Ain Shams University

Fuculty of Engineering - CHEP
CESS Program

Spring 2020

Software Testing, Validation, and Verification (CSE 225)

Final Assessment Research Project

Pizzaro Restaurant
Automation System Test Document

Name: Ahmed Abd ElNasser

ID: 17P8113

Group: 2

TABLE OF CONTENT

1. INDEPENDENT TESTABLE FEATURES

Since the functional test is known to be a requirements-based test that uses the final requirements as the basis for testing. We shall first, carefully examine the user requirements to set up some ITFs.

To make the analysis of features more accessible, the requirements can be categorized or better say classified based on the user type and each user's role in the business process as follows:

I. All Users' Features

ITF01: Log-in and out.

ITF02: Great each user's log-in with his allowed privileges (role-personalized home screens). i.e. When a waiter logs-in he gets a floor status of his assigned tables. But when kitchen staff does, he gets notifications about placed orders. And so on.

ITF03*: Refresh each employee's role-personalized home screens when needed. (when a table is marked ready; a table's order is prepared; a host assigns a waiter to a table; etc.).

The SRS suggests exploiting the logging in and out operations to trigger these updates.

II. Waiter's Features

ITF04: Tables coloring according to the current tables' status. Note that no need here to mention the each-waiter's assigned tables issue as it is presented in (ITF03).

ITF05: Select a table to view it is tab and other options.

ITF06: Manipulate a tab to make orders (Add/Delete items).

III. Manager's Features

ITF07: Create and/or modify profiles.

ITF08: Track employees' activity, including waiters' tables and tabs.

ITF09: Authorize restricted waiter activities.

ITF10: Add and remove (arrange) tables to alter the floor plan.

(*) ITF03 includes the feature of orders to be sent to the kitchen staff on their terminals assuming that this screen update happens once the waiter logs out after placing the order (logging operations trigger updates). It also includes the tables status updates used by hosts and busboys.

2. FORMAL MODELS

Now, after we have decomposed the requirements, extracting the ITFs to design the test on. We move on to the next stage which shall be identifying and selecting representative values for test cases.

But, considering a large multi-functional system like this, the simple input/output transformations cannot adequately describe and represent all the requirements. So, in this section we shall construct supportive formal models for some ITFs as needed to modulate the specifications and describe the system behavior to help with the test design.

Since our test here is a black-box one because we only have the SRS, the following models are based on black-box testing models and techniques.

I. ITF01

Log in screen decision table with 4 combinations of inputs (4 rules) as shown in (Table 1)

Condition	Rule1	Rule2	Rule3	Rule4
Username	Wrong	Correct	Wrong	Correct
Password	Wrong	Wrong	Correct	Correct
Output	Error	Error	Error	Home screen

Table 1 ITF01 Decision Table

II. ITF02

Causes:

C1: Account is waiter

C2: Account is cook

C3: Account is host

C4: Account is busboy

C5: Account is manager

C6: Valid log in

Effects:

E1: Waiter home screen

E2: Cook home screen

E3: Host home screen

E4: Busboy home screen

E5: Manager home screen

E6: Invalid log in error message

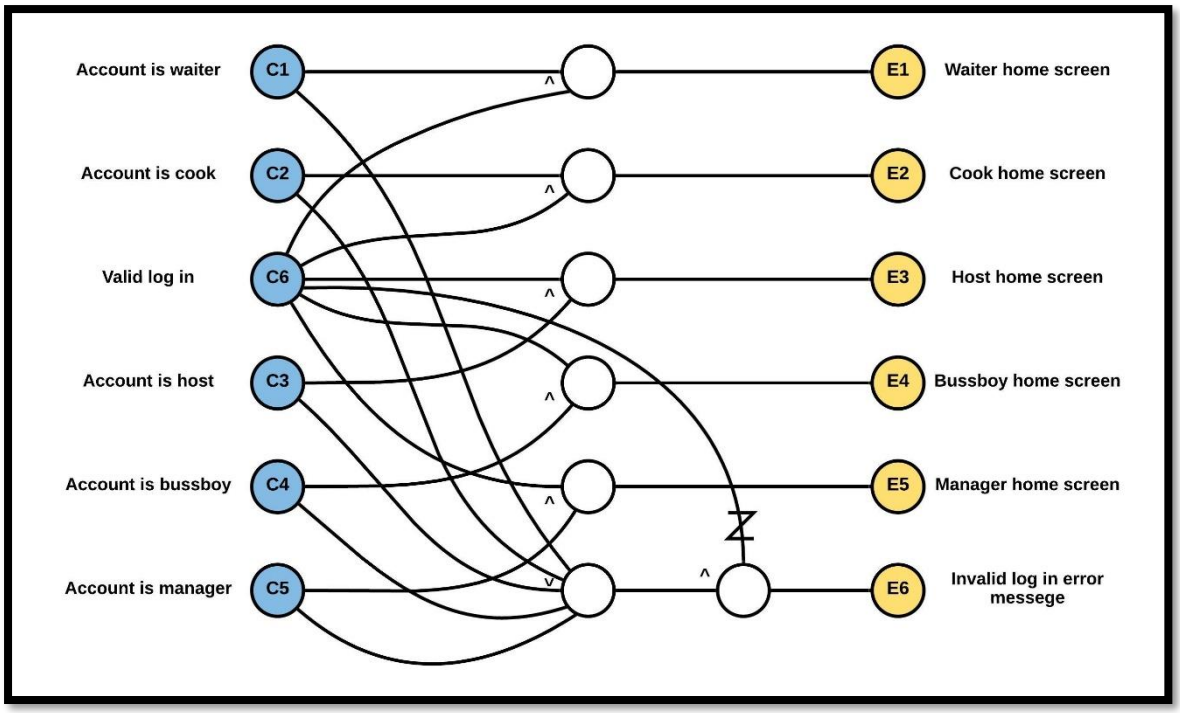

Figure 1 ITF02 Cause-Effect Grapg

#	1	2	3	4	5	6
C1	1	X	X	X	X	X
C2	X	1	X	X	X	X
C3	X	X	1	X	X	X
C4	X	X	X	1	X	X
C5	X	X	X	X	1	X
C6	1	1	1	1	1	0
E1	1					
E2		1				
E3			1			
E4				1		
E5					1	
E6						1

Table 2 ITF02 Decision Table

III. ITF03

To make it simple as it is effective, we will focus in this test on the direct links between some of the staff like: Host ↔ Waiter, and Waiter ↔ Cook. Each certain action these employees do triggers a certain change and updates to others home screen. Mentioning the trigger, we will use the log in and out operation for this task, as each employee is required to log-in, finish their task, and log out.

Note that the SRS assumed the cook and host are be logged-in all the time. So, they cannot be a trigger.

