CS3343 Software Engineering Practice

2022/23 Semester A

Group Project for Group 13

XBOX

Box Storage Management System

Test Report

Conducted by:

DONG Jiajie	56641314
LI Xiaoyang	56638660
SHA Xinchen	56641824
ZHANG Tiantian	56645190
ZHENG Shangkun	56642570
ZHOU Yu	56642568

Dec 4, 2023

Table of Content

1.Hierarchy Diagram

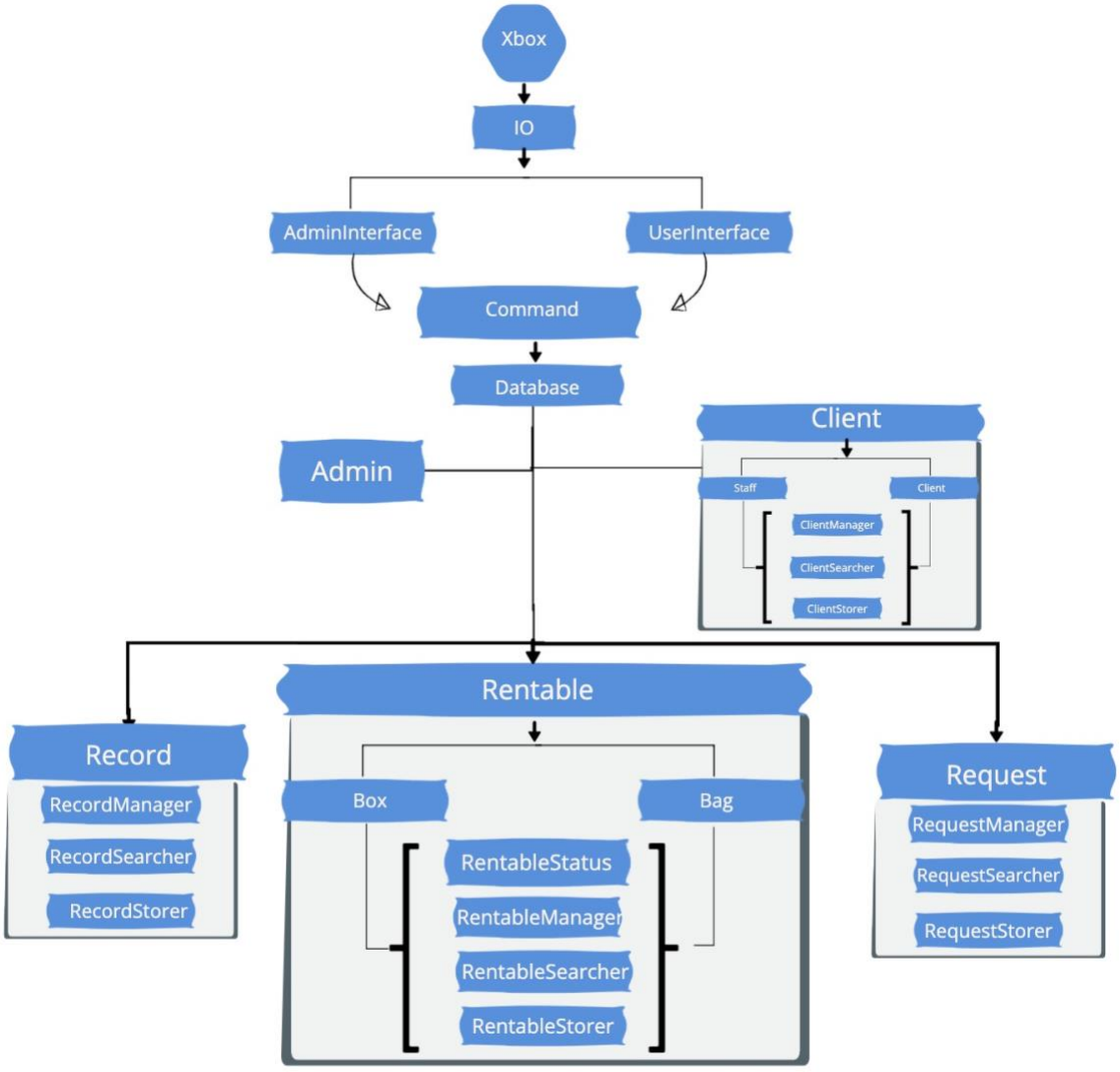

Figure 1. Overall hierarchy diagram

2. Testing Strategy

In this project, since our program's structure is both large and complex, we majorly adopted the sandwich testing strategy, which means the top-down strategy and bottom-up strategy are both adopted through our testing process.

