EAST WEST UNIVERSITY

Department of Computer Science and Engineering
PROJECT REPORT ON

" Hotel Management System Unit Testing"

Course Name: Software Quality Assurance

Course Code: CSE430

Section No: 01

Project Group: 4

Semester: Spring

Under The Guidance of:
Anika Tabassum
Lecturer,
East West University

Submitted By:

Name	ID
Saurov Sikder	**2021-1-60-053**
Rawnak	**2020-1-60-263**
Joy Datta	**2020-3-60-042**
Hossain Sheikh	**2020-1-60-201**

Date of Submission: 23^{rd} January, 2025

Abstract

The Hotel Management Unit Testing is a critical component of the software development process aimed at ensuring the reliability, functionality, and accuracy of the hotel management system. This report provides a detailed overview of the unit testing procedures implemented for the Hotel Management System, outlining the testing objectives, strategies, methodologies, and outcomes.

Introduction

A Hotel Management System is a software solution designed to assist hotels in managing various operations such as bookings, customer relations, billing, and inventory. This project includes numerous classes and functions. We have tested all the functions of these classes, covering possible cases while maintaining interdependencies. To assure the quality of the project, we conducted functionality tests using the Java programming language and the JUnit 4 testing library.

Purpose of Testing

The purpose of this testing phase is to validate functionality, ensure data accuracy, evaluate performance under various conditions, verify integration with other systems, comply with industry standards, gather user feedback, and mitigate risks associated with the Hotel Management System before its deployment. Through unit testing, we aim to identify and address any issues, ensuring smooth and efficient operation.

Objectives

The primary objectives of Hotel Management Unit Testing include:

1. Ensuring the accuracy and reliability of individual units or components.
2. Verifying that each unit performs its intended functions correctly.
3. Identifying and rectifying defects or discrepancies in the units.
4. Evaluating the integration of units with other system components.
5. Assessing the robustness and scalability of the units.

Project Scope

1. **Admin Management:**

   1.1. Add, update, and remove admin records.

   1.2. Manage access controls for system users.

2. **Billing Management:**

   2.1. Generate and update bills for customers.

   2.2. Handle payment processing and invoice generation.

3. **Booking Management:**

   3.1. Reserve, update, or cancel room bookings.

   3.2. Manage booking schedules and availability.

4. **Customer Management:**

   4.1. Add, update, and retrieve customer information.

   4.2. Track customer history and preferences.

5. **Employee Management:**

   5.1. Manage employee details and roles.

   5.2. Schedule and assign employee tasks.

6. **Event Management:**

   6.1. Organize and track hotel events.

   6.2. Assign resources and manage schedules for events.

7. **Feedback Management:**

    7.1. Collect and analyze customer feedback.

    7.2. Generate insights for service improvements.

8. **Inventory Management:**

    8.1. Track and manage inventory levels.

    8.2. Monitor supply and demand trends.

9. **Maintenance Management:**

    9.1. Schedule routine maintenance tasks.

    9.2. Log and resolve maintenance issues.

10. **Menu Management:**

    10.1. Add, update, or remove menu items.

    10.2. Organize menu categories for dining options.

11. **Order Management:**

    11.1. Handle customer food and service orders.

    11.2. Track order statuses and delivery.

12. **Room Management:**

    12.1. Manage room allocation and availability.

    12.2. Update room conditions and types.

Testing Strategy

1. Unit tests will verify individual functionalities.

2. Integration tests will ensure seamless interaction between modules.
3. System tests will evaluate overall system behavior from a user's perspective.

Deliverables

1. Functional hotel management system.
2. Comprehensive test suite covering identified functionalities.
3. Documentation outlining system features, assumptions, and limitations.

Stakeholders

1. **Project Manager:** Oversees project execution and ensures alignment with business objectives.
2. **Test Lead:** Manages the testing squads and coordinates testing activities.
3. **Developers:** Collaborate with testers to fix identified issues and enhance software quality.
4. **Quality Assurance Team:** Ensures that all testing standards and protocols are maintained.
5. **End Users:** Benefit from improved software quality and reliability.

Testing Process

We have tested the functionalities of the project, focusing on Unit Testing. For this purpose, we used Eclipse IDE for Java Developers and external libraries of JUnit 4.

Figure 1: Software and Tools

Functionality Overview

Main classes and functions:

1. **Admin.java class:**

Function name:

1. addPolicy(String)
2. backupSystemData()
3. deletePolicy(int)
4. generateAdminReport()
5. getPolicyDetails(int)
6. getSystemLogs()
7. manageUserRoles(String, String)
8. setHotelPreferences(String, String)
9. SystemLogs(String)
10. updatePolicy(int, String)
11. validateAdminActions(String)

2. **Billing.java class:**

Function name:

1. addPaymentDetails(int, String)
2. applyDiscounts(int, double)
3. createBill(int, double, double)
4. generateInvoice(int)
5. generatePaymentReport()
6. getBillDetails(int)
7. getPaymentHistory(int)
8. processRefund(int, double)
9. trackPendingPayments()
10. validatePaymentMethod(String)

3. **Booking.java class:**

Function name:

1. cancelBooking(int)
2. checkBookingStatus(int)
3. createBooking(int, int, Date, Date)
4. getAvailableRoomsForDate(Date, Date)
5. getBookingByCustomer(int)
6. getBookingDetails(int)
7. getPastBookings()
8. testGetUpcomingBookings()
9. testModifyBooking()
10. testValidateBookingDetails()

4. **Customer.java class:**

Function name:

1. addCustomer(int, String, String, String, boolean)
2. applyDiscount(int, double)
3. checkCustomerMembership(int)
4. deleteCustomer(int)
5. getBookingHistory(int)
6. getCustomerDetails(int)
7. getFeedbackFromCustomer(int)
8. searchCustomer(String)
9. updateCustomerDetails(int, String, String, String)
10. validateCustomerIdentity(String, String)

5. **Employee.java class:**

Function name:

1. addEmployee(int, String, String, double, String)
2. assignShift(int, String)
3. calculateSalary(int, double, double)

4. deleteEmployee(int)
5. generateEmployeeReport(int)
6. getEmployeeDetails(int)
7. getEmployeeSalaryDetails(int)
8. getEmployeeShift(int)
9. promoteEmployee(int, String, double)
10. updateEmployeeDetails(int, String, String, double)

6. Event.java class:

Function name:

1. assignEventStaff(int, List<String>)
2. cancelEventBooking(int)
3. checkEventAvailability(String)
4. createEventBooking(String, String, int, double)
5. generateEventBill(int)
6. getEventDetails(int)
7. getEventHistory(int)
8. getEventReport()
9. updateEventDetails(int, String, String, double)
10. validateEventDetails(int)

7. Feedback.java class:

Function name:

1. addFeedback(int, String, int)
2. analyzeFeedbackTrends()
3. generateFeedbackReport()
4. getFeedbackByCustomer(int)
5. getFeedbackRatings()
6. getFeedbackSummary()
7. respondToFeedback(int, String)
8. updateFeedbackStatus(int, String)
9. validateFeedbackContent(int)
10. viewFeedback()

8. Inventory.java class:

Function name:

1. addItem(String, int, double, String)

2. checkStock(int)
3. generateInventoryReport()
4. getItemDetails(int)
5. getLowStockItems(int)
6. removeItem(int)
7. reorderItems(int, int)
8. trackSuppliers()
9. updateItemDetails(int, String, int, double, String)
10. validateInventoryData(int)

9. Maintenance.java class:

Function name:

1. assignMaintenanceTask(int, String)
2. generateMaintenanceReport()
3. getMaintenanceCosts()
4. getMaintenanceDetails(int)
5. getUpcomingMaintenance()
6. logMaintenanceRequest(String, String, double)
7. scheduleMaintenance(int, String)
8. trackMaintenanceHistory()
9. updateMaintenanceStatus(int, String)
10. validateMaintenanceData(int)

10. Menu.java class:

Function name:

1. addMenuItem(String, String, double, boolean)
2. applyMenuDiscount(double)
3. deleteMenuItem(int)
4. generateMenuReport()
5. getDailySpecials()
6. getMenuItemDetails(int)
7. getPopularItems(int)
8. searchMenuItem(String)
9. updateMenuItemDetails(int, String, String, double, boolean)
10. validateMenuData(int)

11. Order.java class:

Function name:

1. assignOrderToRoom(int, int)
2. cancelOrder(int)
3. createOrder(int, int, List<String>, double)
4. generateOrderBill(int)
5. getOrderDetails(int)
6. getOrderHistory(int)
7. getPendingOrders()
8. trackOrderStatus(int)
9. updateOrder(int, List<String>, double)
10. validateOrderDetails(int)

12. Room.java class:

Function name:

1. addRoom(int, String, double)
2. bookRoom(int)
3. cancelBooking(int)
4. checkAvailability(int)
5. deleteRoom(int)
6. getRoomDetails(int)
7. getRoomPrice(int)
8. getRoomStatus(int)
9. getRoomType(int)
10. updateRoomDetails(int, String, double)

Test classes and Test functions:

1. **AdminTest.java Class:**

Function Name:

1. testAddPolicyAndSystemLogs()
2. testBackupSystemData()
3. testBackupSystemDataAndSystemLogs()
4. testDeletePolicyNotFound()
5. testDeletePolicySuccess()
6. testGenerateAdminReport()
7. testGetPolicyDetailsNotFound()
8. testGetPolicyDetailsSuccess()

9. testGetSystemLogs()
10. testManageUserRolesMultipleRoles()
11. testManageUserRolesSuccess()
12. testSetHotelPreferencesMultiple()
13. testSetHotelPreferencesSuccess()
14. testUpdatePolicyNotFound()
15. testValidateAdminActionsSuccess()

2. **BillingTest.java Class:**

Function Name:

1. testAddPaymentDetailsAlreadyPaid()
2. testAddPaymentDetailsBillNotFound()
3. testApplyDiscountsBillNotFound()
4. testGenerateInvoiceBillNotFound()
5. testGenerateInvoiceSuccess()
6. testGeneratePaymentReport()
7. testGetBillDetailsBillNotFound()
8. testGetBillDetailsSuccess()
9. testGetPaymentHistory()
10. testProcessRefundBillNotPaid()
11. testProcessRefundSuccess()
12. testTrackPendingPayments()
13. testValidatePaymentMethodInvalid()
14. testValidatePaymentMethodValid()

3. **BookingTest.java Class:**

Function Name:

1. testCancelBooking()
2. testCheckBookingStatus()
3. testCreateBooking()
4. testGetAvailableRoomsForDate()
5. testGetBookingByCustomer()
6. testGetBookingDetails()
7. testGetPastBookings()

8. testGetUpcomingBookings()
9. testModifyBooking()
10. testValidateBookingDetails()

4. **CustomerTest.java Class:**

Function Name:

1. testAddCustomerAlreadyExists()
2. testAddCustomerSuccess()
3. testApplyDiscountCustomerNotFound()
4. testApplyDiscountForMember()
5. testApplyDiscountForNonMember()
6. testCheckCustomerMembershipIsMember()
7. testCheckCustomerMembershipNotFound()
8. testCheckCustomerMembershipNotMember()
9. testDeleteCustomerNotFound()
10. testDeleteCustomerSuccess()
11. testGetBookingHistoryNotFound()
12. testGetBookingHistorySuccess()
13. testGetCustomerDetailsNotFound()
14. testGetCustomerDetailsSuccess()
15. testGetFeedbackFromCustomerNotFound()
16. testGetFeedbackFromCustomerSuccess()
17. testSearchCustomerFound()
18. testSearchCustomerNotFound()
19. testUpdateCustomerDetailsNotFound()
20. testUpdateCustomerDetailsSuccess()
21. testValidateCustomerIdentityFail()
22. testValidateCustomerIdentitySuccess()

5. **EmployeeTest.java Class:**

Function Name:

1. testAddEmployee()
2. testAssignShift()
3. testCalculateSalary()
4. testDeleteEmployee()
5. testGenerateEmployeeReport()
6. testGetEmployeeDetails()
7. testGetEmployeeSalaryDetails()

8. testGetEmployeeShift()
9. testPromoteEmployee()
10. testUpdateEmployeeDetails()

6. **EventTest.java Class:**

Function Name:

1. testAssignEventStaff()
2. testCancelEventBooking()
3. testCheckEventAvailability()
4. testCreateEventBooking()
5. testGenerateEventBill()
6. testGetEventDetails()
7. testGetEventHistory()
8. testGetEventReport()
9. testUpdateEventDetails()
10. testValidateEventDetails()

7. **FeedbackTest.java Class:**

Function Name:

1. testAddFeedback()
2. testAnalyzeFeedbackTrends()
3. testGetFeedbackByCustomer()
4. testGetFeedbackSummary()
5. testRespondToFeedback()
6. testRespondToFeedbackInvalidId()
7. testUpdateFeedbackStatus()
8. testUpdateFeedbackStatusInvalidId()
9. testValidateFeedbackContent()
10. testValidateFeedbackContentInvalidId()
11. testViewFeedback()

8. **InventoryTest.java Class:**

Function Name:

1. testAddItem()
2. testCheckStock()
3. testCheckStockInvalidId()
4. testGenerateInventoryReport()
5. testGetItemDetails()
6. testGetItemDetailsInvalidId()
7. testGetLowStockItems()
8. testRemoveItem()
9. testRemoveItemInvalidId()
10. testReorderItems()
11. testReorderItemsInvalidId()
12. testTrackSuppliers()
13. testUpdateItemDetails()
14. testValidateInventoryData()

9. **MainteneceTest.java Class:**

Function Name:

1. testAssignMaintenanceTask()
2. testAssignMaintenanceTaskInvalidId()
3. testGenerateMaintenanceReport()
4. testGetMaintenanceCosts()
5. testGetMaintenanceDetails()
6. testGetMaintenanceDetailsInvalidId()
7. testGetUpcomingMaintenance()
8. testLogMaintenanceRequest()
9. testScheduleMaintenance()
10. testScheduleMaintenanceInvalidId()
11. testTrackMaintenanceHistory()
12. testUpdateMaintenanceStatus()
13. testUpdateMaintenanceStatusInvalidId()
14. testValidateMaintenanceData()

10. **MenuTest.java Class:**

Function Name:

1. testAddMenuItemSuccess()
2. testAddMultipleMenuItems()
3. testApplyMenuDiscount()
4. testApplyMenuDiscountToMultipleItems()
5. testDeleteMenuItemNotFound()
6. testDeleteMenuItemSuccess()
7. testGenerateMenuReport()
8. testGetDailySpecials()
9. testGetMenuItemDetailsNotFound()
10. testGetMenuItemDetailsSuccess()
11. testGetPopularItemsSuccess()
12. testMenuDataPersistence()
13. testSearchMenuItemNoResults()
14. testSearchMenuItemSuccess()
15. testUpdateMenuItemDetailsNotFound()
16. testUpdateMenuItemDetailsSuccess()
17. testValidateMenuDataNotFound()
18. testValidateMenuDataSuccess()

11. **OrderTest.java Class:**

Function Name:

1. testAssignOrderToRoom()
2. testCancelOrder()
3. testCreateOrder()
4. testGenerateOrderBill()
5. testGetOrderDetails()
6. testGetOrderHistory()
7. testGetPendingOrders()
8. testTrackOrderStatus()
9. testUpdateOrder()
10. testValidateOrderDetails()

12. **RoomTest.java Class:**

Function Name:

1. testAddRoom()
2. testBookRoom()
3. testCancelBooking()
4. testCheckAvailability()
5. testDeleteRoom()
6. testGetRoomDetails()
7. testGetRoomPrice()
8. testGetRoomStatus()
9. testGetRoomType()
10. testUpdateRoomDetails()

