goo

Table Of Contents

1. TEST PLAN IDENTIFIER

HMS-TP-001.0

Where:

- HMS stands for Hotel Management System.
- TP indicates it is a Test Plan.
- 001 is a sequential number to distinguish between different test plans (if you have multiple versions or test plans).
- .0 refers to the version number of the test plan.

2. References

- **IEEE 829-2008 Standard**: For test plan structure and guidelines
- **GitHub Repository**: https://github.com/kunaltyagi9/Hotel-Management-System

---

3. Introduction

This is the Master Test Plan for the **Hotel Management System** project. This plan will focus on testing the core functionalities and components of the hotel management system, including customer management, room reservation, check-in/check-out processes, and the integration of the MySQL database with the Swing GUI interface. The system is being developed in Java with a MySQL database backend and a Swing-based graphical user interface (GUI). The purpose of this test plan is to ensure that the application functions as expected and meets the business requirements for a hotel management system, while also ensuring that the integration between components works smoothly.

The project will follow a comprehensive testing strategy with three primary levels of testing:

1. **Unit Testing**: Focuses on testing individual classes and methods to verify that each part of the system functions correctly in isolation.

---

2. **System/Integration Testing**: Verifies that all components (e.g., database, business logic, and GUI) work together as expected. This includes testing the integration between the Java application and the MySQL database as well as the interaction between the business logic and the GUI.
3. **Acceptance Testing**: Ensures the system meets the business requirements and is ready for deployment. This will be done with key stakeholders and is expected to take place after the successful completion of system testing.

The testing process will also include static code analysis through **SonarQube**, defect tracking through **jira**, Unit Testing by **Junit5** and automated GUI testing using **AssertJ-Swing**.

---

4. Test Items

The following is a list, by class and functionality, of the items to be tested for the **Hotel Management System** project:

A. **AddDrivers.java**

- Adds new driver records to the system.
- **Unit Testing**: `AddDriversTest.java`

B. **AddEmployee.java**

- Allows the addition of new employee records (e.g., receptionists, managers).
- **Unit Testing**: `AddEmployeeTest.java`
- **GUI Testing**: `AddEmployeeTestGUI.java`

C. **AddRoom.java**

- Handles room additions and management.
- **Unit Testing**: `AddRoomTest.java`
- **GUI Testing**: `AddRoomTestGUI.java`

D. **CheckOut.java**

- Handles the check-out process for customers, including bill generation.
- **Unit Testing**: `CheckOutTest.java`

E. **CustomerInfo.java**

- Manages customer information, such as contact details and booking history.
- **Unit Testing**: `CustomerInfoTest.java`
- **GUI Testing**: `CustomerInfoTestGUI.java`

G. **Dashboard.java**

- Provides an overview of the system's operational status (e.g., available rooms, current bookings).
- **Unit Testing**: `DashboardTest.java`

H. **Department.java**

- Manages different departments of the hotel (e.g., housekeeping, front desk).
- **Unit Testing**: `DepartmentTest.java`

I. **Employee.java**

- Handles employee records and related business logic.
- **Unit Testing**: `EmployeeTest.java`

K. **Login.java**

- Handles user authentication and login.
- **Unit Testing**: `LoginTest.java`
- **GUI Testing**: `LoginTestGUI.java`

L. **ManagerInfo.java**

- Manages the information of hotel managers.
- **Unit Testing**: `ManagerInfoTest.java`

M. **NewCustomer.java**

- Manages the addition of new customers to the system.
- **Unit Testing**: `NewCustomerTest.java`

N. **PickUp.java**

- Manages hotel pick-up services for customers (e.g., airport transfer).
- **Unit Testing**: `PickUpTest.java`

O. **Reception.java**

- Manages reception operations such as checking in and handling guest requests.
- **Unit Testing**: `ReceptionTest.java`

P. **Room.java**

- Handles room details, availability, and booking.
- **Unit Testing**: `RoomTest.java`

Q. **SearchRoom.java**

- Provides functionality for searching available rooms based on criteria.
- **Unit Testing**: `SearchRoomTest.java`

R. **UpdateCheck.java**

- Manages updating customer check-in and check-out information.
- **Unit Testing**: `UpdateCheckTest.java`

S. **UpdateRoom.java**

- Manages updates to room availability, pricing, and status.
- **Unit Testing**: `UpdateRoomTest.java`

Testing Tools and Integration

- **Unit Testing**: JUnit will be used for testing the individual classes and methods of the system.
- **GUI Testing**: AssertJ-Swing will be used to automate and test the Swing-based graphical user interface of the application.
- **Static Analysis**: SonarQube will be used for static code analysis to ensure code quality and identify potential issues.
- **Defect Tracking**: jira will be used for tracking defects and issues found during the testing process.

