# Test Levels

## Build Tests

### Level 1 – Build Acceptance Test

### Level 2 – Smoke Tests

#### APIs

**/inventory**

Purpose:

The collection of all inventory items.

Anticipated Result:

GET: Retrieve a listing of all existing inventory items. Should support a query parameter "search" that accepts a string by which results can be filtered.

POST: Create a new resource based on the data included in the request body.

**/inventory/:tag**

Purpose:

An individual inventory item resource by asset tag.

Anticipated Result:

GET: Retrieve a representation of the specified resource.

PUT: Update the specified resource with the data included in the request body.

DELETE: Delete the specified resource.

**/inventory/types**

Purpose:

Collection of inventory item types

Anticipated Result:

GET: Retrieve a list of inventory item types. Used to populate inventory type drop-down menus in the app.

**/inventory/:tag/history**

Purpose:

The collection of check-in records for the specified inventory item.

Anticipated Result:

GET: Retrieve a list of representations of the entire check-in history of the specified item. Should support JSON or CSV response format.

POST: Create a new check-in resource for the specified option, based on the data in the request body (in particular, the room ID).

**/inventory/:tag/history/:num**

Purpose:

The check-in resource with the given sequence number (id) for the inventory item with the given ID.

Anticipated Result:

GET: Retrieve a representation of the specified check-in record. Should include details for the room and building.

**/inventory/:tag/history/latest**

Purpose:

An "alias" that points to the latest check-in record for the specified item.

Anticipated Result:

GET: Finds the sequence number of the most recent check-in record for the specified inventory item and does an HTTP redirect to that URL.

**/inventory/reports/by-type**

Purpose:

Inventory report, by type

Anticipated Result:

GET: Retrieve the results of running this report. Should support JSON or CSV response format.

**/inventory/reports/reconciliation**

Purpose:

Inventory reconciliation report

Anticipated Result:

GET: Retrieve the results of running this report. Should support a query parameter "last-checkin" specify how far back an item's last check-in should be for it to appear on the report. Should support JSON or CSV response format.

**/buildings**

Purpose:

The collection of building records.

Anticipated Result:

GET: Retrieve a listing of all existing building resources. Should support a query parameter "search" that accepts a string by which results can be filtered.

POST: Create a new resource based on the data included in the request body.

**/buildings/:id**

Purpose:

An individual building resource

Anticipated Result:

GET: Retrieve a representation of the specified resource.

PUT: Update the specified resource with the data included in the request body.

DELETE: Delete the specified resource.

**/buildings/:id/rooms**

Purpose:

The collection of room resources for the specified building.

Anticipated Result:

GET: Retrieve a listing of all existing room resources associated with the specified building. Should support a query parameter "search" that accepts a string by which results can be filtered.

POST: Create a new room resource, associated with the given building, based on the data included in the request body.

**/buildings/:buildingid/rooms/:roomid**

Purpose:

An individual room resource associated with the specified building resource.

Anticipated Result:

GET: Retrieve a representation of the specified room resource.

PUT: Update the specified room resource with the data included in the request body.

DELETE: Delete the specified room resource. Rooms that have been included as part of an inventory item's check-in history cannot be deleted.

**/buildings/:id/rooms/:roomid/inventory**

Purpose:

An individual room's collection of inventory items.

Anticipated Result:

GET: Retrieve the collection.

### Level 2a – Test Review

Every bug that was “Open” during the previous build, but marked as “Pending, Pass, Not Pass, Fixed, Needs Re-Testing” for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

#### APIs

1. **/inventory**

***Test case 1:***

Description: Retrieve a listing of all existing inventory items.

Assumption: There exists some items.

URL: *http://192.168.33.99/api/inventory*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a listing of all existing inventory items.

Test Code Source: *tests/api/inventory/inventory\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case 2:***

Description: Retrieve a listing of all existing inventory items.

Assumption: There are no items.

URL: *http://192.168.33.99/api/inventory*

Calling Method: *GET*

Anticipated Result Code: *404*

Anticipated Result Datatype: *JSON or HTML:*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/inventory/inventory\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case 3:***

Description: Add a new item.

Assumption: There are no item conflict.

URL: *http://192.168.33.99/api/inventory*

Calling Method: *POST*

Data to post: ['name' => 'PCHS', 'description' => 'Powell Co. High School', 'roomID' => 'ID’]]

Anticipated Result Code: *201*

Anticipated Result Datatype: *JSON.*

Anticipated Result Data: a list of all posted inventory.

