A blue and white logo

Description automatically generated

**School of IT & Business Technologies**

**Bachelor of Business Information Management Cover Sheet and Student Declaration**

This sheet must be signed by the student and attached to the submitted assessment.

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Title:** | **BBIM502 Introduction to Programming** | **Course code:** | **BBIM502** |
| **Student Name:** | **Gurpreet Singh** | **Student ID:** | 764706866 |
| **Assessment No & Type:** | **Assessment 2**  **Project** | **Cohort:** | BBIM7124C |
| **Due Date:** | 14/07/2024 | **Date**  **Submitted:** | 14/07/2024 |
| **Tutor’s Name:** | **Mr Harsh Tiwari** |  |  |
| **Assessment** **Weighting** | 40% |  |  |
| **Total Marks** | 100 |  |  |

**Student Declaration:**

I declare that:

* I have read the New Zealand School of Education Ltd policies and regulations on assessments and understand what plagiarism is.
* I am aware of the penalties for cheating and plagiarism as laid down by the New Zealand School of Education Ltd.
* This is an original assessment and is entirely my own work.
* Where I have quoted or made use of the ideas of other writers, I have acknowledged the source.
* This assessment has been prepared exclusively for this course and has not been or will not be submitted as assessed work in any other course.
* It has been explained to me that this assessment may be used by NZSE Ltd, for internal and/or external moderation.
* If I am late in handing in this assessment without prior approval (see student regulations in handbook), marks will be deducted, to a maximum of 50%.

**Task1**

**1. Add Room:**

**Coding:**

A computer code with black text

Description automatically generated with medium confidence

**Calling Function:**



**Output:**

A screenshot of a computer

Description automatically generated

**Description**

**By entering the room number, type, and price, the user can add a new room to the list of rooms. The add room method includes validation to make sure the room number is distinct, the room type is valid, and the price is a number.**

**2. Delete Room:**

**Coding:**

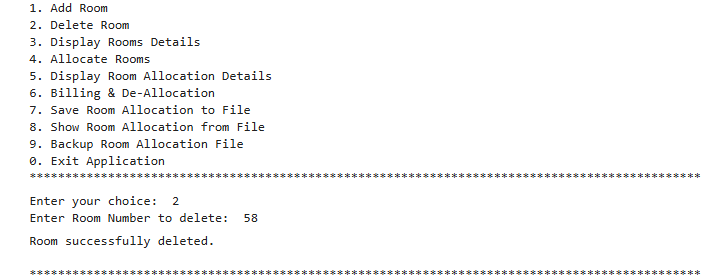
A computer code with colorful text

Description automatically generated

**Calling Function:**



**Output:**



**Description:**

**A room's number can be entered into the delete\_room method to remove it from the list. Together with handling any potential exceptions, it includes validation to make sure the room actually exists.**

**3. Display Room:**

**Coding:**

A close-up of a white background

Description automatically generated

**Calling Function:**



**Output:**

A screen shot of a computer

Description automatically generated

**Description**

**Each room in the list, along with its number, kind, cost, and availability, is printed out in full by the display\_rooms method. As part of managing any exceptions that may occur during the process, it involves error handling.**

**4. Allocate Rooms:**

**Coding:**

A computer screen shot of a program

Description automatically generated

**Calling Function**:



**Output:**

A line of text on a white background

Description automatically generated

**Description**

**Entering the guest's name and room number into the allocate\_room method enables the user to assign a room to them. Once the room's existence and availability have been confirmed, the guest name is assigned and the room's availability status is updated.**

**5. Display Room Allocation Details:**

**Coding:**



**Calling Function:**



**Output**:

A white rectangular frame with blue and black text

Description automatically generated

**Description**

**When a room is allocated but not yet available, the display\_allocation\_details method prints the guest name and room number. For the purpose of addressing any exceptions that may arise along the procedure, it has error handling.**

**6. Billing and De-Allocation:**

**Coding:**

A computer screen shot of a code

Description automatically generated

**Calling Function:**



**Output:**

A screen shot of a computer

Description automatically generated

**Description**

**By inputting the room number, the bill\_and\_deallocate method takes care of charging and deallocating a room. It modifies the availability state of the room, publishes the billing information, and removes the guest's name.**

**7. Allocation to File:**

**Coding:**

A computer code with text

Description automatically generated with medium confidence

**Calling Function:**



**Output:**

A white background with blue and black text

Description automatically generated

**Description**

**Details of rooms allotted are written to a file that is given by self.filename using the save\_room\_allocation\_to\_file method. It covers handling exceptions that may arise during the process, including IOErrors that arise when trying to open or write to the file**.