---

5. Software Risk Issues

There are several parts of the **Hotel Management System** project that involve external dependencies or critical system elements, which must be carefully managed to ensure smooth functioning. These risks are beyond the direct control of the application but can have a significant impact on the project's success.

A. **Database Connectivity and Integration**

- The **MySQL** database connectivity is a critical dependency for the system, and any issues with database connectivity, queries, or data integrity could severely impact the functionality of the hotel management system. The system must handle connections, database transactions, and errors effectively.
- **Risk**: Failure to connect to or integrate with the MySQL database can halt various operations like customer check-ins, bookings, and updates to room availability.

C. **Backup and Recovery of Data**

- The hotel system will rely on MySQL for data storage. Adequate backup and recovery mechanisms must be in place to prevent data loss in case of system crashes, power failures, or other unforeseen events.
- **Risk**: Loss of critical data such as customer records, booking details, and transaction information could lead to significant operational disruption.

D. **Ability to Recover from Failures**

- The system should be able to gracefully handle failures, especially in the middle of important processes such as check-in/check-out, room reservations, or bill generation.
- **Risk**: In the event of an error or system crash, the application should ensure that the state of transactions is properly handled (e.g., preventing double bookings, ensuring partial transactions are rolled back).

E. **Database Security and User Access Control**

- The application needs to implement robust security measures to protect sensitive data, such as customer information, payment details, and employee records. Access control must be defined for different roles (Admin, Employee, Customer) to prevent unauthorized access or data breaches.
- **Risk**: Inadequate security could lead to unauthorized access to critical customer or employee data, putting the system at risk of security breaches or privacy violations.

F. **GUI Performance and Usability**

- The user interface (Swing-based) must be intuitive and responsive, ensuring that employees and customers can efficiently navigate the system for tasks such as booking rooms, checking in/out, and managing customer profiles.
- **Risk**: Slow or confusing user interface elements could result in poor user experience, leading to frustration and potential errors in system use.

G. **System Scalability**

- The system should be able to handle an increasing number of users (both customers and employees) without performance degradation. This includes handling a larger number of transactions, room bookings, and concurrent users.
- **Risk**: If the system is not scalable, it may face performance bottlenecks as the user base or hotel size increases, leading to delays or failures in processing transactions.

H. **Testing and Defect Resolution Delays**

- The development timeline is tight, and any delays in defect resolution or testing could significantly impact the overall project timeline. Comprehensive testing will be required to ensure the system meets its functional and non-functional requirements.
- **Risk**: Delays in testing or defect resolution could push back the deployment date, affecting the project's schedule and deliverables.

6. Features to be Tested

A. **Room Reservation System**

- The process of booking and reserving rooms, including validation of room availability, pricing, and guest details.
- Ensuring that the system correctly handles different types of reservations (e.g., single-night, multi-night, guest preferences, etc.).

B. **Customer Management**

- The functionality that allows the addition, update, and deletion of customer profiles, including contact information, payment details, and booking history.
- Verifying that customer data is correctly stored, retrieved, and updated in the system.

C. **Check-in and Check-out Process**

- Testing the processes for guest check-in and check-out, including the generation of bills, handling of payment methods, and updating room availability.
- Verifying that the system ensures proper data consistency when processing check-in and check-out transactions.

D. **Room Availability and Management**

- Verifying the functionality for managing room availability, pricing, and status (e.g., available, booked, under maintenance).
- Ensuring that the system accurately updates room status and reflects it across all parts of the system in real-time.

E. **Employee Management**

- Testing the features related to employee records, including adding, updating, and deleting employee information.
- Ensuring that the system allows managers to assign roles, track work hours, and manage payroll information.

F. **Reporting and Dashboards**

- The system's ability to generate reports on various operational metrics, such as occupancy rates, revenue reports, employee performance, and customer history.
- Testing the dashboards and reports for accuracy, formatting, and performance.

G. **User Authentication and Access Control**

- Testing the login system for different user roles (Admin, Employee, Customer) to ensure proper access control and secure authentication.
- Verifying that only authorized users can access sensitive areas of the system (e.g., admin functionalities, employee records).

H. **Room Search and Booking Interface**

- The functionality that allows customers to search for available rooms based on specific criteria (e.g., dates, room type, price range).
- Ensuring that the system returns accurate results and allows customers to complete bookings without errors.

I. **Backup and Recovery Mechanisms**

- Testing the backup and recovery features to ensure that the system can safely store and recover data in case of failures or crashes.