Test Code Source:*tests/api/inventory/inventory\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**2. /inventory/:tag**

***Test case 1:***

Description: Delete a specified resource.

Assumption: There’s no such resource existed.

URL: *http://192.168.33.99/api/***/inventory/tag**

Calling Method: *DELETE*

Anticipated Result Code: *404*

Anticipated Result Datatype: *JSON or HTML:*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/* **/inventory/tag***/tag\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case2:***

Description: add a new specified resource.

Assumption: There are no existing resources.

URL: *http://192.168.33.99/api/***/inventory/tag**

Calling Method: *PUT*

Anticipated Result Code: *200*

Anticipated Result Datatype: *JSON:*.

Anticipated Result Data: a list of all existing resources.

Test Code Source: *tests/api/* **/inventory/tag***/tag\_PUT.php;*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case3:***

Description: test get method when the specified items are not existing.

Assumption: DELETE method works..

URL: *http://192.168.33.99/api/***/inventory/tag**

Calling Method: GET

Anticipated Result Code: *404*

Anticipated Result Datatype: *JSON or HTML*.

Anticipated Result Data: Any

Test Code Source: *tests/api/* **/inventory/tag***/tag\_GET.php;*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**3. /inventory/types**

***Test case 1:***

Description: test get method when there is no items..

Assumption: no item exists.

URL: *http://192.168.33.99/api/***/inventory/types**

Calling Method: *GET*

Anticipated Result Code: *404*

Anticipated Result Datatype: *JSON or HTML*.

Anticipated Result Data: Any

Test Code Source: *tests/api/* **/inventory/types***/***types\_GET***.php;*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case2:***

Description: test get method after posting new items..

Assumption: POST works..

URL: *http://192.168.33.99/api/***/inventory/types**

Calling Method: *GET*

Anticipated Result Code: *200*

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of types of existing items.

Test Code Source: *tests/api/* **/inventory/types***/***types\_GET***.php;*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**4. /inventory/tag/history**

***Test case 1:***

Description: test get method when there is no specified resource.

Assumption: there is no resource.

URL: *http://192.168.33.99/api/***/inventory/tag/history**

Calling Method: **GET**

Anticipated Result Code: *404*

Anticipated Result Datatype: *JSON* or HTML

Anticipated Result Data: Any.

Test Code Source: *tests/api/***/inventory/tag/history/history\_GET.php**

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test case2:***

Description: test post method.

Assumption:get works.

URL: *http://192.168.33.99/api/***/inventory/tag/history**

Calling Method: POST

Anticipated Result Code: *200*

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of check-in record..

Test Code Source:*tests/api/* **/inventory/tag/history/history\_POST.php**

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/inventory/:tag/history/:num**

***Test Case 1:*** Pending

Description: Test this api with an existed history id for the inventory item with the existed ID.

Assumption: History id number with *historyid* <**101**> belongs to inventory item with *id* <**1001**>.

URL: *http://192.168.33.99/api/inventory/****1001****/history/****101****/*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a specific inventory history record belongs to the inventory item.

Test Code Source: *tests/api/history\_inventory/ValidHistoryValidItemTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with an fraud history id for the inventory item with the fraud ID.

Assumption: Invalid history id number with *historyid* <**historyid**> belongs to invalid inventory item with *id* <**itemid**>.

URL: *http://192.168.33.99/api/inventory/****itemid****/history/****historyid****/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source:*tests/api/history\_inventory/InvalidHistoryInvalIditemTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api with an fraud history id for the inventory item with the valid ID.

Assumption: Invalid history id number with *historyid* <**historyid**> belongs to valid inventory item with *id* <**itemid**>.

URL: *http://192.168.33.99/api/inventory/****itemid****/history/****historyid****/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source:*tests/api/history\_inventory/InvalidHistoryValidItemTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/inventory/:tag/history/latest**

***Test Case 1:*** Pending

Description: Test this api the latest history for the inventory item with the existed ID.

Assumption: Inventory item with *id* <**1001**>.

URL: *http://192.168.33.99/api/inventory/****1001****/history/****latest****/*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: the latest history record belongs to the inventory item.