For the program components below IO, we managed to write test cases by applying the bottom-up methodology. Since the structure is quite complicated and interdependence occurs quite frequently, it would help us get rid of the tremendous need for test stubs and drivers. Moreover, it enables parallel testing and allows the team members to work independently on the two separate parts, which efficiently shortens the time required for the testing period, but meanwhile requires more effort. Moreover, since the bottom-up is a gradual approach, we can test and fix the bugs in the early coding stage.

For the program components above IO, we applied the top-down methodology. Since its overall structure is not so complicated, it is better to use the top-down method so that the consistency can be well preserved.

3.Testing Process

We began our testing process with the database component, whose overall hierarchy is as follows:

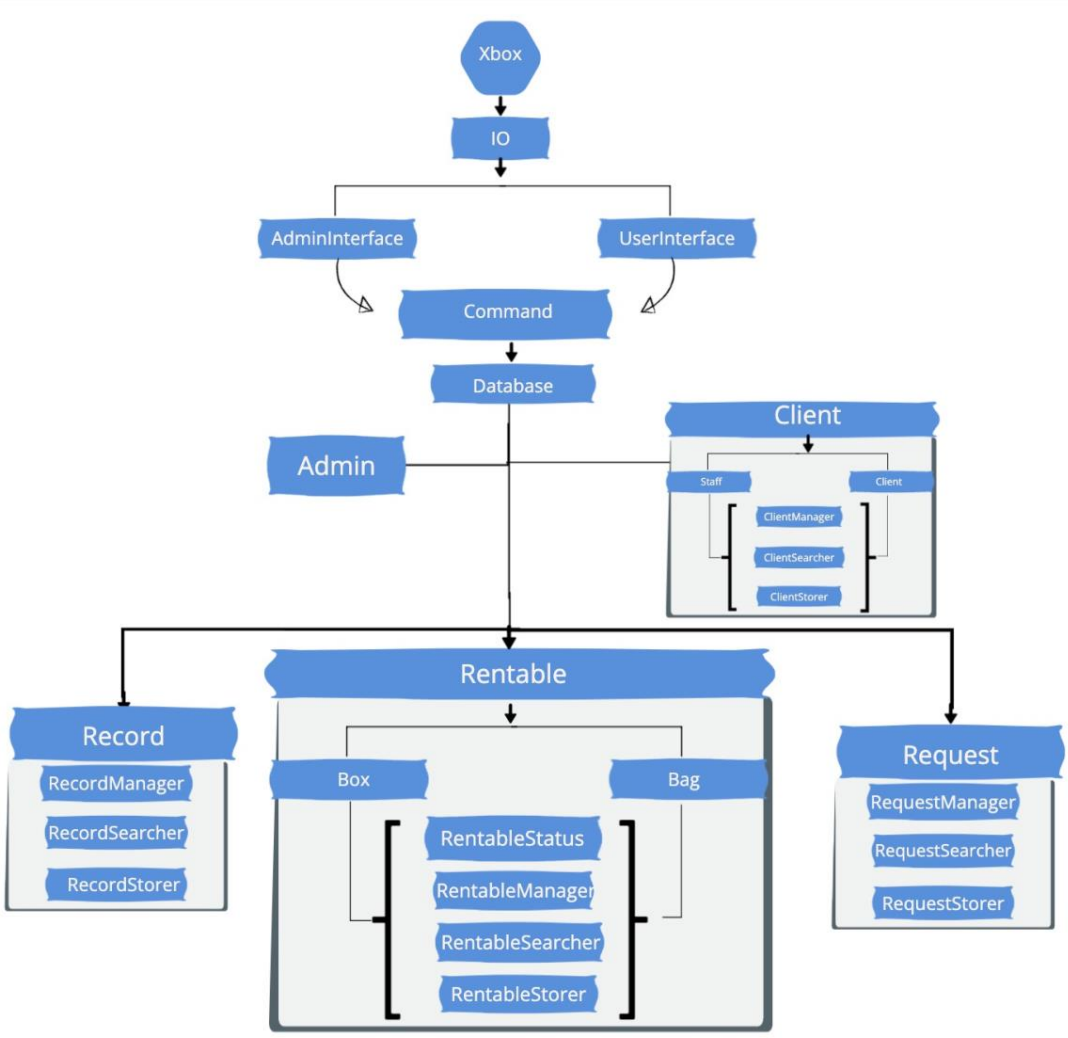

Figure 2 Database hierarchy diagram

First, we did the unit test with the following units:

- Record.java
- Request.java
- Client.java
- RentableStatus.java

Then we started the integration test:

- Record.java + RecordStorer.java + RecordSearcher.java + RecordManager.java

- Request.java + RequestStorer.java + RequestSearcher.java + RequestManager.java

- Client.java + ClientStorer.java + ClientSearcher.java + ClientManager.java

- RentableStatus.java + Rentable.java + RentableStorer.java + RentableSearcher.java + RentableManager.java

- Database.java + Record + Request + Client + Rentable

Note that all those subcomponents are already included in the Record, Request, Client and Rentable components.

After testing the Database, we continue the integration test with other components:

- Command + Database
- AdminInterface + ClientInterface + Command + Database
- IO + AdminInterface + ClientInterface + Command + Database

Finally, we conducted the system test:

- Xbox + IO + AdminInterface + ClientInterface + Command + Database

4. Code Refactoring

We did code refactoring mainly towards the four classes: RecordManager, RentableManager, ClientManager and RequestManager. The sharing idea is to separate the inner ArrayList from Manager and make it as a singleton class.

Here we take the RecordManager.java as an example:

Before:

```
package data;

import java.util.ArrayList;

public class RecordManager {
    private static RecordManager instance=new RecordManager();

    private ArrayList<Record> Recordlist;

    public static RecordManager getInstance()
    {
        return instance;
    }
    public void insert(Record record){
        RecordStorer storer = RecordStorer.getInstance();
        storer.addEntry(record);
    }

    public void delete(Record record){
        RecordStorer storer=RecordStorer.getInstance();
        storer.delEntry(record);
    }
}
```

Figure 3 Sample code before refactoring

After:

```
1  package data;
2
3  public class RecordManager {
4      private static RecordManager instance=new RecordManager();
5
6      public static RecordManager getInstance()
7      {
8          return instance;
9      }
10     public void insert(Record record){
11         RecordStorer storer = RecordStorer.getInstance();
12         storer.addEntry(record);
13     }
14
15     public void delete(Record record){
16         RecordStorer storer=RecordStorer.getInstance();
17         storer.delEntry(record);
18     }
19 }
```

Figure 4.1. Sample code after refactoring

and

```
public class RecordStorer implements XBoxStorer<Record>{
    private ArrayList<Record> recordList;
    private static RecordStorer storer = new RecordStorer();

    private RecordStorer(){
        recordList=new ArrayList<>();
    }

    public static RecordStorer getInstance(){
        return storer;
    }

    public ArrayList<Record> getList(){
        return recordList;
    }
```

Figure 4.2 Sample code after refactoring

Note that we remove the ArrayList in the Manager class, and instead create a new singleton Storer class, which:

- keeps good isolation between data manipulation and storing.
- makes the code structure much clearer
- demonstrates Single Responsibility Principle

5. Test Cases

TestClient.java

Purpose: To test whether the basic methods inside Client.java, ClientStaff.java, ClientStudent.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T001	test_01	Test ClientStaff.getDiscount()	0.8	As expected
T002	test_02	Test ClientStaff.getMaxBorrowedCount()	20	As expected
T003	test_03	Test ClientStaff.getEmailt()	Staff's email String	As expected
T004	test_04	Test ClientStaff.getPassword()	Staff's password String	As expected
T005	test_05	Test ClientStaff.getPhoneNo()	Staff's phoneNo	As expected
T006	test_06	Test ClientStaff.verifyPassword() with correct password	true	As expected

