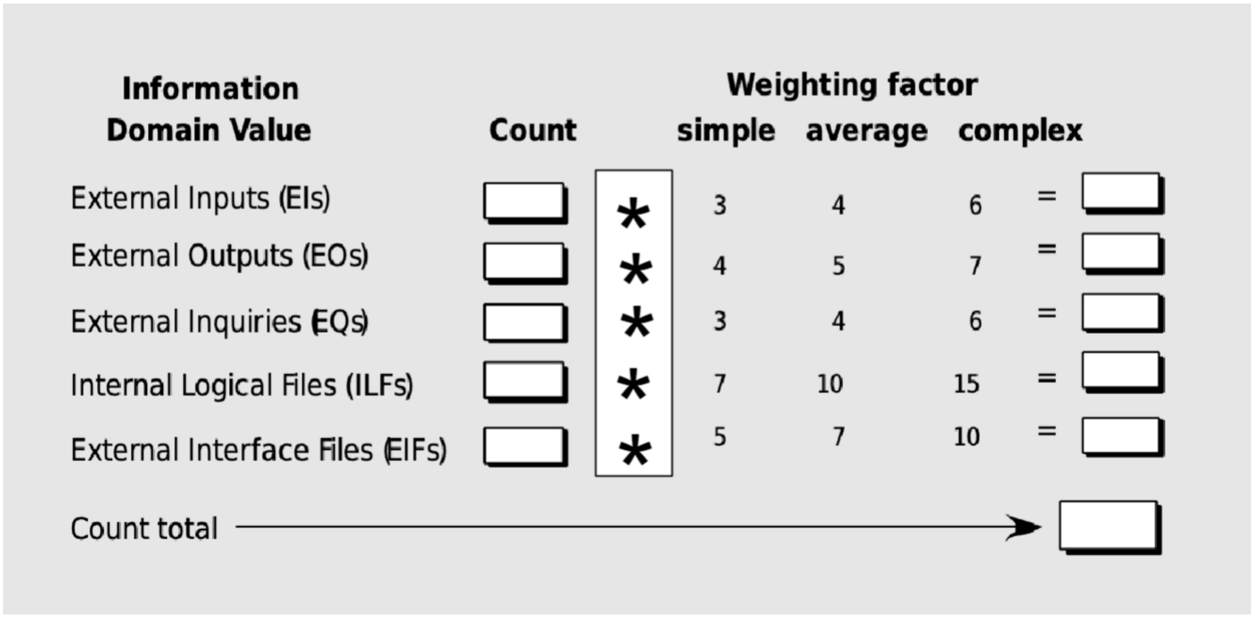
Minute Match: Preston O’Connor
Group 3: Preston, Lawrence, Nathan, Steven
Manager: Aditi



Information Domain Value	Count		Weighting factor simple	average	complex		
External Inputs (EIs)		*	3	4	6	=	
External Outputs (EOs)		*	4	5	7	=	
External Inquiries (EQs)		*	3	4	6	=	
Internal Logical Files (ILFs)		*	7	10	15	=	
External Interface Files (EIFs)		*	5	7	10	=	
Count total						→	

As of the code set up we have so far,

For our previous deadlines, we implemented a solution to use case 1. This was a user who wanted to post a service they are offering.

Internal Logical File

- The application for use case 1 maintains two large internal data tables: the posts and category tables. The posts table holds all user-submitted records and data such as post type (offer or request), category, description, image, timestamp, and optional group tags. The category table holds the range of service types available for users. Both tables are internally managed, created, and updated, so they fall under IFPUG standards as Internal Logical Files (ILFs).

External Interface File:

- There are no valid External Interface Files in this scenario. All data used in the app, such as posts and categories, is internally stored. Although front-end components read from these tables for dynamic presentation, this behavior does not make them EIFs..

External Input:

- The users interact with the system by posting new entries through a form on the page. They indicate whether the entry is an offer of service or a service request, select a category, provide a text description, and optionally upload an image or select a group tag. This data is validated and then added to the posts table. Since the user input is from external to the system and results in an update of internal data.