**8. Allocation From File:**

**Coding:**

A screen shot of a computer

Description automatically generated

**Calling Function:**



**Output:**

A white rectangular frame with blue and black text

Description automatically generated

**Description**

**Details about room allocation are read and displayed by the show\_room\_allocation\_from\_file method from a file that is supplied by self. filename. Various exceptions are handled by it, including normal exceptions that may arise throughout the file reading process, IOError for problems with file operations, and FileNotFoundError when the file doesn't exist.**

**9. Backup Room Allocation File:**

**Coding:**

A screen shot of a computer code

Description automatically generated

**Calling Function:**



**Output:**

A white background with a black text

Description automatically generated with medium confidence

**Description**

**A backup of the room allocation data is produced using the backup\_room\_allocation method. It reads from self.filename first, then writes the contents of the original file to the backup file and produces a new backup file with a timestamped name. The original file (self.filename) is then cleared by opening it in write mode and writing an empty string.**

**Task 3**

**1.**

**Introduction**

The Langham Hotel Management System (LHMS) is a comprehensive program application outlined to streamline and mechanize the fundamental operations of a lodging. This framework is expecting to supplant manual strategies, advertising a more effective and user-friendly approach to overseeing inn room reservations, assignments, charging, and client data. The application is outlined with straightforwardness and strength in intellect, guaranteeing that indeed clients with negligible specialized ability can explore and utilize the framework viably.

**Purpose**

The essential reason of the Langham Hotel Management System (LHMS) is to upgrade the operational productivity of inn administration by mechanizing and rearranging different regulatory errands. The application points to accomplish the taking after objectives:

**2. Pseudocode**

CLASS Room

FUNCTION \_\_init\_\_(room\_number, room\_type, cost)

self.room\_number = room\_number

self.room\_type = room\_type

self.room\_cost = cost

self.is\_available = True

self.customer\_name = None

CLASS HotelMgmtSystem

FUNCTION \_\_init\_\_()

self.list\_of\_rooms = []

self.filename = "LHMS\_764706866.txt"

CALL ensure\_file\_exists()

FUNCTION ensure\_file\_exists()

TRY

IF NOT os.path.exists(self.filename) THEN

OPEN self.filename with mode 'w' AS file

WRITE "" TO file

ENDIF

EXCEPT IOError

PRINT "An IOError occurred while ensuring the file exists."

FUNCTION add\_room()

TRY

READ room\_number FROM user input

IF room\_number ALREADY EXISTS IN self.list\_of\_rooms THEN

PRINT "Room number already exists. Please enter a unique room number."

RETURN

ENDIF

READ room\_type FROM user input

IF room\_type NOT IN ["Single", "Double", "Suite"] THEN

PRINT "Invalid room type. Please enter Single, Double, or Suite."

RETURN

ENDIF

READ cost\_input FROM user input

IF cost\_input IS EMPTY THEN

PRINT "Room price cannot be empty."

RETURN

ENDIF

cost = PARSE cost\_input AS float

CREATE new Room instance WITH room\_number, room\_type, cost

APPEND new Room instance TO self.list\_of\_rooms

PRINT "Room successfully added."

EXCEPT ValueError

PRINT "Invalid input for room cost. Please enter a numeric value."

EXCEPT SyntaxError

PRINT "Syntax error detected!"

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION delete\_room()

TRY

READ room\_number FROM user input

IF room\_number DOES NOT EXIST IN self.list\_of\_rooms THEN

PRINT "Room number does not exist."

RETURN

ENDIF

CREATE new list\_of\_rooms WITHOUT room\_number

PRINT "Room successfully deleted."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION display\_rooms()

TRY

FOR EACH room IN self.list\_of\_rooms DO

PRINT "Number:", room.room\_number, "Type:", room.room\_type, "Price:", room.room\_cost, "Available:", room.is\_available

ENDFOR

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION allocate\_room()

TRY

READ room\_number FROM user input

FOR EACH room IN self.list\_of\_rooms DO

IF room.room\_number EQUALS room\_number THEN

IF room.is\_available THEN

READ customer\_name FROM user input

SET room.customer\_name TO customer\_name

SET room.is\_available TO False

PRINT "Room successfully allocated."

RETURN

ELSE

PRINT "Room is not available."

