



**FACULTY OF ENGINEERING, TECHNOLOGY & BUILT
ENVIRONMENT**

BER2053 :JAVA PROGRAMMING

REPORT (20%)

PROJECT TITLE : Smart Hotel Management System

PREPARED BY : Ayesha Arvag Hussain (1002371911)

Hamzeh Kareh (1002372768)

Ku Kwang Yaw(1002268339)

Mohammed Tariq(1002371596)

Harine Palanirajan Vijayarajkumar (1002371771)

SEMESTER : Jan – Apr 2025

INSTRUCTOR : Dr Mohammad Arif Bin Ilyas

Acknowledgement

We are thankful to Dr Mohammad Arif Bin Ilyas who was our lecturer, who assisted us a lot in accomplishing this project. Our group, Ayesha ,Hamzeh Tariq ,Yaw and Harine , have tried their best to incorporate the goals set and we appreciate the teamwork that every member of the team rendered. This assignment has been very beneficial in terms of knowledge and skills acquisition.

We also appreciate UCSI University for giving us every necessary support and resources that enabled this work.

Table of Contents

Hotel Management System Report.....	4
CHAPTER 1: INTRODUCTION.....	4
1.1 Background of the Study.....	4
1.2 Problem Statement.....	4
1.3 Objectives of the Project.....	4
1.4 Scope and Limitations.....	5
Scope:.....	5
Limitations:.....	6
CHAPTER 2. Literature Review and System Design.....	6
2.1 Overview of Hotel Management Systems.....	6
2.2 Existing Solutions and Their Limitations.....	6
2.3 Justification for Using Java and MySQL.....	7
3. System Analysis and Design.....	7
3.1 Overview of the Smart Hotel Management System.....	7
3.2 Functional and Non-Functional Requirements.....	8
Functional Requirements:.....	8
Non-Functional Requirements:.....	8
3.3 Use Case Analysis.....	8
Actors:.....	8
Primary Use Cases:.....	8
3.4 System Architecture & Database Design.....	9
Database Tables:.....	9
3.5 User Interface Design (Wireframes & Mockups).....	10
4. Methodology.....	11
4.1 Research Approach.....	11
4.2 Development Process (Prototype Development Lifecycle).....	11
4.3 Data Collection Methods.....	11
4.4 Tools and Technologies Used.....	12
5. Implementation and Features.....	12
5.1 Key Features of the Smart Hotel Management System.....	12
5.2 Room Booking and Guest Management.....	13
5.3 Check-in and Check-out Process.....	13
5.4 Payment and Billing System.....	15
5.5 Housekeeping and Staff Management.....	15
5.6 Java GUI Implementation (Swing/AWT).....	16
5.7 Database Connectivity (JDBC with MySQL).....	16
5.8 Exception Handling in the System.....	16
5.9 Use of Collections API for Data Management.....	17
6. Evaluation and Testing.....	18
6.1 Assessment Based on Assignment Rubric.....	18

6.2 Testing Methods (Unit Testing, Integration Testing).....	19
Unit Testing:.....	19
Integration Testing:.....	19
6.3 Performance evaluation and test case results.....	21
7. Challenges and Solutions.....	21
7.1 Technical Challenges Faced.....	21
7.2 Solutions Implemented.....	22
7.3 Lessons Learned.....	22
8. Conclusion and Future Enhancements.....	23
8.1 Summary of Findings.....	23
8.2 Potential Improvements and Scalability.....	23
8.3 Future Research Directions.....	24
9. References.....	25

Table of Figures

figure 1: Use Case Diagram.....	9
figure 2: Database Entity Relationship Diagram.....	10
figure 3: GUI displaying the main menu.....	13
figure 4: UML / system flow diagram.....	14
figure 5: Payment tab and payment confirmation(receipt).....	15
figure 6: Data from payment being stored.....	18
figure 7: Code behind the management system implementation.....	20
figure 8: Results Grid keeping record of all entries within the system.....	21

Hotel Management System Report

1. Introduction

1.1 Background of the Study

Modern hospitality operations have expanded rapidly so businesses need modern management tools to process reservations together with handling customer information and billing and essential operational processes. Hotel firms that maintain their operations through manual record-keeping experience data loss together with errors and operational inefficiencies. The combination of Java programming with MySQL database technology in Hotel Management Systems (HMS) delivers automation which increases operational precision as well as system efficiency and customer-friendly interfaces.

1.2 Problem Statement

Traditional hotels continue using old procedures for booking management which results in data inconsistencies and extended customer support periods as well as double-booked reservations. An automated hotel management system provides a centralized platform which solves operational problems for hotels by creating effective management.

1.3 Objectives of the Project

- To develop a Hotel Management System using Java and MySQL.
- To Implement a graphical user interface (GUI) for easy interaction.
- To Provide functionalities such as room booking, customer check-in/check-out, billing, and record maintenance.
- To Ensure secure data storage and retrieval.
- To Enhance efficiency and reduce operational errors.

1.4 Scope and Limitations

Scope:

Through the system customers can register and manage their data effectively by handling guest information with ease. The system maintains precise customer records throughout booking operations as well as check-in period.

Room booking through this system enables users to track room availability in real-time for both booking reservations and ensuring single-booked accommodations. The system maintains continuous room status updates during every transaction process.

The system computes billable amounts through its invoice generator based on the combination of room category and guest stay duration. The payment references become part of the administrative data storage after confirmation of bookings.

The system operates with staff management along with login authentication support through MySQL-based backend storage. The staff member records including their usernames and passwords are safeguarded in the MySQL database. The system verifies staff members through database queries by checking their submitted credentials. The system blocks unauthorized access to essential features which include booking management as well as check-in/out processing and data viewing. The current system provides basic security functionality because it links system access permissions to the data stored in the database even though advanced role management capabilities are not available. Employees can perform their activities easier through this staff management system since it tracks every system interaction to the specific staff member.

The system achieves secure data management by integrating with the MySQL database. The system protects user and booking data through persistent storage techniques which prevent both data loss and unauthorized access enabling long-term storage and retrieval operations.

Limitations:

The present design structure targets accommodations of small to medium capacity. It does not have the necessary capacity nor infrastructure to handle extensive hotel chains or massive booking operations effectively. This system functions optimally in properties that have small to medium guest flows alongside moderate room capacities.

This system lacks the capability to connect with external booking services along with online reservation platforms. Staff members need to manually input all booking information that comes from external sources. The absence of data integration creates extra work for administrators while producing potential inconsistencies because online and in-person reservations do not sync properly. Hotel staff need to manage their entire guest booking process through the internal system because there are no external integration capabilities.