Test Case Table

Main Class Name: Admin	**Test Class Name: AdminTest**

Table 1: Test Case Table for Admin Class

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
updatePolicy	testUpdatePolicyNotFound	1	99, "New policy description"	Policy not found	Policy not found
deletePolicy	testDeletePolicySuccess	2	Add policy, then deletePolicy(1)	Policy ID 1 removed successfully	Policy ID 1 removed successfully

deletePolicy	testDeletePolicyNotFound	3	deletePolicy(99)	Policy not found	Policy not found
getPolicyDetails	testGetPolicyDetailsSuccess	4	Add policy, then getPolicyDetails(1)	Policy details printed without exceptions	Policy details printed without exceptions
getPolicyDetails	testGetPolicyDetailsNotFound	5	getPolicyDetails(99)	Policy not found	Policy not found
manageUserRoles	testManageUserRolesSuccess	6	"john_doe", "Manager"	Role updated successfully for john_doe	Role updated successfully for john_doe
manageUserRoles	testManageUserRolesMultipleRoles	7	"john_doe", "Manager", "jane_doe", "Receptionist"	Multiple roles managed successfully	Multiple roles managed successfully
getSystemLogs	testGetSystemLogs	8	Perform an action, then getSystemLogs()	Logs printed without exceptions	Logs printed without exceptions
generateAdminReport	testGenerateAdminReport	9	Add a policy and manage roles, then generate report	Report contains correct counts of policies and roles	Report contains correct counts of policies and roles
validateAdminActions	testValidateAdminActionsSuccess	10	"Add Policy"	Returns true for valid action	Returns true for valid action
setHotelPreferences	testSetHotelPreferencesSuccess	11	"Check-in Time", "2:00 PM"	Preference set successfully	Preference set successfully
setHotelPreferences	testSetHotelPreferencesMultiple	12	Set multiple preferences	All preferences updated correctly	All preferences updated correctly

backupSystemData	testBackupSystemData	13	None	System data backup completed successfully	System data backup completed successfully
addPolicy and systemLogs	testAddPolicyAndSystemLogs	14	Add a policy	Logs updated after adding a policy	Logs updated after adding a policy
backupSystemData and systemLogs	testBackupSystemDataAndSystemLogs	15	Perform a backup	Logs updated after performing a backup	Logs updated after performing a backup

Main Class Name: Billing	**Test Class Name:BillingTest**

Table 2: Test Case Table for Billing Class

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
addPaymentDetails	testAddPaymentDetailsBillNotFound	1	99, "Cash"	Bill not found	Bill not found
addPaymentDetails	testAddPaymentDetailsAlreadyPaid	2	1, "Credit Card", then 1, "Cash"	Payment method remains "Credit Card".	Payment method remains "Credit Card".
generateInvoice	testGenerateInvoiceSuccess	3	1	Invoice generated successfully	Invoice generated successfully
generateInvoice	testGenerateInvoiceBillNotFound	4	99	Bill not found	Bill not found
applyDiscounts	testApplyDiscountsBillNotFound	5	99, 20	Bill not found	Bill not found

processRefund	testProcessRefundSuccess	6	1, 100 after adding payment	Refund processed successfully	Refund processed successfully
processRefund	testProcessRefundBillNotPaid	7	2, 50 (unpaid bill)	Bill not paid yet	Bill not paid yet
validatePaymentMethod	testValidatePaymentMethodValid	8	"Credit Card", "Cash"	Returns true for valid payment methods	Returns true for valid payment methods
validatePaymentMethod	testValidatePaymentMethodInvalid	9	"Bitcoin"	Returns false for invalid payment method	Returns false for invalid payment method
trackPendingPayments	testTrackPendingPayments	10	None	Pending payments listed (e.g., Bill ID 2)	Pending payments listed (e.g., Bill ID 2)
generatePaymentReport	testGeneratePaymentReport	11	None	Report contains all payment details	Report contains all payment details
getPaymentHistory	testGetPaymentHistory	12	1	Payment history for customer ID 1 retrieved correctly	Payment history for customer ID 1 retrieved correctly
getBillDetails	testGetBillDetailsSuccess	13	1	Bill details printed without exceptions	Bill details printed without exceptions
getBillDetails	testGetBillDetailsBillNotFound	14	99	Bill not found	Bill not found

Table 3: Test Case Table for Booking Class

Main Class Name: Booking	Test Class Name: BookingTest

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
createBooking	testCreateBooking	1	3, 103, today, tomorrow	Booking created successfully; exists in the system	Booking created successfully; exists in the system
cancelBooking	testCancelBooking	2	1	Booking canceled successfully; removed from system	Booking canceled successfully; removed from system
modifyBooking	testModifyBooking	3	1, tomorrow, dayAfterTomorrow	Booking modified successfully	Booking modified successfully
getBookingDetails	testGetBookingDetails	4	1	Booking details retrieved successfully	Booking details retrieved successfully
checkBookingStatus	testCheckBookingStatus	5	1	Booking status retrieved successfully	Booking status retrieved successfully
getUpcomingBookings	testGetUpcomingBookings	6	None	Upcoming bookings retrieved successfully	Upcoming bookings retrieved successfully
getPastBookings	testGetPastBookings	7	None	Past bookings retrieved successfully	Past bookings retrieved successfully

getBookingByCustomer	testGetBookingByCustomer	8	1	All bookings for the customer retrieved	All bookings for the customer retrieved
getAvailableRoomsForDate	testGetAvailableRoomsForDate	9	today, tomorrow	List of available rooms retrieved	List of available rooms retrieved
validateBookingDetails	testValidateBookingDetails	10	1 (valid booking) and 999 (invalid booking)	Valid booking returns true, invalid booking returns false	Valid booking returns true, invalid booking returns false