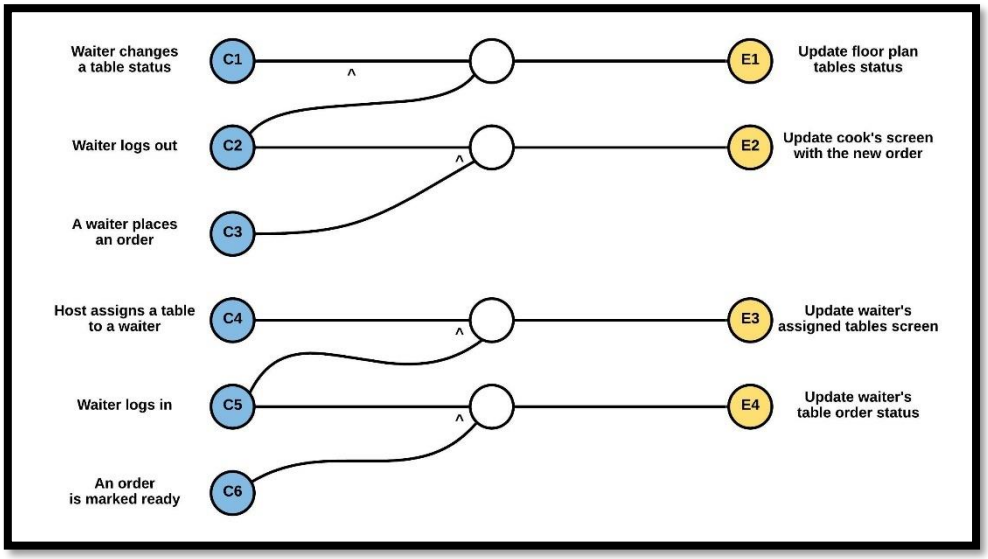

Figure 2 ITF03 Cause-Effect Graph

#	1	2	3	4
C1	1	X	X	X
C2	1	1	X	X
C3	X	1	X	X
C4	X	X	1	X
C5	X	X	1	1
C6	X	X	X	1
E1	1			
E2		1		
E3			1	
E4				1

Table 3 ITF03 Decision Table

IV. ITF04

This test is concerned with waiter's assigned table coloring according to their assigned status which should be one and only one of enumerated fixed values.

The SRS has defined the table states, but no available data about the possible transitions.

So, I will assume that:

- There is a variable (TS) associated with each table object, that represents its status from an enumerated type: Status {OPEN, OCCUPIED, DIRTY}
- The possible state transitions are only according to the FSM.

A Finite State Machine model would be typical for this test (Figure 3).

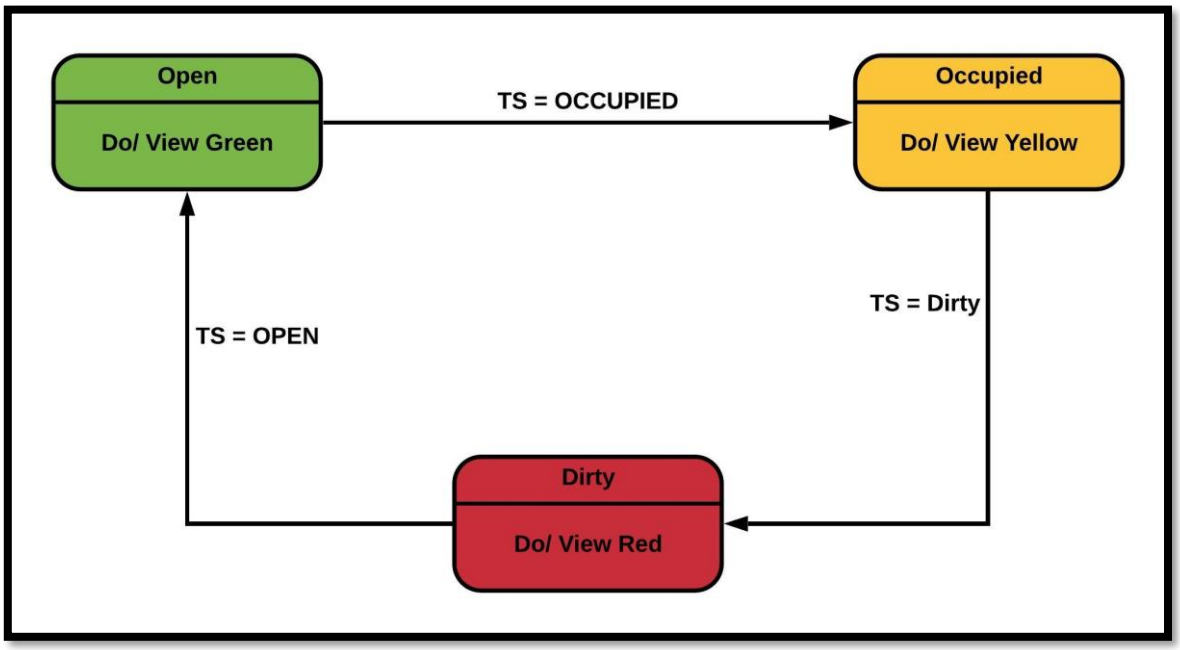

Figure 3 ITF04 State Diagram

V. ITF05

Waiter's privileges allow him only to view and manipulate the tabs of his assigned tables. He has no access to other tables.

We can easily test this using representative values for assigned and non-assigned tables.

VI. ITF06

Testing the order options. The following assuming are not stated clearly in the SRS but are needed for this test:

- Each item in the menu is associated with a quantity counter and a flag indicating either it is available for ordering or not.
- Among the waiter assigned tables in ITF05, he can make orders for occupied yellow tables only. This is a rational assumption.

A Cause-Effect Graph followed by a Decision table:

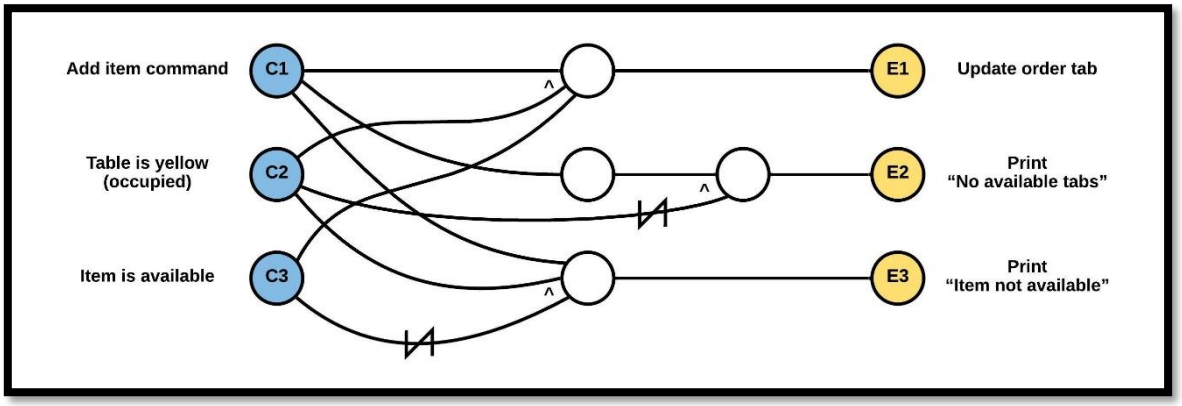

Figure 4 ITF06 Cause-Effect Graph

#	1	2	3
C1	1	1	1
C2	1	0	1
C3	1	X	0
E1	1		
E2		1	
E3			1

Table 4 ITF06 Decision Table

VII. ITFs related to the manager

Since the SRS provides poor description for these features. It will mostly rely on some assumptions and I may use the Ad Hoc Exploratory Testing method for error guessing.

i. ITF07

An ambiguous feature because of the following:

- The SRS did not provide what kind of modifications can be applied on the profiles.
- The data attributes associated with each profile is not defined.

I assume that:

- Each employee profile contains his username, ID, password, SSN, and email Address.
- It is like any other account usual modifications (Change username or password, change email address, change employee's job tittle, etc.)

Causes:

(1) Creating new profile.

(2) Using duplicated data.

(3) Modify account.

(4) Email validation.

Effects:

1) Successful creation.

3) Data duplication error.

2) Successful modification.

4) Data validation error.

#	1	2	3	4
C1	1	1	X	X
C2	0	1	X	X
C3	X	X	1	1
C4	X	X	1	0
E1	1			
E2		1		
E3			1	
E4				1

Table 5 ITF07 Decision Table

ii. ITF08

An ambiguous feature because:

- The domain of employees' activities is not fully clarified.
- The tracking method is not mentioned at all. Is it updated regularly? or is it like a daily report? or what? If we assume it is updated by a trigger using log in and out operations like ITF03. What about the host and the kitchen staff who are (according to the SRS) logged-in all the time.