T007	test_07	Test ClientStaff.verifyPassword() with wrong password	false	As expected
T008	test_08	Test ClientStaff.toString	Staff's info String	As expected
T009	test_09	Test ClientStaff.toJSONString	Staff's info String in JSON format	As expected
T010	test_19	Test ClientStaff.changeBorrowedCount(-2)	Staff's borrowedCount is subtracted by 2	As expected
T011	test_10	Test ClientStudent.getDiscount()	0.9	As expecte
T012	test_11	Test ClientStudent.getMaxBorrowedCount()	10	As expected
T013	test_12	Test ClientStudent.getEmailt()	Student's email String	As expected
T014	test_13	Test ClientStudent.getPassword()	Student's password String	As expected
T015	test_14	Test ClientStudent.getPhoneNo()	Student's phoneNo	As expected
T016	test_15	Test ClientStudent.verifyPassword() with correct password	true	As expected
T017	test_16	Test ClientStudent.verifyPassword() with wrong password	false	As expected

T018	test_17	Test ClientStudent.toString	Student’s info String	As expected
T019	test_18	Test ClientStudent.toJSONString	Student’s info String in JSON format	As expected
T020	test_20	Test ClientStudent.changeBorrowedCount(-1)	9	As expected

TestClientManager.java

Purpose: To test whether the client list inside ClientManager.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T021	test_01	Insert one client and test whether it is in the client list	true	As expected
T022	test_02	Delete one client and test whether it is in the client list	false	As expected

TestClientSearcher.java

Purpose: To test whether the search function inside ClientSearcher.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T023	test_01	ClientSearcher. *getInstance*().searchAll()	list of all clients	As expected
T024	test_02	Search an existing client by ClientSearcher.searchByKeyword()	the client's info String	As expected
T025	test_03	Search an inexisting client by ClientSearcher.searchByKeyword()	null	As expected

TestClientStorer.java

Purpose: To test whether the search function inside ClientSearcher.java can work well

Test case ID	Test Name	Input Description	Expected Result	Actual Output
T026	test_01	Test whether it can read a student client from the database	student info string	As expected
T027	test_02	Test whether it can read a staff client from the database	staff info string	As expected
T028	test_03	Test whether it can read an incorrect client from the database	null	As expected
T029	test_04	Test whether it can write a student client into the database	the JSON object that contains client's info String	As expected
T030	test_05	Test whether it can write a staff client into the database	the JSON object that contains client's info String	As expected
T031	test_06	Test whether it can write an incorrect student client into the database	null	As expected
T032	test_07	Test whether it can read the JSON file from the database	same client list in the database	As expected

T033	test_08	Test whether it can write the JSON file to the database	same client list stored in the system	As expected

TestCmdRequestRentable.java

Purpose: To test whether commands inside CmdRequestRentable.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T034	test_01	borrow rentables that exceed the max count	"[Error] No more than 10 BAG per user!"	As expected
T035	test_02	borrow rentables that have nonpositive value	"[request list]\n"	As expected
T036	test_03	borrow one bag for one month	"[request list]\n" + String.format("> %-10s%tF\n", "BAG0001", systemDate.getDayAfterNMonth("1"))	As expected
T037	test_04	borrow two bags for one month	"[request list]\n" + String.format("> %-10s%tF\n", "BAG0002", systemDate.getDayAfterNMonth("1")) + String.format("> %-10s%tF\n", "BAG0003", systemDate.getDayAfterNMonth("1"))	As expected

T038	test_05	test undo function	undo borrowing two bags	As expected
T039	test_06	test redo function	redo borrowing two bags	As expected

TestCmdRequestReturn.java

Purpose: To test whether commands inside CmdRequestReturn.java can function well				
Test case ID	**Test Name**	Input Description	Expected Result	Actual Output
T040	test_01	return rentables with empty input	"[Return list]\n"	As expected
T041	test_02	return rentables with existing rentable ID	"[Return list]\n" + String.format("> send checkin notification [%s]\n", "BAG7777")	As expected
T042	test_03	return rentables with inexisting rentable ID	String.format("Rent record [%s] not found", "BOX9999")	As expected
T043	test_04	return rentables with incorrect client email	String.format("Rent record [%s] not found", "BAG7777")	As expected
T044	test_05	return rentables that have been returned before	String.format("Checkin notification[%s]'s been sent", "BAG7777")	As expected
T045	test_06	test undo and redo function	undo and then redo returning the bag	As expected

