

# CS361: Assignment 2: Environment Setup, Course Project Plan, and Sprint 1 Plan (for Milestone #1)

**Overview**

Now that you’ve been introduced to the microservices concept, start planning your course project (except for the microservice you’ll later make for a teammate) and your Main Program (Milestone #1). It’s OK to change your plan later!

This assignment has three parts:

* **Part 1**: Environment Setup. Initialize your GitHub repository and investigate task management systems to organize your project tasks.
* **Part 2**: Course Project Plan. Write all the user stories you would like to be part of your course project (except for the microservice you’ll later make for a teammate). It’s OK if you don’t implement all of them this term.
* **Part 3**: Sprint 1 Plan. Select at least three user stories to implement during Sprint 1 (for your Main Program / Milestone #1). Define detailed requirements for each user story.

**Note these minimum requirements for Milestone #1:**

* Your Main Program implementation must offer value to users
* At least three user stories are completed
* All features that are part of the milestone must be working. The milestone must not have partially completed features.
* It allows users to interact (e.g., provide input, push buttons, etc.)
* Reflects each of the Inclusivity Heuristics
* Reflects three quality attributes of your choice (i.e., satisfies the non-functional requirements you write for each quality attribute)

*Hint: If you choose “usability” or “inclusivity” as a quality attribute, your corresponding non-functional requirement can involve the Inclusivity Heuristics.*

**Part 1: Environment Setup**

Set up your development environment. In addition to an IDE or code editor (choose any you prefer), start a GitHub repository and choose a task management system.

Complete each item below by replacing the highlighted text (**Usability note**: double-click the text to select it).

1. **GitHub Repository**

Create a GitHub account if you don’t already have one, create a Git repository hosted on GitHub. This first repository will be for your Main Program. Make a **test commit**. The test commit should show up on GitHub.

1. What is your GitHub username?

|  |
| --- |
| *Ashton-Haviland* |

1. Provide a screenshot of your test commit.

|  |
| --- |
|  |

Later, it would make sense for you to set up additional repositories, one for each of the microservices you’ll implement.

**2) Spike: Task Management Systems**

For your course project, you will be using a task management system to keep track of development tasks. Spike at least **three** task management systems you could use.

A spike is a quick, directed effort focused on getting a question answered. Performing a spike can help you make intelligent decisions. Spikes do take time upfront, but they can also save you from making a bad choice that takes much more time to recover from. This portion of the assignment provides an opportunity to do a spike while making a relatively low-stakes decision (which task management system to use).

Examples of task management systems you could spike: Trello, Jira, Asana.

**Requirements for the task management systems**:

* Software specifically designed for task management
* Support for collaboration, task definition/deletion/updating, task priorities, task due dates, assigning people to tasks, setting task status, and the ability to label/tag tasks or put them in different columns.

To do a spike, you need to research the task management systems and also (1) try to **use** them, (2) **evaluate** them based on specific criteria, (3) **compare** them, and (4) **decide** which to use.

1. Which task management systems did you spike?

|  |
| --- |
| *Jira* |
| *Trello* |
| *Asana* |

1. **Try** each system. Create a task then update it, assign it, delete it, etc. **Screenshot** your task in each system and paste it below. Name the tasks **"CS361 Test Task"**.

|  |
| --- |
|  |
|  |
|  |

1. For each, **evaluate** against at least these criteria:

**Ease of use**. Ex: Is it intuitive to learn? Easy to remember how to use it? Do you find yourself making lots of errors trying to use it? Are there tutorials and documentation?

**Name of system 1:** *Jira*

**Evaluation of system 1’s ease of use (2+ sentences):** *Was very difficult overall to find how to set up sprints/scrum, and where there tasks etc where. Good amount of customization, but is bright mode, and the sheer amount of customizability/recessed options make it hard to operate.*

**Name of system 2:** *Trello*

**Evaluation of system 2’s ease of use (2+ sentences):** *Simplistic with intuitive controls for click and edit, less errors made but still some. Good documentation and tutorials.*

**Name of system 3:** *Asana*

**Evaluation of system 3’s ease of use (2+ sentences):** *Like the coloration and grouping, the tasks system was a little unintuitive but easy enough. Accidentally entered a 30-day free trials instead of free account though.*

**Speed/responsiveness**. Ex: Does it take an annoyingly long time to log in

/ load / create new projects / etc. or is it peppy?