So, I assume that:

- I have a possible visualization for this, that closely matches any other system "activity tracking". Since the manager has access to all accounts, so, each account will be associated with something like a **log file** that records all the operations that are executed by the employee owning that account.
- As for the waiters' activities, each table or tab (order) has a unique number.

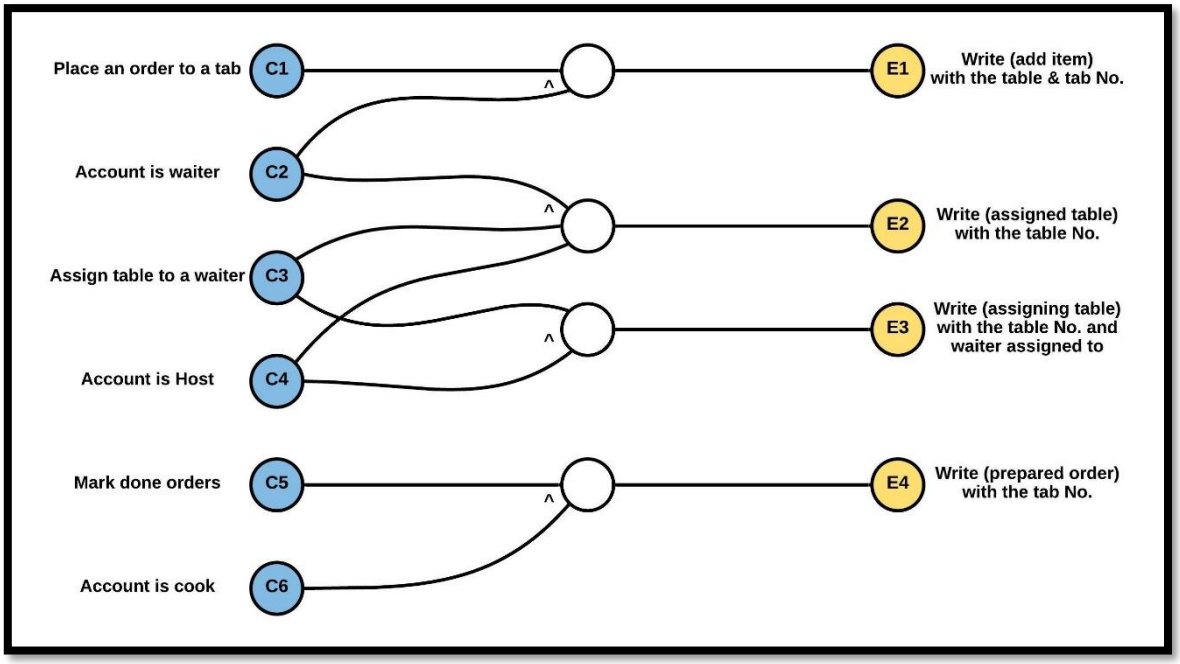

Figure 5 ITF08 Cause-Effect Graph

This test will be concerned with making sure that each operation type is recorded correctly in the right place where:

- **Causes:** mostly combinations of the operation and its executer.
- **Effects:** updates (writes) to the corresponding account log file.

#	1	2	3	4
C1	1	X	X	X
C2	1	1	X	X
C3	X	1	1	X
C4	X	1	1	X
C5	X	X	X	1
C6	X	X	X	1
E1	1			
E2		1		
E3			1	
E4				1

Table 6 ITF08 Decision Table

iii. ITF09

An ambiguous feature. Why? Because of the following:

- To test this feature, we need first to identify two things that are not clearly mentioned in the SRS: What are the exact restricted activities, and what are the possible authorizable one.
- Another important point, are those expanding activity features already designed and implemented by programmers? Or only the authorized part is implemented and when a manager wants to add new activities, he shall call the company to bring up their team.

In later case, and as a tester, I have nothing to do here.

But assuming the first case, such test is not concerned with the features themselves either or not they are working properly. But, with the "Authorization" issue. A waiter is privileged and have access only to the authorized features for him.

It is not a real plan, but I am left with no other option but to use intuition to set some inputs that might cause the program to fail:

- Using a waiter account, try to access a non-authorized activity.
- Try to restrict waiter features, then assert its non-accessibility.

iv. ITF10

Another ambiguous feature because:

- When does the effect of the floor plan changes take place? And how does it affect the home screens of the other employees?

I assume to:

- Deal with this one using the **log out operation trigger**. When the manager changes the floor plan organization or tables arrangement, He logs out. After that, the home screens that views the floor plan gets updated.
- The manager can only re-arrange free tables (GREEN).

A Cause-Effect graph would be typical here. Followed by a Decision table.

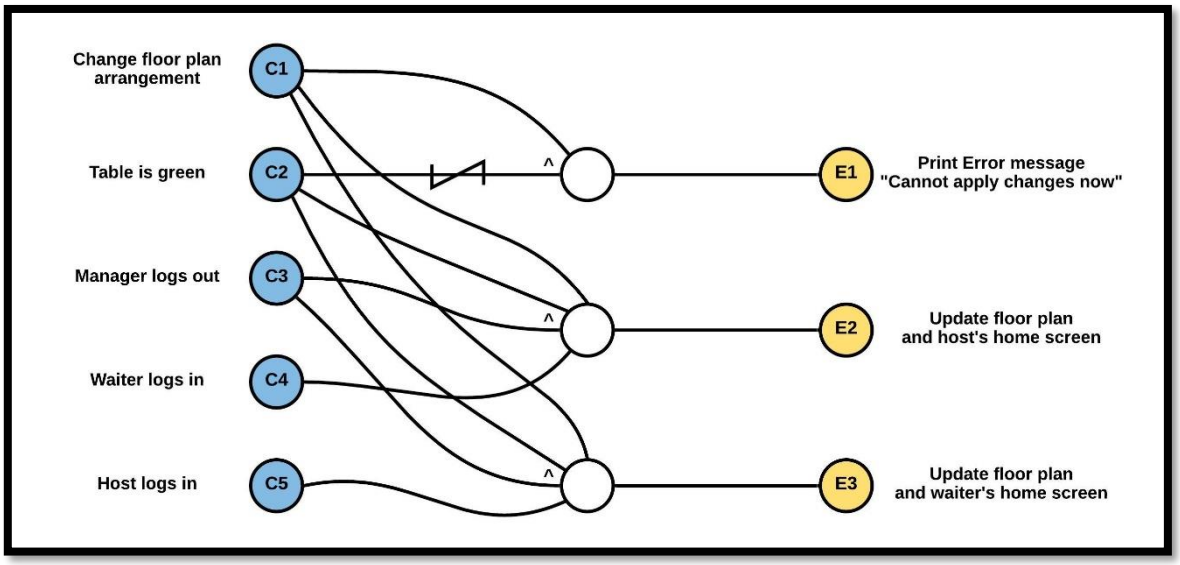

Figure 6 ITF10 Cause-Effect Graph

#	1	2	3
C1	1	1	1
C2	0	1	1
C3	X	1	1
C4	X	1	X
C5	X	X	1
E1	1		
E2		1	
E3			1

Table 7 ITF10 Decision Table

3. TEST SPECIFICATIONS

Before jumping to test specifications from which we are going later to generate the test cases, and following the test methodology, here's how it's going to work: first, we examine the ITFs one by one and with the guidance of the SRS, we obtain some clues (variables, operations, pre and post conditions) which will lead us to the test requirements, and from there we're going to set the test specifications. In all these steps, I will follow the formal format of specifications. I also will assure the variety when using the same variable in different tests.

I. ITF01

i. Inputs / Variables

(1) Username (String)

(2) Password (String)

ii. Validated pre-conditions

(P1.1) Username length > 0

(P1.2) Username does not contain numbers or special characters.

(P1.4) On failure, print "Invalid log in"

(P2.1) Password length (>= 4 characters) and (<= 8 characters)

(P2.2) password contains at least one digit and one special character.