TestCmdStoreRentable.java

Purpose: To test whether commands inside CmdStoreRentable.java can function well				
Test case ID	**Test Name**	Input Description	Expected Result	Actual Output
T046	test_01	store rentables with empty input	"[stored list]\n[unused list]\n" + String.format("> %s is requested but not used\n", "BOX0001")	As expected
T047	test_02	store rentables with existing rentable ID	"[stored list]\n" + String.format("> %s will be stored for %s\n", "BOX0001", "xyli45-c@my.cityu.edu.hk") + "[unused list]\n"	As expected
T048	test_03	store rentables with inexisting rentable ID	String.format("Request[%s] is not found", "BOX9999")	As expected
T049	test_04	store rentables with incorrect client email	String.format("Request[%s] is not found", "BOX0001")	As expected

T050	test_05	test undo and redo function	undo and then redo storing the bag	As expected

TestConfirmPayment.java

Purpose: To test whether commands inside CmdConfirmPayment.java can function well				
Test case ID	**Test Name**	Input Description	Expected Result	Actual Output
T051	test_01	pay rentables with full amount	"[Payment]\n" + String.format(">%-5s\t$%.2f\n", "BOX2333", allRentables[0].getPrice()) + String.format("\ndiscount: %.0f percent off\n", (1-allClients[0].getDiscount()) * 100) + String.format("total: $%.2f\n", allRentables[0].getPrice() * allClients[0].getDiscount())	As expected
T052	test_02	pay rentables with discounts	"[Payment]\n" + String.format("\ndiscount: %.0f percent off\n", (1-allClients[0].getDiscount()) * 100) + String.format("total: $%.2f\n", 0.0)	As expected
T053	test_03	test undo and redo function	undo and then redo paying the bags	As expected

TestConfirmReturn.java

Purpose: To test whether commands inside CmdConfirmReturn.java can function well				
Test case ID	**Test Name**	Input Description	Expected Result	Actual Output
T054	test_01	return rentable with empty input	"[Checkin list]\n"	As expected
T055	test_02	return rentable with correct rentable ID	"[Checkin list]\n" + String.format("> %s\n", "BOX9526")	As expected
T056	test_03	return rentable with inexisting rentable ID	String.format("Record [%s] is not found!", "BOX9999")	As expected
T057	test_04	return rentable with inexisting request	String.format("[Error] Checkin notification [%s] not found!", "BOX3141")	As expected

T058	test_05	test undo and redo function	undo and then redo confirming the return of the bags	As expected

TestDatabase.java

Purpose: To test whether the basic methods inside Database.java can function well				
Test case ID	**Test Name**	Input Description	Expected Result	Actual Output
T059	test_01	initialize the database by reading from JSON files	Same client, record, rentable, request lists in the JSON files	As expected
T060	test_02	Store the database by writing to JSON files	Same client, record, rentable, request lists stored by ClientStorer, RecordStorer, RentableStorer, RequestStorer.	As expected

TestInterface.java

Purpose: To test whether the basic methods inside AdminInterface.java and UserInterface.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T061	Test_login	Test login with the correct email and password	String.format("Login Success <%s>","admin@xbox.com.hk") String.format("Login Success <%s>","test")	As expected
T062	Test_login_2	Test login with incorrect email or incorrect password	String.format("[Error] No user <%s>, please register first!" "[Error] The password is invalid!"	As expected
T063	Test_register	Test register with the valid email and password	String.format("Register Success <%s>", "1")	As expected
T064	Test_register_2	Test register with the invalid email and password	"[Error] Please fill in all blank fields!" String.format("[Error] Email address <%s> has been registered!"	As expected

T065	Test_AmdinPage	Test Admin page functions	realize all the admin functions by clicking these six buttons	As expected
T066	Test_AdminPage_2	Test Admin page undo and redo functions	undo and then redo the other six functions	As expected
T067	Test_AdminPage_3	Test Admin page functions with incorrect input	report all the error messages of admin functions by clicking these six buttons	As expected
T068	Test_UserPage	Test User page functions	realize all the user functions by clicking these six buttons	As expected
T069	Test_UserPage_2	Test User page undo and redo functions	undo and then redo the other six functions	As expected
T070	SystemTest	Test the main system call	No error incurred	As expected

TestIO.java

Purpose: To test whether the IO methods inside xbox.java, AdminInterface.java and UserInterface.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T071	Test_login_or_register	Test entering the login and register pages	enter the login and register pages by clicking buttons	As expected
T072	Test_login	Test entering the interfaces after logining	enter different interface pages by emails	As expected
T073	Test_login_2	Test reporting errors when logining	"No Entry"	As expected
T074	Test_register	go back to home pages after registering successfully	successfully register accounts and go back to home pages	As expected
T075	Test_register_2	Test reporting errors when registering	"Error"	As expected

