Agile:

As a vanilla git power-user that has never seen GiggleBit before, I want a simple interface, so that I can do work efficiently right from the beginning.

As a team lead onboarding an experienced GiggleGit user, I want to be able to easily assign permissions, so that the experienced user can start work as soon as possible.

New user story:
As a new GiggleGit user, I want a diagram that explains how merges are managed by memes, so that I can understand how GiggleGit merges work.

- Task: Draw a diagram that shows how memes manage merges and implement it into the interface
    - Ticket 1
        - Title: Design the diagram
        - Description: Draw a diagram showing how merges are managed by memes that can easily be understood to a new GiggleGit user
    - Ticket 2
        - Title: Implement the diagram into the user interface
        - Description: Implement the diagram explaining how memes manage merges into the user interface in a way where it is easy to access

“As a user I want to be able to authenticate on a new machine”

- This is not a user story because it lacks describing why the user wants this feature.
- This is a functional requirement.

Formal Requirements

Goal: Launch user studies to see how difficult it is to understand what “syncing with a snicker” means.

Non-goal: The user study will not receive feedback about the base GiggleGit packages.

Requirements:

Non-functional requirement 1: Ensure that only authorized users can modify anything within SnickerSync.

Functional requirements:

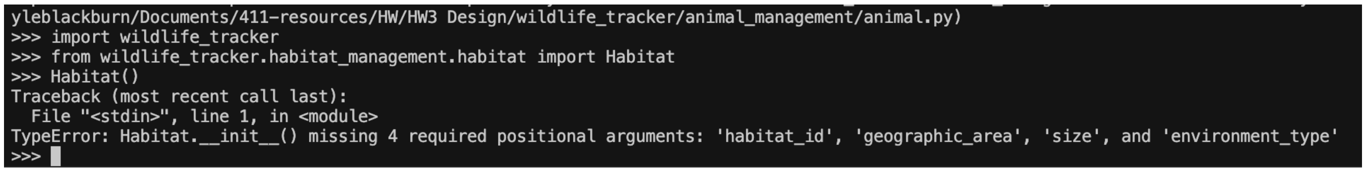
- 1: Implement a role-based access system where PMs and authorized users can access and modify SnickerSync settings while non-authorized users cannot
- 2: Implement logging so that any changes to SnickerSync are recorded

Non-functional requirement 2: Ensure that SnickerSync supports the random assignment of users into both control and experimental groups for testing

Functional requirements:

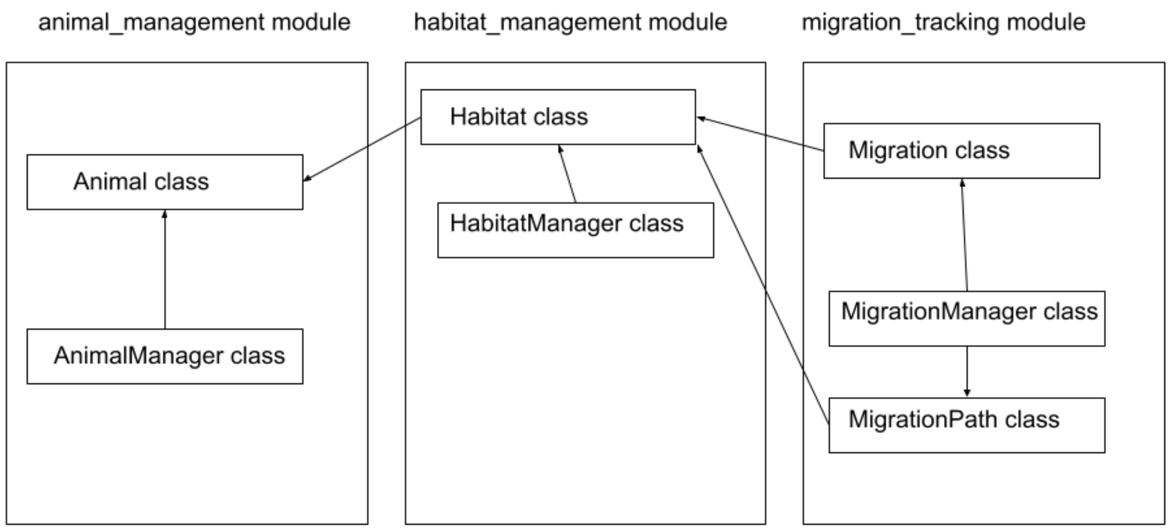
- 1: Build-in a mechanism to automatically assign users at creation to either the control or experimental group
- 2: Create a mechanism that tracks and records performance for users in both the control and experimental group

Importable Module Proof:



```
yleblackburn/Documents/411-resources/HW/HW3 Design/wildlife_tracker/animal_management/animal.py)
>>> import wildlife_tracker
>>> from wildlife_tracker.habitat_management.habitat import Habitat
>>> Habitat()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Habitat.__init__() missing 4 required positional arguments: 'habitat_id', 'geographic_area', 'size', and 'environment_type'
>>> 
```

Dependencies Diagram:


animal_management module
habitat_management module
migration_tracking module
Animal class
AnimalManager class
Habitat class
HabitatManager class
Migration class
MigrationManager class
MigrationPath class