RETURN

ENDIF

ENDIF

ENDFOR

PRINT "Room number does not exist."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION display\_allocation\_details()

TRY

FOR EACH room IN self.list\_of\_rooms DO

IF NOT room.is\_available THEN

PRINT "Number:", room.room\_number, "Guest:", room.customer\_name

ENDIF

ENDFOR

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION bill\_and\_deallocate()

TRY

READ room\_number FROM user input

FOR EACH room IN self.list\_of\_rooms DO

IF room.room\_number EQUALS room\_number THEN

IF NOT room.is\_available THEN

PRINT "Billing: Number:", room.room\_number, "Guest:", room.customer\_name, "Price:", room.room\_cost

SET room.is\_available TO True

SET room.customer\_name TO None

PRINT "Room successfully deallocated."

RETURN

ELSE

PRINT "Room is already available."

RETURN

ENDIF

ENDIF

ENDFOR

PRINT "Room number does not exist."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION save\_room\_allocation\_to\_file()

TRY

OPEN self.filename with mode 'w' AS file

FOR EACH room IN self.list\_of\_rooms DO

IF NOT room.is\_available THEN

WRITE "Room Number:", room.room\_number, "Guest:", room.customer\_name, "Type:", room.room\_type, "Price:", room.room\_cost TO file

PRINT "Room allocations saved to file."

EXCEPT IOError

PRINT "An IOError occurred while saving room allocations to the file."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION show\_room\_allocation\_from\_file()

TRY

OPEN self.filename with mode 'r' AS file

content = READ file

PRINT "Room Allocation from File:"

PRINT content

EXCEPT FileNotFoundError

PRINT "The file was not found."

EXCEPT IOError

PRINT "An IOError occurred while reading from the file."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION backup\_room\_allocation()

TRY

SET backup\_filename TO "LHMS\_Studentid\_Backup\_" + datetime.now().strftime('%Y%m%d\_%H%M%S') + ".txt"

OPEN self.filename with mode 'r' AS original\_file

content = READ original\_file

OPEN backup\_filename with mode 'w' AS backup\_file

WRITE content TO backup\_file

OPEN self.filename with mode 'w' AS original\_file

WRITE "" TO original\_file

PRINT "Backup created:", backup\_filename

EXCEPT FileNotFoundError

PRINT "The file was not found."

EXCEPT IOError

PRINT "An IOError occurred while creating a backup."

EXCEPT Exception AS e

PRINT "An error occurred:", e

FUNCTION run()

PRINT "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

PRINT " LANGHAM HOTEL MANAGEMENT SYSTEM"

PRINT " MENU"

WHILE True DO

PRINT "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

PRINT "1. Add Room"

PRINT "2. Delete Room"

PRINT "3. Display Rooms Details"

PRINT "4. Allocate Rooms"

PRINT "5. Display Room Allocation Details"

PRINT "6. Billing & De-Allocation"

PRINT "7. Save Room Allocation to File"

PRINT "8. Show Room Allocation from File"

PRINT "9. Backup Room Allocation File"

PRINT "0. Exit Application"

PRINT "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

TRY

READ choice FROM user input AS integer

IF choice EQUALS 1 THEN

CALL add\_room()

ELSE IF choice EQUALS 2 THEN

CALL delete\_room()

ELSE IF choice EQUALS 3 THEN

CALL display\_rooms()

ELSE IF choice EQUALS 4 THEN

CALL allocate\_room()

ELSE IF choice EQUALS 5 THEN

CALL display\_allocation\_details()

ELSE IF choice EQUALS 6 THEN

CALL bill\_and\_deallocate()

ELSE IF choice EQUALS 7 THEN

CALL save\_room\_allocation\_to\_file()

ELSE IF choice EQUALS 8 THEN

CALL show\_room\_allocation\_from\_file()

ELSE IF choice EQUALS 9 THEN

CALL backup\_room\_allocation()

ELSE IF choice EQUALS 0 THEN

PRINT "Exiting application..."

BREAK

ELSE

PRINT "Invalid choice! Please try again."

ENDIF

EXCEPT ValueError

PRINT "Invalid input! Please enter a numeric value."

EXCEPT SyntaxError

PRINT "Syntax error detected!"

EXCEPT IndexError

PRINT "Index out of range!"

EXCEPT NameError

PRINT "Variable name not defined!"

EXCEPT TypeError

PRINT "Invalid data type!"

EXCEPT OverflowError

PRINT "Numeric calculation exceeded limits!"

EXCEPT EOFError

PRINT "End of file reached unexpectedly!"

EXCEPT ImportError

PRINT "Module not found!"