- Verifying that backup operations are performed regularly and that recovery processes restore the system to its last consistent state.

K. **Database Integration and Data Integrity**

- Testing the integration between the Java application and the MySQL database to ensure data consistency, reliability, and security.
- Verifying that all data entered into the system is correctly stored, updated, and retrieved from the database.

L. **User Interface (UI) and Usability**

- Testing the Swing-based GUI for responsiveness, ease of navigation, and overall user experience.
- Verifying that the interface is intuitive and that users can perform all necessary actions without confusion or errors.

M. **System Performance**

- Ensuring that the system can handle a large number of users, bookings, and transactions without performance degradation.
- Conducting load testing to assess the system's capacity to manage high traffic and concurrent users.

---

7. Features Not to be Tested

The following is a list of areas that will not be specifically addressed in the testing of the **Hotel Management System**. Testing for these areas will be indirect as a result of other testing efforts:

A. **Non-Hotel Management Functionalities**

- Features not directly related to hotel management (e.g., marketing, forecasting, external third-party systems) are outside the scope of this project. Any data or reports generated for such purposes will be indirectly tested as part of the system's core functionalities (e.g., customer reservations, billing, and room management).
- **Reason**: These are external processes or modules that do not impact the fundamental hotel operations such as bookings, check-in/check-out, or room management.

B. **Network Security**

- Network security concerns, including the configuration of firewalls, intrusion detection, and encrypted communication between systems, will not be tested within this project scope. Security aspects of the network (e.g., securing communication between the application and MySQL database) will be indirectly tested via user authentication and database access control.
- **Reason**: Security-related features at the network level will be managed and tested by the IT/network team, and are not part of the application-level testing.

C. **Backup and Disaster Recovery Procedures**

- While the backup functionality of the database is tested (as part of data integrity), operational aspects related to system-wide disaster recovery procedures, including database and system-level restoration following a crash or disaster, will not be explicitly tested.
- **Reason**: These aspects are handled by infrastructure and system administrators and are outside the scope of the application's functionality testing.

E. **Customer-Side Features (Non-Critical Booking/Reservation) Functions**

- Features like promotional offers, loyalty programs, and customer engagement features will not be directly tested in this phase. These may be handled in future phases or updates of the project.
- **Reason**: These are additional features that may be implemented later, and current testing focuses on core functionality such as booking, check-in, and room management.

F. **Operational Aspects of Hotel Management (e.g., Staff Scheduling)**

- Internal operational management processes, such as staff scheduling, housekeeping management, or inventory management, are not part of the scope for this phase of the project. These features may be tested in future versions of the system if they are implemented.
- **Reason**: These features are either not yet implemented or are considered beyond the current phase of testing.

G. **Integration with Legacy Systems**

- The hotel management system may need to integrate with legacy systems for functions such as employee payroll, housekeeping updates, etc. However, testing for legacy system integrations will not be done as part of this project.

- **Reason**: Legacy system integration is outside the scope of this project, and will be handled by a separate team if necessary.

---

8. Approach

8.1 Testing Levels

The testing for the **Hotel Management System** will consist of **Unit Testing**, **System/Integration Testing**, and **Acceptance Testing**. The levels of testing will be structured as follows:

Unit Testing:

- Unit testing will be performed by the development team. Each individual class or module (such as `AddRoom.java`, `Login.java`, `CheckOut.java`) will undergo unit testing to verify that they work as expected in isolation.
- Proof of unit testing (test case lists, sample outputs, data printouts, and defect information) must be provided by the developer to the team leader before passing it on for system testing.
- Unit tests will be written and executed by the developers, with oversight from the team leader. These tests will validate basic functionality such as input validation, data processing, and expected outputs.

System/Integration Testing:

- System/Integration testing will be performed by the **test manager** and the **development team leader**, with assistance from the individual developers as necessary.
- This testing will involve validating the integration between different components, such as the **Hotel Management System** and the MySQL database, and checking the interactions between core functionalities like room bookings, customer management, and billing.
- This testing will ensure that all components work together in a unified manner. Defects discovered during this phase will be categorized based on severity, and any critical defects must be fixed before proceeding further.
- The goal of this testing is to ensure that different modules integrate correctly, data flows seamlessly between them, and the system behaves as expected in a real-world environment.

