We used OUnit and manual tests to test the functionality of our game. The basic functionality of the game was tested through glass box OUnit2 testing, however, the more complicated functionality of the game was tested manually.

Plant.ml

Most of the functions from Plant.ml were tested through glass box Ounit testing for each type of plant. For example, feed, water, and neglect functions were tested for all the plant types since all the plants have different attributes and were created differently. However, the random tragedies in Plant.ml were tested manually through the game because they take in randomly generated values. Those functions would be hard to test through OUnit testing because of it. While playing the game, we were able to manually test these types of functions.

Garden.ml

For Garden.ml, we used OUnit testing to see if the feed, water, and neglect functions from Plant.ml would get applied in the proper way. In addition, we also tested whether a plant would get removed from the garden after calling the remove_plant function. The rest of the functions included some sort of visualization in the terminal, like print_garden. So, we mainly tested those functions manually. We went through all the menu options and tried all the possible options in order to see if the visualization parts would work. In addition, other functions like dragon and unicorn that included the use of a random number generator were tested through manual testing too.

Inventory.ml

For Inventory.ml, we used OUnit testing to check that plants would not be added to the inventory since they should be added directly to the garden instead. Additionally, we used OUnit testing to ensure that non-plant items (e.g., beef) would be added to the inventory after being purchased. We also tested that adding a ladybug to the garden would increment its count in the inventory. I manually tested that item counts in the inventory were properly initialized and incremented.

Store.ml

For Store.ml, we used OUnit testing to determine that buying plants incremented the plant_count attribute of the garden and that buying a ladybug would add it to the inventory. We manually tested that the store randomly applied discounts to items and that buying from the store would subtract money from the user as expected.

Recipe.ml

For Recipe.ml, all the functions (have_ingredients, missing_ingredients, update_inventory, create_recipe, and sell_recipe) were tested through glass box OUnit testing. To do this, we created our own recipes that aren't part of the menu, and tested it. Additionally, we tested manually to see if after creating a recipe, it would add it to the inventory and subtract the ingredients from the inventory too. We also tested the sell function in a similar way, where we checked to see if the money in the garden would increase by the recipe sale price after selling it.