(P2.3) On failure, print "Invalid log in".

iii. Assumed pre-conditions

(1.1) Username is a string

(2.1) Password is a string

iv. Clues / Post-conditions

```
if ((P1.1) and (P1.2)) then {
        if ((P2.1) and (P2.2)) then {
                The username and password formats are valid
                if match (username, password) then { return home screen }
                else { return "Log in error" }
        }
} else {
        The username format is not valid
        return "Invalid username format"
}
```

v. Test requirements

Note that pre-conditions on variables could have been combined like saying (Invalid username or password), but it is recommended to handle each variable's pre-conditions separately.

No.	Test Requirement	Covers
R01.1	Empty Username and password fields. Inv inputs error	inputs pre-condition
R01.2	Invalid username format. Invalid inputs error	username pre-condition
R01.3	Valid username format.	username pre-condition
R01.4	Invalid password format. Invalid inputs error	password pre-condition
R01.5	Valid password format	password pre-condition
R01.6	Wrong username (not in database). Username doesn't exist error	post-condition clue
R01.7	Correct username, wrong password. Password doesn't match error	post-condition clue
R01.8	Wrong username and password. Login error	post-condition clue
R01.9	Correct username and password. Home screen	post-condition clue

Table 8 ITF01 Test Requirements

vi. Test specifications

No.	Test Specification	Covers
S01.1	username = "", password = ""	R01.1
S01.2	username = "emp123", password = "abcd_01"	R01.2
S01.3	username = "notOmar", password = "notMe_01"	R01.3, R01.5 R01.6, R01.8
S01.4	username = "Kamal", password = "123456789"	R01.3, R01.4 R01.7
S01.5	username = "Salem", password = "yesMe_01"	R01.3, R01.5 R01.9

Table 9 ITF01 Test Specifications

II. ITF02

i. Inputs / Variables

(1) Correct valid username (String)

(2) Correct, matching, valid password (String)

ii. Pre-conditions

Same preconditions on the inputs set in ITF01.

iii. Clues / Post-conditions

if (successful log in) then home screen = role-personalized page.
if (returned home screen not the one associated with the profile) then return “internal error”

iv. Test requirements

No.	Test Requirement
R02.1	Valid manager profile
R02.2	Valid host profile
R02.3	Valid waiter profile
R02.4	Valid cook profile

Table 10 ITF02 Test Requirements

v. Test specifications

No.	Test Specification	Covers
R02.1	username = “Ahmed”, password = “man_02”	R02.1
R02.2	username = “Mostafa”, password = “host_02”	R02.2
R02.3	username = “Karim”, password = “wait_02”	R02.3
R02.4	username = “Ali”, password = “chef_02”	R02.4

Table 11 ITF02 Test Specifications

III. ITF03

i. Inputs / Variables

(1) An action that requires screens to update

(2) A log-in or out operation (trigger)

ii. Assumed pre-conditions

(P1.1) The action is authorized to its executer.

iii. Clues / Post-conditions

if (table assigned by host) and (waiter logs in) then { waiter's home screen is updates }
if (waiter order) and (waiter logs out) then { cook's home screen is updated with new order }
if (order is ready) and (waiter logs in) then { notify the waiter with done orders }
if (table status change) and (waiter logs out) then { floor plan is updated with table colors }
if any updating action is not followed by a trigger then { no change (update is not applied) }

iv. Test requirements

No.	Test Requirement
R03.1	A table is assigned to a waiter, waiter logs in
R03.2	An order is placed, waiter logs out
R03.3	An order is marked ready, waiter log in
R03.4	A waiter changes a table status, waiter logs out

Table 12 ITF03 Test Requirements

v. Test specifications

No.	Test Specification	Covers
S03.1	Table no.11 assigned to waiter "Samir"	R03.1
S03.2	Order with id #289 containing "Soup, Rice"	R03.2
S03.3	Order with id #365 is ready	R03.3
S03.4	Table no.27 changed from "ready" to "occupied"	R03.4

Table 13 ITF03 Test Specifications

IV. ITF04

i. Inputs / Variables

Table status

ii. Assumed pre-conditions

Table status is one of an enumerated type

iii. Clues / Post-condition

if (TS = OPEN) Then { table color = GREEN }
else if (TS = OCCUPIED) { then table color = YELLOW }
else if (TS = DIRTY) { then table color = RED }

iv. Test requirements

No.	Test Requirement
R04.1	A transition tour to detect any output (color) error

Table 14 ITF04 Test Requirements

v. Test Specifications

No.	Test Specification	Covers
S04.1	Transition tour is: OPEN.OCCUPIED.DIRTY	R04.1

Table 15 ITF04 Test Specifications

V. ITF05

i. Inputs / Variables

Selected table in waiter's home screen

ii. Clues / Post-conditions

if (table selected is assigned (colored)) then { next view = table options screen }
if (selected table not assigned (not colored)) then { return error message (table not assigned) }

iii. Test requirements

No.	Test Requirement
R05.1	Selecting an assigned table
R05.1	Selecting a non-assigned table

Table 16 ITF05 Test Requirements

iv. Test specifications

No.	Test Specification	Covers
S05.1	Table no.15	R03.1
S05.2	Table no.20 Error	R03.2

Table 17 ITF05 Test Specifications

VI. ITF06

i. Inputs / Variables

(1) Selected ASSIGNED table.

(2) A menu item to add.

ii. Validated pre-conditions

Selected table status is yellow (OCCUPIED)

iii. Assumed pre-conditions

The item is a menu item (among the item categories offered by the restaurant)

iv. Clues / Post-conditions

If (table is yellow) then {
    If (added item's availability flag == true) then { Item can be added, Update order tab }
    } else { Item cannot be added, Return "Item is not available" }
} else {
    Table is not occupied. Therefore, it has no current order tabs.
    return "No order tab available" }

v. Test requirements

No.	Test Requirement
R06.1	Table is not yellow. Error
R06.2	Table is yellow, Item is not available in the inventory. Error
R06.3	Table is yellow, item is available in the inventory

Table 18 ITF06 Test Requirements

vi. Test specifications

No.	Test Specification	Covers
S06.1	Table no.18 which is GREEN	R06.1
S06.2	Yellow table no.6, Item "Soup" is not available	R06.2
S06.3	Yellow table no.8, Item "Rice" is available	R06.3

Table 19 ITF06 Test Specifications

VII. ITF07

i. Inputs / Variables

(1) New Profile

(2) Data field (ex. Employee's ID)

(3) Email address

ii. Validated pre-conditions

(P1) New profile is unique

(P2) Personalized data must be unique like the ID.

(P3) Email address must be valid.

iii. Clues / Post-conditions

If (new profile is a duplicate of another) then {
Duplicate profiles are indistinguishable
return "Profile already exists" }
if (new ID exists in another account) then { return "ID already assigned to another employee
if (new email is invalid) then { return "invalid email" }

iv. Test specifications

No.	Test Specification
S07.1	Duplicate an existing account. Error
S07.2	New ID = "PRM289", which was assigned before. Error
S07.3	New email = . Error

Table 20 ITF07 Test Specifications

VIII. ITF08

i. Inputs / Variables

An action done on the system by an employee

ii. Assumed pre-conditions

The action is authorized to its executer.

iii. Clues / Post-conditions

if (an action takes place on the system) then { log file updated }
else { return log file error }

iv. Test requirements

No.	Test Requirement
R08.1	Waiter's profile, Placed order.
R09.2	Waiter's profile, assigned table
R09.3	Host's profile, table assigning
R09.4	Cook's profile, prepared order

Table 21 ITF08 Test Requirements

v. Test specifications

No.	Test Specification	Covers
S08.1	Profile ID "PRW015", Table no.23, Order id #256	R08.1
S08.2	Profile ID "PRW017", Assigned table no.14	R08.2
S08.3	Profile ID "PRH050", "PRW019", Table no.3	R08.3
S08.4	Profile ID "PRC060", Order id #128	R08.4

Table 22 ITF08 Test Specifications

IX. ITF09

i. Inputs / Variables

A new authorized feature by the manager

(F1) View other waiter's tabs

(F2) Refuse assigned table by the host

ii. Assumed Pre-conditions

Feature is implemented.

iii. Clues / Post-conditions

if (feature is authorized) then { waiter's home screen updates with feature enables }
else if (feature is disabled) then { waiter's home screen updates with the feature disabled }

iv. Test specifications

No.	Test Specification
S09.1	Authorize a new feature
S09.2	Un-authorize a feature

Table 23 ITF09 Test Specifications

X. ITF10

i. Inputs / Variables

Here is another ambiguity about this feature:

- How does the manager re-arrange the floor plan? Is it as simple as swiping tables?

