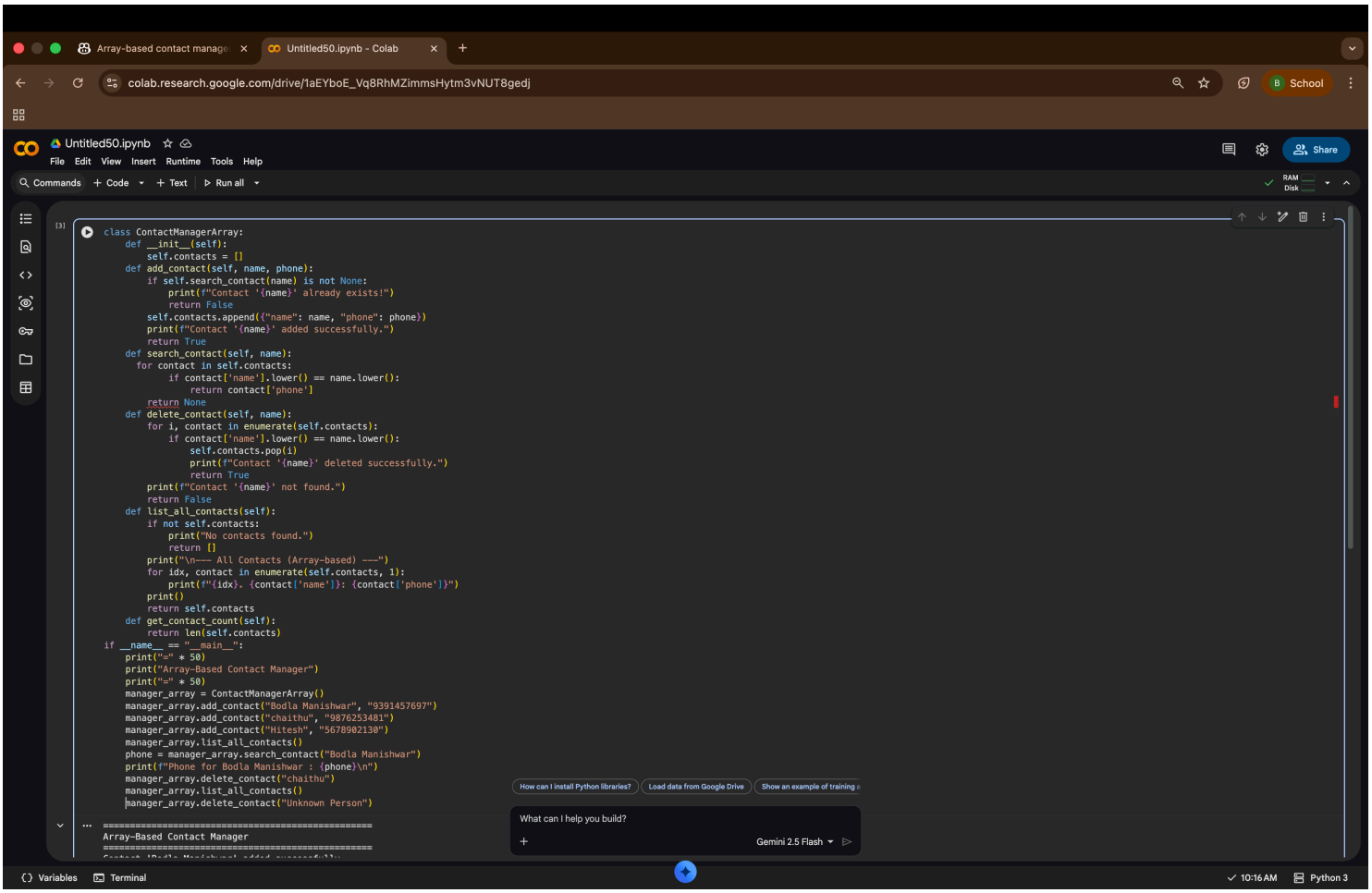
ASSIGNMENT-11

NAME: B.MANISHWAR
HALL TICKET:2303A51276
BATCH:05

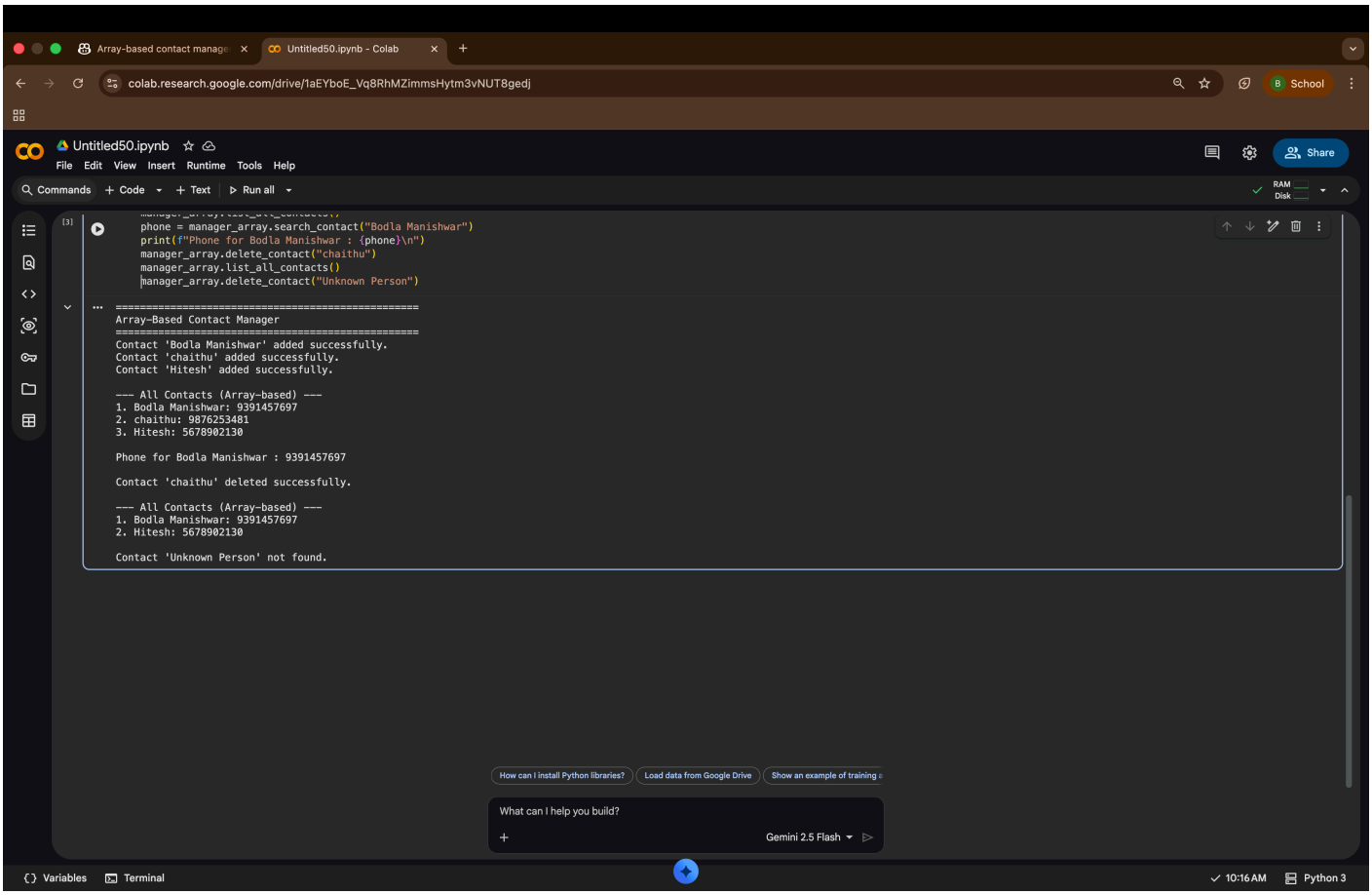
Task 1

PROMPT:
Generate a Python Smart Contact Manager application for a student club.Implement it using both an array (Python list) and a linked list.Each contact should store a name and phone number.Include methods to add, search, and delete contacts in both implementations.

CODE:



OUTPUT:



CODE EXPLANATION:

