

CS361: Assignment 6: Sprint 2 Plan (for Microservice A)

# Overview

Plan the microservice you’ll make for your teammate(s). That includes defining how to **request** and **receive data** from the microservice.

# Instructions

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

1. Which teammate will you implement Microservice A for? Name the teammate.

|  |
| --- |
| *Scott Ladd* |

1. Describe the Microservice A that you will implement for the teammate listed above. (e.g., the Microservice A I will implement in Sprint 2 for my teammate will spell-check phrases provided by their Main Program)

|  |
| --- |
| The microservice I will implement in Sprint 2 for my teammate will average an array of numeric ratings sent via an HTTP POST request and return the result. |

1. What is your **Sprint Goal**? (e.g., fully implement the spell-checker microservice) The Sprint Goal must clearly communicate what the microservice will do.

|  |
| --- |
| Fully implement a ratings-averager microservice that takes a JSON array of numbers via HTTP POST and returns the average. |

1. Define **at least three user stories** for this Sprint. Provide your user stories and their functional and non-functional acceptance criteria (and associated quality attributes).

**Requirements for Microservice A:**

* You must implement at least three user stories.
* Each user story must have a name.
* Each user story must use the “As a… I want to... so that…” format.
* Each user story must have at least one functional acceptance criterion.
* All functional acceptance criteria must use the “Given… when… then…” format.
* At least one of the user stories must have an associated quality attribute and non-functional acceptance criterion.
* All non-functional requirements must be testable.

**First user story**

|  |
| --- |
| (Front of index card)  *User Story Name:* Submit Ratings  As a user, I want to submit a list of numeric ratings to the microservice so that it can calculate the average rating. |
| (Back of index card)  **Acceptance criteria**  Functional requirements  *Given* a JSON array of numeric values, *when* I submit it to the /average endpoint via POST, *then* the response should return the correct arithmetic mean of the values.  Quality attributes & Non-functional requirements  Quality Attribute: Performance  The average must be computed with a response time of less than 100ms for arrays of up to 1,000 numbers. |

**Second user story**

|  |
| --- |
| (Front of index card)  *User Story Name: Handle Invalid Input*  As a developer, I want the microservice to handle invalid or malformed input gracefully so that it doesn’t crash or return misleading data. |
| (Back of index card)  **Acceptance criteria**  Functional requirements  *Given* an invalid or non-numeric array, *when* I send a POST request to the /average endpoint, *then* the service should return a clear error message with HTTP status 400.  Quality attributes & Non-functional requirements  Quality Attribute: Reliability/Usability  The service must validate input and provide human-readable error messages for common input errors. |

**Third user story**

|  |
| --- |
| (Front of index card)  User Story Name: Zero Ratings Case  As a user, I want to know the behavior of the microservice when submitting an empty array so that I can handle edge cases in my application. |
| (Back of index card)  **Acceptance criteria**  Functional requirements  *Given* an empty array, *when* I send it via POST to the /average endpoint, *then* the service should return a predefined response (e.g., 0 or "error": "no ratings").  Quality attributes & Non-functional requirements  Quality Attribute: Consistency  The microservice should maintain consistent return types even in edge cases. |

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

|  |
| --- |
|  |

1. What kind of **communication pipe** will the microservice use? (e.g., text files, REST API)

|  |
| --- |
| REST API using HTTP and JSON |

1. How will other programs be able to **request data** from the microservice and what request parameters will be required? Give an example call using pseudocode or actual code.

|  |
| --- |
| POST body with key ratings, containing an array of numeric values. |
| The microservice will listen for HTTP POST requests at the /average endpoint. Clients must send a JSON body like {"ratings": [4, 5, 3, 5]}. The server will parse the array, calculate the average, and return it. |
| import requests  response = requests.post("http://localhost:5000/average", json={"ratings": [4, 5, 3, 5]})  print(response.json())  # Expected: {"average": 4.25} |

1. How will other programs be able to **receive data** from the microservice and what data will the microservice provide? Give an example call using pseudocode or actual code.

|  |
| --- |
| JSON |
| The microservice will respond to HTTP POST requests with a JSON object containing the calculated average. In case of an error, it will return a JSON error message. |
| Request: POST /average  Body: {"ratings": [2, 3, 4]}  Response:  Status Code: 200  Body: {"average": 3.0} |

**This would be a good time to make a new repository to house the microservice.**

# Submission

PDF or Word format via Canvas.

**You must follow instructions at Modules > “Attach a Document to "Text Entry" Field”.**

# 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.