Acceptance Testing:

- Acceptance testing will be carried out by the **end users**, including employees and hotel management, with the assistance of the **test manager** and **development team leader**.
- This phase will focus on validating that the system meets the requirements and works in parallel with the existing manual processes, such as manual check-ins and billing. Acceptance testing will run in parallel with the hotel's current system for at least **one month** after system integration testing is completed.
- A limited group of users will participate in this initial testing phase to ensure the system can handle real-world operations. The focus will be on functionality, usability, and ensuring that the system meets the expectations of the hotel staff and management.
- All critical and major defects must be corrected and verified by the customer test representative before final acceptance.

8.2 Configuration Management/Change Control

- Programs and configurations will be managed using version control to ensure the correct version is being tested. Any updates or changes to the codebase will be tracked and managed through the **change control process**.
- All unit and integration testing will be performed on the development environment (local system or staging server). Once the system reaches a stable state, **acceptance testing** will be conducted in parallel with the live environment (production server), but with strict controls to prevent actual updates to the live data.
- Once all tests have passed, the code will be migrated to the **production environment**, following a structured deployment process to ensure that the hotel management system runs smoothly without affecting live operations.

8.3 Test Tools

The following tools will be used during the testing phases:

- **JUnit**: For unit testing of individual classes and methods (e.g., verifying functionality for `AddRoom.java`, `Login.java`, `Employee.java`, etc.). JUnit will provide the foundation for validating core functionalities at the code level.
- **AssertJ-Swing**: For testing the **Swing-based GUI**, verifying that the interface elements such as buttons, text fields, and labels work correctly and interact as expected with the backend logic.
- **jira**: For defect tracking, jira will be used to log and track the defects found during testing. Developers will use jira to resolve issues, and testers will ensure that issues are fixed before passing them to the next phase of testing.
- **SonarQube**: Used for static analysis to ensure that the code adheres to coding standards and to identify potential vulnerabilities or code smells early in the development process.

8.4 Meetings

- The test team will meet bi-weekly to evaluate progress, discuss any discovered defects, and identify potential issues early in the development cycle.
- The test team leader will also meet with the **development team** and **project manager** every two weeks to provide updates on testing progress and discuss any issues encountered.
- These meetings will be used to ensure alignment between testing efforts and project goals and to make adjustments as necessary.
- Additional meetings can be scheduled in case of urgent issues or critical defects that need immediate attention.

8.5 Measures and Metrics

The following metrics will be collected and reported regularly to track the progress of testing and defect resolution:

- **Defects by module and severity**: This will include a breakdown of defects by severity (e.g., critical, major, minor) and by module (e.g., `CheckOut.java`, `Room.java`, `SearchRoom.java`).
- **Defect Origin**: Each defect will be tracked to determine whether it originated from requirements, design, or coding issues.
- **Time spent on defect resolution**: Time taken to resolve critical and major defects will be tracked to ensure that the team is addressing high-priority issues promptly.
- **Number of times a program is submitted for testing**: This metric tracks how often a program is submitted to the test team, which will help assess the stability and maturity of each module.
- **Defects found at higher levels**: Defects that should have been found during unit testing but were identified during system or acceptance testing will be documented to help improve future testing efforts.

---

9. Item Pass/Fail Criteria

The test process will be considered complete when the following conditions are met:

- **Functionality**: All core functionalities of the Hotel Management System (such as room booking, customer management, check-in/check-out process, employee management, billing, and reporting) pass the unit, system/integration, and acceptance testing phases.

- **Data Integrity**: The integration between the application and the MySQL database will be tested, ensuring that all data (e.g., customer information, room booking details, transaction history) is correctly stored, retrieved, and updated. Any discrepancies in data between the system and the database will result in a fail.
- **Usability**: The **Swing-based GUI** will be validated for usability and user experience. All GUI elements (e.g., buttons, forms, dropdowns) should be functional and responsive, as tested using **AssertJ-Swing**.
- **Performance**: The system should meet the specified performance standards (e.g., room search should return results within 3 seconds, check-out transactions should be processed within 5 seconds).
- **Error Handling**: The system should handle errors gracefully (e.g., invalid login attempts, incomplete booking details) and provide appropriate feedback to users.
- **Test Comparison**: In the case of integrating with external systems (like payment gateways), the system will undergo **parallel testing** to ensure that data is synchronized and the expected output matches the manual or legacy process.

Once these conditions are met, the application will be considered for production release. Specifically:

- **Initial Pass**: When the core functionalities have been validated, and critical defects have been fixed, the system will be considered ready for deployment to the **first group of users**.
- **Fail Criteria**: If critical defects or functionality gaps remain after testing, the system will fail, and the issues will need to be resolved before moving forward to the acceptance phase. For instance, if room booking or customer check-in cannot be completed, it will fail the test process.