EXCEPT Exception AS e

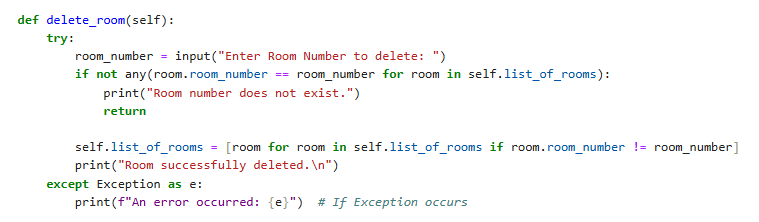
PRINT "An unexpected error occurred:", e

### 

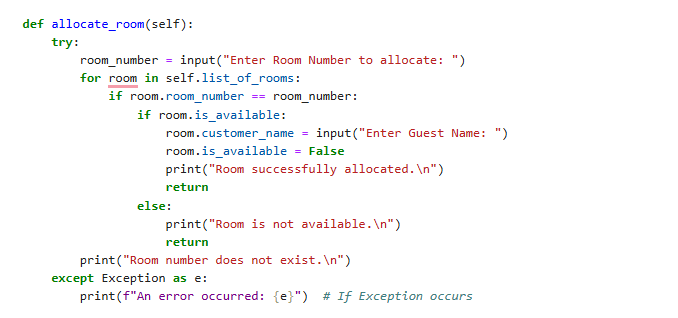
**3. Best Coding Practices and Standards**