The system functions exclusively for individual hotel business locations. The system lacks capabilities for centralized management of multiple branches or properties. The current system design does not allow hotels with multiple locations to handle their properties together since it lacks both multi-site management capabilities and data sharing across locations.

2. Literature Review

2.1 Overview of Hotel Management Systems

A hotel management system is software designed to streamline administrative tasks, such as room reservations, staff management, billing, and customer records. Modern HMS solutions integrate with databases and user-friendly interfaces to improve operational efficiency.

2.2 Existing Solutions and Their Limitations

Hotel management solutions operate in two distinct forms which are cloud-based platforms and standalone software applications. Available hotel management systems experience four main drawbacks:

The standard commercial platforms available in the market do not allow users to customize their interfaces. These platforms need to serve multiple customers so they provide set interfaces and workflows. Hotels that need to customize their systems based on procedures or

branding find commercial platforms too inflexible and challenging to modify unless they involve developer assistance.

presence of security vulnerabilities in online systems. When hotel management software operates in the cloud it faces external security threats such as data breaches and unauthorized access unless strict measures of encryption and access control are implemented. The absence of proper security measures generates substantial threats to operational security as well as guest privacy protection.

Most current hotel management systems present complex interfaces which create problems for non-technical staff. The system presents challenges for staff members with minimal computer experience because they struggle to operate it properly thus creating operational delays along with inaccurate data entry and drawn-out training periods.

2.3 Justification for Using Java and MySQL

The Java environment delivers independence from platforms together with robust performance and it features an extensive development ecosystem for developing GUI applications using Swing or JavaFX. The hotel records can be managed through MySQL which delivers secure and efficient and scalable relational database management capabilities to users. Java integrates perfectly with MySQL to create an optimal solution when developing dependable HMS. The analysis presents an effective hotel management system development based on Java and MySQL for resolving prevalent hotel industry problems through a systematic method.

3. System Analysis and Design

3.1 Overview of the Smart Hotel Management System

The Smart Hotel Management System is a system where lodge operations are simplified by room reservation, checking in and checking out programs, billing, and worker control within one capability. The system allows real time tracking room availability to prevent overbook, offers automated guest check in / check out for enhanced experience as well as individual guest profile for customized service. The system also allows multiple payment options and enhances housekeeping and staff duty management to improve operational efficiency.

3.2 Functional and Non-Functional Requirements

Functional Requirements:

- Give guests permission to reserve rooms based on room availability.
- Activate check-in and check-out procedures with real-time status.
- Process billing and said pay method.
- Keep track of all guest records and booking history.
- Give administrative access for staff management.

Non-Functional Requirements:

- Guarantee high system reliability and availability
- Give a simple user friendly interface for resort staff.
- Protect sensitive information with encryption.
- Support scalability for future expansion.

3.3 Use Case Analysis

Actors:

- Guest
- Receptionist
- Hotel Manager
- Housekeeping Staff

Primary Use Cases:

1. **Guest Booking:** A guest enters company information and an available room is selected.
2. **Check-in:** The receptionist checks out the booking against a record there, marks the room, thereby declaring it occupied.
3. **Check-out & Billing:** The receptionist does the total billing, makes payment and room status updates.

4. **Room Management:** The system updates the room availability based on the records of the guest check-in and out.
5. **Housekeeping Management:** Housekeeping personnel are automated with cleaning schedules and updates the room status once cleaning is complete.
6. **Admin Panel:** Hotel Manager Bookings, staff allocation & report comes across.

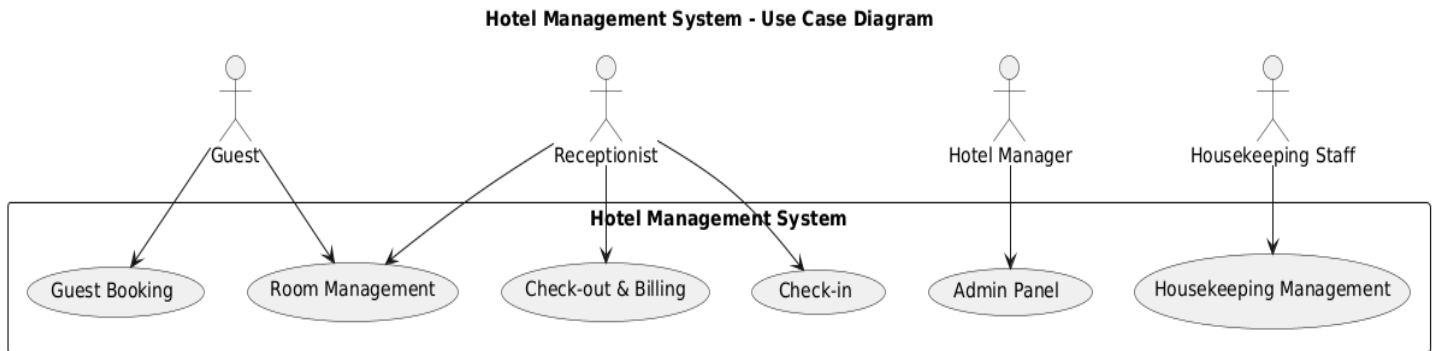


figure 1: Use Case Diagram

3.4 System Architecture & Database Design

The system operates using a **three-tier structure**:

1. **Presentation Layer (Frontend):** Java Swing/AWT-based GUI for user interaction.
2. **Business Logic Layer:** The Java-based Business Logic Layer controls all application logic functions including booking processes and check-in/check-out and payments management.
3. **Data Layer (Database):** The database component uses MySQL to store all information about rooms, guests, transactions and housekeeping work.

Database Tables:

- **Rooms:** The Rooms table contains information about room identification numbers along with types and statuses and nightly pricing rates.
- **Guests:** The Guest table holds guest information with name and contact data together with booking identification.
- **Bookings:** The Bookings system handles reservations through its fields which include booking ID and guest ID along with check-in/check-out dates and room ID.
- **Payments:** Tracks transactions (payment ID, guest ID, amount, payment method).

- **Housekeeping:** Housekeeping personnel oversee both the cleaning schedule and the room status maintenance.