---

10. Suspension Criteria and Resumption Requirements

Testing will be suspended and resumed based on the following conditions:

A. Unavailability of Key Test Resources (e.g., Distributors or Test Users)

- If key users or distributors (those performing parallel testing) are not available at the start of the pilot phase, the testing will be delayed. A minimum of **three hotel departments or users** should be ready and able to test before the system enters the pilot phase. No new modules or features will be added during this delay.

- **Resumption Requirement**: Testing will resume once at least three test users or departments are available, and the system is able to be tested in a real-world scenario.

B. Missing Test Data or Dependencies (e.g., Database)

- If **real test data** (e.g., customer bookings, employee records, transactions) is missing or if the database schema is not fully integrated with the application, the testing will be delayed. Testing cannot proceed until a sufficient data set is available to validate the system's functionality, especially for critical functionalities like billing or reporting.
- **Resumption Requirement**: Testing will resume once test data is populated, and the data flow between the application and the MySQL database is fully functional. This includes making sure that data integrity is maintained throughout the system's processes.

C. External System Integration Issues (e.g., Payment Gateway)

- If external systems (such as **payment gateways** for online booking payments or third-party APIs for room availability) are not integrated or tested, this will result in a delay. The **hotel booking system** relies on these integrations for full functionality.
- **Resumption Requirement**: Testing will resume once the integration with external systems is complete and verified. This may include testing external APIs, payment processing, and any other third-party service required by the system.

D. Critical Defects Found During Testing

- If **critical defects** are identified during any of the testing phases (e.g., system crashes during booking, failure to process transactions, data corruption), testing will be suspended until the defects are fixed.
- **Resumption Requirement**: Testing will resume once the critical defects are resolved, and the system is stable enough for further testing. This includes verifying that the defects have been addressed and retesting the affected functionality.

E. Environmental Issues (e.g., System Downtime, Server Issues)

- If **server downtime** or environmental issues occur (e.g., unavailability of MySQL database servers, network issues), testing may be suspended.
- **Resumption Requirement**: Testing will resume once the environmental issues are resolved and the testing environment is stable. This includes ensuring that all necessary systems are back online and functional.

11. Test Deliverables

The following deliverables will be provided at different stages of testing for the **Hotel Management System** project:

A. Acceptance Test Plan

- A detailed **Acceptance Test Plan** will be created to guide the end users during the acceptance testing phase. This will include test cases that validate the core functionalities of the system (e.g., room booking, check-in/check-out, billing), along with any specific business logic expected by the users.
- The plan will outline the test objectives, entry and exit criteria, required resources, and the schedule for user testing.

B. System/Integration Test Plan

- A **System/Integration Test Plan** will be provided to detail how the components of the Hotel Management System will be tested together. This will include integration points with the **MySQL database**, **Swing GUI**, and any third-party services (e.g., payment gateways or external APIs).
- The plan will define test scenarios for the **system's overall functionality**, data flow between components, and interface behavior across different modules (such as room management, employee management, and customer data).

C. Unit Test Plans/Turnover Documentation

- For each class and module (e.g., **AddRoom.java, AddEmployee.java, CustomerInfo.java**), a **Unit Test Plan** will be provided. This will document the specific unit tests executed by developers to verify individual components before system-wide testing.
- The **Turnover Documentation** will include:
    - A list of all unit tests run and their results.
    - Details of any defects or issues found during unit testing, including resolution.
    - A summary of the code changes or updates made based on unit test feedback.

D. Screen Prototypes

- **Screen Prototypes** will be developed for key UI elements (using **Swing**), including the **room booking interface**, **customer check-in/check-out forms**, **employee management screens**, and any other relevant user interfaces.

- These prototypes will be reviewed with stakeholders and will serve as the baseline for **GUI testing** (using **AssertJ-Swing**), ensuring that the system meets the user interface requirements before development continues.

E. Report Mock-Ups

- **Report Mock-Ups** will be created to simulate the reports generated by the system (e.g., booking reports, payment reports, employee shift reports). These mock-ups will outline the layout and content of reports before implementing the actual report generation features.
- These mock-ups will be used for **user feedback** and to ensure that the final reports meet the business requirements.

F. Defect/Incident Reports and Summaries

- **Defect/Incident Reports** will document any issues found during testing (unit, system/integration, or acceptance) along with the status of each defect (open, in-progress, resolved, closed).
- A **Defect Summary** will be provided that groups defects by severity and priority, with descriptions of the issues and steps taken to resolve them.
- These reports will also include the results of **static analysis** from **SonarQube** and any relevant **jira defect tracking**.