Test Code Source: *tests/api/history\_inventory/LatestHistoryValidItemTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api the latest history for the inventory item with the fraud ID.

Assumption: Invalid inventory item with *id* <**itemid**>.

URL: *http://192.168.33.99/api/inventory/****itemid****/history/****latest****/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source:*tests/api/history\_inventory/LatestHistoryInvalidItemTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/inventory/reports/by-type**

***Test Case 1:*** Pending

Description: Test this api when there is a valid report in JSON or CSV format.

Assumption: There is a valid report in format <**CSV**>.

URL: *http://192.168.33.99/api/inventory/reports/****csv****/*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON or CSV*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/reports\_inventory/ValidReportsTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api when there is a fraud report not in JSON or CSV format.

Assumption: Invalid report in format <**by-type**>.

URL: *http://192.168.33.99/api/inventory/reports/****by-type****/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source:*tests/api/reports\_inventory/InvalidReportsTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/inventory/reports/reconciliation**

***Test Case 1:*** Pending

Description: Test this api a query parameter "last-checkin" when there is a valid report in JSON or CSV format.

Assumption: There is a valid report in format CSV.

URL: *http://192.168.33.99/api/inventory/reports/reconciliation/*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON or CSV*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/reports\_reconciliation\_inventory/ValidReportsReconciliationTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api a query parameter "last-checkin" when there is a fraud report in JSON or CSV format.

Assumption: There is a fraud report in format CSV.

URL: *http://192.168.33.99/api/inventory/reports/reconciliation/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON or HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/reports\_reconciliation\_inventory/InvalidReportsReconciliationTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/buildings**

***Test Case 1:*** Pending

Description: Test this api when there is valid building resource without search parameter.

Assumption: There is at least one valid building resource in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of building resource .

Test Code Source: *tests/api/buildings/ValidBuildingTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with when there is no valid building resource without search parameter.

Assumption: There is no valid building resource in the database

URL: *http://192.168.33.99/api/buildings/*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings/ValidBuildingTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api when there is valid building resource with search parameter..

Assumption: There is valid building resource matching the search parameter in the database. There is matching record for MATCHING string.

URL: [*http://192.168.33.99/api/buildings*](http://192.168.33.99/api/buildings/)*?search=MATCHING*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of building resource .

Test Code Source: *tests/api/buildings/ValidBuildingTestSearch\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 4:*** Pending

Description: Test this api when there is no valid building resource with search parameter..

Assumption: There is no valid building resource matching the search parameter in the database. There is no matching record for NOMATCHING string.

URL: [*http://192.168.33.99/api/buildings*](http://192.168.33.99/api/buildings/)*?search=NOMATCHING*

Calling Method: *GET*

Anticipated Result Code: *204*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings/ValidBuildingTestSearch\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 5:*** Pending

Description: Test this api when create a valid building resource

Assumption: The building name in data in the request body does not exist in the database. PCHS is not in the database yet.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)

Calling Method: *POST*

Data to Post: ['name' => 'PCHS', 'description' => 'Powell Co. High School']

Anticipated Result Code: *201*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a building resource corresponding to the posted data.

Test Code Source: *tests/api/building/ValidBuildingTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 6:*** Pending

Description: Test this api when there is confliction.

Assumption: The building name in data in the request body does exist in the database. PCHS is in the database now.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)

Calling Method: *POST*

Data to Post: ['name' => 'PCHS', 'description' => 'Powell Co. High School']

Anticipated Result Code: *400*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source:*tests/api/building/ValidBuildingTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 7:*** Pending

Description: Test this api when there is no content in the request body.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)

Calling Method: *POST*

Data to Post: ['name' => '', 'description' => '']

Anticipated Result Code: *405*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building/NoContentTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 8:*** Pending

Description: Test this api when there is partial content in the request body.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)

Calling Method: *POST*

Data to Post: ['name' => '', 'description' => '']

Anticipated Result Code: *405*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building/PartialContentTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/buildings/:id**

***Test Case 1:*** Pending

Description: Test this api with an existed building id.

Assumption: Building with id <**3320**> exits in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: building resource description with id <**3320**>.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with a fraud building id.

Assumption: Building with id <**3320**> does not exist in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api with an existed building id to update building resource.