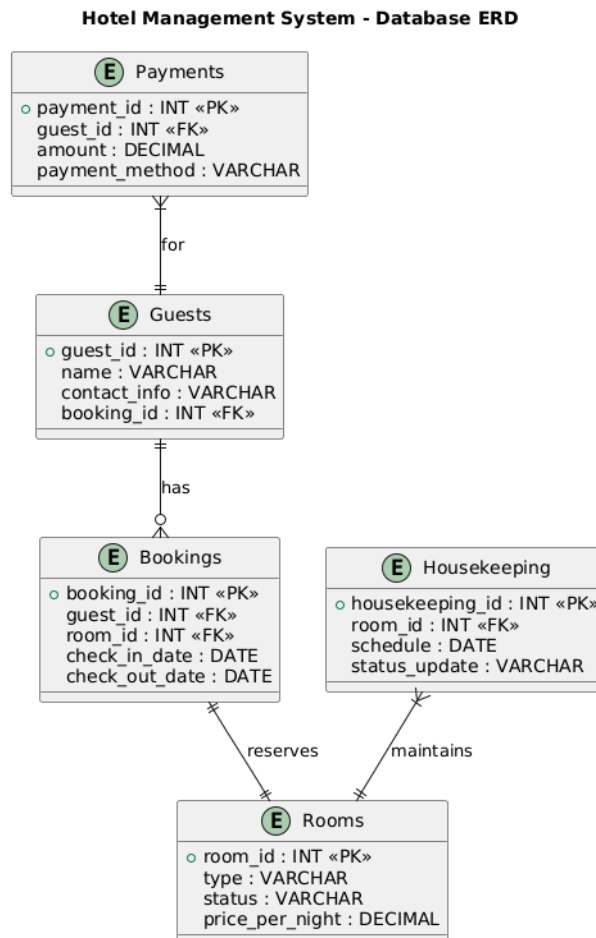


figure 2: Database Entity Relationship Diagram

3.5 User Interface Design (Wireframes & Mockups)

The system interface features an easy-to-use design approach, key UI components include:

- **Main Dashboard:** The main dashboard gives users one-click access to book rooms and perform check-in and check-out and billing operations.
- **Booking Form:** Through the Booking Form users can provide their information along with selecting their desired room options..
- **Check-in/Check-out Interface:** The interface features a check-in/checkout section which shows guest information while updating room availability.
- **Payment Processing Screen:** Facilitates transactions with various payment methods.

- **Admin Panel:** The admin panel gives users an overview of bookings with room availability as well as housekeeping tasks.

4. Methodology

4.1 Research Approach

Combination of qualitative and quantitative research were realised to determine the hotel management challenges. Surveys and interviews with hotel staff were carried out to directly identify operational inefficiencies, while existing hotel management systems' case studies were reviewed to gain insight into the best practices.

4.2 Development Process (Prototype Development Lifecycle)

The system uses a development process which implements a **prototype development lifecycle**:

1. **Requirement Gathering:** The method of requirement gathering aims to determine core features by engaging stakeholders.
2. **Design & Prototyping:** Creating initial UI wireframes and database schemas.
3. **Implementation:** Developing core functionalities in Java and MySQL.
4. **Testing & Iteration:** Unit testing occurs first followed by a process of refinement which uses feedback to improve the system.
5. **Deployment:** The system reaches completion for practical implementation during deployment.

4.3 Data Collection Methods

- **Interviews with Hotel Staff:** Hotel staff interviews help understand the weaknesses of present management practices.
- **User Surveys:** Gathering feedback on system usability and feature preferences.
- **System Logs & Analytics:** Tracking system performance and user interactions for improvements.

4.4 Tools and Technologies Used

- **Programming Language:** Java (Swing/AWT for GUI, JDBC for database connectivity)
- **Database Management System:** MySQL
- **Development Environment:** Java IDE Eclipse
- **Version Control:** GitHub for source code management
- **Testing Tools:** JUnit for unit testing

This structured approach ensures the system meets hotel management needs while maintaining efficiency and scalability.

5. Implementation and Features

5.1 Key Features of the Smart Hotel Management System

The Smart Hotel Management System operates as an operational system for hotels to improve both hotel operations and guest experiences. Key features include:

- **Automated Room Booking & Guest Management:** The automated room booking system with guest management delivers instant booking capabilities and efficient service scheduling for guest needs.
- **Seamless Check-in & Check-out:** The hotel benefits from digital check-in capabilities which eliminate manual work along with automated check-out systems.
- **Integrated Payment & Billing System:** The integrated payment and billing system combines multiple payment alternatives into a system that auto-generates billing documentation.
- **Housekeeping & Staff Management:** The optimization of housekeeping and staff management includes methods to schedule room cleaning in addition to employee assignment procedures.
- **Data Analytics & Reports:** The hotel monitoring system creates performance data along with user preference details through its analytical report functions.