G. Test Logs and Turnover Reports

- **Test Logs** will be maintained to document the execution of test cases, including inputs, outputs, results, and any anomalies or issues encountered.
- **Test Turnover Reports** will summarize the testing progress at different stages, detailing:
    - The scope of tests performed.
    - Any blockers or delays encountered during testing.
    - The overall readiness of the system for deployment based on testing results.
    - A final sign-off indicating that testing has been completed successfully for each test phase (unit, system/integration, acceptance).

12. Remaining Test Tasks

Task	Assigned To	Status
Create Acceptance Test Plan	TM (Test Manager), PM (Project Manager), Client	Pending
Create System/Integration Test Plan	TM (Test Manager), PM (Project Manager), Dev (Development Team)	Pending
Define Unit Test Rules and Procedures	TM (Test Manager), PM (Project Manager), Dev (Development Team)	Pending
Define Turnover Procedures for Each Level	TM (Test Manager), Dev (Development Team)	Pending
Verify Prototypes of Screens	Dev (Development Team), Client, TM (Test Manager)	Pending
Verify Prototypes of Reports	Dev (Development Team), Client, TM (Test Manager)	Pending

- **TM**: Test Manager
- **PM**: Project Manager
- **Dev**: Development Team
- **Client**: Client/End User

13. Environmental Needs

A. Access to Development and Production Systems

- **Development System**: Access to the development environment where the system is being built and unit testing is conducted. This system should have the necessary configurations and test data available to simulate real-world usage.
- **Production System**: Access to the production environment (if applicable), where the final version of the application will run. The system should be configured with actual customer data to ensure accurate system behavior during **System/Integration** and **Acceptance Testing**.

B. Database Access (MySQL)

- Access to the **MySQL database** that holds hotel-related data such as room availability, customer details, employee information, bookings, and payments. This includes both development and production instances.
- The database will be necessary for testing data flow, CRUD operations (Create, Read, Update, Delete), and ensuring data consistency during all testing phases.

C. Access to Swing GUI Environment

- A functional **Swing GUI** setup for all front-end components like room booking interfaces, check-in/check-out forms, and employee management screens. The development and testing environments should be equipped with the proper Swing libraries to simulate real user interaction with the application during **GUI Testing** (using **AssertJ-Swing**).

D. Test Devices for GUI Testing

- Access to devices or virtual machines running different operating systems (Windows, macOS, etc.) for testing the **Swing GUI** across different environments to ensure compatibility. This is necessary for **cross-platform testing** and to ensure the user interface functions as expected.

E. Access to Test Data for Functional Testing

- **Test data** must be prepared and available for testing purposes, such as:
    - Customer information for booking and check-in.
    - Room types, availability, and rates for room management.
    - Employee data for managing hotel staff and payroll.
    - Booking data for verifying reservation workflows.

- **Test cases** should use this data to validate the core functionalities like **room booking**, **payment processing**, **employee management**, and **report generation**.

F. Backup and Recovery Systems

- **Backup systems** for database and application recovery, ensuring the integrity of data during **System/Integration Testing** and **Acceptance Testing**. This includes ensuring that all transactions (e.g., bookings, payments) are recoverable after system failures.

G. Bug Tracking and Static Analysis Tools

- **jira**: Used for tracking defects and incidents reported during testing. The tool will help in managing defect life cycles, including reporting, prioritization, and resolution.
- **SonarQube**: To perform **static code analysis** and ensure code quality standards are maintained. It will also be used to identify vulnerabilities and other potential issues during the **Unit Testing** phase.

H. Test Automation Tools

- **JUnit**: For unit testing purposes, particularly for testing individual classes (like **AddRoom.java**, **AddEmployee.java**, etc.) and ensuring their expected behavior.
- **AssertJ-Swing**: For automating the testing of **Swing-based GUIs**, ensuring that the application's front-end is working properly, including all interactions, buttons, and input fields

---

.

14. Staffing and Training Needs

For the **Hotel Management System** project, effective staffing and training are crucial to ensure that all aspects of testing, development, and project implementation are thoroughly covered. The following staffing requirements and training needs will help ensure the success of the project:

A. Testing and Development Team Staffing

- **Test Manager (TM)**: The project should have a dedicated **Test Manager** responsible for overseeing all testing efforts. If a separate full-time tester is not available, the **Test Manager** will assume this role and be involved in both development and testing.