T076	Test_AdminPage	Test entering different functions part in the Admin page by clicking buttons	successfully entering different functions pages in the admin interface	As expected
T077	Test_AdminPage_2	Test reporting errors when clicking the undo or redo buttons in the Admin page	"Error"	As expected
T078	Test_confirmPayment	Test entering the confirmpayment page and changing back to admin interface by clicking buttons	successfully entering and changing back to admin interface	As expected
T079	Test_confirmPayment_2	Test reporting errors in the confirmpayment page	"Error"	As expected
T080	Test_confirmReturn	Test entering the confirmreturn page and changing back to admin interface by clicking buttons	successfully entering and changing back to admin interface	As expected
T081	Test_confirmReturn_2	Test reporting errors in the confirmreturn page	"Error"	As expected
T082	Test_SearchClient	Test entering the searchclient page and changing back to admin interface by clicking buttons	successfully entering and changing back to admin interface	As expected
T083	Test_SearchClient_2	Test reporting errors in the searchclient page	"Error"	As expected
T084	Test_SearchItem	Test entering the searchitem page and changing back to	successfully entering and changing back	As expected

		admin interface by clicking buttons	to admin interface	
T085	Test_SearchItem_2	Test reporting errors in the searchitem page	"Error"	As expected
T086	Test_UserPage	Test entering different functions part in the User page by clicking buttons	successfully entering different functions pages in the user interface	As expected
T087	Test_UserPage_2	Test reporting errors when clicking the undo or redo buttons in the User page	"Error"	As expected
T088	Test_requestBox	Test entering the requestBox page and changing back to user interface by clicking buttons	successfully entering and changing back to admin interface	As expected
T089	Test_requestBox_2	Test reporting errors in the requestBox page	"Error"	As expected
T090	Test_returnBox	Test entering the returnBox page and changing back to admin interface by clicking buttons	successfully entering and changing back to admin interface	As expected
T091	Test_returnBox_2	Test reporting errors in the returnBox page	"Error"	As expected
T092	Test_storeBox	Test entering the storeBox page and changing back to admin interface by clicking buttons	successfully entering and changing back to admin interface	As expected

T093	Test_storeBox_2	Test reporting errors in the storeBox page	"Error"	As expected

TestRecord.java

Purpose: To test whether the basic methods inside Record.java can function well

Test case ID	Test Name	Input Description	Expected Result	Actual Output
T094	test_01	r.getClient().toString() r.getRentable().toString() r.getDue().toString() r.getPaymentStatus()	the client string the rentablie sting the due date string the payment status string in the record	As expected
T095	test_02	Record string without payment	"BAG001 unpaid 123@abc.com 1970-01-01"	As expected
T096	test_03	Record string with payment	"BAG001 paid 123@abc.com 1970-01-01"	As expected
T097	test_04	Test store record to json object	"{\"client\":{\"email\":\"123@abc.com\",\"phoneNo\":\"1234\",\"password\":\"123\",\"type\":\"staff\"},\"rentable\":{\"id\":\"001\",\"status\":{\"status\":\"Available\"},\"type\":\"BA	As expected

			G\"},\"dueDate\":114514,\"isPaid\":false}"	

TestRecordManager.java

Purpose: To test whether the client list inside RecordManager.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T098	test_01	Insert one record and test whether it is in the record list	true	As expected
T099	test_02	Delete one record and test whether it is in the record list	false	As expected

TestRecordSearcher.java

Purpose: To test whether the search function inside RecordSearcher.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T100	test_07	RecordSearcher. *getInstance*().searchAll()	list of all records	As expected
T101	test_01	Search an existing record by RecordSearcher.searchByKeyword()	the record's info String	As expected
T102	test_02	Search an inexisting record by RecordSearcher.searchByKeyword()	null	As expected
T103	test_03	Search all existing records by RecordSearcher.searchAllByKeyword("123@abc.com")	all record's info Strings that belong to "123@abc.com"	As expected
T104	test_04	Search inexisting records by RecordSearcher.searchAllByKeyword("123@ac.com")	null	As expected
T105	test_05	Search all existing records by RecordSearcher.searchAllByKeyword(d)	all record's info String that belongs to d	As expected

