**Test Plans for Gardening Application**

There are two parts of the project: Garden App and Garden Engine.

Garden Engine manages the plants within its virtual garden. It takes commands like edit plant and move plant. It changes the state of the garden accordingly. It also takes commands like get plant, and returns certain properties of the requested plant. Garden App takes inputs from user and interacts with the Garden Engine, then shows the outputs from Garden Engine to the user. It also interacts with the Android platform for storage, notifications, etc.

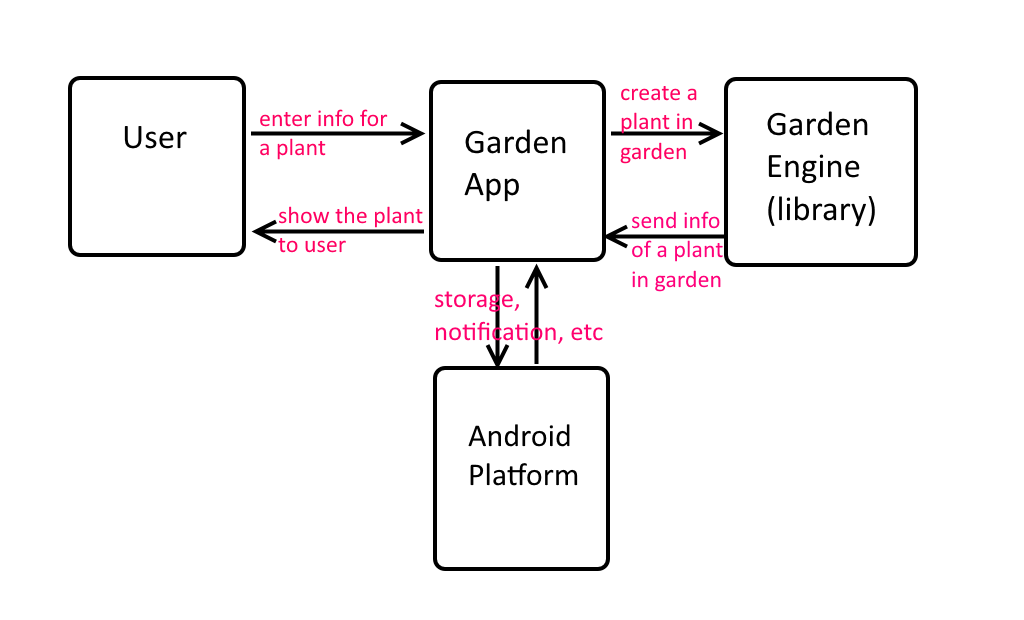


Figure A

Tests will be done using JUnit and Espresso, as well as manual testing on an actual Android device and code inspections.

There are two sets of tests: Garden App tests and Garden Engine tests.

Engine tests focus on the Garden Engine. They test whenever a command is sent to the engine, the engine should take the proper actions. UI tests focus on the App user interface. The app should also interact with engine and android platform correctly.

Code inspections will be done before a merge. All functions should be well documented using the Javadoc format. All classes within these two projects should only implement functions within their scope. This will ensure good encapsulations. This will help us keep up the scalability of the project.

# Unit Tests for Garden Engine

There should be a test for almost every single method in the classes using JUnit. Edge cases will also be tested. This will ensure the reliability of the engine, so that the app can assume the library works, making it easier to debug the app.

# Acceptance Tests for Garden Engine

1. **Engine received a “get all plants” command – Engine should return a list of plant objects within the garden.**
2. **Engine received a “get plant” command, with a plant identifier as parameter – Engine should return the specific plant within the garden (null if plant not found).**
3. **Engine received a “create plant” command, with some properties of the plant to create – Engine should create such plant in its garden.**
4. **Engine received an “edit plant” command, with some properties of the plant to edit, along with a plant identifier as parameters – Engine should edit the properties of the specified plant in its garden.**

To test “create plant” and “get all plants”, we can first call “create plant” multiple times to add some arbitrary plants into the garden, then call “get all plants” to see if the plants returned have the same properties as the ones that were entered.

To test “edit plant” and “get plant”, we first call “edit plant” to edit all the properties of a specific plant. Then we call “get plant” to get the plant we just modified. We check the properties of the plant and see if they have been modified.

# Unit Tests for Garden App

There should be a test for almost every single method in the classes using JUnit. Edge cases will also be tested.

# Acceptance Tests for Garden App

The engine must pass the test first in order to test the garden app. Espresso will be used to test all the functionalities of UI.

1. **User tapped “add plant” button – App should display a screen to allow user to add a plant.**
2. **User tapped “edit plant” button – App should display a screen to allow user to edit properties of the specified plant.**
3. **User tapped “save plant” button in edit plant screen – App should save the plant properties to storage.**
4. **User launched the app – App should read the garden and its plants from storage and reconstruct the virtual garden.**
5. **User tapped “show garden” button – App should display a visual representation of the garden, with its plants inside.**

These tests can be run in order. Espresso will be used to test every single requirement. Also some manual testing will be done on an actual Android device, to ensure the performance of the app.