From our Wiki:

Sugar is a learning platform for children. It teaches kids how to interact and work a computer on their own. It has a bunch of interactive activities and puzzles. It's internet independent and runs on most computers running Linux, also can run on a Raspberry Pi. It has a simple UI and very easy to use. Sugar is developed with Python. Using sugar labs would be a good learning experience by being able to learn how to create simple games with Python.

Sugar Labs is the open source software which supports Sugar. Sugar Labs was worked on in the Google Summer of Code from 2016 to 2018. It is a member project of the Software Freedom Conservancy. According to Sugar Labs, Sugar is a collection of tools which they encourage the users to appropriate and create new activities to contribute to the network of Sugar users. It allows for children to write, read, or make music, classroom-learning style activities, automatic data backups in a “Sugar Journal,” and for easy open source activity sharing. Sugar aims to be friendly to teachers and children through a simple GUI and intuitive means of creating and experiencing content on their platform. They strive to create labs/learning environments around the world in regional ways by adapting Sugar activities to local curricula and language.

To begin working with Sugar Labs, we had to load an image of the live build into a virtual machine. We had some minor snags in the installation, which made us optimistic about the tasks to come. Surprisingly, learning to navigate our way sugar labs proved to be more difficult than expected. Sugar Labs has its own terminal as well as a bunch of development tools. We found the built in test cases, but we were not sure that we were able to run them properly. So we got in touch with a GitHub contributor, asking him about running testing. He was under the impression that we intended to contribute to the source, and also suggested that we did not have the resources available to learn the required things needed for running the Sugar Toolkit unit tests. He also gave us very relevant and important information on how many lines of code each of the Sugar components contained.

However, it did allow us to infer how we would be interacting with this software, because we are going to need to choose a component to run testing on. The options are Sugar Toolkit (Developer Tools), Sugar (Actual software), Datastore (the automatic backups), Core Sugar Activities (pre-made activities that run on Sugar), and the library of public contributors to Sugar (a lot of lines of code). We could probably run a series of unit tests on one of the Sugar activities alone, however, whether or not this would fit within the scope of the project is up for discussion. The github contributor we got into contact with suggested looking into the Datastore component, probably because it contains the least amount of lines. The tests that we found in the installation are for Sugar itself, and supposedly should be able to be run by anyone who uses Sugar for testing purposes. Although, it seems rather strange that they would not contain any logging or output when the test case runs properly, as we found through our experimentation.

Eventually, we were able to run the tests and identify the testing procedure in Sugar Labs. Sugar Labs has a number of pre-existing test cases that are prebuilt into the installation. These test cases are designed in order to test major functions of the Sugar Labs software.

Upon a successful test, the test case exits normally and often nothing is output to the terminal. Some test cases, such as test_mime.py, which output the success or failure of a test along with the time it took to complete. Accessing the test cases is not difficult as each module's prebuilt tests are stored within the /tests/ directory within said module's directory. Within the test directories, occasionally subdirectories will exist with supplemental files needed for testing. One test, test_backup.py, tests where the backup exists and is functioning properly. Another test, test_user_profile attempts to create multiple, varied user profiles. If all tests provide acceptable results, then the user can imply that Sugar Labs is operating normally.

After running that testing, we attempted to get Sugarizer running. Sugarizer is an application-based GUI that runs Sugar. We got Sugarizer running on Google Cloud Platform and ran some tests. We launched a GCP cloud instance, started up a web server, and ran Sugarizer from there as a web application. We then ran the test/index.html and tested the mochajs unit test on the datastore functions. This was a much easier environment for us to test in due to the fact that it uses nodejs. It also allows us to see passes and failures in more detail. Ideally, this is the Sugar environment that we would be able to conduct our own testing in.