T106	test_06	Search inexisting records by RecordSearcher.searchAllByKeyword(**new** Date(114))	null	As expected

TestRecordStorer.java

Purpose: To test whether the search function inside RecordStorer.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T107	test_01	Insert one record entry and test whether it is in the record list	true	As expected
T108	test_02	Delete one record entry and test whether it is in the record list	false	As expected
T109	test_03	Test whether it can read a record from the JSON object	record info string	As expected
T110	test_04	Test whether it can put a record to the JSON object	JSON object that contains record info	As expected
T111	test_05	Test whether it can read the JSON file from the database	same record list in the database	As expected
T112	test_06	Test whether it can write the JSON file to the database	same record list stored in the system	As expected

TestRentable.java

Purpose: To test whether the basic methods inside Rentable.java can function well

Test case ID	Test Name	Input Description	Expected Result	Actual Output
T113	test_01	Rentable.getType()	"BAG”	As expected
T114	test_02	Rentable.toString()	"BAG0001 Available "	As expected
T115	test_03	Rentable.getPrice()	80	As expected
T116	test_04	Rentable.toJSONString()	"{\"id\":\"0001\",\"status\":{\"status\":\"Available\"},\"type\":\"BAG\"}"	As expected
T117	test_05	Rentable.getType()	"BOX”	As expected
T118	test_06	Rentable.toString()	"BOX0001 Occupied "	As expected
T119	test_07	Rentable.getPrice()	100	As expected

T120	test_08	Rentable.toJSONString()	"{\"id\":\"0001\",\"status\":{\"status\":\"Occupied\",\"client\":{\"email\":\"yzhou@gmail.com\",\"phoneNo\":\"46464646\",\"password\":\"password\",\"type\":\"student\"},\"due\":" + dueStr +"},\"type\":\"BOX\"}"	As expected
T121	test_09	Rentable.getStatus().getStatus()	"Occupied"	As expected

TestRentableAllocator.java

Purpose: To test whether the client list inside RentableAllocator.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T122	test_01	Add one box and borrow one box	no error incurs	As expected
T123	test_02	borrow one box with zeo remaining boxes	incur ExNoSufficientRentable	As expected

TestRentableManager.java

Purpose: To test whether the client list inside RentableManager.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T124	test_01	Insert one rentable and test whether it is in the rentable list	true	As expected
T125	test_02	Delete one rentable and test whether it is in the rentable list	false	As expected

TestRentableSearcher.java

Purpose: To test whether the search function inside RentableSearcher.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T126	test_01	RentableSearcher. searchByKeyword("BOX0001") RentableSearcher. searchAllByKeyword("BOX0001") RentableSearcher. searchByKeyword("BAG0001")	No errors or exceptions are incurred	As expected

TestRecordStatus.java

Purpose: To test whether the search function inside RecordStatus.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T127	test_01	RentableStatusAvailable() .toString()	"Available"	As expected
T128	test_02	RentableStatusAvailable().getStatus()	"Available"	As expected
T129	test_03	RentableStatusAvailable().toJSONString()	"{\"status\":\"Available\"}"	As expected
T130	test_04	RentableStatusOccupied(due,student1) .toString()	"Borrowed by yzhou@gmail.com on "+due.toString()	As expected
T131	test_05	RentableStatusOccupied(due,student1) .getStatus()	"Occupied"	As expected
T132	test_06	RentableStatusOccupied(due,student1) .toJSONString()	"{\"status\":\"Occupied\",\"client\":{\"email\":\"yzhou@gmail.com\",\"phoneNo\":\	As expected

			"46464646\",\"password\":\"password\",\"type\":\"student\"},\"due\":"+due.getTime()+"}"	
T133	test_07	RentableStatusPending(student1).toString()	"held by yzhou@gmail.com"	As expected
T134	test_08	RentableStatusPending(student1).getStatus()	"Pending"	As expected
T135	test_09	RentableStatusPending(student1).toJSONString()	"{\"status\":\"Pending\",\"client\":{\"email\":\"yzhou@gmail.com\",\"phoneNo\":\"46464646\",\"password\":\"password\",\"type\":\"student\"}}"	As expected
T136	test_10	RentableStatusRequested(student1).toString()	"Requested by yzhou@gmail.com"	As expected
T137	test_11	RentableStatusRequested(student1).getStatus()	"Requested"	As expected

