user stories. Here are some requirements:

1. All epics for all core features are entered and in priority order
2. All user stories that you've written to this point are entered
3. Icebox is prioritized
4. You've estimated the effort points (as a team) for enough of the highest priority stories to make 1.5 or 2 sprints worth (see below for how many points this is)
5. The stories for the previous item all are very well defined and have tasks. As the video on the Moodle page states, the expectation is that all but the simplest user stories will have 250+ words of description/specification/acceptance criteria (the "what" and "why"). Tasks are written to itemize the steps needed to fully implement

and test (manually) the story (the "how"). If you find that you don't yet know some of the "what" or the "how" for stories that will go into sprint 2 that's totally normal -- following Agile you are not expected to know everything up front. Just note that where appropriate in your description and you'll tackle that when it comes up in the next grooming or planning meeting.

6. You may want to select one team member to act as the Scrum Master to keep your meeting on task and working efficiently. We haven't practiced Scrum Master skills yet so you'll have to wing it.
7. Select and commit user stories for Sprint 1. This is a two week, 100% capacity Sprint, so you have quite a bit of time. Be sure to have the first few user stories be ones that prove your architecture and get everything running smoothly before trying to add significant functionality. Handle dependencies the best way you can. If your project will have individual user accounts then now is the time to set it up. If you didn't do it for the class project here's a link to the writeup that should help: [Using ASP.NET Identity](#)

You need to have enough stories in the Sprint so that **each team member has 6 effort points per sprint**. That's  $6 * 4 = 24$  effort points total if you have 4 team members.

Let's go ahead and put some labels on our effort points:

- 1: extra small (XS) -- often only slices the UI, or barely the controller
  - 2: small (S) -- usually slices UI and controller and might add a model. More complicated CSS work here or non/simple AJAX JS could bring a story up to a 2.
  - 4: medium (M) -- usually slices everything and includes more complicated model work (multiple tables added or sliced with FK's), or it adds complexity in the front end (JS, usually with AJAX) or the back end (using an external API). Think of 4 points as one week's worth of work for a student who passed CS 460 with a solid B grade. That's about one of our homeworks from last term except you're doing it all independently and it needs to both look really good and be fully tested.
  - 8: large (L) -- a big one; takes the whole sprint for one developer. **Not allowed until Sprint 3 and beyond.**
8. Ideally you want each team member to select their own stories on a just-in-time basis, limiting WIP (work in progress) to one story per person at a time. For this first sprint, however, go around and choose the entire sprint's worth of stories for each person. But you only work on one user story at a time!
  9. Be ready for the start of the sprint on Monday. Plan to have 4 points worth of stories done per developer within the first week, and earlier if there are dependencies with a teammate's story. **Do not under any circumstances think that you have 2**

**weeks to do this work and then wait until late in week 2 to start or do all your work. Imagine if you were a professional developer and did that in your job.**

**What would happen?** If you are sick or otherwise unable to work on your project, immediately alert your teammates and then your project advisor. Teammates should do what they can to continue with the project. We will deal with events like this on a case-by-case basis.

10. At the end of the sprint we'll have a Sprint Grading Sheet for you to fill out and have ready at the Sprint Review Meeting. This will have a place to grade your sprint planning and then individual lines for grading each developer's user stories. Watch for it next week.