Assumption: Building with id <**3320**> exits in the database. The updated school name is unique.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Data to Post: ['name' => 'NewSchool', 'description' => 'Adding new schools for test']

Calling Method: *PUT*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: building resource description with id <**3320**>.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 4:*** Pending

Description: Test this api with a fraud building id to update building resource.

Assumption: Building with id <**3320**> does not exit in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Data to Post: ['name' => 'NewSchool', 'description' => 'Adding new schools for test']

Calling Method: *PUT*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 5:*** Pending

Description: Test this api with a valid building id to update building resource, but there will be confliction in school name if the new data is updated.

Assumption: Building with id <**3320**> does exist in the database. A record with name ‘NewSchool’ has already been in the data base.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Data to Post: ['name' => 'NewSchool', 'description' => 'Adding new schools for test']

Calling Method: *PUT*

Anticipated Result Code: *409*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 6:*** Pending

Description: Test this api with an existed building id to delete building resource.

Assumption: Building with id <**3320**> exits in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Calling Method: *DELETE*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 7:*** Pending

Description: Test this api with a fraud building id to delete building resource.

Assumption: Building with id <**3320**> does not exist in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320*

Anticipated Result Code: *404*.

Calling Method: *DELETE*

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/buildings\_id/ValidBuildingIDTest\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/buildings/:id/rooms**

***Test Case 1:*** Pending

Description: Test this api with an existed building id.

Assumption: building with *id* <**3320**> exists in the database

URL: *http://192.168.33.99/api/buildings/****101****/rooms*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of room resource belong to building <**101**>.

Test Code Source: *tests/api/building\_id\_rooms/ValidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with a fraud building id.

Assumption: invalid *building id* <**BuildingID**> .

URL: [*http://192.168.33.99/api/buildings/****RoomID****/rooms/*](http://192.168.33.99/api/buildings/RoomID/rooms/)

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms/IValidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api with an existed building id to create new room resource

Assumption: building with *id* <**3320**> exists in the database, newly added room is not in the database yet.

URL: *http://192.168.33.99/api/buildings/****3320****/rooms*

Calling Method: *POST*

Anticipated Result Code: *201*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of room resource belong to building <**3320**>.

Test Code Source: *tests/api/building\_id\_rooms/ValidRoomTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 4:*** Pending

Description: Test this api with an existed building id to create new room resource

Assumption: building with *id* <**3320**> exists in the database, newly added room is in the database now.

URL: *http://192.168.33.99/api/buildings/****3320****/rooms*

Calling Method: *POST*

Anticipated Result Code: *409*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms/RoomConflictionTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 5:*** Pending

Description: Test this api with a fraud building id to create new room resource

Assumption: building with *id* <**3320**> does not exist in the database

URL: *http://192.168.33.99/api/buildings/****3320****/rooms*

Calling Method: *POST*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms/ValidRoomTest\_POST.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/buildings/:buildingid/rooms/:roomid**

***Test Case 1:*** Pending

Description: Test this api with an existed room id which belong to an existed buildings id.

Assumption: Room with *roomid* <**101**> belongs to building with *id* <**3320**>.

URL: *http://192.168.33.99/api/buildings/****101****/rooms/****3320***

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of resource belong to the room.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with an fraud room id and an fraud buildings id.

Assumption: invalid *roomid* <**RoomID**> and invalid *id* <**BuildingID**>.

URL: [*http://192.168.33.99/api/buildings/****RoomID****/rooms/****BuildingID****/*](http://192.168.33.99/api/buildings/RoomID/rooms/BuildingID/)

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/InvalidBuildingRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api with an fraud room id and an existed buildings id.

Assumption: invalid *roomid* <**RoomID**> and valid *id* <**BuildingID**>.

URL: *http://192.168.33.99/api/buildings/****RoomID****/rooms/****BuildingID***

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidBuildingInvalidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 4:*** Pending

Description: Test this api with an existed building id and existing room id to delete room resource in the building

Assumption: valid *roomid* <**RoomID**> and valid *id* <**BuildingID**>.

URL: *http://192.168.33.99/api/buildings/****RoomID****/rooms/****BuildingID***

Calling Method: *DELETE*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidBuildingValidRoom\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 5:*** Pending

Description: Test this api with a fraud building id to delete room resource in the building

Assumption: Building with id <**3320**> does not exist in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320/rooms/101*

Anticipated Result Code: *404*.