I assume that:

- The floor plan is designed and drawn using an external third-party tool, and then uploaded to the restaurant system to apply it.
- The uploaded file must, of course, be of a certain type with a certain extension.

So, the inputs are:

(1) The new floor plan drawing.

(2) An update trigger (log-in or out operation)

ii. Validated pre-conditions

(P1.1) The uploaded drawing file has a (.xxx) extension.

iii. Clues / Post-conditions

```
if (file extension is accepted) then {
        if (changed tables are not green) then {
                File is uploaded accepted and changes will not be lost. But,
                return "cannot apply changes now, try to apply later"
        } else if (changed tables are green) then {
                Apply the new floor plan
                if (host or waiter logs in) then {
                home screens update
                }
        }
} else { File is not accepted, return "cannot read floor plan drawing" }
```

iv. Test requirements

No.	Test Requirement
R10.1	New floor plan drawing with invalid extension
R10.2	New floor plan drawing with valid extension
R10.3	Changes include Non-GREEN tables
R10.4	Changes only includes GREEN tables
R10.5	Waiter log-in operation after successful changes
R10.6	Host log-in operation after successful change

Table 24 ITF10 Test Requirements

v. Test Specifications

No.	Test Specification	Covers
S10.1	File name = newDesign.xyz	R10.1
S10.2	File name = NewDesign.xxx, YELLOW tables included	R10.2, R10.3
S10.3	File name = LatestDesign.xxx, only GREEN tables	R10.2, R10.4
S10.4	Waiter logs in, Host logs in	R10.5, R10.6

Table 25 ITF10 Test Specifications

4. TEST CASES

After formulating models, and deriving test specifications from them according to test requirements. It is time now to generate the test cases to validate the system with each one's main three parts.

Case No.	Input/s	System State	Output/s
I. ITF01			
C01.1	username = “” password = “”	Log-in screen	Invalid log-in data
C01.2	username = “emp123”, password = “abcd_01”	Log-in screen	Invalid username
C01.3	username = “notOmar” password = “notMe_01”	Log-in screen	username does not exist
C01.4	username = “Kamal”, password “123456789”	Log-in screen	Password does not match
C01.5	username = “Salem” password = “yesMe_01”	Log-in screen	Home screen
II. ITF02			
C02.1	username = “Ahmed” password = “man_02”	Log-in screen, the used profile metrics are correct	Manager home screen
C02.2	username = “Mostafa” password = “host_02”	Log-in screen, the used profile metrics are correct	Host home screen
C02.3	username = “Karim” password = “wait_02”	Log-in screen, the used profile metrics are correct	Waiter home screen
C02.4	username = “Ali” password = “chef_02”	Log-in screen, the used profile metrics are correct	Cook home screen
III. ITF03			
C03.1	Table no.11 waiter “Samir”	Host's assign table to a waiter from his H. screen	Table no.11 at Samir's home screen is colored
C03.2	Order with id #289 containing “Soup, Rice”	Waiter places an order at the table options screen	Cook's home screen is updated with new order
C03.3	Order with id #365 marked ready	Cook marks an order as ready after it is prepared	Waiter's H.S updated with a notification
C03.4	Table no.27 “ready” to “occupied”	Waiter changes table state at the floor plan screen	Update floor plan for both waiter and host
IV. ITF04			
C04.1	Transition tour = OPEN. OCCUPIED.DIRTY	Table options screen, “change status” option	Table color = GREEN. YELLOY.RED

Case No.	Input/s	System State	Output/s
V. ITF05			
C05.1	Selected table no. = 15	Waiter's assigned tables home screen	Table options screen
C05.2	Selected table no. = 20	Waiter's assigned tables home screen	Error message = Table is not accessible
VI. ITF06			
C06.1	Selected table no = 18 Table color = GREEN	Table options screen, "place an order" option	Error message = No available tabs
C06.2	Selected table no. = 6 Table color = YELLOW Order items = Soup	Table options screen, "place an order" option	Error message = Item is currently unavailable
C06.3	Selected table no. = 8 Table color = YELLOW Order items = Rice	Table options screen, "place an order" option	Item is added successfully. Order tab is updated
VII. ITF07			
C07.1	New account credentials = duplicated ones	Create new profile option screen on manager's H.S	Error message = Account already exists
C07.2	Assigned ID = "PRM289"	Modify profile data option screen on manager's H.S	Error message = ID is taken
C07.3	New email = Ahmed@notGmail.com	Modify profile data option screen on manager's H.S	Error message = Invalid email address
VIII. ITF08			
C08.1	Profile ID = PRW015 Table no. = 23 Order id #256	Waiter makes an order. Manager tracks his activity on his profile's log file	Log file is updated with the executed operation details
C08.2	Profile ID = PRW017 Assigned table no = 14	Manager tracks a host's activity on his log file	Log file is updated with the executed operation
C08.3	Host ID = PRH050 Waiter ID = PRW019 Table no. = 3	Manager tracking activity	Log files of both host and waiter are updated with the operation
C08.4	Profile ID = PRC060 Order id #128	Cooks marks ready order. Manager tracks his activity	Log file is updated with the executed operation

Case No.	Input/s	System State	Output/s
IX. ITF09			
C09.1	A new authorized activity for a waiter	Manager options	Feature enabled at waiter's home screen
C09.2	Restrict an authorized feature = view tabs	Manager options	The feature is no longer available
X. ITF10			
C10.1	Uploaded file name = newDesign.xyz	Manager options. Upload a new design to alter the floor plan.	Error message = unreadable file format
C10.2	File name = NewDesign.xxx Changed tables = 16, 19 Table 16 is YELLOE	Manager options. Upload a new design to alter the floor plan.	Error message = Cannot apply changes now, table is under use
C10.3	File name = LatestDesign.xxx Changed tables = 8, 14 Both are green	Manager options. Upload a new design to alter the floor plan.	Changes accepted and will be applied after a successful log-out
C10.4	A waiter log in A host log in	Host & waiter home screens	Home screens are updated with the new floor plan

Table 26 Test Suite

5. SYSTEM TESTING

In this part, several system testing techniques are proposed, focusing on three points:

- System testing techniques used.
- Why choosing this technique, and what are the benefits of applying it.
- How to apply this technique on this system.

NOTE THAT, The SRS mentioned (by illustration, not clearly spilled) that the waiter terminal which deals with tabs is connected to a server for record keeping. Meanwhile, it has not provided any clear information about the inter-communications structure, so, I assume that:

- The terminals communicate with each other using a local network.
- The whole network is connected to a DB server to record all the activities for tracking.
- Only the manager has access to this archive.

I. Performance Testing

Performance testing, as known, is applied to evaluate some investigation measures and verify them, like speed (responsiveness), scalability, stability, and confidence (reliability). To evaluate the system's performance, we will go through some tests as follows.

i. Load testing

This test aims to put the system under heavy demand by executing the largest tasks it can handle and measure its response. this test is applied to:

- Determine how the system will behave under both normal and anticipated heavy load.
- Measure the maximum capacity of operations the system can deal with. (consecutive loads leading to "limit testing")
- Determine which specific operational elements can cause the system to degrade.