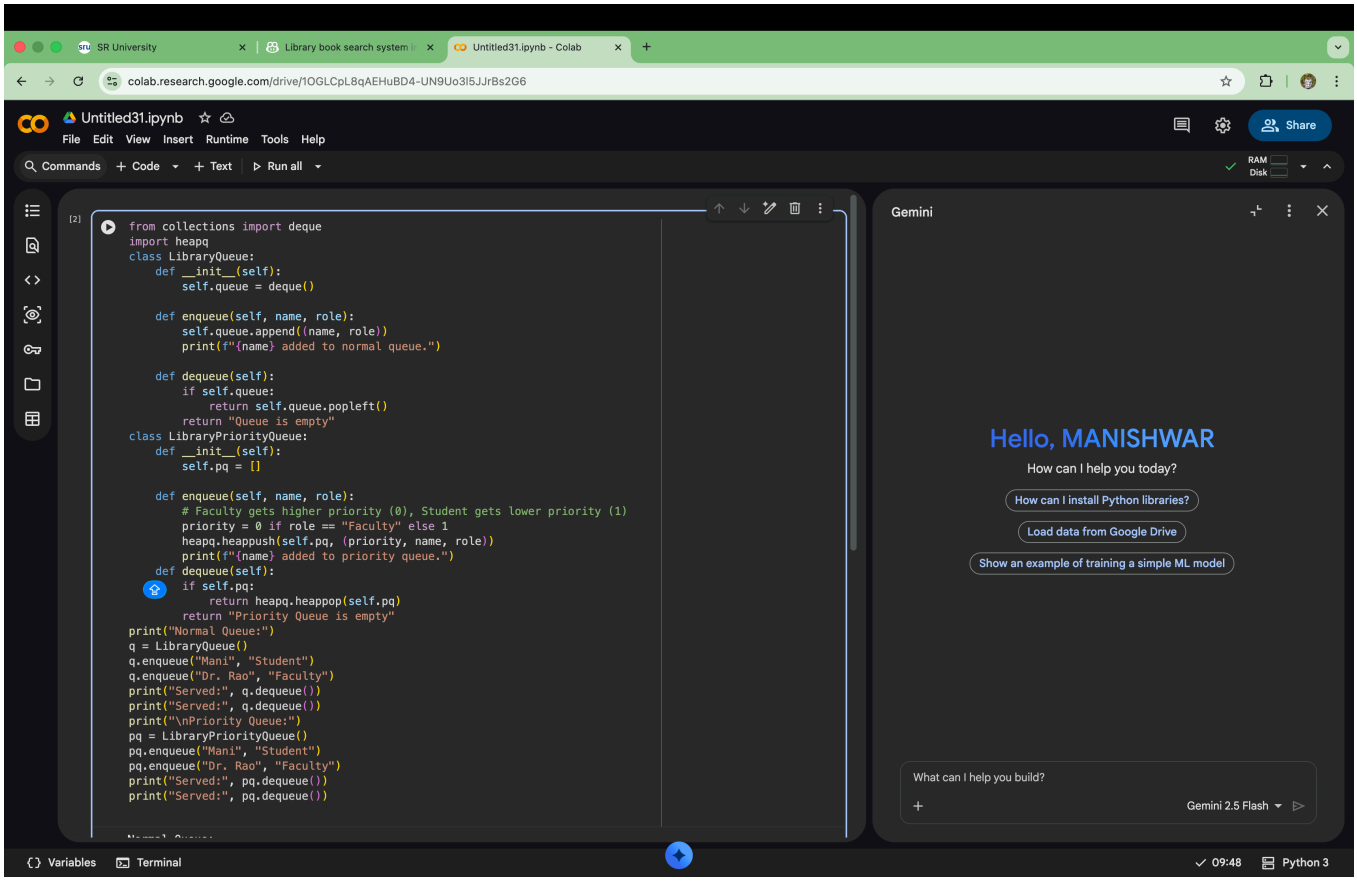
1) The program creates a Contact Manager using a Python list to store contacts as dictionaries with name and phone number.
2) The “add_contact()” method adds a new contact to the list using the “append()” function.
3) The “search_contact()” method checks each contact one by one to find a matching name.
4) The “delete_contact()” method finds the contact and removes it from the list using the `del` statement.
5) Adding is fast (O(1)), but searching and deleting take more time (O(n)) because the list must be scanned.

TASK 2:

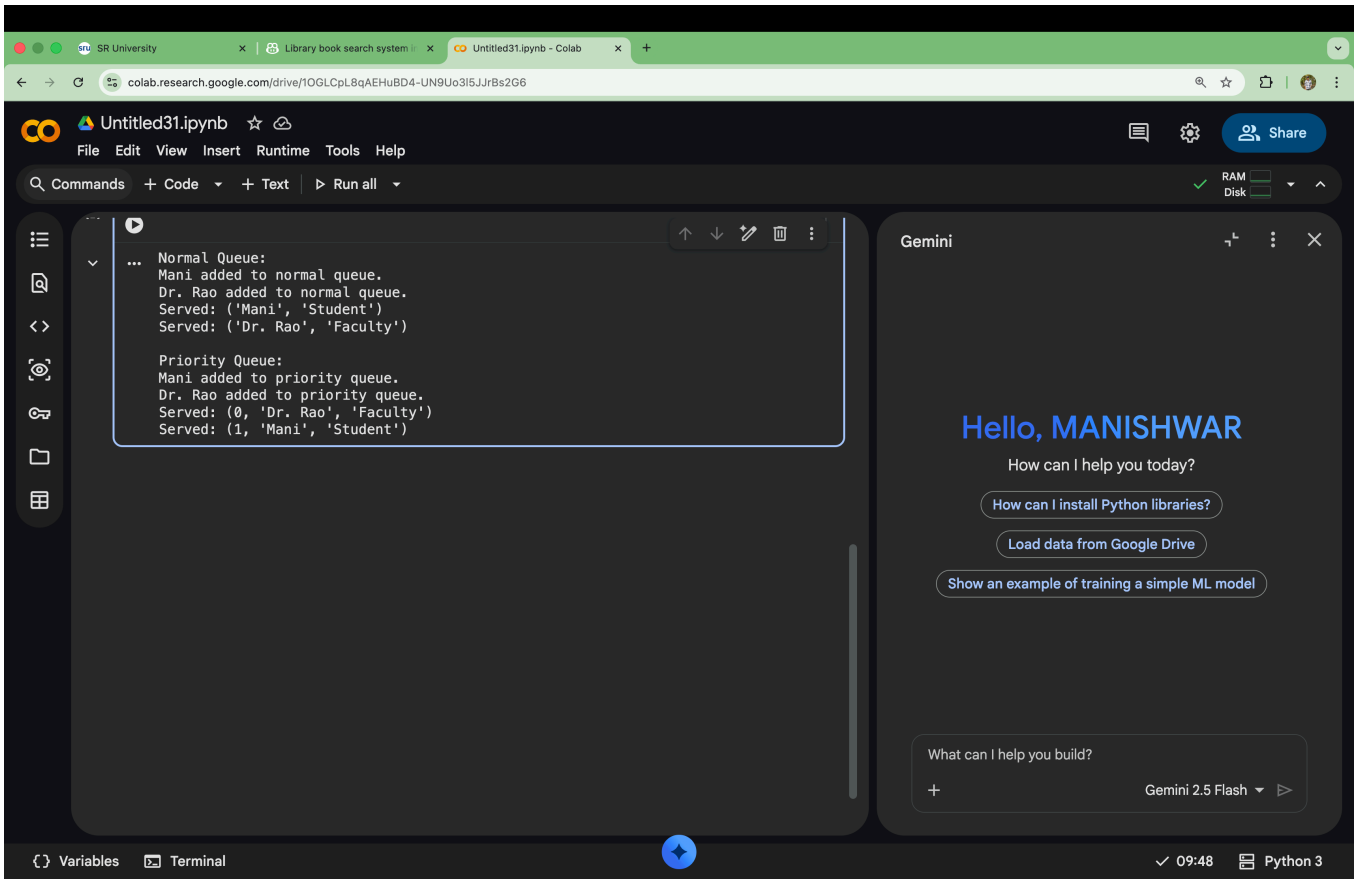
PROMPT:

Generate a Python Library Book Request System using Queue and Priority Queue.Implement enqueue() and dequeue() methods and prioritize faculty requests over student requests with test cases.

CODE:


SR University
Library book search system
Untitled31.ipynb - Colab
colab.research.google.com/drive/1OGLCpL8qAEHuBD4-UN9Uo3l5JJrBs2G6
Untitled31.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[2]
from collections import deque
import heapq
class LibraryQueue:
def __init__(self):
self.queue = deque()
def enqueue(self, name, role):
self.queue.append((name, role))
print(f"{name} added to normal queue.")
def dequeue(self):
if self.queue:
return self.queue.popleft()
return "Queue is empty"
class LibraryPriorityQueue:
def __init__(self):
self.pq = []
def enqueue(self, name, role):
# Faculty gets higher priority (0), Student gets lower priority (1)
priority = 0 if role == "Faculty" else 1
heapq.heappush(self.pq, (priority, name, role))
print(f"{name} added to priority queue.")
def dequeue(self):
if self.pq:
return heapq.heappop(self.pq)
return "Priority Queue is empty"
print("Normal Queue:")
q = LibraryQueue()
q.enqueue("Mani", "Student")
q.enqueue("Dr. Rao", "Faculty")
print("Served:", q.dequeue())
print("Served:", q.dequeue())
print("\nPriority Queue:")
pq = LibraryPriorityQueue()
pq.enqueue("Mani", "Student")
pq.enqueue("Dr. Rao", "Faculty")
print("Served:", pq.dequeue())
print("Served:", pq.dequeue())
Gemini
Hello, MANISHWAR
How can I help you today?
How can I install Python libraries?
Load data from Google Drive
Show an example of training a simple ML model
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
09:48
Python 3

OUTPUT:


SR University
Library book search system
Untitled31.ipynb - Colab
colab.research.google.com/drive/1OGLCpL8qAEHuBD4-UN9Uo3l5JJrBs2G6
Untitled31.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
Normal Queue:
Mani added to normal queue.
Dr. Rao added to normal queue.
Served: ('Mani', 'Student')
Served: ('Dr. Rao', 'Faculty')
Priority Queue:
Mani added to priority queue.
Dr. Rao added to priority queue.
Served: (0, 'Dr. Rao', 'Faculty')
Served: (1, 'Mani', 'Student')
Gemini
Hello, MANISHWAR
How can I help you today?
How can I install Python libraries?
Load data from Google Drive
Show an example of training a simple ML model
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
09:48
Python 3

CODE EXPLANATION:

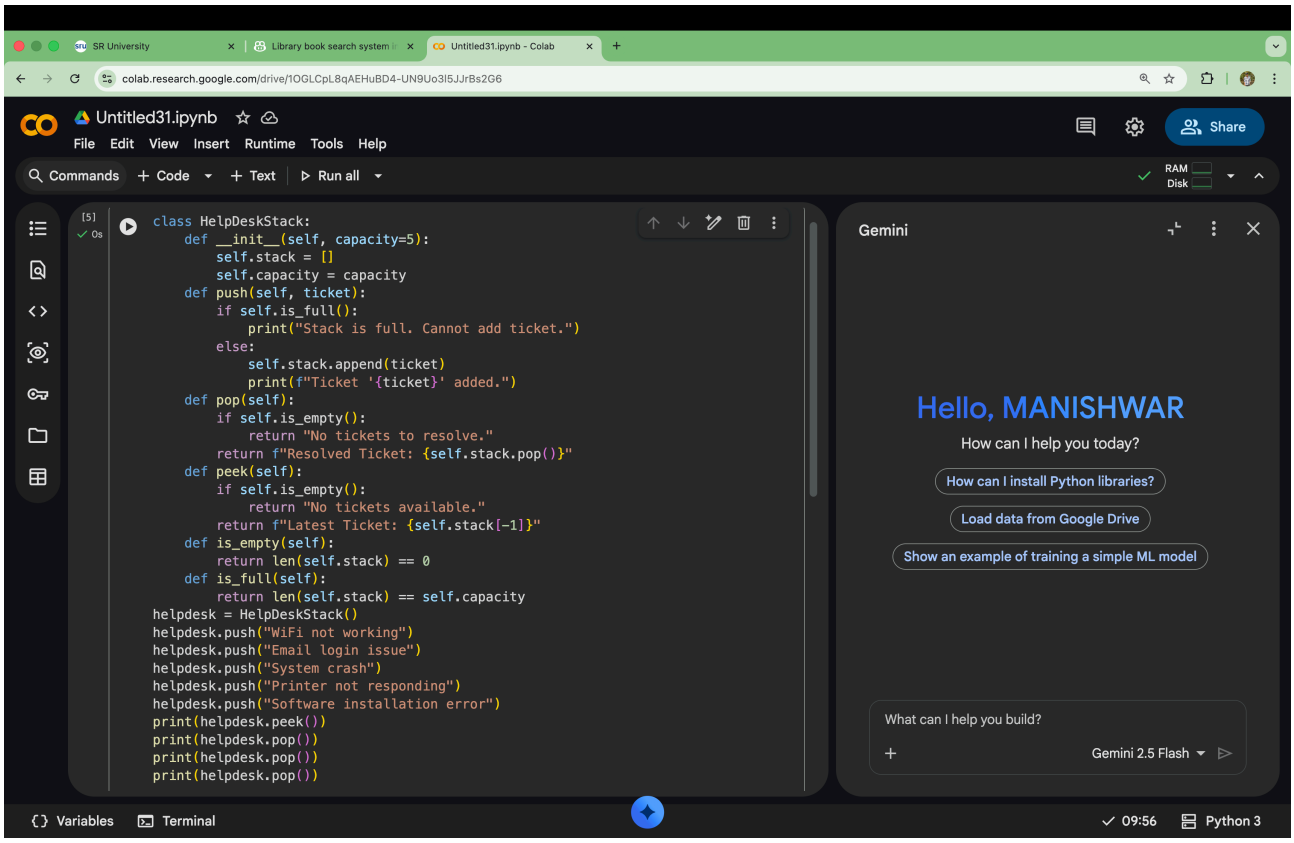
1) The normal queue uses FIFO (First-In-First-Out), so requests are served in the order they arrive.
2) The enqueue() method adds requests to the queue, and dequeue() removes the first request.
3) The priority queue uses the heapq module to assign priority values.
4) Faculty requests are given higher priority (0) than student requests (1), so they are served first.
5) This ensures correct prioritization while maintaining efficient insertion and removal operations.