External Output

- The front page dynamically displays a list of posts contributed by users. Each entry holds formatted information such as the date, description, category, optional image, and membership in a group. This data is retrieved from the posts table and presented in the front end, often with presentation and formatting logic introduced.

External Inquiry:

- Users can query or filter by category, group, or keyword. These actions read from the posts table without changing it and display the result directly on the interface. Since there is no calculation, conversion, or writing to the database, this type of interaction is an External Inquiry.

T1 – Internal Logical File

- Effort: 3 person-days
- Duration: 2 days
- Dependencies: None

T2 – External Input (form UI + backend endpoint)

- Effort: 3 person-days
- Duration: 2 days
- Dependencies: T1

T3 – External Output (display feed)

- Effort: 2 person-days
- Duration: 2 days
- Dependencies: T1, T2

T4 – External Inquiry (search/filter UI + route)

- Effort: 1.5 person-days
- Duration: 1 day
- Dependencies: T1

T5 – Basic Integration Testing

- Effort: 2 person-days
- Duration: 2 days
- Dependencies: T2, T3, T4

A comparison of the deadlines we can see vs the ones we implemented:
For the 9 days when we were working, the guidelines and implementation closely adhered to the overall project timeline. In total, our team spent about 12 days on the whole use case. This included the initial setup and several brainstorming sessions so that we would be on the same page with the code implementation. We also had to spend a couple of days debugging unexpected issues, which included asking for assistance from an individual on the team who had implemented parts of the function. Overall, I believe that our team stayed on track with this setup and arrangement, as the features were not too time-consuming or difficult to implement within the timeframe.

For the upcoming Use case. I will take a full-stack approach and implement use case 4 of our project. Where the user is selecting a ranking for someone they received help from

Internal Logical File:

- Our application's category table and posts table are Internal Logical Files. They are collections of related data, recognizable to a user, which the system generates, employs, and maintains itself. The posts table stores all user-content information like post type (offer or request), category, text description, image URL, timestamp, and optional group association. The category table stores various service types available for posts.

External Interface File:

- In this particular implementation, there are no actual External Interface Files. All that is consumed and referenced by the application is created and stored internally. While some components do transfer data from one module to another (e.g., front end reading from category or posts tables), this is not an EIF because it has to be stored by an external application.

External Input:

- The user makes input through a form that allows them to publish a new post. This includes selecting whether the post is an offer or request for assistance, selecting a category, completing a text description, optionally adding an image, and choosing a group where applicable. Upon submission, this information is validated and saved to the posts table.

External Output:

- All posts inserted by users are displayed on the main page in a dynamic layout. Each post includes a formatted timestamp, text, category label, image (if any), and corresponding group tags. This information is fetched from the posts table and displayed graphically through dynamic HTML or front-end components.

External Inquiry

- The user can look for or sift posts by category, group, or keyword from the interface. Such operations request data from the database and pull suitable post data, which are displayed to the user without modifying the content. There is no calculation, derivation, or updating, only retrieval and presentation of data stored.

T1 – Internal Logical File

- Effort: **1 person-day** *(reduced since tables are mostly complete)*
- Duration: **1 day**
- Dependencies: None

T2 – External Input (form UI + backend endpoint)

- Effort: **1.5 person-days** *(partial logic already exists)*
- Duration: **1.5 days**
- Dependencies: T1

T3 – External Output (display feed)

- Effort: **2 person-days (look at with another member to see any overlap with their section)**
- Duration: **2 days**
- Dependencies: T1, T2

T4 – External Inquiry (search/filter UI + route)

- Effort: **1.5 person-days (same person covers their updated routes and meaning with me)**
- Duration: **1.5 days**
- Dependencies: T1

T5 – Basic Integration Testing

- Effort: **2 person-days**
- Duration: **2 days**
- Dependencies: T2, T3, T4

This is a 12-day span, which is good because it provides me with an additional 2 days before the group's agreed cut off of integration to look over any errors that we may encounter for the implementation and set up. It lso allows for time for some implementation where I need to reference another users work to see how I go about implementing throught there ideology and code.