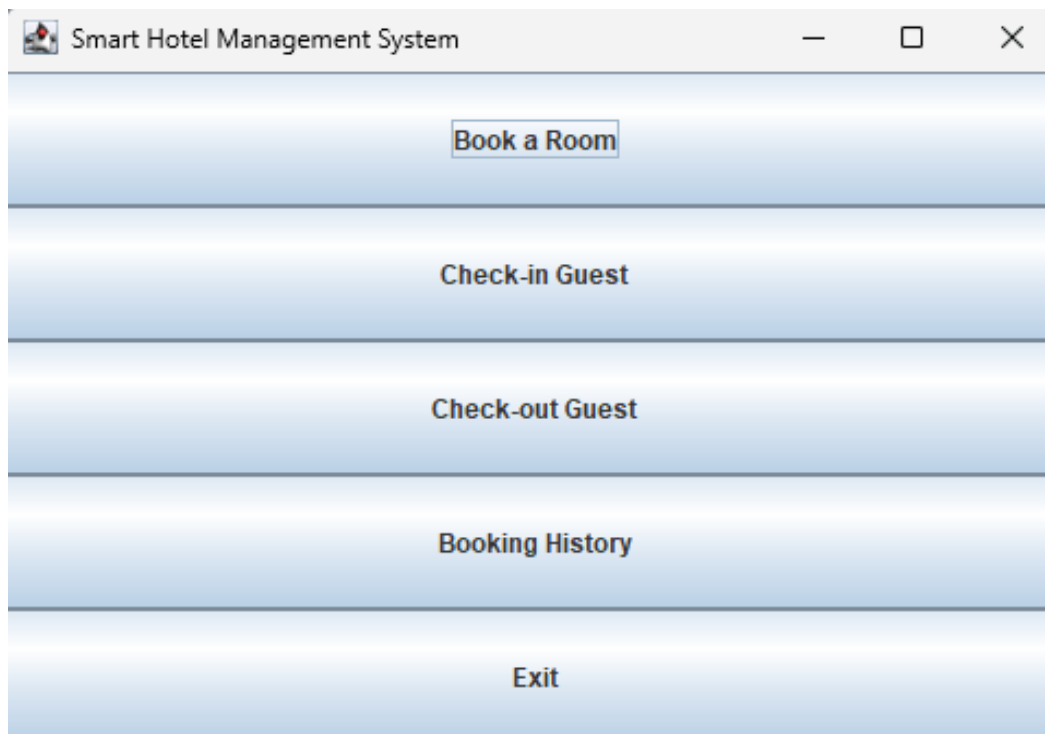


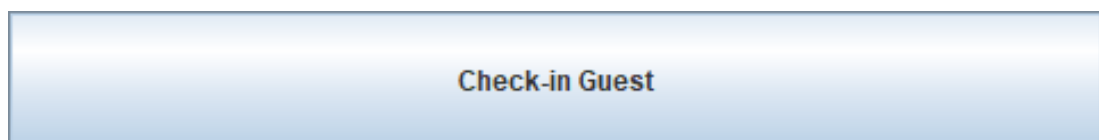
figure 3: GUI displaying the main menu

5.2 Room Booking and Guest Management

- The hotel enables online bookings and also allows guests to book rooms at reception.
- Session time updates room reservations to avoid overbookings.
- Personal information together with previous accommodations and preference choices gets stored under guest profiles.

5.3 Check-in and Check-out Process

- The hotel provides two options for check-in which includes reception service and self-service kiosks that distribute digital key cards.
- **Check-in:** Guests can check in via a reception desk or self-service kiosk, with digital key card issuance.



- **Check-out:** Bills are automatically generated based on room charges and additional services used.

Check-out Guest

Hotel Room Booking System - Activity Diagram

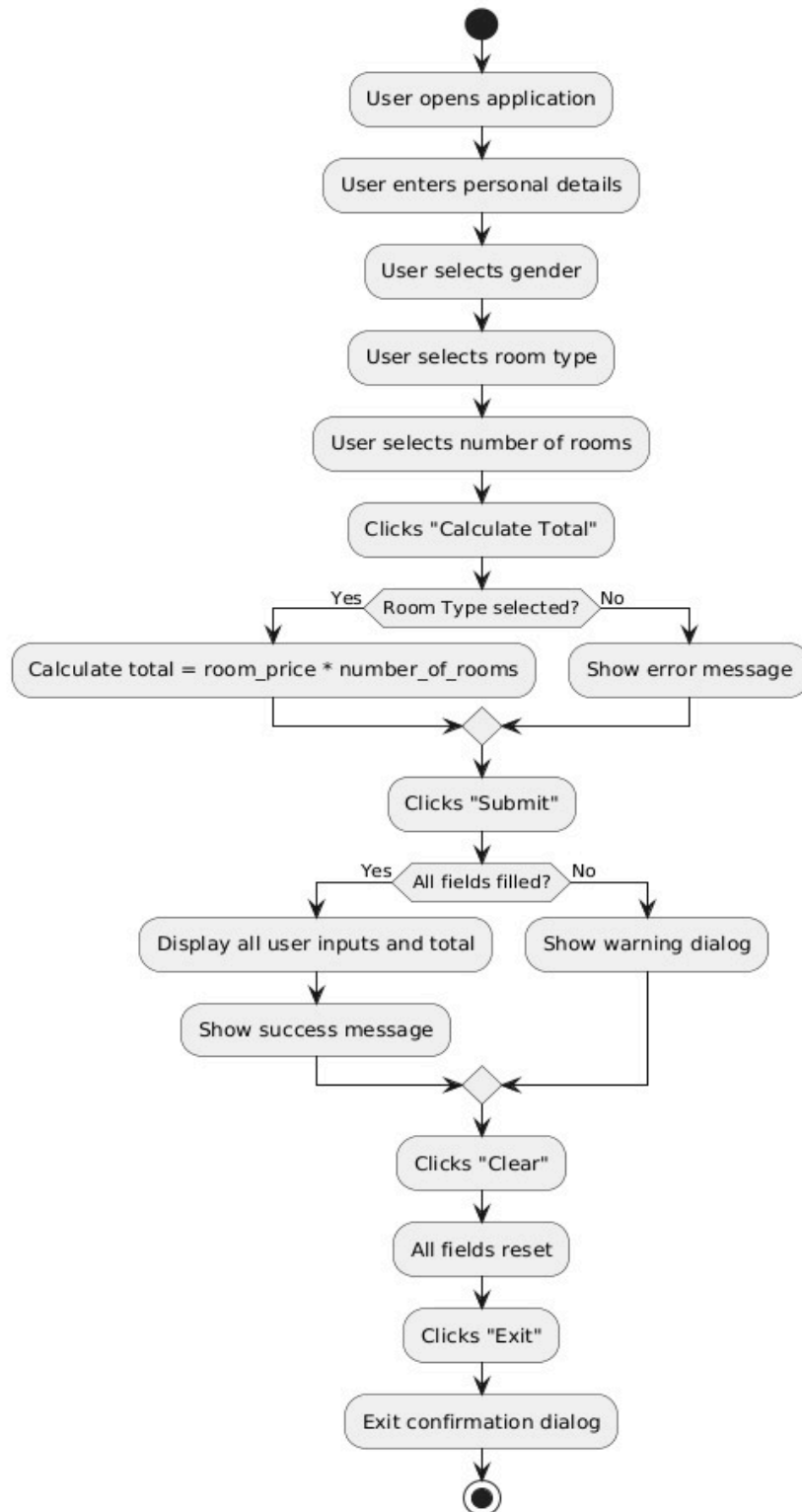


figure 4: UML / system flow diagram

5.4 Payment and Billing System

- The hotel accepts payments through credit or debit cards and mobile solutions and digital wallets.
- The system produces electronic invoices which display both tax specifications and itemized charging information.
- The system records accounting and reporting transactions through maintained logs.

