# Find an Expert Sprint 2 Evaluation

---

Spencer Plant

## Updates to Prior Deliverables

### User Stories:

- I think the user story numbers were mixed or changed up since Sprint 1, because the stories I referenced in the Sprint 1 deliverable do not match up with the current user stories. That said, it seems as though my feedback from Sprint 1 has been acknowledged here.
- User story 1.07 is missing

### Storyboards:

- Unchanged. UI not yet implemented.

### Architecture Design:

- Unchanged.

### Planning:

- US 1.01 moved from Sprint 2 deliverable to Sprint 4 deliverable as per my suggestion.
- More evenly distributed tasks across sprints as per my suggestion.

## What is New

### Code:

If I don't mention a file specifically, you can assume I looked at it and saw no issues.

- **blueprints/data_ingestion.py** is very readable and understandable thanks in part to variable names, but mostly because of the concise inline comments

- **common/exceptions.py**: I notice the three exception classes you have defined are identical in their functionality. This could just be because you are saving it for a later sprint. The names are pretty explanatory, but the file could benefit from comments describing each exception.

- **common/utils.py:** in the *generate_names_from_json* function, the regex could benefit from a comment briefly describing its purpose. Same goes for the regex at the top of the FacultyNames class. I assume that the last two lines of this file are for testing purposes, with JNelson.Amaral hardcoded; maybe say this.

- **components/data_pipeline/tasks/get.py:** *is_requirement_satisfied* returns True if the data passed to it is null? This is strange to me. I also notice that the *data* parameter passed to the *run* function is unused.

- Good use of object-oriented design in **components/data_pipeline/tasks/** directory, having each class inherit from a generic *Task* class.

- **components/data_pipeline/workflow.py:** There is a break from your comment standard that you've adopted. In the *run_next* method, same-line comments are used, whereas for the rest of the project thus far, comments have had their own dedicated lines.

- **components/key_generation/key_generation_approach.py:** unfinished. Is this going to be "TextRank" that you mention in the GitHub wiki?

- It looks like **components/key_generation/** classes could all inherit from an *Approach* class or something similar. Also, reorder the class methods to be consistent. For example, RakeApproach has *__init__, generate_keywords,* then *get_id*. To be consistent, the other classes in this directory would have these methods appear in the same order.

- **components/data_ingestor.py** has a differently spelled class name ("DataIngest**e**r vs. data_ingest**o**r.py). Probably a good idea just to keep things consistent.

- **models.py:** there is inconsistent spacing above class definitions. Namely, *Keywords* only has one blank line above it as opposed to the other classes which all have two.

# Unit Tests

- Tests pass.

- The difference between *test_generic_approach.py* and *test_key_generation_approach.py* isn't immediately apparent to me. One creates a "GenericApproach" and the other creates a "KeyGenerationApproach" of type "key-generic". My first intuition tells me these tests are identical in what they test.

- test_models.py doesn't test if the entire Faculty object created in TestFaculty is accurate. While you can probably safely assume that the other fields are saved, it would be better to assert that all the fields were set properly.

# Functional Test

The demo was good and hitch-free; everything performed as expected. The fact that three separate cURL commands had to be made felt a little tedious, but it showed off each step's output effectively. I suggest combining the commands into a single command to expedite the process, or, build a UI for the user.

# Consistency with Design

The delivered code is consistent with the overall architecture design provided in the Sprint 1 deliverables.

- Scrapp class implementation doesn't have a "keywords" array as specified in UML
- "OrcIdScraper" in UML specifies "get_contents" method, but implementation has different name ("get_scrapps").
  - Same for "ResearchIdScraper"
  - Same for "ProfileScraper"
  - Same for "Scraper"
- Search Engine: mostly tackled in Sprint 3.
- Key Generator Class diagram can be updated to reflect the three different approaches now being pursued.
- Publication, Faculty, and Grant classes showcased in Data Ingester diagram aren't implemented. **data_ingestion.py** contains two classes not in the diagram, "FacultyAPI" and "FacultyListAPI".

# Quality Considerations

- I don't think security is as much of a concern for this application as others, as there isn't really any logging-in of any sort.
- Where there is future work planned, the outline of the code is completed. Good structure, easy to make additions to.
- Nice modularity in instances like tasks and key generation approaches. If in the future other tasks or key-gen approaches were desired, it would be relatively easy to add them to the setup, from the way things look.
- I realize your client mentioned that a UI isn't entirely necessary, so long as there are instructions on how to use the tool. While that leniency is welcome, a functional UI would make every user's life much easier. Nobody enjoys reading documentation (as I write you a full paper of documentation).
- The comment standard you have adopted in this project is mostly consistent. I suggest following PEP 8 comment styling as this is a widely used standard.
- Code coverage is good, as with more coverage, the less likely unexpected bugs sneak into the final product. That's a good tool you've included for going forward.