1. **Comments**

**Header comments:**

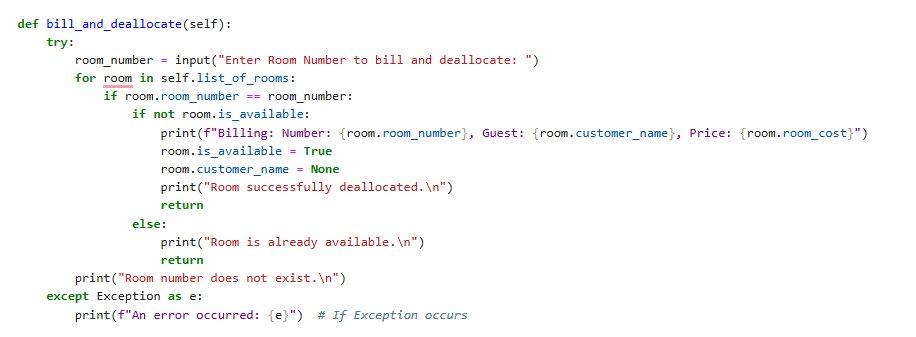
****

**Single Line and Multiple Line comment:**

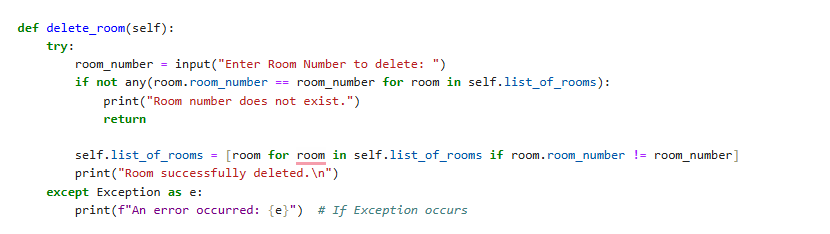
****

1. **Identifier Naming:**

**Variable and Function Names:**

****

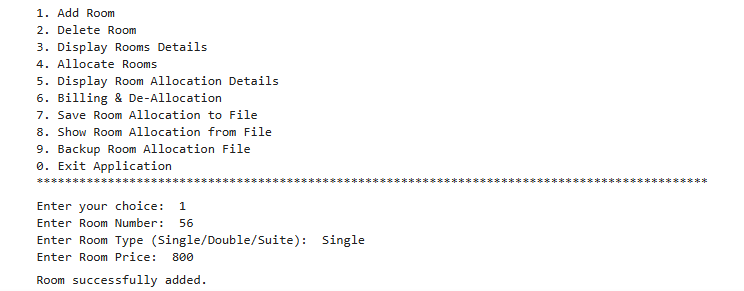
1. **Indentation and Syntax**

****

1. **Documenting Function**
2. **GitHub Repository Evidence:**

## User Guide:

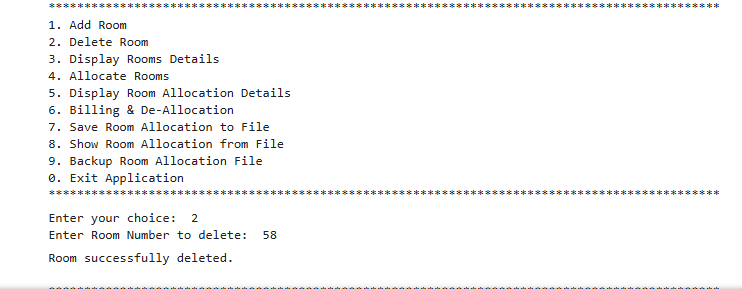
**1. Add Rooms:**



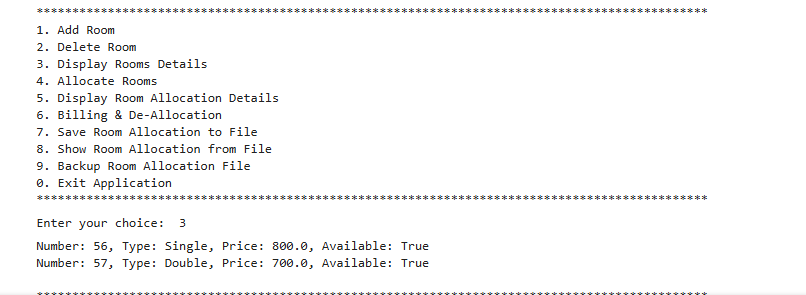
A screen shot of a computer

Description automatically generated

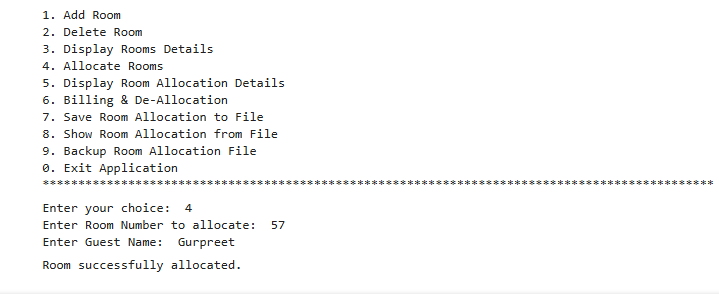
**2. Delete Room:**



**3. Display Room Details:**



**4. Allocate room:**



**5. Display Room Allocation details:**

A white background with blue and black text

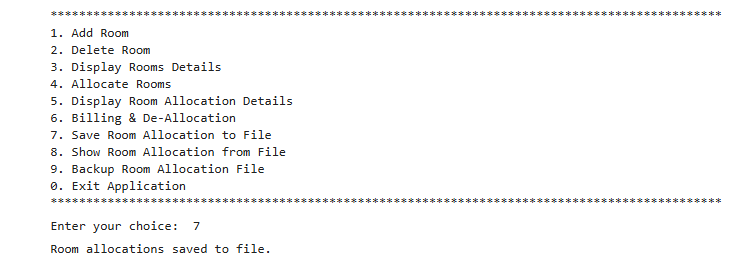
Description automatically generated

**6. Billing and De-allocation**:

A screen shot of a computer

Description automatically generated

**7. Save room Allocation to file:**

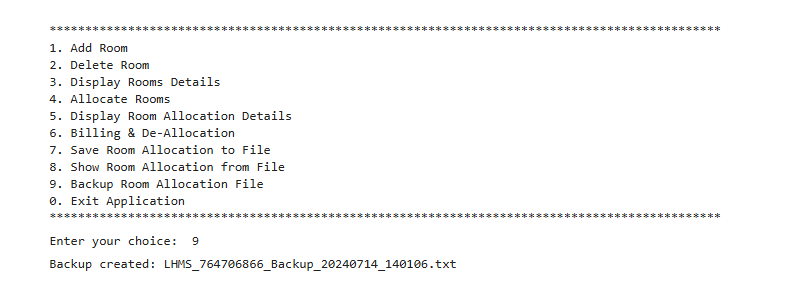


**8. Show Room Allocation from file:**

A white rectangular frame with blue and black text

Description automatically generated

**9. Backup File:**



**Self-reflection**

Structured Error Handling: Implemented try-except blocks across methods to handle specific exceptions like Value Error and File Not Found Error. This ensures robust error management, preventing crashes and providing clear feedback to users.

Input Validation: Enhanced input validation for room numbers, types, and costs to enforce data integrity. This ensures that only valid inputs are accepted, improving overall reliability and user experience.

Refined File Handling: Introduced dedicated functions for file operations, including creation, reading, writing, and backup of room allocation data. This promotes code clarity, reduces redundancy, and ensures consistent file management practices.

Documentation and Comments: Documented each method with clear comments explaining functionality, parameters, and exception handling. This enhances code readability, facilitates maintenance, and improves collaboration among developers.

**Student signature:** **A close-up of a sign

Description automatically generated**

**(Gurpreet Singh)**

**Date: 14/07/2024**

|  |  |  |
| --- | --- | --- |
| **Tutor only to complete** |  |  |
| **Assessment result:** | **Mark /100** | **Grade** |

© NZSE - BBIM505 Professional Practice and Communications Assessment 1 [Report] v1.4