**CS673 Software Engineering** 

**Team 1 - Med Tracker**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Andrew Gieraltowski | Team leader | *AJG* | 2022/9/11 |
| Yuan Wang | Requirement Leader | *Yuan Wang* | 2022/9/11 |
| Yuan Wang | Design and Implementation Leader | *Yuan Wang* | 2022/9/11 |
| Haiyang Lu | QA leader | Haiyang Lu | 2022/9/11 |
| Andrew Gieraltowski | Configuration Leader | *AJG* | 2022/9/11 |
| Haiyang Lu | Security Leader | Haiyang Lu | 2022/9/11 |
| [Divya Thomas](mailto:divthomas22@gmail.com) | Design and Implementation Leader | *Divya Thomas* | 2022/9/11 |
| [Divya Thomas](mailto:divthomas22@gmail.com) | QA Leader | *Divya Thomas* | 2022/9/11 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **0.1** | **Andrew Gieraltowski** | **9/22/22** | **Initial Submission** |
| **0.2** | **Andrew Gieraltowski** | **10/10/22** | **Updated tests** |
| **0.3** | **Haiyang Lu** | **10/11/22** | **Updated tests** |

[Testing Summary](#_sm5odwyvuk3j)

[Manual Tests Reports](#_pqso2mbjyzx4)

[Automated Testing Reports](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Testing Summary

**Unit Testing**

For this project, we have chosen Junit as our unit test framework. Our goal is 100% code coverage for all included modules excluding the included libraries. Each module will have a paired java file called [module]Test.java that links the original module implementation as well as the Junit framework. There is currently no plan to integrate CI/CD into our github repository so the unit tests will need to be run and verified locally on developers machines.

**Activity Testing**

Activity tests will be implemented for all the existing interfaces. For every function and activity within the interface, there will be a test that is called to check if that part is launching successfully for each time the application is operated.

**Integration testing**

Integration tests will be designed and documented on a per task basis. Each task created for the sprint should include the details of how the change should be tested using the application.

**System Testing**

System tests will be performed on newly created features to ensure that the full system is operating as expected with the addition of a new feature. The tests will consist of functional testing aimed at the new and surrounding features.

**Acceptance Testing**

New features will need to be evaluated against the requirements defined for that feature. So after system testing, the new features will need to be tested to make sure that they align with the requirements defined for that feature. The features will also need to be tested to ensure they meet the performance requirements acceptable for that type of feature, (i.e. a database call can’t take more than 1 second to load).

**Regression Testing**

Regression tests will only be performed after a release has been made. The team will verify that previously created functionality of the application not covered by current system tests are operational before submission and after the release binary is built. Hotfixes may need to occur if an issue is found during regression testing. In that event, a new release will be created and the entire regression test suite will need to be run again.

# Manual Testing Report

## User Select Test

* New
* Android Phone/Emulator
* High
* Dependencies: None
* Preconditions: App installed
* Input Data: User Information
* Test steps:
  + Open the android application on phone/emulator
  + Attempt to select patient as account type
    - Verify that the patient UI is shown
  + Press the button to switch to caretaker mode
    - Verify that the caretaker UI is shown
* Postconditions: App does not crash
* Expected output: User should be able to transition between account types and the correct user interface should be displayed depending on the account type
* Actual Output:
* Pass/Fail:
* Bug ID/link:
* Additional Notes

## Add Medication Test

* New
* Android Phone/Emulator
* High
* Dependencies: None
* Preconditions: App installed
* Input Data: User Information
* Test steps:
  + Navigate to the add medication tab in the application
    - Verify that the add medication tab is viewable
  + Enter in the following information into the name of the medication
    - -1
    - Abc
    - 00123
    - Leave it empty
  + Verify that the app is able to handle to each input accordingly
  + Enter in a valid date on the calendar popout
  + Enter a valid time on the time selector popout
  + Enter in a valid dosage (float)
  + Press the add medication button
    - Verify that the success screen is displayed
* Postconditions: The database is updated with the medication information and the success prompt is displayed
* Expected output: Database should be updated with the medication information and the medication should be displayed to the user on the application
* Actual Output:
* Pass/Fail:
* Bug ID/link:
* Additional Notes

## Remove medication test

* New
* Android Phone/Emulator
* High
* Dependencies: None
* Preconditions: App installed, medications already added
* Input Data: Medication information (name)
* Test steps:
  + On the medication tab, select a medication
  + Select the delete button
  + Verify that the medication no longer exists in the database
  + Verify that the medication no longer displays
* Postconditions: Patient’s calendar is updated and displays correct information
* Expected output: User should be able to delete medication on the patients calendar
* Actual Output:
* Pass/Fail:
* Bug ID/link:

## Main Page Activity test

* New
* Android Phone/Emulator
* High
* Dependencies: None
* Preconditions: App installed
* Input Data:
* Test steps:
  + Navigate to the testLaunch class
  + Right click to open the menu
  + In the menu choose the run ‘testLaunch()’
  + Check if the application’s main page is successfully open
* Postconditions: The UI for the main page is set up already.
* Expected output: User should be able to interact with every function on the main page
* Actual Output:
* Pass/Fail:
* Bug ID/link:

(Consider using additional spreadsheet for more test cases)

# 

# 

# 

# 

# 

# 

# 

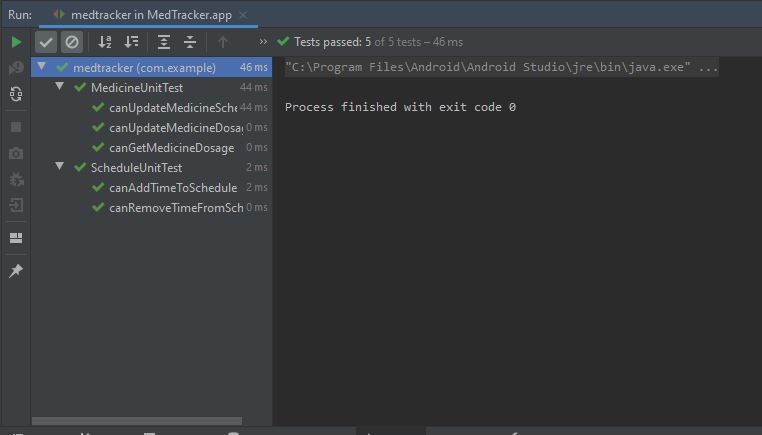
# 

# 

# 

# Automated Testing Report

Unit test code resides under the ***/src/test/java/com/example/medtracker*** folder. The unit tests leverage Junit for all modules and are only available to run locally on developer machines. No CI/CD is set up at this time.



**Figure 1:** Android studio unit test successful completion

# Testing Metrics

This project will be evaluated on unit test coverage. The goal is to have at least 80 percent unit tests covered.

# References

# Glossary