**Name of system 1:** *Jira*

**Evaluation of system 1’s speed/responsiveness (2+ sentences):** *Login was weirdest here, took more time to login/get to boards. Creating new projects was unintuitive.*

**Name of system 2:** *Trello*

**Evaluation of system 2’s speed/responsiveness (2+ sentences):** *Login was easy enough, new projects are simply new boards. Different views/project easy to enter.*

**Name of system 3:** *Asana*

**Evaluation of system 3’s speed/responsiveness (2+ sentences):** *Easy to find and create projects/coordinate teams. Login was standard fair, but did accidentally create a free trial account.*

**Feature set**. Ex: Besides the required features, does the system have other features you are likely to need?

**Name of system 1:** *Jira*

**Evaluation of system 1 (2+ sentences):** *Includes backlog/code development inbuilt, as well as a timeline. Features could prove useful especially for sloppy development/actual introduction of secondary parties in the coding group.*

**Name of system 2:** *Trello*

**Evaluation of system 2 (2+ sentences):** *The views and boards are nice organizational method that could help with brainstorming. These views could also help with any revisionary steps, though a lack of a link to the code hinders it slightly.*

**Name of system 3:** *Asana*

**Evaluation of system 3 (2+ sentences):** *While you can link files, and it has a calendar view, it isn’t specifically for coding projects, and as such is a little clunky overall for these side benefits.*

**Relevance/popularity**. Ex: Is it likely you will ever see the task management system again after the course?

**Name of system 1:** *Jira*

**Evaluation of system 1’s relevance popularity (2+ sentences):** *Once more fairly likely as 65,000+ companies use Jira.*

**Name of system 2: Trello**

**Evaluation of system 2’s relevance popularity (2+ sentences):** *Fairly likely as as Trello has 50 million users as of October 2019.*

**Name of system 3:** *Asana*

**Evaluation of system 3’s relevance popularity (2+ sentences):** *Fairly likely as Asana has millions of users and 150,000 customers(companies?).*

1. **Compare** the systems by **ranking** them based on the criteria above. Best to worst for each criterion. **List or table format**.

**System 1 name:** *Jira*

**System 1 ease of use:** *3*

**System 1 speed/responsiveness:** *3*

**System 1 feature set:** *2*

**System 1 relevance/popularity:** *3*

**System 2 name:** *Trello*

**System 2 ease of use:** *1*

**System 2 speed/responsiveness:** *2*

**System 2 feature set:** *1*

**System 2 relevance/popularity:** *1*

**System 3 name:** *Asana*

**System 3 ease of use:** *2*

**System 3 speed/responsiveness:** *1*

**System 3 feature set:** *3*

**System 3 relevance/popularity:** *2*

1. Which system is the **highest ranked?**

|  |
| --- |
| *Trello* |

Decide which task management system you’re going to use and use it to complete Parts 2 and 3.

**Part 2: Course Project Plan**

Write the entire set of user stories for your course project (except for the microservice you’ll later make for a teammate). Put the user stories in a **Product Backlog** column/section/category of your task management system, or a label/tag the user stories “Product Backlog”. **You probably won’t have to finish implementing the entire Product Backlog this term**.

Complete each item below by replacing the highlighted text (**Usability note**: double-click the text to select it).

1. **Product Goal and Backlog**

You’ll be using *some* Scrum methods in this course. Unfortunately, the Scrum Master and Product Owner roles don’t work well in this course setting. You will, however, experience Scrum Events and Artifacts.

1. What is your **Product Goal** for your course project? This includes your Main Program, Microservice A that your teammate will later implement for you, and Microservices B, C, and D. It does NOT include the Microservice A you will make for a teammate.

|  |
| --- |
| *Develop a desktop app that stores user notes as well as consumed media with attached notes for future reference, and optional customizable study timer to increase studying efficiency.* |

The Scrum Guide (<https://scrumguides.org/scrum-guide.html>) doesn’t give a detailed description of the Product Goal: “**describes a future state**”, “**long-term objective**”.