The strategy of applying such test goes as follows:

- Steadily, and using automated tools, increase the load applied to certain points in the system, that actually, in real time scenario, will normally experience a heavy load.
- Test the single ordering communication between waiter ↔ cook, by increasing the rate of placed orders per hour. Reversely, test large numbers of orders marked ready per hour.
- Apply the same test but involve another parameter by increasing the number of waiter accounts making orders, and number of cooks receiving them.
- Since the system updates are triggered by logging, it is important to put the system under test of excessive logging in and out operations from several and different types of profiles and see how much it can handle them quickly and responsively.
- Simulate a rush hour scenario, by increasing the overall number of operations handled by the system (Assigning tables, regular changes in table status).
- Putting the system under heavy load will also cause the number of order tabs to increase. So, we also need to test that they are saved successfully.

ii. Limit testing

This test comes after the load test, and simply, aims to push the system to its limit in order to:

- Determine the maximum (peak point) of the system, after which the system starts to degrade and struggle with problems.
- Later in stress testing, to test how the system degrades, is it a dramatic or peaceful?

Here is how we are going to do this:

- Test every limit of the system's different features and behaviors specified in the SRS.
- Maximum number of profiles (with their different types) can run on the system.
- Maximum rate of placed orders per hour the system can handle, including tabs.
- Maximum rate of shifting table status.
- Maximum Log in and out operation per certain amount of time.
- For the activity tracking feature, test the maximum successive activities the log file can track and save.
- Maximum size of the floor plan drawing (.xxx) file the manager can upload (recap this assumption)

iii. Stress testing

This test aims to answer the question: what happens if the applied load surpassed the peak point of the system? How will the system behave if the normal operational capacity exceeded its limits? So, this test is applied to:

- Measure the stability of the system.
- Make sure that when the system fails, it can recover gracefully.

To apply this test:

- Test the system under conditions beyond its limits (number of orders per hour it can process, number of logging operations)
- Double the number of profiles running on the system: test if the system breaks due to excessive operations from large number of profiles.
- Make random shutdown and restarts to the local network devices ports and the server connectors.
- Test the system under an abnormal resources demand:
    - Low memory conditions.
    - Server and/or network faults.
    - Unusually high data transfer rate on the network.

iv. Soak testing

Since the system will be up and running for almost all day, also involves a database and record tracking, it is important to apply this test in order to:

- Verify system's performance and provide a stability measure over long period of time.
- Identify if there exist any issue with the log files that tracks activities.
- Test the database resource utilization.

How to apply soak testing:

- After the system is put into operation by users for an extended period of time, we trace the performance of the system among this period as a factor in time.
- Plot a chart illustrating this factor and by examine and analysis, does the system hold a steady performance over this time period or it suffers from stability issues?
- It is important to note that in order to apply this test, we need to maintain a certain level of user concurrency during the testing period. Assume an average number of profiles of each type.

An illustrated example of the plotted chart (Figure 7)

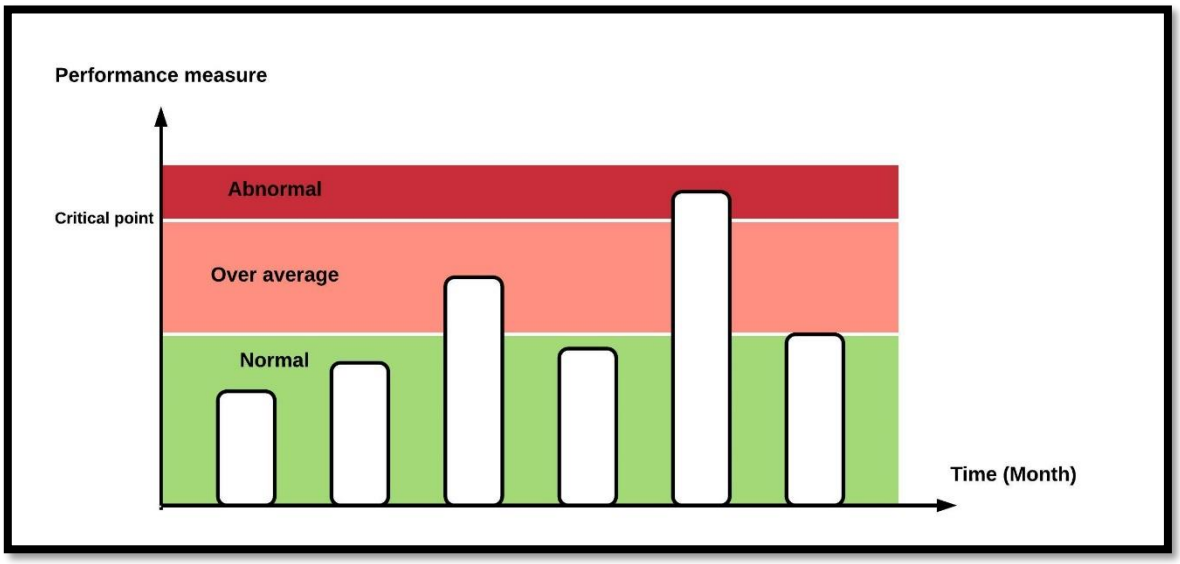

Figure 7 Soak Testing Performance Chart

v. Spike testing

A sudden increment in the load on the system is very expected to happen during rush hours or an offer day. The number of operations executed is supposed to increase suddenly in these situations, so a spike testing will aim to test the performance stability of the system when it experiences such spikes.

How to apply it:

- Somehow similar to the load test cases, except that instead of increasing steadily, an extreme load here is applied in a sudden instead after the system was running on an average load.

II. Recovery Testing

When the system experiences a crash or a hardware failure for any reason, this test aims to:

- Test the ability of the system to recover even if its configuration files got corrupted.
- When it recovers, what is the status of the system in terms of files and data.

It is a crucial issue to test, because assume that the testing is running in an environment suffering from random breakouts where power cuts suddenly, the test should check that:

- Will the system even be able to correctly go live again or it is so fragile that a sudden failure can cause a permanent damage.
- When the system wakes up again, if it did, how well its recovery will be considering the user data:
  - Can the placed orders be recovered? or the orders queue has vanished.
  - The assigned tables for each waiter and the tables status. Are they recovered is they were before failure? Or the home screens have been reset?

  It is an important matter for a system like this when it recovers from a crash to maintain all the user data status as it was before the crash. Otherwise, it would be a mess for the working staff.

- Consider the same recovering issue with the president data repository, are the data and files in the database damaged or corrupted?

III. Configuration Testing

The SRS seems confused about this issue. No determination on neither the specific hardware components nor the OS the system will run on. So, assuming that these configurations are stated, this test will:

- Test on all required hardware configurations.
- Using virtualization technologies, test on all required OS versions.
- Combine the above two tests with each other as much as we can.
- Try testing relevant configurations to the ones required.

IV. Compatibility Testing

Since the system will run on the terminal as standalone, the only compatibility issue will be with the network and server connections.

This test can be applied to:

- Test if there are specific network interfaces recommended to use.
- If the menu includes items images, test if there are specific types are best for the system to deal with (jpg, jpeg, or png and so on)

V. Usability Testing

Such system is in an imperious need to a user-friendly interface to help accelerating and smoothening the tasks each employee does on it.