Table 4: Test Case Table for Customer Class

Main Class Name: Customer	**Test Class Name: CustomerTest**

Main Function	**Test Function**	**Test Case ID**	**Inputs**	**Expected Output**	**Actual Output**
addCustomer	testAddCustomerSuccess	1	3, "Alice Johnson", "alice@example.com", "5556667777", true	Customer added successfully	Customer added successfully

addCustomer	testAddCustomerAlreadyExists	2	1, "John Doe", "john@example.com", "1234567890", true	"Customer already exists."	"Customer already exists."
deleteCustomer	testDeleteCustomerSuccess	3	2	Customer deleted successfully	Customer deleted successfully
deleteCustomer	testDeleteCustomerNotFound	4	99	"Customer not found."	"Customer not found."
updateCustomerDetails	testUpdateCustomerDetailsSuccess	5	1, "John Updated", "john_updated@example.com", "1112223333"	Customer details updated successfully	Customer details updated successfully
updateCustomerDetails	testUpdateCustomerDetailsNotFound	6	99, "New Name", "new_email@example.com", "0000000000"	"Customer not found."	"Customer not found."
getCustomerDetails	testGetCustomerDetailsSuccess	7	1	Customer details displayed successfully	Customer details displayed successfully
getCustomerDetails	testGetCustomerDetailsNotFound	8	99	"Customer not found."	"Customer not found."
searchCustomer	testSearchCustomerFound	9	"Jane Smith"	"Found Customer ID: 2"	"Found Customer ID: 2"
searchCustomer	testSearchCustomerNotFound	10	"Nonexistent Name"	"Customer not found."	"Customer not found."
getBookingHistory	testGetBookingHistorySuccess	11	1	Booking history displayed successfully	Booking history displayed

					successfully
getBookingHistory	testGetBookingHistoryNotFound	12	99	"Customer not found."	"Customer not found."
checkCustomerMembership	testCheckCustomerMembershipIsMember	13	1	Membership status: true	Membership status: true
checkCustomerMembership	testCheckCustomerMembershipNotMember	14	2	Membership status: false	Membership status: false
checkCustomerMembership	testCheckCustomerMembershipNotFound	15	99	"Customer not found." Membership: false	"Customer not found." Membership: false
applyDiscount	testApplyDiscountForMember	16	1, 100.0	"Discounted amount: 90.0"	"Discounted amount: 90.0"
applyDiscount	testApplyDiscountForNonMember	17	2, 100.0	"No discount applicable. Amount: 100.0"	"No discount applicable. Amount: 100.0"
applyDiscount	testApplyDiscountCustomerNotFound	18	99, 100.0	"Customer not found."	"Customer not found."
validateCustomerIdentity	testValidateCustomerIdentitySuccess	19	"john@example.com", "1234567890"	Customer identity validated successfully	Customer identity validated successfully
validateCustomerIdentity	testValidateCustomerIdentityFail	20	"invalid@example.com", "0000000000"	"Customer identity could not be validated."	"Customer identity could not be validated."
getFeedbackFromCustomer	testGetFeedbackFromCustomerSuccess	21	1	Feedback displayed successfully	Feedback displayed successfully
getFeedbackFromCustomer	testGetFeedbackFromCustomerNotFound	22	99	"Customer not found."	"Customer not found."

Table 5: Test Case Table for Employee Class

Main Class Name: Employee	Test Class Name: EmployeeTest

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
addEmployee	testAddEmployee	1	3, "Charlie", "Cleaner", 2000, "Morning"	added successfully	added successfully.
deleteEmployee	testDeleteEmployee	2	1	deleted	deleted
updateEmployeeDetails	testUpdateEmployeeDetails	3	1, "Alice Johnson", "General Manager", 6000	Updated	Updated
getEmployeeDetails	testGetEmployeeDetails	4	1	Successfully print	Successfully print
assignShift	testAssignShift	5	1, "Night"	Assigned	Assigned
getEmployeeShift	testGetEmployeeShift	6	2	found	found
calculateSalary	testGetEmployeeSalaryDetails	7	1	Detailed found salary	Detailed found salary

calculateSalary	testCalculateSalary	8	2, 500, 100	Succesfully calculate salary	Succesfully calculate salary
promoteEmployee	testPromoteEmployee	9	2, "Senior Receptionist", 700	promoted	promoted
generateEmployeeReport	testGenerateEmployeeReport	10	1	Report generated	Report generated
deleteEmployee	testDeleteNonexistentEmployee	11	99	No found	No found
updateEmployeeDetails	testUpdateNonexistentEmployee	12	99, "Nonexistent", "None", 0	Not found	Not found
assignShift	testAssignShiftToNonexistentEmployee	13	99, "Morning"	Not found	Not found
assignShift	testGetNonexistentEmployeeShift	14	99	Doesnot change	Doesnot change
calculateSalary	testGetNonexistentEmployeeSalaryDetails	15	99	Donot found salary details	Donot found salary details
calculateSalary	testCalculateSalaryForNonexistentEmployee	16	99, 500, 100	Not calculated	Not calculated
promoteEmployee	testPromoteNonexistentEmployee	17	99, "None", 0	Not promoted	Not promoted
generateEmployeeReport	testGenerateReportForNonexistentEmployee	18	99	Not generated	Not generated