T138	test_12	RentableStatusRequested(student1) .toJSONString()	"{\"status\":\"Requested\",\"client\":{\"email\":\"yzhou@gmail.com\",\"phoneNo\":\"46464646\",\"password\":\"password\",\"type\":\"student\"}}"	As expected

TestRentableStorer.java

Purpose: To test whether the search function inside RentableStorer.java can work well

Test case ID	Test Name	Input Description	Expected Result	Actual Output
T139	test_01	RentableStorer.getManager().get("BOX").contains(box1)	true	As expected
T140	test_02	RentableStorer.getList("BOX").contains(box2)	true	As expected
T141	test_03	.RentableStorer.getManager().containsValue(box1)	false	As expected
T142	test_04	Test whether it can read an available rentable from the JSON object	rentable info string	As expected
T143	test_05	Test whether it can read an occupied rentable from the JSON object	rentable info string	As expected
T144	test_06	Test whether it can read a requested rentable from the JSON object	rentable info string	As expected

T145	test_07	Test whether it can read a pending rentable from the JSON object	rentable info string	As expected
T146	test_08	Test whether it can read an incorrect rentable from the JSON object	null	As expected
T147	test_09	Test whether it can read an incorrect rentable from the JSON object	null	As expected
T148	test_10	Test whether it can put a rentable to the JSON object	JSON object that contains rentable info	As expected
T149	test_11	Test whether it can read the JSON file from the database	same rentable list in the database	As expected
T150	test_12	Test whether it can write the JSON file to the database	same rentable list stored in the system	As expected
T151	test_13	RentableStorer.getManager().get("TEMP"). contains(r3)	true	As expected

TestRequest.java

Purpose: To test whether the basic methods inside Request.java can function well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T152	test_01	r.getClient().toString() r.getRentable().toString() r.getDue().toString()	the client string the rentablie sting the due date string in the request	As expected
T153	test_02	Test request string	"BAG001 123 @abc.com 1970-01-01"	As expected
T154	test_03	Test store request to json object	"{\"client\":{\"email\":\"123@abc.com\",\"phoneNo\":\"1234\",\"password\":\"123\",\"type\":\"staff\"},\"rentable\":{\"id\":\"001\",\"status\":{\"status\":\"Available\"},\"type\":\"BAG\"},\"dueDate\":114514}"	As expected

TestRequestManager.java

Purpose: To test whether the client list inside RequestManager.java can function well

Test case ID	Test Name	Input Description	Expected Result	Actual Output
T155	test_01	Insert one request and test whether it is in the request list	true	As expected
T156	test_02	Delete one request and test whether it is in the request list	false	As expected

TestRequestSearcher.java

Purpose: To test whether the search function inside RequestSearcher.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T157	test_07	RequestSearcher. *getInstance*().searchAll()	list of all requests	As expected
T158	test_01	Search an existing request by RequestSearcher.searchByKeyword()	the request's info String	As expected
T159	test_02	Search an inexisting request by RequestSearcher.searchByKeyword()	null	As expected
T160	test_03	Search all existing requests by RequestSearcher.searchAllByKeyword("123@abc.com")	all request's info Strings that belong to "123@abc.com"	As expected
T161	test_04	Search inexisting requests by RequestSearcher.searchAllByKeyword("123@ac.com")	null	As expected

TestRequestStorer.java

Purpose: To test whether the search function inside RequestStorer.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T162	test_01	Insert one request entry and test whether it is in the request list	true	As expected
T163	test_02	Delete one request entry and test whether it is in the request list	false	As expected
T164	test_03	Test whether it can read a request from the JSON object	request info string	As expected
T165	test_04	Test whether it can put a request to the JSON object	JSON object that contains request info	As expected
T166	test_05	Test whether it can read the JSON file from the database	same request list in the database	As expected
T167	test_06	Test whether it can write the JSON file to the database	same request list stored in the system	As expected

TestUndoable.java

Purpose: To test whether the search function inside Undoable.java can work well				
Test case ID	**Test Name**	**Input Description**	**Expected Result**	**Actual Output**
T168	test_01	undo with empty undo list	"[Error] Nothing to undo!"	As expected
T169	test_02	redo with empty redo list	"[Error] Nothing to redo!"	As expected
T170	test_03	Test whether it can undo and redo after requesting a rentable	No errors and exceptions are incurred	As expected