Example Product Goal: “Develop a desktop app that listens to what people are saying and automatically shows content that might be relevant to their conversation.”

1. **Write user stories** for your **entire** **course project** (except for the microservice you will make for your teammate). Use INVEST to guide you.

**Assignment requirements for Product Backlog user stories**:

* Each has a **name** that briefly describes the functionality (e.g., “Login”)
* Each uses the **“As a… I want to… so that…” format** (explained in textbook)
* Each is about **functionality** and not about the quality of the functionality or a constraint (user stories are functional requirements, not non-functional requirements)
* Total of at least **10** user stories (you will not have to implement all of these)
* As a set, must have **no obvious violations of INVEST**
* **User story 1:** *Movie Notes*
* **User story 1:** “As an avid movie consumer, I want to take notes on specific movies so that I remember and learn from the movies I watch.”
* **User story 2: Categories**
* **User story 2:** “As a student I want to better organize my notes so that I can study my subjects better, and review them more efficiently”
* **User story 3:** *Edit*
* **User story 3:** “As a student I want to be able to edit my old notes so that I can fix the gaps in my understanding.”
* **User story 4:** *Add*
* **User story 4:** “As a student I want to take notes so that I can efficiently review and recall what I’ve learnt.”
* **User story 5:** *Delete*
* **User story 5:** “As a user I want to delete old notes so that Remove Redundant/incorrect information.”
* **User story 6:** *Group read*
* **User story 6:** “As a student I want to want to read a grouping of notes so that I can study for a specific exam.”
* **User story 7:** *Find note*
* **User story 7:** “As a user I want to Navigate through my notes so that I can find information relevant to my question.”
* **User story 8:** *Add movie*
* **User story 8:** “As a Movie watcher I want to Add to the list of watched movies and create a relevant note so that take notes on the movie I just watched.”
* **User story 9:** *Movies By Genre*
* **User story 9:** “As a Movie aficionado I want to see my movies by genre so that I can decide what to watch tonight.” format
* **User story 10:** *Timer*
* **User story 10:** “As a student I want to use an inbuilt timer so that efficiently focus on my studying.”

Enter the user stories into your task management system in a **Product Backlog column/section/category**, or with a “Product Backlog” label/tag. Paste a **screenshot** below so that the grader can confirm you added the stories.

|  |
| --- |
|  |

**2) Quality Attributes**

Quality attributes can help guide the entire development of your project. They can remind you (and other developers) what aspects of your project matter the most and can help you decide which features to implement and in what way.

**Select the top three quality attributes you care about** for your course project. See <https://en.wikipedia.org/wiki/List_of_system_quality_attributes> **for ideas.**

1. **Which three quality attributes did you choose? Name** and **define** each**.**

* **Quality attribute 1:** *Integrity*
* **Quality attribute 1 definition:** *How frequently software loses data.*
* **Quality attribute 2:** *Maintainability*
* **Quality attribute 2 definition:** *Amount of effort needed for developers to update, refactor, or otherwise modify the software’s code.*
* **Quality attribute 3:** *Memorability*
* **Quality attribute 3 definition:** *Amount of time users must spend relearning functionality.*

1. **Why did you choose these quality attributes?** Explain how each quality attribute is particularly relevant to your project (1+ sentence per quality attribute)

* **Why quality attribute 1 is relevant to your project:** *As a note taking application, any loss of data is inexcusable unless end user specifically wished for data to be lost.*
* **Why quality attribute 2 is relevant to your project:** *Continuous need for upkeep would reduce operability of simple note taking software.*
* **Why quality attribute 3 is relevant to your project:** *As a note-taking software for those looking to reduce time, memorability is key as having hidden features/convoluted menu’s that require relearning is anathema to time management.*

**Part 3: Sprint 1 Plan (for Milestone #1)**

Next, move some user stories from your Product Backlog to your Sprint Backlog, or change the label/tag to “Sprint Backlog”. These will be the user stories you WILL implement during Sprint 1 (for Milestone #1 / your Main Program) and comprise your Sprint Plan. Your Milestone #1 Main Program implementation must offer value to users.