So, we apply this test to:

- Determine how much the user interfaces are organized logically and easy to use.
- See if common tasks (assigning tables, making orders, changing table status, etc.) are fast and simple to execute.
- Are users satisfied with this interface, or it is un-comfy and frustrate them.
- Report any bugs or problems encountered with the interface (e.g. miss-arranging items)

To apply this test:

- Bring up real users to try the system and use it to perform some tasks. Expose to them the important tasks interfaces like the floor plan of the restaurant and the ordering menu.
- Note the things that may obstruct them while doing their regular tasks.
- Get their feedback on the user interface and any suggested recommendations to improve it (e.g. if the place order hit button can be located better on the screen, if an icon can be larger, if the floor plan view can be altered or rotated, and so on)

VI. Documentation testing

This test is not discussed, but it was, actually, applied on the level of internal documentation. To be specific, the document you are reading right now has been subjected to:

- Read test by examining its clarity, accuracy, and organization.
- Functional test by verifying the document.

VII. Acceptance Testing

This a benchmark test, applied on three test phases sequentially as follows:

i. Pilot testing

Users will employ the system under these conditions of controlled operational environment:

- Only certain users will be involved (Managers or high-level profiles)
- Small-scale experience of certain system functionalities:
  - Logging in and out operations.
  - Hosts assigning tables to waiters.
  - Waiters placing orders.
  - Managers tracking employees' activities using log files.
- The network used for communications between terminals is of a certain bandwidth.

ii. Alpha testing

An in-house pre-release test. Since this test is done by real users, so, it aims to:

- Discover errors caused by users' misuse of the system.

This test is done under these circumstances:

- The presence of system professionals.
- A controlled operational environment of network and hardware with certain organization-standard specifications.
- Using simulated data. (Dummy employee profiles, floor plan, and restaurant menu).

iii. Beta testing

Also a pre-release test, but this time, adding a new parameter to the test process which is the user environment the system will actually run on. So this test aims to:

- Discover the problem caused by user environment.

  Which in fact, the SRS at first, hesitantly proposed multiple options without being specific, from using handhold devices like tablets to setting up limited stationary computer terminals. But, at last, the SRS seemed to go with the second option.

This test is done under these circumstances:

- No presence of system professionals this time.
- Running in the user environment using real data related to the restaurant (real user profiles, the floor plan of the restaurant, and the restaurant menu)

VIII. Deployment / Installation Testing

The final test is to deploy the software by installing it on the target system, and test it against:

- User hardware configurations including computer terminal devices and network components.
- Using various versions of third-party components like database and file system that is supposed to keep the log files.
- Testing various versions of the operating system and resident software installed on the users' computer terminals.

6. SECURITY TESTING

Recap these requirements from the SRS:

- Computer terminals will use a keyboard for user authentications.
- Some terminals must be physically secured because its users are always logged in.
- An illustration of the use of a server for record-keeping.

Also recap these assumptions:

- Terminals communicate with each other using a local network set up and installed at the restaurant.
- The system is connected to a DB server for activity records keeping and data archiving (tabs, profiles data, and other restaurant data).

Added assumptions:

A security matter question, is the DB server also installed locally? I assume that:

- The restaurant budget cannot afford such option.
- The system's network is connected to an external DB server using a web server.
- The local network uses wired connections with ethernet cables connected using some networking devices.

I. Security Architecture

iv. Sub-systems structure

According to these requirements and assumption. A **Multi-Tier Architecture** reveals the decomposition of the system into components and sub-systems illustrated as in (Figure 8)

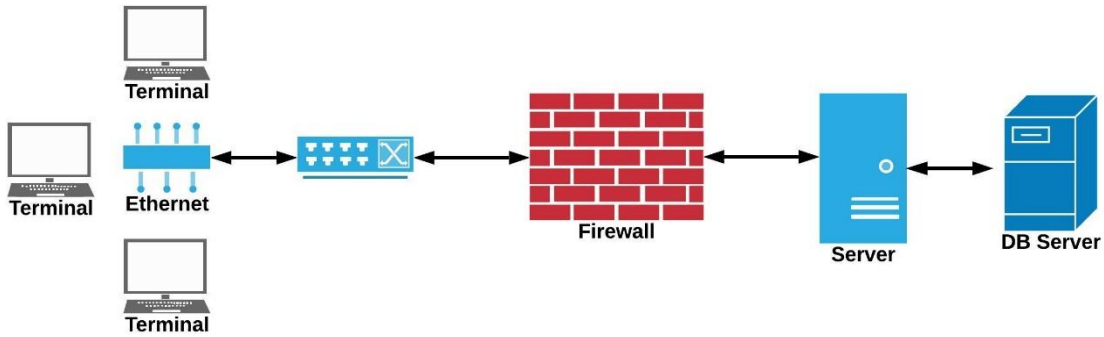

Figure 8 System Architecture

Using cables instead of a wireless network is to prevent any attempt for connecting to the network except for the connected authenticated terminals, unless an attacker got himself a wired access to the network which is physically hard to achieve.

v. Proposed security architecture

Security architecture provides a description to the security mechanisms placing in the system's architecture. In other words, it represents a security view of the system architecture.

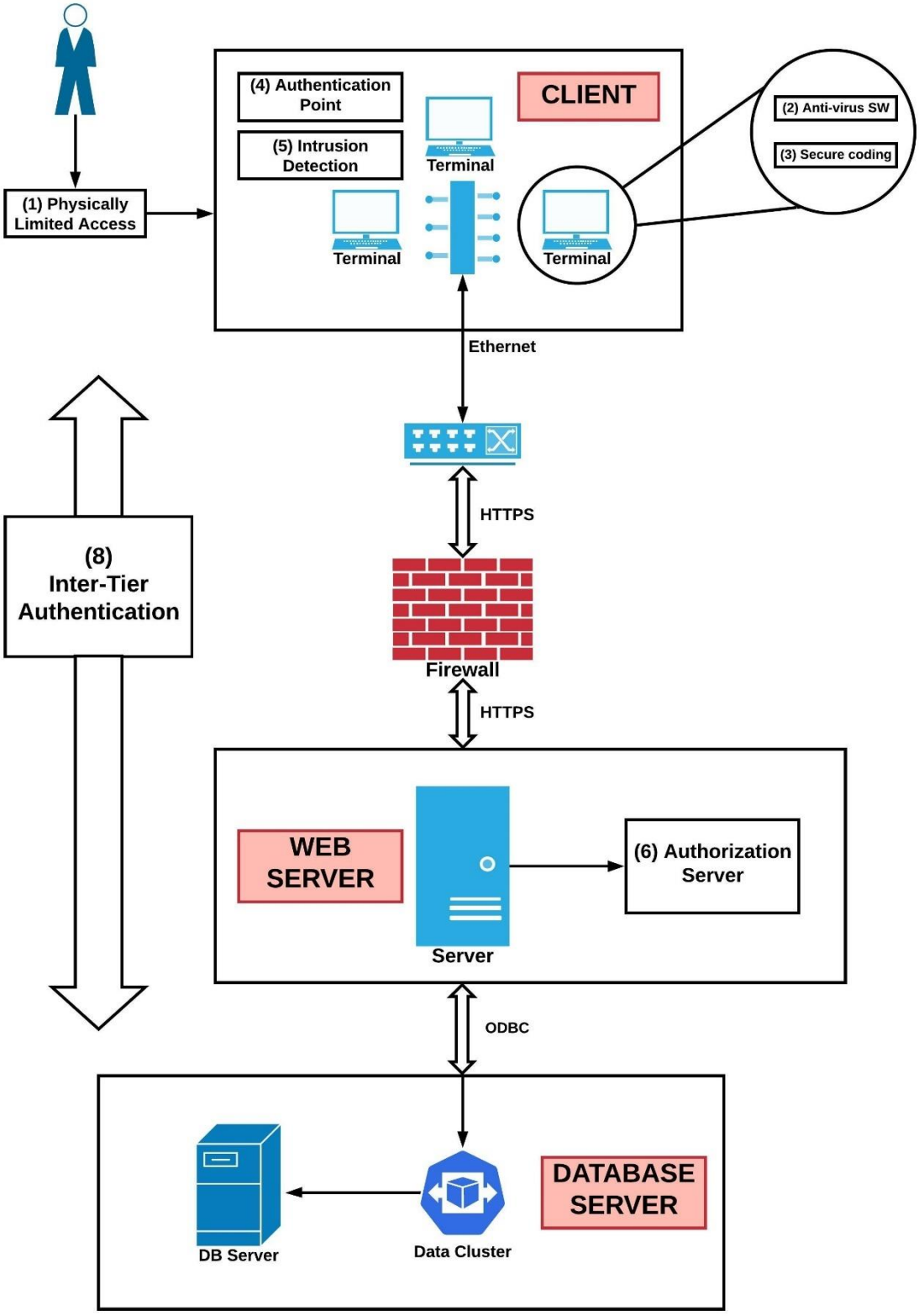

Figure 9 Security Architecture

vi. Proposed security architecture explained

(1) Physically limited access: The computer terminals used by the staff is to be placed in a secure area where they can be supervised.