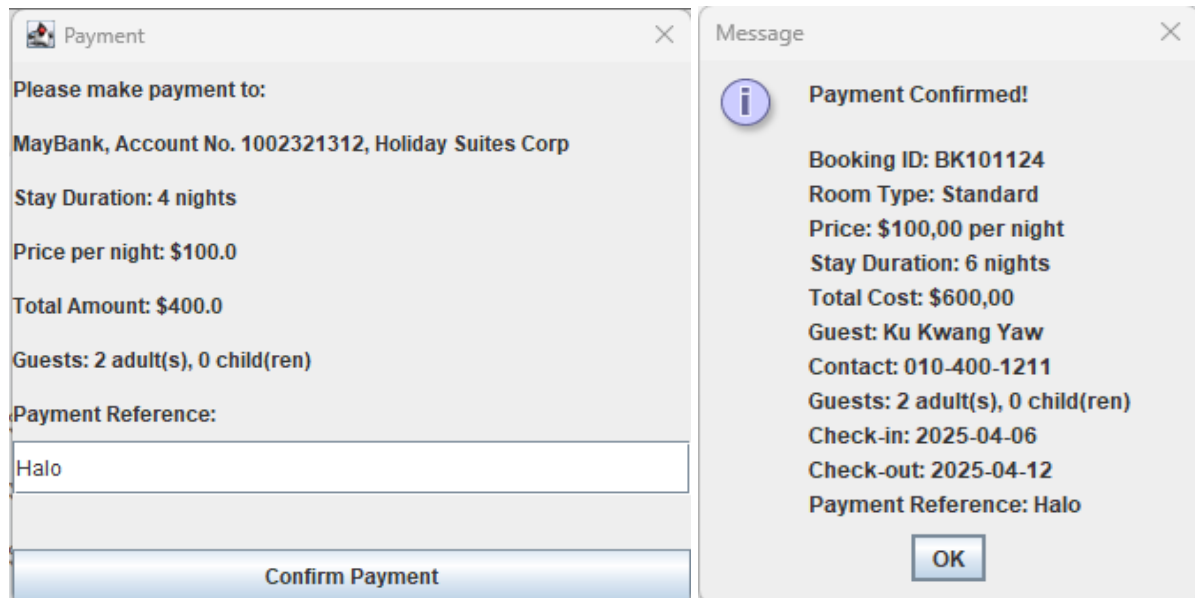


figure 5: Payment tab and payment confirmation(receipt)

5.6 Java GUI Implementation (Swing/AWT)

The Smart Hotel Management System builds its graphical user interface (GUI) through implementation of Java Swing. We selected Swing because of its flexible integration capabilities within Java-based programs. Features of this system appear through multiple windows and panels which present different functions that include room booking along with check-in and check-out operations as well as payment handling and management features.

This interface consists of multiple interacting elements which include JTextFields, JComboBoxes, JButtons while JTables display and capture system data. We intentionally made the GUI approach minimal and user-friendly to serve both expert and novice user

groups. Each component used Layout managers including GridLayout and BorderLayout for aligning elements consistently regardless of window dimension. Java Swing enabled a major development of usable features which enhanced the total user-system interaction experience.

5.7 Database Connectivity (JDBC with MySQL)

JDBC enables the system to connect with MySQL database storage while maintaining data security at all times. A single Java class known as DatabaseHelper.java exists to control database connections within the system. The method within this class builds a connection to hotel_system database through JDBC driver by using configuration-defined credentials.

Booking data management through SQL operations enables the system to perform INSERT, SELECT and UPDATE commands. After a user finishes booking through the GUI system it instantly inputs guest information along with reservation data to MySQL bookings table. The data integration through JDBC provides consistent data records with extended storage duration and simplified scaling capabilities for future development. Through JDBC connectivity with MySQL the application functions as a complete system that keeps front-end information in step with backend database storage.

5.8 Exception Handling in the System

The Smart Hotel Management System depends on exception handling to maintain reliability and durability. The application uses try-catch blocks in strategic positions to handle errors which emerge during database connectivity and SQL execution as well as input validation operations.

The system provides dual error detection methods through detailed stack trace output to the Eclipse console for developer debugging and user-friendly error notifications through JOptionPane dialog displays. The implementation protects user experience flow alongside enabling developers to locate and diagnose technical problems during development. Exception handling effectively stops software failure and keeps systems running smoothly throughout real-time operation.

5.9 Use of Collections API for Data Management

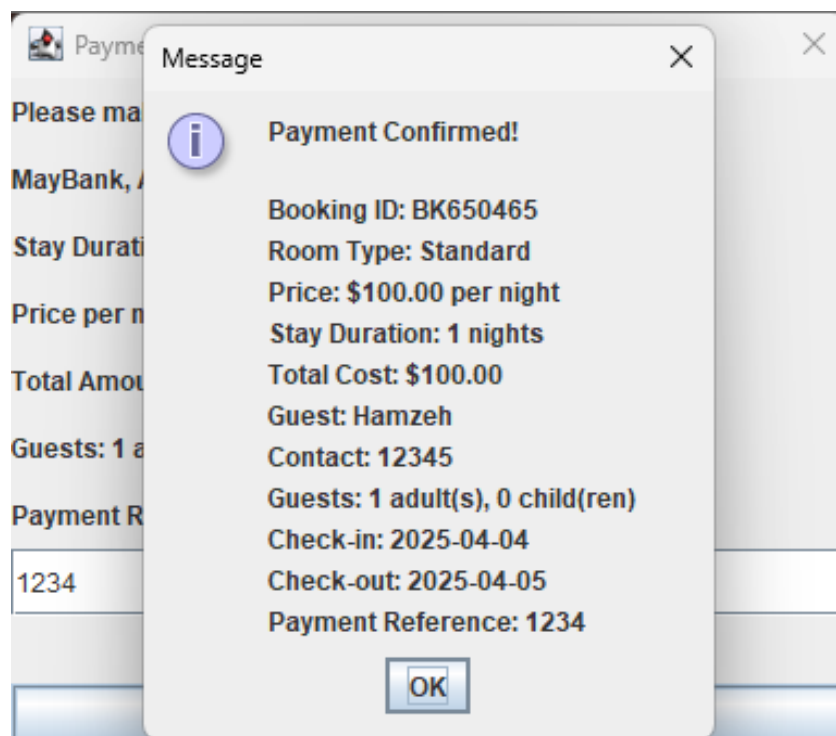
The application extracted extensive data management from Java Collections API before implementing MySQL integration. The application utilized HashMap and ArrayList classes temporarily for booking record storage as well as track history and booking ID to guest details associations. The application stored booking information by using HashMap<String, Booking> which searched bookings through identifier strings but maintained the Booking sequence with ArrayList<Booking>.

The application maintains its dependence on collections even after it adopted MySQL for database operations. The application uses collections to store temporary data received from the database along with providing structures for table population and implementing filtering capabilities. SelectedItems through collections help the system accomplish faster data access and enhanced memory control while extending the system's flexibility in managing live and stored data.