Calling Method: *DELETE*

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/InvalidBuilding\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 6:*** Pending

Description: Test this api with a valid building id but a fraud room id to delete room resource in the building.

Assumption: Building with id <**3320**> do exist in the database, but room id <**101**> is not in the building.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320/rooms/101*

Anticipated Result Code: *404*.

Calling Method: *DELETE*

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidBuildingInvalidRoom\_DELETE.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 7:*** Pending

Description: Test this api with a valid building id and valid room id to update building resource.

Assumption: Building with id <**3320**> and room id <**101**> exits in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320/rooms/101*

Calling Method: *PUT*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: room resource description with id <**3320**> and room id <**101**>.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidBuildingValidRoom\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 8:*** Pending

Description: Test this api with a fraud building id to update room resource.

Assumption: Building with id <**3320**> does not exit in the database.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320/rooms/101*

Calling Method: *PUT*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/InvalidBuilding\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 9:*** Pending

Description: Test this api with a valid building id but a fraud room id to update room resource.

Assumption: Building with id <**3320**> does exit in the database, but room id <**101**> is not associate with it.

URL: [*http://192.168.33.99/api/buildings/*](http://192.168.33.99/api/buildings/)*3320/rooms/101*

Calling Method: *PUT*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_id\_rooms\_id/ValidBuildingInvalidRoom\_PUT.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

**/buildings/:id/rooms/:roomid/inventory**

***Test Case 1:*** Pending

Description: Test this api with an existed room id which belong to an existed buildings id.

Assumption: Room with *roomid* <**101**> belongs to building with *id* <**3320**>.

URL: *http://192.168.33.99/api/buildings/****101****/rooms/****3320****/inventory*

Calling Method: *GET*

Anticipated Result Code: *200*.

Anticipated Result Datatype: *JSON*.

Anticipated Result Data: a list of inventory belong to the room.

Test Code Source: *tests/api/building\_room\_inventory/ValidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 2:*** Pending

Description: Test this api with an fraud room id and an fraud buildings id.

Assumption: invalid *roomid* <**RoomID**> and invalid *id* <**BuildingID**>.

URL: *http://192.168.33.99/api/buildings/****RoomID****/rooms/****BuildingID****/inventory*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_room\_inventory/InvalidBuildingRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

***Test Case 3:*** Pending

Description: Test this api with an fraud room id and an existed buildings id.

Assumption: invalid *roomid* <**RoomID**> and valid *id* <**BuildingID**>.

URL: *http://192.168.33.99/api/buildings/****RoomID****/rooms/****BuildingID****/inventory*

Calling Method: *GET*

Anticipated Result Code: *404*.

Anticipated Result Datatype: *JSON* or *HTML*.

Anticipated Result Data: Any.

Test Code Source: *tests/api/building\_room\_inventory/ValidBuildingInvalidRoomTest\_GET.php*

Test logs:

|  |  |  |
| --- | --- | --- |
| Datetime | Result | Comments |
|  |  |  |

## Milestone Tests

### Level 3 – Critical Path Integration Tests

Critical Path test cases are targeted on features and functionality that the user will see and use every day.

Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop, but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

Login

Scan

Check In

Device

Record Device

View Device History

Delete Device

Modify Device

User

Create User

…

**<need to be modified>**

### Level 3a – Test Review

Login

Scan

Check In

Device

Record Device

View Device History

Delete Device

Modify Device

User

Create User

…

**<need to be modified>**

## Release Tests

### Level 4 – Standard Tests

Test Cases that need to be run at least once during the entire test cycle for this release. These cases are run once, not repeated as are the test cases in previous levels. Functional Testing and Detailed Design Testing (Functional Spec and Design Spec Test Cases, respectively). These can be tested multiple times for each Milestone Test Cycle (Iteration, Final Release, etc.).

Standard test cases usually include Installation, Data, GUI, and other test areas.

### Level 5 – Suggested Test

These are Test Cases that would be nice to execute, but may be omitted due to time constraints.

Most Performance and Stress Test Cases are classic examples of Suggested test cases (although some should be considered standard test cases). Other examples of suggested test cases include WAN, LAN, Network, and Load testing.

# Level Review

|  |  |  |
| --- | --- | --- |
| level | complete | comments |
| 1 | N/A |  |
| 2 | N/A |  |
| 3 | N/A |  |
| 4 | N/A |  |
| 5 | N/A |  |