# Restaurant Management Application

*Project Testing and Acceptance Plan*

Washington State Unviersity

**Team One**

Amrit Banga

Tucker Brisbois

Zijian Zhang

[07/05/2024]

**Note**: Recall that this writing assignment says:

Length = 3  pages text (or more because space goes fast once you start using lists and forms for user feedback) + appendixes as needed.

Not included in 3 page count:

- Cover page
- table of contents
- pictures
- tables
- images
- diagrams

Posted as a single self-contained file (no links to outside resources.)

Posted as a PDF file.

Typed single-spaced.

Typed with black text.

Typed with #11 font size.

Typed using Arial font.

Typed with one inch margins on sides, top and bottom.

**Please erase this page in your final document.**

# TABLE OF CONTENTS

# I. Introduction

## I.1. Project Overview

Provide a brief summary of your software that is being tested. Outline all major functionality for which testing is crucial.

The software being tested will be the fully functional Restaurant Management Application. This version of the application will be fully functional minus UI finalization. The functions that will be tested will be the create account, log in system, 2 factor authentication, add business system, inventory management system.

## I.2. Test Objectives and Schedule

Describe, at a high level:

- the approach (the test plan objectives),
- required resources, and
- the schedule of the testing activities
    - major work activities
    - products to be delivered - code, containers, documentation, virtual machines, etc.
    - major milestones.

In order to test these functions, a fully functional build of the application will be required. The testing will be done on the virtual iPhone tester which is provided by XCode when the application is running. Most of the testing will be pass or fail tests, with screenshots providing the evidence. For instance, for the create an account system, the test will enter in new email and password, and if the account is created and pops up in Firebase, then it is a pass. The testing will commence from the beginning of the application flow.

## I.3. Scope

Describe the scope and purpose of this document in a short paragraph.

The purpose of this document is to outline the plans and steps for testing the Restaurant Management Application.

# II. Testing Strategy

Describe the overall approach to testing and provide the overall flow of the testing process. An example is provided in Appendix A.

Will you be using Continuous Integration (CI) and/or Continuous Delivery (CD) in your testing? If you're not using CI or CD, make a *very* strong case for your decision.

For testing any particular function, the required tests will identified. The required outcome for any given test will also be noted and the compared against what happens during the test. If the outcomes match, then the test will pass. If they differ, then the test will fail. Screenshots will be taken before, during, and after. This process will be repeated with each function.

The approach taken will be continuous delivery (with the delivery location being the main branch of the github repo).

# III. Test Plans

Describe the plan for testing your project in the context of the following testing activities. You may include additional test activities, if necessary.

For each of the following activity, describe how the testing will be conducted. What would be the sequence of events, and how will the testing activity take place? Please refer to the CptS422 class notes for details on testing strategies.

## III.1. Unit Testing

The primary goal of unit testing is to take the smallest unit of testable software in the application, isolate it from the remainder of the code, and test it for bugs and unexpected behavior.

For unit testing, we will be testing each specific function of our application by themselves, not paying attention to how it interacts with other objects. This is to ensure the primary, basic functions work before trying to integrate them with others.

## III.2. Integration Testing

Integration testing detects faults that have not been detected during unit testing by focusing on small groups of components. Two or more components are integrated and tested, and when no new faults are revealed, additional components are added to the group.

The majority of our integration testing will see if the functions mentioned above integrate with our database to make sure that the information is saved and can be reused by the user at a later time.

## III.3. System Testing

System testing is a type of black box testing that tests all the components together, seen as a single system to identify faults with respect to the scenarios from the overall requirements specifications. Entire system is tested as per the requirements.
During system testing, several activities are performed:

### III.3.1. Functional testing:

Test of functional requirements (from requirements specification). The goal is to select those tests that are relevant to the user and have a high probability of uncovering a failure.

### III.3.2. Performance testing:

Performance tests check whether the nonfunctional requirements and additional design goals from the design document are satisfied. In stress testing, system is stressed beyond its specifications to check how and when it fails.

### III.3.3. User Acceptance Testing:

Acceptance testing and installation testing check the system against the project agreement. The purpose is to confirm that the system is ready for operational use. During acceptance test, end-users (customers) of the system compare the system to its initial requirements (if necessary) with help by the developers.

# Testing Plans:

Tested Aspect: Add Item

Expected Result: When add item button is clicked, a page will pop up. Once the page is filled out, the user can click “add” and the new item will appear at the bottom of the chart.
Observed Result:
Test Result:
Test Case Requirements: None

Tested Aspect: Remove Item
Expected Result: When the red ‘X’ on an item is clicked, the entire row item should disappear
Observed Result:
Test Result:
Test Case Requirements: An item must already exist in the inventory page

Tested Aspect: Edit Item
Expected Result: When the blue pencil icon in an items row is clicked, an edit page will pop up. The user should be able to edit any of the information on the page and when the user clicks “Save”, the new, updated information will appear in the row.
Observed Result:
Test Result:
Test Case Requirements: An item must already exist in the inventory page

## IV. Environment Requirements

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Identify special test tools needed.

No special tools will be needed in order to perform the tests. In XCode, when a new build is generated, a virtual iPhone with the application downloaded will be provided to the developer. This will be the location for the testing and will provide the developer a visual landscape to see. All the hardware that is currently in use (computers) will all that will be needed.

## V. Glossary

Define technical terms used in the document.

## VI. References

Cite your references here. Please use one style for the references. You’re welcome to choose between: IEEE and Chicago style formats. I highly recommend using scholar.google.com to help with the formatting. Seriously, scholar.google.com is an incredibly powerful tool to both find citations, and to generate well formatted citations for papers/materials you’ve already found.

For the papers you cite give the authors, the title of the article, the journal name, journal volume number, date of publication and inclusive page numbers. Giving only the URL for the journal is not appropriate.

For the websites, give the title, author (if applicable) and the website URL. Here’s a format for it:

- http://www.easybib.com/reference/guide/apa/website

# Appendix-A

# Example Testing Strategy:

1. Identify the requirements to be tested. All test cases shall be derived using the current Software Requirements Specification.

2. Identify which particular test(s) will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the revised Test Plan document.

8. Successful unit testing is required before the unit is eligible for component integration/system testing.

9. Unsuccessful testing requires a bug form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.