Table 6: Test Case Table for Event Class

Main Class Name: Event	**Test Class Name: EventTest**

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
createEventBooking	testCreateEventBooking	1	"Birthday Party", "2024-12-25", 103, 2000	Event booking created successfully with ID: 1	Event booking created successfully with ID: 1
cancelEventBooking	testCancelEventBooking	2	1 (id)	Event ID 1 has been cancelled.	Event ID 1 has been cancelled.
updateEventDetails	testUpdateEventDetails	3	1, "Wedding Ceremony", "2024-12-30", 5500	Updated successfully.	Updated successfully.
assignEventStaff	testAssignEventStaff	4	"John", "Doe"	Staff Assigned: John, Doe	Staff Assigned: John, Doe
getEventDetails	testValidateEventDetails	5	999	Not found	Not found
getEventHistory	testGetEventHistory	6	101	Event history found	Event history found
checkEventAvailability	testCheckEventAvailability	7	2024-12-29	Not available	Not available

Table 7: Test Case Table for Feedback Class

Main Class Name: Feedback	**Test Class Name: FeedbackTest**

Main Function	Test Function	Test case ID	Inputs	Expected Output	Actual Output
addFeedback	testAddFeedback	1	customerId=1, feedbackContent="Great service!", rating=5	Feedback added successfully with ID 1. feedbackDatabase contains entry with correct data.	Feedback added successfully with ID 1. feedbackDatabase contains entry with correct data.
		2	customerId=-1, feedbackContent="", rating=-1	Output: "Invalid input. Feedback not added."	Output: "Invalid input. Feedback not added."
viewFeedback	testViewFeedback	3	customerId=1, feedbackContent="Great service!", rating=5	Output contains "Great service!"	Output contains "Great service!"
		4	No feedback added	Output: "--- Feedback List ---\nNo feedback found."	Output: "--- Feedback List ---\nNo feedback found."
updateFeedbackStatus	testUpdateFeedbackStatus	5	feedbackId=1, newStatus="Reviewed"	Status of feedback ID 1 updated to "Reviewed".	Status of feedback ID 1 updated to "Reviewed".
	testUpdateFeedbackStatusInvalidId	6	feedbackId=999, newStatus="Reviewed"	Output: "Feedback not found."	Output: "Feedback not found."
getFeedbackByCustomer	testGetFeedbackByCustomer	7	customerId=1, feedbackContent="Great service!", rating=5	Output contains "Great service!"	Output contains "Great service!"
		8	customerId=999	Output: "--- Feedback from Customer ID: 999 ---\nNo	Output: "--- Feedback from Customer ID: 999 -

				feedback found."	--\nNo feedback found."
getFeedbackSummary	testGetFeedbackSummary	9	Add feedback with statuses "Pending" and "Reviewed"	Summary includes: "Total Feedback: 2", "Pending: 1", "Reviewed: 1".	Summary includes: "Total Feedback: 2", "Pending: 1", "Reviewed: 1".
		10	No feedback added	Summary includes: "Total Feedback: 0", "Pending: 0", "Reviewed: 0".	Summary includes: "Total Feedback: 0", "Pending: 0", "Reviewed: 0".
analyzeFeedbackTrends	testAnalyzeFeedbackTrends	11	customerId=1, rating=5; customerId=2, rating=2	Trend analysis output includes "Rating: 5	Count: 1" and "Rating: 2
		12	No feedback added	Output: "--- Feedback Trends ---\nNo feedback data available."	Output: "--- Feedback Trends ---\nNo feedback data available."
validateFeedbackContent	testValidateFeedbackContent	13	feedbackId=1, feedbackContent="Great service!"	Output: "Feedback ID 1 validation: true".	Output: "Feedback ID 1 validation: true".
	testValidateFeedbackContentInvalidId	14	feedbackId=999	Output: "Feedback not found."	Output: "Feedback not found."
respondToFeedback	testRespondToFeedback	15	feedbackId=1, response="Thank you for your feedback!"	Response is logged, status of feedback ID 1 updated to "Reviewed".	Response is logged, status of feedback ID 1 updated to "Reviewed".
	testRespondToFeedbackInvalidId	16	feedbackId=999, response="Thank you for your feedback!"	Output: "Feedback not found."	Output: "Feedback not found."

Table 8: Test Case Table for Inventory Class

Main Class Name: Inventory	Test Class Name: InventoryTest

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
addItem	testAddItem	1	"Laptop", 10, 50000, "TechSupplier Inc."	Item added successfully with ID: 1	Item added successfully with ID: 1
		2	"", -10, -50000, ""	Item added successfully (but data is invalid and needs validation)	Item added successfully (but data is invalid and needs validation)
removeItem	testRemoveItem	3	1	Item ID 1 removed successfully.	Item ID 1 removed successfully.
	testRemoveItemInvalidId	4	999	Item not found.	Item not found.
updateItemDetails	testUpdateItemDetails	5	1, "Gaming Laptop", 8, 60000, "TechSupplier Inc."	Item ID 1 updated successfully.	Item ID 1 updated successfully.
		6	999, "", -10, -50000, ""	Item not found.	Item not found.
getItemDetails	testGetItemDetails	7	1	Details of item with ID 1 printed: Name: Laptop, Quantity: 10, Price: 50000.	Details of item with ID 1 printed: Name: Laptop, Quantity: 10, Price: 50000.
	testGetItemDetailsInvalidId	8	999	Item not found.	Item not found.

