**Team 4 - Milestone 4: Essential Test Plan**

**A. Goals and Exit Criteria (Nick) (Complete)**

i. In order for us to meet our goals to be out of the test phase, we expect our product to do the following: First, it should accurately display the number of people in the store. For the sake of our prototype, this will be simulated. Should a change be made to the store’s capacity, it should also be able to update that in real time. Finally, the pickup-from-store feature will contain a basic search feature to see if items are in stock and use a linked list to create a basket of goods tied to the users account for pick-up later in the day.

ii. Regarding robustness and our products ability to complete the tasks at hand, I believe that our targets for robustness are in line with our testing parameters as successful completion of these elements in the test phase will coincide with our product creating robust or unique value.

iii. Our complete time table can be found on our GitHub repository, and it is in line with the deadlines for the class as well.

iv. Regarding performance, our product should be able to update in nearly real time with individuals entering the store via some counter mechanism that will be simulated in our code to represent people entering and exiting the store, akin to concert venues or other limited capacity locations such as nightclubs or political rallies that take count of every entry and exit.

**B. Items to Be Tested/Inspected (Nick) (Finished)**

i. The primary executable component that will require rigorous testing and inspection will be the Java code that thus far has been only prototyped, checking that the following works

1. The Program accepts credentials and accurately redirects the user to either the customer mode or manager mode
2. The program can be executed on a mobile device
3. When an update to the Covid rules are made on the manager end, reflections occur nearly instantly on the customer side.

ii. Since our program is mainly the executable, we have little to no requirements outside of the executable beyond its ability to work on other devices.Other non-code related items, such as grammar & ease of use will also be considered.

**C. Test Process/ Methodologies (Santhoshini)**

i. Unit test/Functional test/Acceptance test/Regression test/ and so on, methodologies

ii. Inspections/reviews methodologies

Different steps involved in inspection are

Overview : A overview is given

Preparation : Information is gathered for inspection

Planning : Planning starts when inspection is started

Meeting : All the issues are collected

Follow up : A follow up is done on the bug fixes

rework : All the issues are fixed

iii. Black-box testing (e.g., Input domain test, boundary value testing)

1.Equivalence Partitioning

Customers of age above 15 are allowed to sign up to the app. This is verified while entering the date of birth during the signup.

2.Boundary Value Analysis

Check if there are more than 30 customers in the store . If there are more than 30 customers then application should display a message stating no in person pick up right now ,please try after some time

iv. White-box testing (e.g., control path testing, data flow testing)

1.Branch Testing

Consider small code

INPUT A

IF A > = 30

PRINT “ Please Visit later”

ELSE

PRINT “Please visit the store”

Pass different values for A to check if the conditions are getting satisfied

2.Data Flow analysis Testing

Consider small code

1 INPUT A

2 IF A > = 30

3 PRINT “ Please Visit later”

4 ELSE

5 PRINT “Please visit the store”

if a is greater than A then order of execution should be 1,2,3 otherwise 1,2,4,5

v. Test metrics (e.g., code coverage, branch coverage, number of problem by severity)

Total number of test cases

Number of passed test cases

Number of failed test cases

vi. Test-bug report-fix-retest process

create a bug report

Maintain the status of the bugs

Share the details with the developer and once the bugs are fixed retest

**D. Major Test Scenarios and Test Cases**

i. Boundary value and input domain test cases

ii. Control path and dataflow test cases

iii. Integration and intermodular test cases

Major Test Scenarios:

1. Check the Login Functionality
2. Check the Product Description Page
3. Check the Payments Functionality
4. If the app comes with a users’ settings features, check if the app changes when some form of change is affected by the user.
5. Check that the app is still working as intended after the successful update of the app.

Component testing:

When a customer types into the search bar, the software will populate search suggestions, dynamically generate filters once a user selects a search suggestion, display the search results as a list, show item details when a user clicks on a link from the list, allow a user to add an item to their cart, display the cart when a user wants to view the cart, and carry out a transaction when a user clicks on a link from the list. Other user groups, such as administrators and suppliers, will have functionality handled by other components. Those that allow an administrator to produce a report, pull a report, remove a report, and edit a report are examples of components that might be utilized by administrators. Vendors will also be able to add new goods and be warned when inventory levels are low. Component-level testing will guarantee that the code is manageable, effective, resilient, performant, and free of flaws.

**E. Resources**

i. People (number of, skills, etc.)

ii. Tools (for measurement, defect management, etc.)

iii. Systems (test execution platform, test case development, etc.)

**F. Schedule - (Bryan) -** [**Please see Team 4 Project Plan**](https://docs.google.com/spreadsheets/d/1mGG9qHcyyTj4lnpE-PKUmnkchwIebzpX/edit#gid=1789924259)

**G**. **Risks - (Bryan)**

* **Security**: Our program does not have high severity security breach risks, however we do support and deal with public health. It is somewhat complicated because there are no government rules against actual store capacity. However if there was we would need to make sure to be compliant in the following ways:
  + **Secure user login.**
    - **Mitigation:** 2FA is a common day good mitigation for secure login.
  + **Secure data storage and management**
    - **Mitigation**: Our plan is to use Amazon S3 buckets for data storage. Part of that service that Amazon provides includes safety and redundancy measures to protect our data.
  + **High level of confidence that when a capacity at the store is shown to be safe to the user that it is actually safe!**
    - **Mitigation:** Test For Edge Cases: For this we would need to test all the edge cases and do extensive bug testing.

For all of our security Risks a post mortem software update plan needs to be put in place and maintained. This is because the paths and methods in which our program could become breached are constantly changing, and also the OS are being updated which could open up new attack windows for bad actors.

* **Risks**
  + **Low** **Risks**:
    - **Program Planning / Health**: Our program has been tracking very nicely to plan. System architecture is established and we are refining the class structure and user flows.
  + **High Risks**
    - **Coding**: Because our team is small and with limited knowledge on mobile systems code, our code base will take some time to complete. (It is unlikely we will be able to complete the program to the scale at which we have designed it). More resources needed.
      * We have until end of semester for coding and our original plan was to start this around milestone 4 timing but really pick it up after the personal paper presentations. We are likely to stay on track to that.
    - **Testing**: Due to our limited resources and being required to plan, code and test: It is unlikely we will be able to establish automated UI testing to run our code against. We will be able to run basic UI testing to confirm the flows work as intended.

**H. Progress information and changes from Milestone 3 - (Bryan)**

* **Review of progress since the previous milestone.** 
  + Understanding more of the complexities of mobile code bases.
  + System consideration from a testing perspective and Risk.
  + Refinement of use cases.
  + Refined program scope / tasks to align with the newest learned information.
  + Made a team plan for how and when to work on the code.
  + Updated program plan.
* **Revised schedule and plan for the remainder of the project.** 
  + (Complete and committed to github)