(2) Anti-virus software: The computer security of the terminal devices may not be our main concern as testing team because it is the management interest to use highly secured devices. But, to avoid any virus damage to the system, we need to make sure that these computers at least contain a powerful Anti-virus.

(3) Secure coding techniques: are to be used to limit the system vulnerabilities.

(4) Authentication: An essential part of the software security. Each employee has a username and a security code (password), these data must pass by an authentication check before accessing the system.

(5) Intrusion Detection: An effective way to detect any attempt of intrusive access to the network. The computer terminals have IP addresses which are known to the network. Although the network is wired, but assume that someone succeeded to hock himself in. His device will be recognized as an intruder and the network administrators shall be notified to take actions.

(6) Authorization server: Security domains needs a keep of a proper boundary between them. An authentication server is an access policy applier. It is an engine or a factory to mint OpenID connect (Token) which are used to authenticate users with the software system.

(7) The use of multi-tier architecture and a data cluster: we could have used a single server to server as web and DB at the same time. But this multiplicity in tiers eliminates a single point of failure and provides an enhanced security by limiting the attacker's chances of taking control over the entire system by breaking a single point.

(8) Inter-tier authentication: Before initiating any connection or data transfer between tiers. They must identify and authenticate each other to ensure that no attacker can fake a tier identity get between them. Such authentication is applied using some mechanisms like mutual SSL or IP validation.

(9) The use of HTTPS connection protocols: is to encrypt the network traffic. HTTPS provides an end-to-end secure encrypted connection between client (SW system) and server. The two protocols TLS and SSL are used to protect the communication's confidentiality and integrity over HTTPS. Also, the use of HTTPS protects the transferred data over the network.

(10) The use of a separated data server: is meant to be a way of protection to keep the data stored within a back-end DB server, because it is any attacker's main target to steal important data of the system. There are multiple ways to protect the data resting on the server, including using updated encryption algorithms like AES with a strong encryption key (128 bits or higher).

(11) Logging: An important feature to the system that is also related to security issues. Alongside with users' activities and operations on the system, log files also identify all security incidents either by system users or external attackers. The later case is much important because we can trace back the attacker using these logs. Log files keep the security events, but no sensitive information is logged. The logs are encrypted strongly, and only administrators have access to these files (managers), and we have discussed such issue before.

II. Security Techniques

Before recommending the testing techniques to be used for the security test. We will briefly pass through the **Threat Modeling Process;** first, to identify the assets. And after reviewing the security architecture, possible threats are identified. According to this process, we can then recommend some testing techniques to use.

Recap from the SRS that:

- The manager has the privileges of accessing the database.
- This access includes profiles manipulation and financial matters.
- Other staff users have restricted privileges considering their tasks on the system.

Recap the assumptions that:

- Employees activities are tracked and recorded in log files associated with their profiles.
- Database resides encrypted at a DB server. Meaning that, unless you have an administrative power, you cannot access it.

i. Assets

So, what is so important about a restaurant serving food, that attracts attackers' attention. In fact, the services and operations executed on the system (assigning tables, making orders, and so on) are not that interesting for an attacker, unless he is playing around!!! Therefore, we are not highly expecting a DDOS attack or something like that.

So, the most important asset for such a system is, typically like any other system, the data base:

- **Database:** the data base of this system holds all employees' profiles containing their personal data, alongside with the whole operational activities of the restaurant, including orders tabs which is a critical financial subject.

ii. Threats

Possible threats (with each one's category) that can threaten the database security:

- **Spoofing identity:** Bypassing the authentication point (4) and gain access to a manager account. This is a critical threat that can compromise the system to real danger.
- **Repudiation:** Assume there could be some manipulative actions with order tabs performed by waiters. They will deny, however, log files are there for a reason!
- **Elevation of privilege:** Shamelessly, this is an internal attack. An employee may grant an access to perform some actions that are not authorized to him.

A very dangerous threat: gaining the log in credentials of an administrator account might not be enough for the attacker because he might get caught and the password gets changed, so he can grant him self an administrative access by:

- Planting rootkit on the server.

iii. Recommended security testing techniques

Based on the threat modeling, the following security testing techniques are recommended to apply:

- **OS Hardening:** because the system is connected to a network, it is important to:
  - Disable or restrict unused ports.
  - Lock down the ports.
  - Install root certificate.
  - Keep the OS up to date.
- **Vulnerability Scanning:** Back doors can cause real threat to the security of the system by opening a window for the attackers to take control over the system, so, we should:
  - Scan and identify both known and unknown vulnerabilities.
- **Port scanning:** For networking purpose again, it is recommended for the open ports on each computer terminal used, to be identified and located.
- **Firewall Rule Testing:** Recap that the security architecture contains a protective firewall between the network and the web server. Therefore, to avoid web attacks, it is considerable to:
  - Set the firewall rules appropriately and pay attention for conflicting rules.
  - Discover any administrative backdoors or tunnels.
- **SQL Injection:** Consider this as the most critical test since the database is the highest priority asset in the system. The system should prevent any attempt to execute unauthorized SQL commands exploiting any database security vulnerability.
- **Network Scanning:** The local network connecting the computer terminals must of course be secured enough, so we need to:
  - Identify the active hosts.
  - Collecting the connected devices IP addresses, OS details, system architecture, and running services.
  - Collecting network user and group names if any.

  Recap that the security architecture contains an intrusion detection mechanism that can make a good use of such collections.

- **Password Cracking:** Specially for the high-profile accounts owned by managers.
  - Collecting passwords from transmitted data over the network.
  - Using brute-force attack by trying every combination in password dictionaries.
  - Identify weak passwords. Recap that there a pre-condition to prevent users from setting weak passwords to their accounts.
- **Penetration Testing:** This test is targeted to hunt down security vulnerabilities that might be existing in the system. We do that by:
  - Simulating possible attacks from malicious sources.
  - Network and vulnerabilities scanning
  - Simulate attack from someone who is unfamiliar with the system.
- **Other techniques to consider:**
  - Random Data Testing
  - IP spoofing
  - Packet Sniffing
  - Virtual Network Testing

REFERENCES

[1] Andreas Spillner, Tilo Linz, Hans Schaefer. *Software Testing Foundations.* Rocky Nook. Santa Barbara. 2014, Mar 19.

[2] Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black. *Foundations of Software Testing.* Cengage Learning Emea. 2008, Jan 28.

[3] Mauro Pezze, Michal Young. *Software Testing and Analysis: Process, Principles and Techniques.* Wiley. 2007, Apr 13.

[4] Roger S. Pressman, Bruce R. Maxim. *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education. 2014, Jan 23.

[5] Thomas J. Ostrand, Marc J. Balcer. *The Category-Partition Method for Specifying and Generating Functional Tests.* Communications of the ACM. 1988, Jun.

[6] Bernhard K. Aichernig, Wojciech Mostowski, and others. *Model Learning and Model-Based Testing.* Department of Informatics, University of Leicester, UK