checkStock	testCheckStock	9	1	Stock for Item ID 1: 10	Stock for Item ID 1: 10
	testCheckStockInvalidId	10	999	Item not found.	Item not found.
reorderItems	testReorderItems	11	1, 5	Reordered 5 units for Item ID 1	Reordered 5 units for Item ID 1
	testReorderItemsInvalidId	12	999, 5	Item not found.	Item not found.
getLowStockItems	testGetLowStockItems	13	10	Items with stock below 10 printed: Chair.	Items with stock below 10 printed: Chair.
		14	-5	No items found with stock below -5 (invalid threshold).	No items found with stock below -5 (invalid threshold).
generateInventoryReport	testGenerateInventoryReport	15	Inventory contains 2 items	Report printed with all inventory details.	Report printed with all inventory details.
		16	Inventory is empty	Report shows no items in inventory.	Report shows no items in inventory.
validateInventoryData	testValidateInventoryData	17	1	Returns true.	Returns true.
		18	999	Returns false.	Returns false.
trackSuppliers	testTrackSuppliers	19	Inventory contains items from 2 suppliers	Printed list: TechSupplier Inc., FurnitureCo.	Printed list: TechSupplier Inc., FurnitureCo.
		20	Inventory is empty	Printed message: No suppliers to track.	Printed message: No suppliers to track.

Table 9: Test Case Table for Maintenance Class

Main Class Name: Maintenance	Test Class Name: MainteneceTest

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
logMaintenanceRequest	testLogMaintenanceRequest	1	"Fix AC", "2024-12-30", 200.00	Maintenance request logged successfully with ID: 1	Maintenance request logged successfully with ID: 1
	testLogMaintenanceRequest	2	"", "2024-12-30", 200.00	Error: Description cannot be empty	(Captured from output: Error or Exception Message)
updateMaintenanceStatus	testUpdateMaintenanceStatus	3	1, "In Progress"	Request ID 1 status updated to: In Progress	Request ID 1 status updated to: In Progress
	testUpdateMaintenanceStatusInvalidId	4	999, "In Progress"	Maintenance request not found.	Maintenance request not found.
getMaintenanceDetails	testGetMaintenanceDetails	5	1	Details of Request ID 1 displayed	Details of Request ID 1 displayed
	testGetMaintenanceDetailsInvalidId	6	999	Maintenance request not found.	Maintenance request not found.
scheduleMaintenance	testScheduleMaintenance	7	1, "2024-12-29"	Request ID 1 scheduled for: 2024-12-29	Request ID 1 scheduled for: 2024-12-29
	testScheduleMaintenanceInvalidId	8	999, "2024-12-29"	Maintenance request not found.	Maintenance request not found.

assignMaintenanceTask	testAssignMaintenanceTask	9	1, "John Doe"	Request ID 1 assigned to: John Doe	Request ID 1 assigned to: John Doe
	testAssignMaintenanceTaskInvalidId	10	999, "John Doe"	Maintenance request not found.	Maintenance request not found.
validateMaintenanceData	testValidateMaintenanceData	11	1	TRUE	TRUE
	testValidateMaintenanceData	12	Id= 999	FALSE	FALSE
trackMaintenanceHistory	testTrackMaintenanceHistory	13	Log 2 requests: "Fix AC", "Repair plumbing"	Maintenance history displayed with both requests	Maintenance history displayed with both requests
getUpcomingMaintenance	testGetUpcomingMaintenance	14	Log 2 requests, update 1 to "In Progress"	Display upcoming maintenance for pending requests	Display upcoming maintenance for pending requests
generateMaintenanceReport	testGenerateMaintenanceReport	15	Log 2 requests	Report displayed with all requests	Report displayed with all requests
getMaintenanceCosts	testGetMaintenanceCosts	16	Log 2 requests with costs 200.00, 150.00	Total Maintenance Costs: 350.00	Total Maintenance Costs: 350.00

Table 10: Test Case Table for Menu Class

Main Class Name: Menu	Test Class Name: MenuTest

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
addMenuItem	testAddMenuItemSuccess	1	"Cheeseburger", "Food", 150.0, true	Menu item added successfully	Menu item added successfully
addMenuItem	testAddMultipleMenuItems	2	"Cheeseburger", "Food", 150.0, true and "Latte", "Beverage", 80.0, false	Both menu items added successfully	Both menu items added successfully
deleteMenuItem	testDeleteMenuItemSuccess	3	1	Menu item deleted successfully	Menu item deleted successfully
deleteMenuItem	testDeleteMenuItemNotFound	4	99	"Menu item not found."	"Menu item not found."
updateMenuItemDetails	testUpdateMenuItemDetailsSuccess	5	1, "Veggie Burger", "Food", 140.0, true	Menu item details updated successfully	Menu item details updated successfully
updateMenuItemDetails	testUpdateMenuItemDetailsNotFound	6	99, "New Item", "Food", 100.0, false	"Menu item not found."	"Menu item not found."
getMenuItemDetails	testGetMenuItemDetailsSuccess	7	1	Menu item details displayed successfully	Menu item details displayed successfully

getMenuItemDetails	testGetMenuItemDetailsNotFound	8	99	"Menu item not found."	"Menu item not found."
searchMenuItem	testSearchMenuItemSuccess	9	"cheese"	Search results displayed successfully	Search results displayed successfully
searchMenuItem	testSearchMenuItemNoResults	10	"Pizza"	"No menu items found."	"No menu items found."
generateMenuReport	testGenerateMenuReport	11	N/A	Menu report displayed successfully	Menu report displayed successfully
applyMenuDiscount	testApplyMenuDiscount	12	10 (percentage discount)	Discount applied to the item's price	Discount applied to the item's price
applyMenuDiscount	testApplyMenuDiscountToMultipleItems	13	10 (percentage discount)	Discount applied to prices of all menu items	Discount applied to prices of all menu items
getPopularItems	testGetPopularItemsSuccess	14	6 (popularity threshold)	Only items with popularity > 6 displayed	Only items with popularity > 6 displayed
validateMenuData	testValidateMenuDataSuccess	15	1	Menu item data validated successfully	Menu item data validated successfully
validateMenuData	testValidateMenuDataNotFound	16	99	"Menu item not found."	"Menu item not found."
getDailySpecials	testGetDailySpecials	17	N/A	Daily specials displayed successfully	Daily specials displayed successfully

menuDatabase	testMenuDataPersistence	18	"Cheeseburger", "Food", 150.0, true	Menu item persisted in database correctly	Menu item persisted in database correctly