6. Evaluation and Testing

6.1 Assessment Based on Assignment Rubric

Evaluation of the Smart Hotel Management System occurred according to criteria from the course assignment rubric that rated functionality together with GUI design and database use and innovative approaches and code quality and documentation standards. Core operational requirements involving live room reservations and customer check-ins and check-outs and payment handling and MySQL database storage persistency are included in the system. A functional prototype demonstrates all fundamental elements through its responsive Java Swing interface. Standardization of code architecture as well as system upkeep were core design principles and objectives.



Result Grid													
Filter Rows:													
Edit: Export/Import: Wrap Cell Content:													
booking_id	name	contact	room_type	room_price	check_in_date	check_out_date	stay_duration	total_cost	payment_reference	num_adults	num_children	checked_in	checked_out
BK123456	Alice Smith	0123456789	Deluxe	150.00	2025-04-05	2025-04-08	3	450.00	REF123456	2	1	0	0
BK650465	Hamzeh	12345	Standard	100.00	2025-04-04	2025-04-05	1	100.00	1234	1	0	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

figure 6: Data from payment being stored

6.2 Testing Methods (Unit Testing, Integration Testing)

Unit Testing:

JUnit framework served as the testing tool to perform unit tests that evaluated the individual logical sections of the system. The assessment tests checked the precision of distinct operations including duration measurement and price calculations tied to room types and booking days along with form data validity. The development team achieved code reliability by testing individual program units while using JUnit framework under valid and extreme input conditions.

A combination of different check-in and check-out date scenarios including same-day reservations and invalid time periods was provided to test that stay duration calculations detected possible errors appropriately. The system validated name as well as contact number and payment reference entries to prevent invalid or incorrect data from making it to the database layer. The tests performed on individual units were essential during initial development stages to detect logical problems before other modules were integrated.

Integration Testing:

The integration tests checked the proper data flow and communication between the Java Swing GUI and the application logic and the MySQL database which used JDBC as the connector. Real user interactions with the system served as the basis for testing at this level to verify the application's operation as one unified system.

The testing procedure required users to perform a full booking workflow that included room reservations and MySQL data entry before validating proper storage and retrieval of the information in the MySQL database. The JDBC functionality received critical testing through integration testing to verify stable database connections along with proper SQL command construction and execution and appropriate handling of errors that included duplicate booking IDs and connection failures.

- Booking creation through the GUI: Filling in the booking form with sample data and submitting it through the interface.
- Database insertion via JDBC: Verifying that the booking details were accurately inserted into the bookings table.

- Viewing the booking in MySQL Workbench: Running SQL queries to confirm that the booking record exists in the database and matches the user input.

```

1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.*;
4 import java.text.ParseException;
5 import java.text.SimpleDateFormat;
6 import java.util.*;
7 import java.util.concurrent.TimeUnit;
8 import java.util.List;
9 import java.util.ArrayList;
10 import java.util.Calendar;
11 import java.sql.Connection;
12 import java.sql.PreparedStatement;
13 import java.sql.SQLException;
14 public class Hotel {
15     private JFrame mainFrame;
16     // Mock data for available rooms
17     private String[] roomTypes = {"Standard", "Deluxe", "Suite"};
18     private double[] roomPrices = {100.0, 150.0, 250.0};
19     // Data structures to store bookings and history
20     private Map<String, Booking> bookings = new HashMap<>();
21     private List<Booking> bookingHistory = new ArrayList<>();
22     public Hotel() {
23         prepareGUI();
24     }
25     private void prepareGUI() {
26         mainFrame = new JFrame("Smart Hotel Management System");
27         mainFrame.setSize(500, 350);
28         mainFrame.setLayout(new GridLayout(5, 1));
29         mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         JButton bookRoomBtn = new JButton("Book a Room");
31         JButton checkInBtn = new JButton("Check-in Guest");
32         JButton checkOutBtn = new JButton("Check-out Guest");
33         JButton historyBtn = new JButton("Booking History");
34         JButton exitBtn = new JButton("Exit");
35     }

```

Problems Javadoc Declaration Search Console

Hotel [Java Application] C:\Users\User\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2-

☑ Booking saved to MySQL!

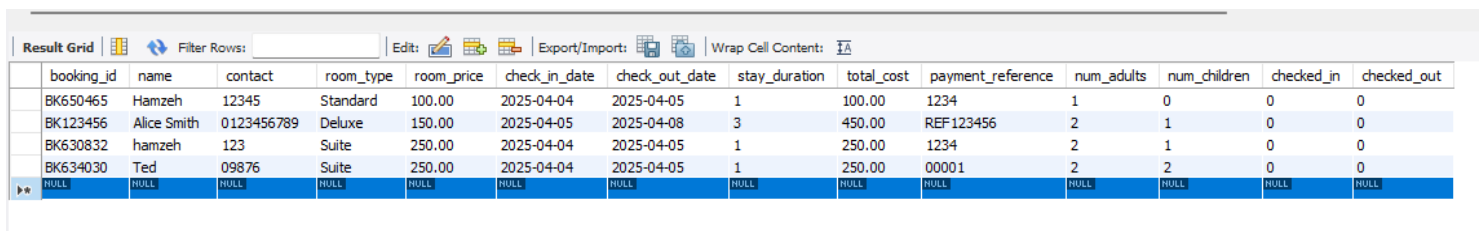
figure 7: Code behind the management system implementation

6.3 Performance evaluation and test case results

The Smart Hotel Management System underwent testing under normal conditions to verify system functionality together with its responsiveness along with data consistency. The system operated its essential modules without causing hardware failures or database corruption or data synchronization problems. The hotel booking data stored precisely in MySQL databases showed proper status modification through the graphical user interface.

A performance test used multiple booking testing scenarios that displayed full user interaction and inserted data into the database before finishing the process. The system operated successfully through more than 10 successive tests that produced regular system performance.

The GUI interface components displayed within one second of screen loading time. The JDBC platform enabled quick execution of MySQL database operations which included record insertions and retrievals in the local testing environment. System testing demonstrated complete stability because no connection problems or performance issues occurred during the examination phase.



booking_id	name	contact	room_type	room_price	check_in_date	check_out_date	stay_duration	total_cost	payment_reference	num_adults	num_children	checked_in	checked_out
BK650465	Hamzeh	12345	Standard	100.00	2025-04-04	2025-04-05	1	100.00	1234	1	0	0	0
BK123456	Alice Smith	0123456789	Deluxe	150.00	2025-04-05	2025-04-08	3	450.00	REF123456	2	1	0	0
BK630832	hamzeh	123	Suite	250.00	2025-04-04	2025-04-05	1	250.00	1234	2	1	0	0
BK634030	Ted	09876	Suite	250.00	2025-04-04	2025-04-05	1	250.00	00001	2	2	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

figure 8: Results Grid keeping record of all entries within the system

7. Challenges and Solutions

7.1 Technical Challenges Faced

- Integrating Java with MySQL: At the start Eclipse failed to detect the JDBC connector that integrated Java with MySQL. Trying to establish a connection between systems produced several errors.
- Real-time data update: After both booking and check-out processes the system automatically kept all status updates in real-time.