    - **Role**: The Test Manager will be responsible for overseeing **unit testing**, **system/integration testing**, and **acceptance testing**. They will also coordinate with the development team to resolve defects and ensure testing schedules are adhered to.
    - **Full-time Tester**: It is recommended to assign a **full-time tester** during the **System/Integration Testing** and **Acceptance Testing** phases. The tester will help execute test cases, track defects, and ensure proper documentation of the results.
    - **Duration**: The full-time tester will be engaged approximately **four months** into the project.
- **Developers (Dev)**: The development team will be responsible for coding, debugging, and initial unit testing of individual classes and components (e.g., **AddRoom.java**, **Login.java**, **HotelManagementSystem.java**, etc.).

    - **Role**: Developers will participate in **unit testing** and also support **system/integration testing** when necessary.
- **Project Manager (PM)**: The **Project Manager** will oversee the overall progress of the project, ensuring that testing activities are aligned with the project timeline and objectives.

B. Training Needs

Training will be required for different roles involved in the project, including developers, testers, and end-users. The training needs are outlined below:

1. **Testing Staff and Developers**

    - **Test Manager and Developer Training**:
        - **Basic Testing Tools**: Developers and testers will be trained on the use of testing tools such as **JUnit** (for unit testing) and **AssertJ-Swing** (for automating GUI testing). This will ensure efficient execution of unit and integration tests, as well as testing of the front-end user interface.
        - **Test Documentation**: Developers and testers will be trained on how to document test cases, test results, defects, and how to report issues using **jira**. This will ensure that all issues are tracked efficiently throughout the development and testing phases.
        - **Version Control and Change Management**: Training on the use of **SonarQube** for static code analysis and maintaining version control through tools like Git will be provided to ensure code quality and tracking of changes.

2. **End-User Training (Client Side)**

    - **Sales Administration Staff**: Training will be provided to the sales administration staff on using the **Swing GUI** interfaces (for tasks like checking in guests, managing room bookings, and processing payments). They will also be trained on understanding and utilizing the **generated reports**.
        - **Training Content**: This will include the **room booking process**, **payment processing**, **employee management**, and **report generation** functionalities. Training will be hands-on, and end-users will be given mock scenarios to practice.
3. **Operations Staff Training**

    - **Developer and Operations Training**: At least one developer and one member of the operations team will be trained on installing and configuring the application on the production systems. This will ensure smooth deployment and post-deployment maintenance.
    - **Operations Training**: The operations team will also be trained on system **backup/recovery processes** and handling **database access** to ensure that the system is properly maintained and secure.
4. **Training for Bug and Issue Management**

    - **Bug Tracking (jira)**: Developers and testers will be trained on using **jira** for reporting and tracking defects during testing. This will ensure that all defects are categorized, prioritized, and tracked until resolution.
    - **Issue Resolution**: Developers and the test manager will be trained on how to handle issues, prioritize them based on severity, and collaborate on resolving critical and major defects before moving forward with each testing phase.
5. **Client and Distributor Training**

    - **Distributors**: If applicable, training will be provided to distributors who will interact with the **Hotel Management System** through external systems like booking platforms or payment gateways. These distributors will be trained on system interfaces for data exchange (e.g., EDI processes, if any).
    - **Client-Side Training for Deployment**: Before going live, the project manager, test manager, and client team should conduct a final review of the application. This is important to ensure that the client is fully capable of operating the system post-launch.

C. Resource Requirements

- **Test Environment Setup**: A dedicated test environment, including access to development and production systems, must be available to conduct the required training and testing activities.
- **Hardware/Software Resources**: The development and testing staff will require access to development machines, databases (MySQL), and testing tools (JUnit, AssertJ-Swing, SonarQube).
- **Communication Tools**: Collaboration tools such as **Slack**, **Microsoft Teams**, or other internal communication channels should be set up for quick discussions and issue resolution.

---

15. Responsibilities

Task	TM	PM	Dev Team	Test Team	Client
Acceptance Test Documentation & Execution	X	X	X	X	X
System/Integration Test Documentation & Execution	X	X	X		
Unit Test Documentation & Execution	X	X	X		
System Design Reviews	X	X	X	X	X
Detail Design Reviews	X	X	X	X	X

Test Procedures and Rules	X	X	X	X	
Screen & Report Prototype Reviews	X	X	X	X	
Change Control and Regression Testing	X	X	X	X	X

Roles Overview:

- **TM (Test Manager)**: Oversees all testing-related activities, responsible for creating test plans, ensuring proper execution, and documentation.
- **PM (Project Manager)**: Assists in creating and reviewing test documentation, ensures that the project timeline and resources are managed properly.
- **Dev Team**: Implements the system, reviews designs, writes and executes unit tests, and supports other testing activities.
- **Test Team**: Focuses on the execution of test cases, documenting results, and supporting integration and system tests.
- **Client**: Participates in review processes, provides feedback, and assists with executing acceptance tests, ensuring that the system meets business requirements.