Table 11: Test Case Table for Order Class

Main Class Name: Order	**Test Class Name: OrderTest**

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
createOrder	testCreateOrder	1	"Burger", 101, 201, items, 15.99	created successfully with ID: 1	created successfully with ID: 1
updateOrder	testUpdateOrder	2	Coke	Order ID 1 updated successfully.	Order ID 1 updated successfully.
cancelOrder	testCancelOrder	3	Burger", 101, 201, 15.99	Order ID 1 has been cancelled.	Order ID 1 has been cancelled.
generateOrderBill	testValidateOrderDetails	4	"Burger", 101, 201, items, 15.99	Validated	Validated
getOrderDetails	testGetOrderDetails	5	1(order id)	Found	Found

getPendingOrders	testGetPendingOrders	6	101(order id)	Order ID: 1, Status: Pending	Order ID: 1, Status: Pending
generateOrderBill	testGenerateOrderBill	7	"Burger", 101, 201, items, 15.99	Total Price: 15.99	Total Price: 15.99
getOrderHistory	testGetOrderHistory	8	"Pizza", "Salad",101, 202, items2, 20.99	Order History for Customer ID: 101 Order ID: 1, Status: Pending Order ID: 2, Status: Pending	Order History for Customer ID: 101 Order ID: 1, Status: Pending Order ID: 2, Status: Pending
trackOrderStatus	testTrackOrderStatus	9	101(id)	Order ID: 2, Status: Pending	Order ID: 2, Status: Pending

Table 12: Test Case Table for Room Class

Main Class Name: Room	**Test Class Name: RoomTest**

Main Function	Test Function	Test Case ID	Inputs	Expected Output	Actual Output
testAddRoom	testAddRoom	1	103, "Single", 150.0	Room added successfully.	Room added successfully.
testAddRoom	testDeleteRoom	2	104, "Standard", 180.0	Room booked.	Room booked.
updateRoomDetails	testUpdateRoomDetails	3	101, "Premium Deluxe", 250.0	Updated successfully.	Updated successfully.

checkAvailability	testCheckAvailability	4	101	Availability: Available	Availability: Available
getRoomDetails	testGetRoomDetails	5	101	Not found	Not found
bookRoom	testBookRoom	6	102	Booked successfully	Booked successfully
cancelBooking	testCancelBooking	7	101	Output: "Cancel booking."	Output: "Cancel booking."
getRoomStatus	testGetRoomStatus	8	101	booked	booked
getRoomType	testGetRoomType	9	" Deluxe"	“Room-type : deluxe”	“Room-type : deluxe”
getRoomPrice	testGetRoomPrice	10	999	Room Price: 200.0	Room Price: 200.0

Expected Outcomes

- Improved management and allocation of hotel resources.
- Enhanced guest satisfaction through streamlined booking and service processes.
- Optimized staff performance and communication within hotel operations.
- Higher operational efficiency and reliability of hotel services.
- Data-driven insights for better decision-making and strategic planning.
- Improved detection and resolution of software defects.
- Enhanced collaboration and communication within testing teams.
- Streamlined testing processes leading to faster release cycles.
- Higher overall software quality and reliability.

Summarization

Serial No	Class Name	Number of functions it contains	Test Class Name	Number of test functions it contains	No of test cases we covered
1	Admin.java	11	AdminTest.java	15	15
2	Billing.java	10	BillingTest.java	14	14
3	Booking.java	10	BookingTest.java	10	10
4	Customer.java	10	CustomerTest.java	22	22
5	Employee.java	10	EmployeeTest.java	10	18
6	Event.java	10	EventTest.java	10	7
7	Feedback.java	10	FeedbackTest.java	11	16
8	Inventory.java	10	InventoryTest.java	14	20
9	Maintenance.java	10	MaintenanceTest.java	14	16
10	MenuTest.java	10	MenuTest.java	18	18
11	Order.java	10	OrderTest.java	10	9
12	Room.java	10	RoomTest.java	10	10
Total	**12 Classes**	**121 Functions**	**12 Test Classes**	**158 Test Functions**	**175 Test Cases**

Conclusion

The Hotel Management System is a comprehensive tool designed to streamline hotel operations with various functionalities, including booking, billing, customer management, and more. Through rigorous testing using Eclipse and JUnit, the system is optimized for reliability, real-time monitoring, and operational efficiency. This ensures enhanced service quality, reduced operational costs, and improved customer satisfaction.

Task Part:

Name	ID	Contribution Part
Rawnak	2020-1-60-263	AdminTest.java, BookingTest.java, BillingTest.java
Saurov Sikder	2021-1-60-053	FeedbackTest.java, InventoryTest.java, MainteneceTest.java RoomTest.java,
Hossain Sheikh	2020-1-60-201	EventTest.java, OrderTest.java, EmployeeTest.java
Joy Datta	2020-3-60-042	CustomerTest.java, MenuTest.java