- GUI and database syncing: The process of maintaining database form input data consistency with MySQL storage demanded thorough backend method development and strict input validation.
- MySQL connector version mismatch: The use of MySQL 9.2.0 with compatible libraries caused a compatibility problem in connector versions.

7.2 Solutions Implemented

- The Eclipse build path received its JDBC .jar through manual addition so developers could create the DatabaseHelper.java class as their connection logic center.
- The system executed SQL insert/update commands in real-time after booking actions had been confirmed.
- GUI components sent data through structured channels to the database following ongoing validation which protected system data integrity.
- The project version compatibility worked due to proper configuration of Eclipse with the newest stable mysql-connector-j-9.2.0 version.

7.3 Lessons Learned

- Java developers need complete knowledge of JDBC to build connections between Java applications and database systems.
- Both server-side and client-side configurations need to be checked to fix debugging database connection errors during development.
- The separation of user interactions from database access logic will build structured code and prevent graphical user interface delays.
- The splitting of responsibilities between team members produces accelerated performance for problem resolution and execution.

8. Conclusion and Future Enhancements

8.1 Summary of Findings

The Smart Hotel Management System proves to be a transformative modern hotel infrastructure that applies automated digital technologies as its core foundation. The system boosts operational performance along with maintaining high quality standards in guest interactions. Automatic management of operations like room booking together with check-in/check-out procedures as well as billing and housekeeping tasks helps staff deliver better customized support to hotel guests. The hotel managers obtain enhanced trust and operational transparency through secure and transparent payment processing options. Online data analysis provides business leaders the opportunity to make strategic decisions through real-time data insights enabling better resource optimization and business outcomes. The system's centralized control framework decreases operational waiting times in hotel daily activities and enables modern hospitality environments focused on customer needs.

8.2 Potential Improvements and Scalability

The current benefits of the Smart Hotel Management System can be expanded through additional improvement features and scalability solutions. Smart Hotel Management System benefits from AI-driven guest recommendations as its main improvement method. Through analyzing guest actions the system develops tailored recommendations about dining options along with spa treatments and room enhancement possibilities. AI-based chatbots serve as a solution to handle regular inquiries which helps decrease human intervention while improving guest-technology interaction.

The transition to cloud-based management represents a crucial enhancement because it provides centralized control for operation of hotel chains across multiple properties. This arrangement would match data properly and improve operational control and create a smooth journey between different properties for guests. Cloud solutions grant better security measures while enabling expansion possibilities and remote accessibility for employees and front-office staff and management teams.

The inclusion of mobile-friendly features including mobile check-in and digital key access together with contactless payment options would make the hotel align itself with contemporary hospitality practices. These mobile-first services streamline operations while offering guests a more convenient and hygienic experience.

8.3 Future Research Directions

Potential future enhancements for the Smart Hotel Management System aim to enhance its functionality and scalability as well as improve user experience. New additions will both modernize the system framework and solve existing operational restrictions.

AI-driven personalization features constitute a major advancement direction that the system requires. The system can study guest choices together with booking activity to deliver customized suggestions for upgrades and spa services as well as dining options. Hotel operators can benefit from incorporating AI chatbots to handle routine customer requests and provide continuous support that frees up staff work hours.

Making the system operate on cloud-based infrastructure stands as one of the primary system upgrades. Centralized data management becomes feasible through this system especially for hotel chains operating multiple outlets. Cloud-based implementation enables users to access information more easily and provides reliable data backup alongside extensive scalability parameters that simplify expansion without restructuring major aspects of the operation.

The system would gain significant benefits from mobile platform improvements since these enhancements deliver better guest comfort and satisfaction experiences. Modernizing guest experience together with operational streamlining can be achieved through adding mobile check-in capability and digital room access functions and contactless payment features. The proposed enhancements create benefits that match what customers in the hospitality industry want while improving both efficiency levels and safety measures.

These future updates will capitalize on the existing system foundation to achieve enhanced operational flexibility and automated guest-programmed services and deliver improved guest service quality.

9. References

1. Hotel Management System Overview - Smith, J. (2022). Hotel Operations and Management. Pearson Publications.
2. Java Programming for Enterprise Applications - Lee, K. (2021). Java for Beginners to Experts. McGraw Hill.
3. MySQL Database Design Principles - Brown, T. (2020). SQL & Databases. O'Reilly Media.
4. Modern Hospitality Industry Challenges - Johnson, R. (2023). The Digital Transformation of Hotels. Springer.
5. Comparison of Hotel Management Systems - Research Paper: "A Study on Hotel Management Software," International Journal of Computer Science, Vol. 15, No. 2, 2023
6. Stringam, B. B., & Gerdes, J. H. (2021). Hotel and guest room technology. University of South Florida (USF) M3 Publishing, 17(9781732127593), 6.
7. Dzikria, I., & Solihin, M. L. S. (2023). The Role of Task-Technology Fit on the Design and Use of a Hotel Management System. Journal of Information Technology and Cyber Security, 1(2), 41-52.
8. Chen, G. (2018, February). Analysis and Design of Five-Star Hotel Management Information System Based on UML. In 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (pp. 104-109). IEEE.
9. Tie, J., Jin, J., & Wang, X. (2011, July). Study on application model of three-tiered architecture. In 2011 Second International Conference on Mechanic Automation and Control Engineering (pp. 7715-7718). IEEE.
10. Senjaya, R., Sweetania, D., Herawati, M. S., Sularsih, P., & Rosemalatriasari, A. (2023). Designing UI/UX Reservation Application Savero Hotel Depok With Design Thinking Method Using Figma. Jurnal Inovatif: Inovasi Teknologi Informasi dan Informatika, 6(1), 1-14.
11. Lockyer, C., & Scholarios, D. (2004). Selecting hotel staff: why best practice does not always work. International journal of contemporary hospitality management, 16(2), 125-135.
12. Rodríguez-Antón, J. M., del Mar Alonso-Almeida, M., Celemín, M. S., & Rubio, L. (2012). Use of different sustainability management systems in the hospitality industry. The case of Spanish hotels. Journal of Cleaner Production, 22(1), 76-84.
13. Koh, W. S., & Hassim, Y. M. M. (2021). Hotel reservation management system. Applied Information Technology And Computer Science, 2(2), 973-992.