1. What is your **Sprint Goal**? The Sprint Goal must clearly communicate what you plan to work on (e.g. what pages, what functionality, etc)

|  |
| --- |
| *CRUD operations* |

1. Select **at least three** user stories from your Product Backlog and move them to your Sprint Backlog, or re-label/tag them “Sprint Backlog”. Because you will be implementing these user stories during the Sprint, you need to write more specific requirements in the form of **acceptance criteria**.

Acceptance criteria can cover both functional and non-functional requirements. The non-functional requirements can serve to carry through your intention to reflect quality attributes.

Some developers write their user stories on 3” by 5” index cards: The user story name and “As a” format goes on the front of the card and the acceptance criteria can go on the back. **Example**:

|  |
| --- |
| (Front of index card)  **Automatic IMDB** (“As a … I want to … so that …”)  **As a user** speaking during a conversation, **I want to** automatically see the IMDB.com webpage for the movie I’m talking about **so that** I can continue with my conversation and examine the webpage as needed. |
| (Back of index card)  **Acceptance criteria**  Functional requirements (“Given… when… then…”)   * **Given** a person is speaking in English at 60 dB or louder **when** the software is at least 80% sure it knows what movie the person is talking about, **then** it will open and focus the default web browser and navigate to the movie’s IMDB.com webpage.   Quality attributes & Non-functional requirements   * Responsiveness: Once the software is 80% sure about what movie is being spoken about, it will display the movie’s IMDB.com webpage within 3 seconds. |

Use this format to fill out each of your Sprint Backlog user stories.

**Assignment requirements for Sprint Backlog user stories:**

* For each of the three (or more) user stories…
  + The front of the card must contain the user story’s name and “As a… I want to… so that…” format
  + The back of the card must contain at least one functional requirement and each functional requirement must use the “Given… when… then…” format.
* Each of your three quality attributes must appear at least once on a user story’s “back of index card” and must be converted to a non-functional requirement.
* All of the functional and non-functional requirements must be testable.

Later, you will be asked to show that your functional and non-functional requirements are met.

**First user story**

|  |
| --- |
| (Front of index card)  *Add*  *UserStory ”* As a student I want to take notes so that I can efficiently review and recall what I’ve learnt.” |
| (Back of index card)  **Acceptance criteria**  Functional requirements   * *Given A user has input command add when the software detects command then input user with fields for note addition.*   Quality attributes & Non-functional requirements   * *Memorability: Each field will be as intuitive and nondescript as possible.* |

**Second user story**

|  |
| --- |
| (Front of index card)  *Find Note*  *UserStory* “As a user I want to Navigate through my notes so that I can find information relevant to my question.” |
| (Back of index card)  **Acceptance criteria**  Functional requirements   * *Given a user has input View Notes when software detects command then displays user input notes*   Quality attributes & Non-functional requirements   * *Integrity: Once software is sure of user input/note, displays note’s full input data.* |

**Third user story**

|  |
| --- |
| (Front of index card)  *Delete*  *UserStory* “As a user I want to delete old notes so that Remove Redundant/incorrect information.” |
| (Back of index card)  **Acceptance criteria**  Functional requirements   * *Given a user has input delete when view a card then system prompts to ensure correct inputs, then deletes card. Format*   Quality attributes & Non-functional requirements   * *Maintainability: User can shore up information/incorrect inputs.* |

1. Take a **screenshot** that shows you’ve moved these user stories into a Sprint Backlog in your task management system.

|  |
| --- |
|  |

Your **Definition of Done** for Sprint would typically include, “The acceptance criteria are satisfied for all Sprint Backlog user stories.” You aren’t required to write your DoD or put it in your task management system.

This would also be **a good time to break each of your user stories into a list of specific tasks** you need to complete. Task management systems are, as you might imagine, a great place to do that!

**Submission**

PDF or Word format via Canvas.

**Grading**

You are responsible for satisfying all criteria listed in the Canvas rubric for this assignment.

**Questions?**

Please ask via Ed so that others can benefit from the answer.