---

16. Shedule

The following testing activities are scheduled within the project timeline. Specific dates and times for each activity are defined in the overall project plan. The personnel required for each task, including the test team, development team, management, and customer, will be coordinated by the project manager in collaboration with the development and test team leaders.

1. **Review of Requirements Document**
   Assigned to: Test Team, Development Team
   Duration: 1 week
   The test team will review the requirements document with the development team to

---

understand the project scope, objectives, and to begin defining test cases.

2. **Development of Master Test Plan**
   Assigned to: Test Manager
   Duration: 2 weeks
   The test manager will create the Master Test Plan, which will include testing strategies, objectives, and timelines. The plan will undergo at least two reviews to ensure completeness.

3. **Review of System Design Document**
   Assigned to: Test Team
   Duration: 1 week
   The test team will review the system design document to gain a better understanding of the overall application structure. This will help further define the testing objectives and the specific test cases needed.

4. **Development of System/Integration and Acceptance Test Plans**
   Assigned to: Test Manager, Development Team, Test Team
   Duration: 3 weeks
   The System/Integration and Acceptance Test Plans will be developed by the test manager, with input from the development and test teams. These plans will be reviewed at least twice to ensure their effectiveness and alignment with project goals.

5. **Review of Detail Design Documents**
   Assigned to: Test Team
   Duration: 1 week
   The test team will review the detail design documents to gain a deeper understanding of the individual program structure. This will further refine the test cases and ensure that testing is aligned with the final application structure.

6. **Unit Testing**
   Assigned to: Development Team
   Duration: Concurrent with Development
   Unit testing will be performed by the development team during the development phase. The test team will provide feedback, and any issues will be documented and resolved as part of the ongoing development cycle.

7. **System/Integration and Acceptance Testing**
   Assigned to: Test Team, Development Team, Client
   Duration: 4 weeks
   After the development and unit testing phases, System/Integration and Acceptance

Testing will take place. The client will be involved in validating the tests, ensuring the application meets all requirements and expectations.

---

17. Planning Risks and Contingencies

a. **Limited Availability of Local Development Resources**

Given that the project is being developed in Pakistan, a potential risk could be the **limited availability of skilled Java developers** and **database administrators** familiar with the specifics of MySQL, Swing for GUI, and integrating external services. If developers with the required skill set are unavailable, it could cause delays in project milestones. In this case, the contingency plan would involve **recruiting additional resources** or **upskilling current team members** through training.

b. Language and Localization Challenges

The system must cater to both local staff and customers, which may require significant localization efforts, especially for Urdu and regional language support in the GUI. However, **language inconsistencies or UI bugs** due to translations or regional preferences could cause delays in acceptance testing. A contingency would involve having **dedicated language experts** or **localization specialists** who are well-versed in the cultural and linguistic nuances of Pakistan, ensuring that the system works seamlessly in multiple languages.

c. Integration with Local Payment Gateways

Integrating with **local payment gateways** (such as Easypaisa, JazzCash, or bank APIs) might be challenging due to **compatibility issues or changing regulations**. If the integration fails or is delayed, it could impact the overall functionality of the HMS, especially in the **payment processing module**. The contingency plan would include **identifying multiple payment gateway providers**, ensuring the system can fall back on an alternative provider if one is unavailable or facing issues.

d. **Unpredictable Regulatory Changes**

Pakistan’s hospitality industry might face **frequent regulatory changes** (such as new taxes or legal requirements for hotel operations, including guest tracking regulations by NADRA). If these changes occur during the development phase, they could necessitate significant updates to the system's features. The contingency plan would be to **build flexibility into the software**

---

architecture, allowing the system to be easily updated in response to regulatory changes without major rework.

e. Data Security and Privacy Risks

Data security is a crucial aspect, particularly in handling sensitive guest information such as passport details and payment information. Any **data breaches** or failures to comply with data privacy laws (e.g., Personal Data Protection Bill in Pakistan) could delay the project and result in reputational damage. A contingency plan would be to **implement strong encryption, regular security audits**, and **compliance checks** with local and international data protection standards, ensuring that sensitive data is secure and in compliance with legal frameworks.

---

18. Approvals

- Project Sponsor - Zain
- Development Management - Noor
- EDI Project Manager - Abdullah
- RS Test Manager - Faraz
- RS Development Team Manager - Abdul hassan
- Reassigned Sales - Gul Ahmed
- Order Entry EDI Team Manager - Abu Bakr