Dyshkantiuk, O., Salamatina, S., Polishchuk, L., Komarnytskyi, I., Tserklevych, V., & Nedobiichuk, T. (2020). Modern hotel business management tools. *International Journal of Advanced Research in Engineering and Technology*, 11(6).

14. Course Hero 3-software requirements notes.pdf - Software Engineering and Development; Software Requirements. The Hong Kong Polytechnic University.COMP.COMP1.advchiwaic.12/19/2021.3-software_requirements_notes.pdf <https://www.coursehero.com/file/123449172/3-software-requirements-notespdf/>
15. Dzikria, I., & Solihin, M. L. S. (2023). The Role of Task-Technology Fit on the Design and Use of a Hotel Management System. *Journal of Information Technology and Cyber Security*, 1(2), 41-52.
16. Baker, S., & Bradley, P. (2014). *Principles of Hotel Front Office Operations*. Cengage Learning.
17. Ivanov, S. (2014). *Hotel Revenue Management: From Theory to Practice*. Zangador.
18. Kasavana, M. L., & Cahill, J. J. (2010). *Managing Technology in the Hospitality Industry*. American Hotel & Lodging Educational Institute.
19. Law, R., Buhalis, D., & Cobanoglu, C. (2014). *Progress on Information and Communication Technologies in Hospitality and Tourism*. *International Journal of Contemporary Hospitality Management*, 26(5), 727-750.
20. Singh, A. (2021). *How Smart Hotel Technology is Transforming the Hospitality Industry*. HospitalityNet. Retrieved from www.hospitalitynet.org
21. TechRadar. (2023). *Best Hotel Management Software for 2024*. Retrieved from www.techradar.com
22. Forbes. (2022). *The Future of Smart Hotels and AI in Hospitality*. Retrieved from www.forbes.com
23. PwC. (2021). *Hospitality Industry Insights: The Role of Digital Transformation in Hotels*. Retrieved from www.pwc.com
24. Statista. (2023). *Hotel Industry Digitalization Trends & Market Growth*. Retrieved from www.statista.com

RUBRIC:

	Unsatisfactory (0 to 4 marks)	Satisfactory (5 to 6marks)	Good (7 to 8marks)	Excellent (9 to 10 marks)	Marks Scored
Delivery PLO1	<ul style="list-style-type: none"> Completed less than 70% of the requirements. Not delivered on time or not in correct format 	<ul style="list-style-type: none"> Completed between 70-80% of the requirements. Delivered on time, and in correct 	<ul style="list-style-type: none"> Completed between 80-90% of the requirements. Delivered on time, and in correct format 	<ul style="list-style-type: none"> Completed between 90-100% of the requirements. Delivered on time, and in correct format 	
Documentation Algorithm PLO1 & PLO2	<ul style="list-style-type: none"> No or incomplete documentation included. 	<ul style="list-style-type: none"> Basic documentation has been completed including descriptions of all variables. Purpose is noted for each method. Little to non discussion and analysis is provided 	<ul style="list-style-type: none"> Clearly documented including descriptions of all variables. Specific purpose is noted for each algorithm and data structure. Acceptable discussion and analysis. 	<ul style="list-style-type: none"> Clearly and effectively documented including descriptions of all variables. Specific purpose is noted for each method, algorithm, data structure, input requirements, and output results. discussion and analysis is provided 	
Coding Standards PLO2	<ul style="list-style-type: none"> No name, date, or experiment title included Poor use of white space (indentation, blank lines). Disorganized and messy 	<ul style="list-style-type: none"> Includes name, date, and experiment title. White space makes program easy to read. Organized work. 	<ul style="list-style-type: none"> Includes name, date, and experiment title. Good use of white space. Organized work. Good use of variables 	<ul style="list-style-type: none"> Includes name, date, and experiment title. Excellent use of white space. Creatively organized work. Excellent use of variables (Unambiguous 	

	Poor use of variables • (ambiguous naming).	• Good use of variables • (unambiguous naming).	(unambiguous naming)	naming).	
Total					

Presentation (PLO1, PLO2):

	Unsatisfactory (0 to 2 marks)	Satisfactory (3 marks)	Good (4 marks)	Excellent (5 marks)	Marks Scored
Comprehension	<ul style="list-style-type: none"> • Presenters did not understand the topic. • Majority of questions answered by only one member or majority of information incorrect. 	<ul style="list-style-type: none"> • Few members showed good understanding of some parts of the topic. • Only some members accurately answered questions. 	<ul style="list-style-type: none"> • Most showed a good understanding of the topic. • All members able to answer most of audience questions. 	<ul style="list-style-type: none"> • Extensive knowledge of the topic. • Members showed complete understanding of assignment. Accurately answered all questions posed. 	
Presentation Skills	<ul style="list-style-type: none"> • Minimal eye contact by more than one member focusing on small part of audience. • The audience was not engaged. • Majority of presenters spoke too quickly or quietly making it difficult to understand. • 	<ul style="list-style-type: none"> • Members focused on only part of audience. • Sporadic eye contact by more than one presenter. • The audience was distracted. • Speakers could be heard by only half of the audience. • Body language was 	<ul style="list-style-type: none"> • Most members spoke to majority of audience; steady eye contact. • The audience was engaged by the presentation. • Majority of presenters spoke at a suitable volume. 	<ul style="list-style-type: none"> • Regular/constant eye contact, The audience was engaged, and presenters held the audience's attention. • Appropriate speaking volume & body language. 	

	Inappropriate/disinterested body language.	distracting.	<ul style="list-style-type: none"> Some fidgeting by member(s). 		
Contents	<ul style="list-style-type: none"> The presentation was a brief look at the topic but many questions were left unanswered. Majority of information irrelevant and significant points left out. 	<ul style="list-style-type: none"> The presentation was informative but several elements went unanswered. Much of the information irrelevant; coverage of some of major points. 	<ul style="list-style-type: none"> The presentation was a good summary of the topic. Most important information covered; little irrelevant info. 	<ul style="list-style-type: none"> The presentation was a concise summary of the topic with all questions answered. Comprehensive and complete coverage of information. 	
Total					