TASK 3:

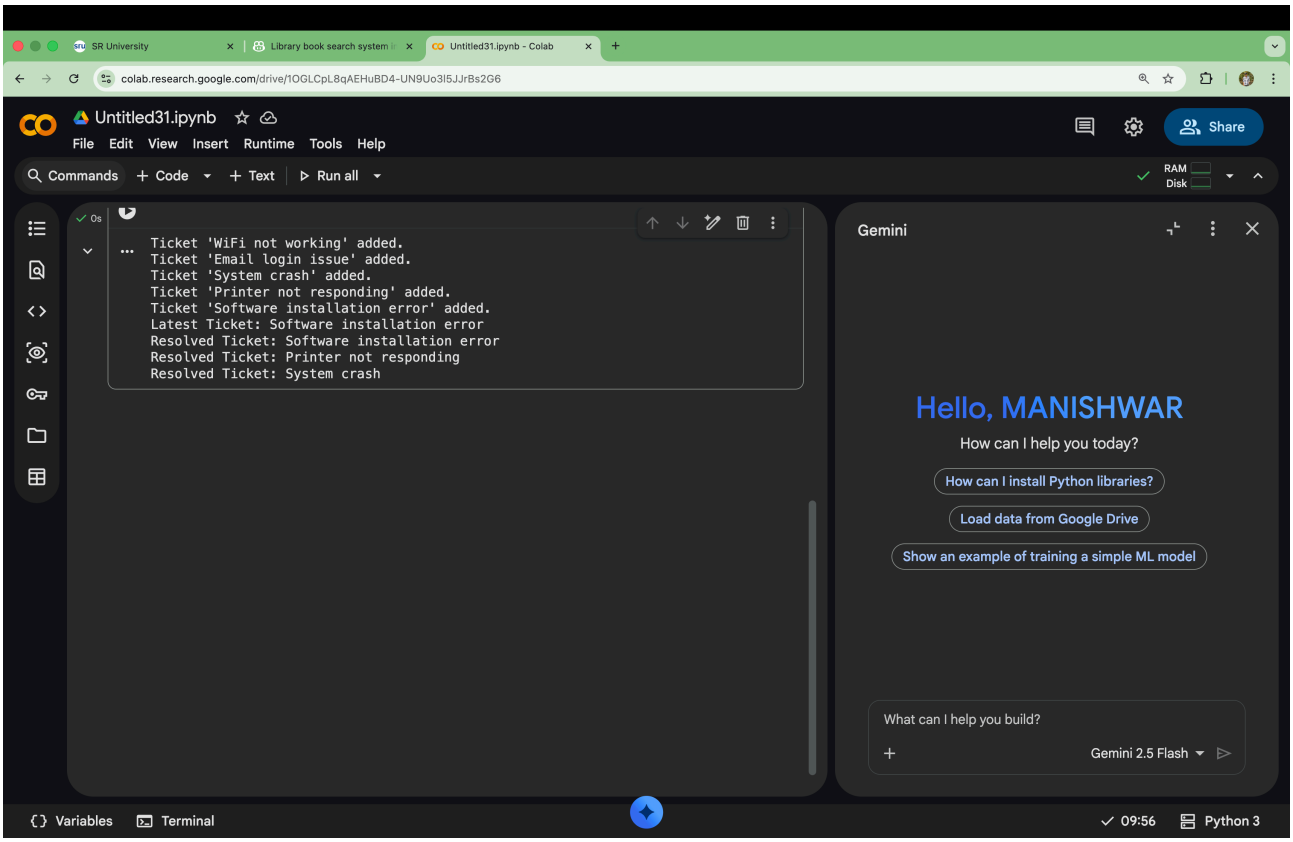
PROMPT:

Generate a Python Emergency Help Desk system using a Stack data structure.Implement push(), pop(), peek(), is_empty(), and is_full() methods and simulate five support tickets.

CODE:



OUTPUT:



EXPLANATION:

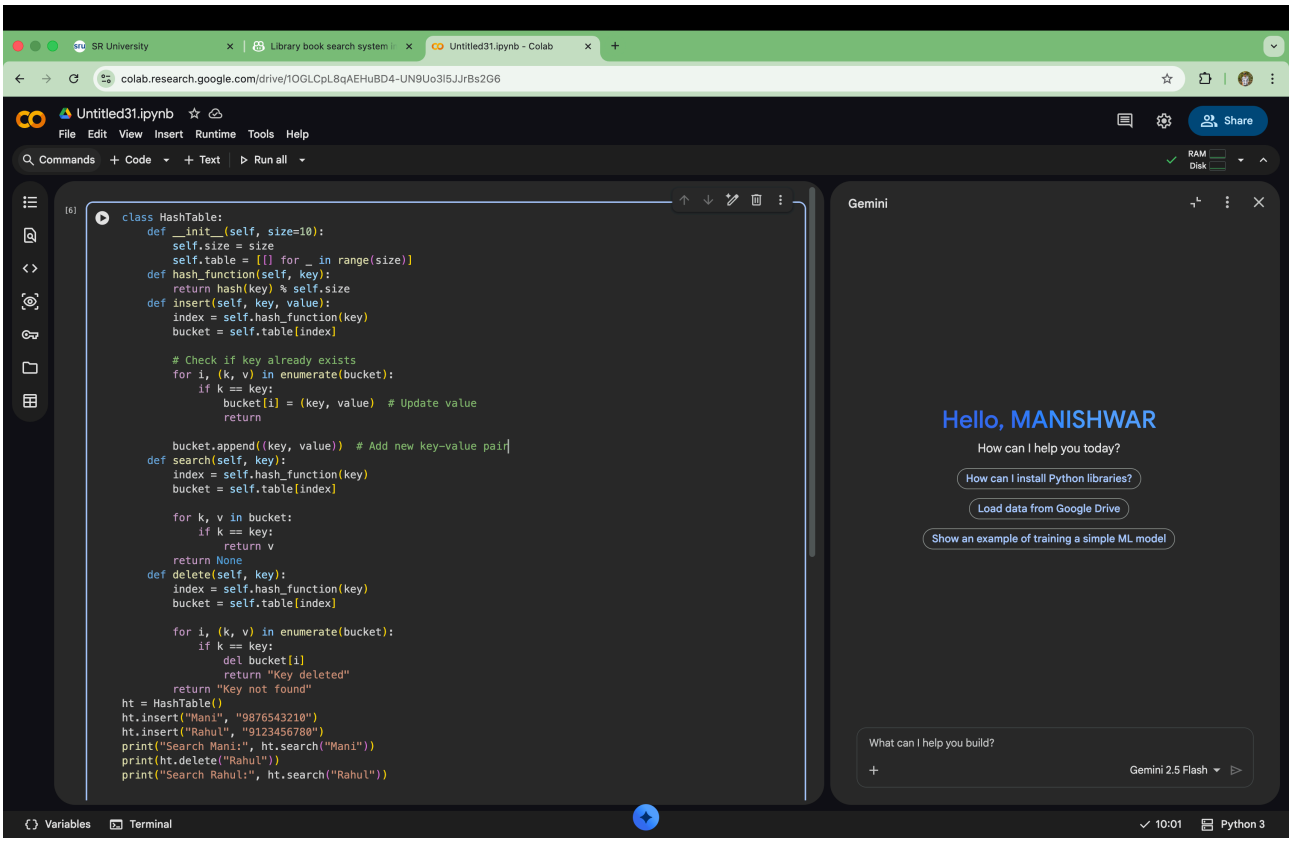
1) The stack follows the LIFO (Last-In, First-Out) principle, meaning the most recent ticket is resolved first.
2) The push() method adds a new ticket to the top of the stack if it is not full.
3) The pop() method removes and returns the most recently added ticket.
4) The peek() method shows the latest ticket without removing it from the stack.
5) The is_empty() and is_full() methods check the stack status to prevent errors during operations.

TASK 4:

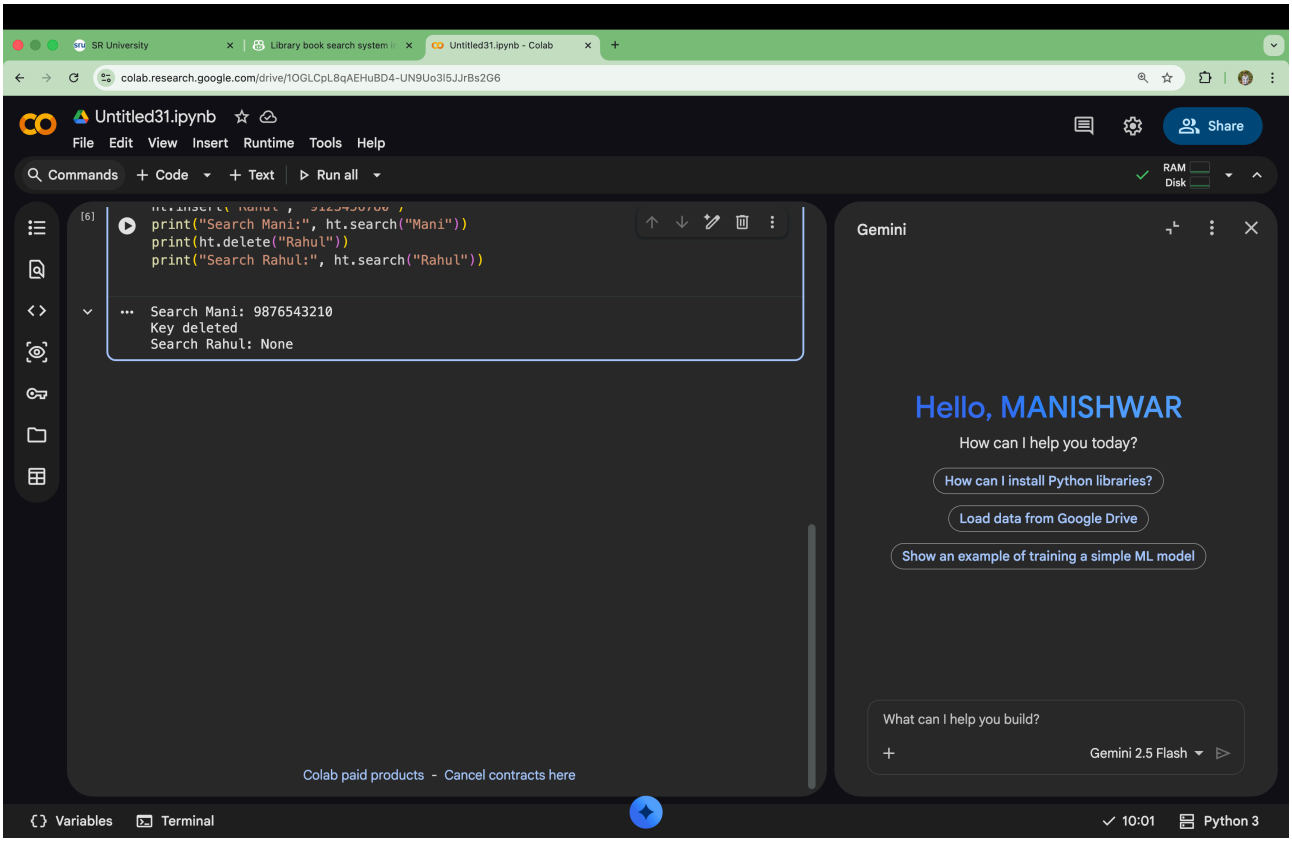
PROMPT:

Generate a Python Hash Table implementation using chaining for collision handling.Include insert, search, and delete methods with proper comments and test cases.

CODE:


SR University
Library book search system
Untitled31.ipynb - Colab
colab.research.google.com/drive/1OGLCpL8qAEHuBD4-UN9Uo3I5JJrBs2G6
Untitled31.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[6]
class HashTable:
def __init__(self, size=10):
self.size = size
self.table = [[] for _ in range(size)]
def hash_function(self, key):
return hash(key) % self.size
def insert(self, key, value):
index = self.hash_function(key)
bucket = self.table[index]
# Check if key already exists
for i, (k, v) in enumerate(bucket):
if k == key:
bucket[i] = (key, value) # Update value
return
bucket.append((key, value)) # Add new key-value pair
def search(self, key):
index = self.hash_function(key)
bucket = self.table[index]
for k, v in bucket:
if k == key:
return v
return None
def delete(self, key):
index = self.hash_function(key)
bucket = self.table[index]
for i, (k, v) in enumerate(bucket):
if k == key:
del bucket[i]
return "Key deleted"
return "Key not found"
ht = HashTable()
ht.insert("Mani", "9876543210")
ht.insert("Rahul", "9123456780")
print("Search Mani:", ht.search("Mani"))
print(ht.delete("Rahul"))
print("Search Rahul:", ht.search("Rahul"))
Gemini
Hello, MANISHWAR
How can I help you today?
How can I install Python libraries?
Load data from Google Drive
Show an example of training a simple ML model
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
10:01
Python 3

OUTPUT:


SR University
Library book search system
Untitled31.ipynb - Colab
colab.research.google.com/drive/1OGLCpL8qAEHuBD4-UN9Uo3I5JJrBs2G6
Untitled31.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[6]
print("Search Mani:", ht.search("Mani"))
print(ht.delete("Rahul"))
print("Search Rahul:", ht.search("Rahul"))
Search Mani: 9876543210
Key deleted
Search Rahul: None
Colab paid products - Cancel contracts here
Gemini
Hello, MANISHWAR
How can I help you today?
How can I install Python libraries?
Load data from Google Drive
Show an example of training a simple ML model
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
10:01
Python 3

EXPLANATION:

1) The hash table stores data in buckets (lists) to handle collisions using chaining.

2) The hash_function() converts a key into an index within the table size.

3) The insert() method adds key-value pairs and updates the value if the key already exists.

4) The search() method checks the correct bucket and returns the value if found.

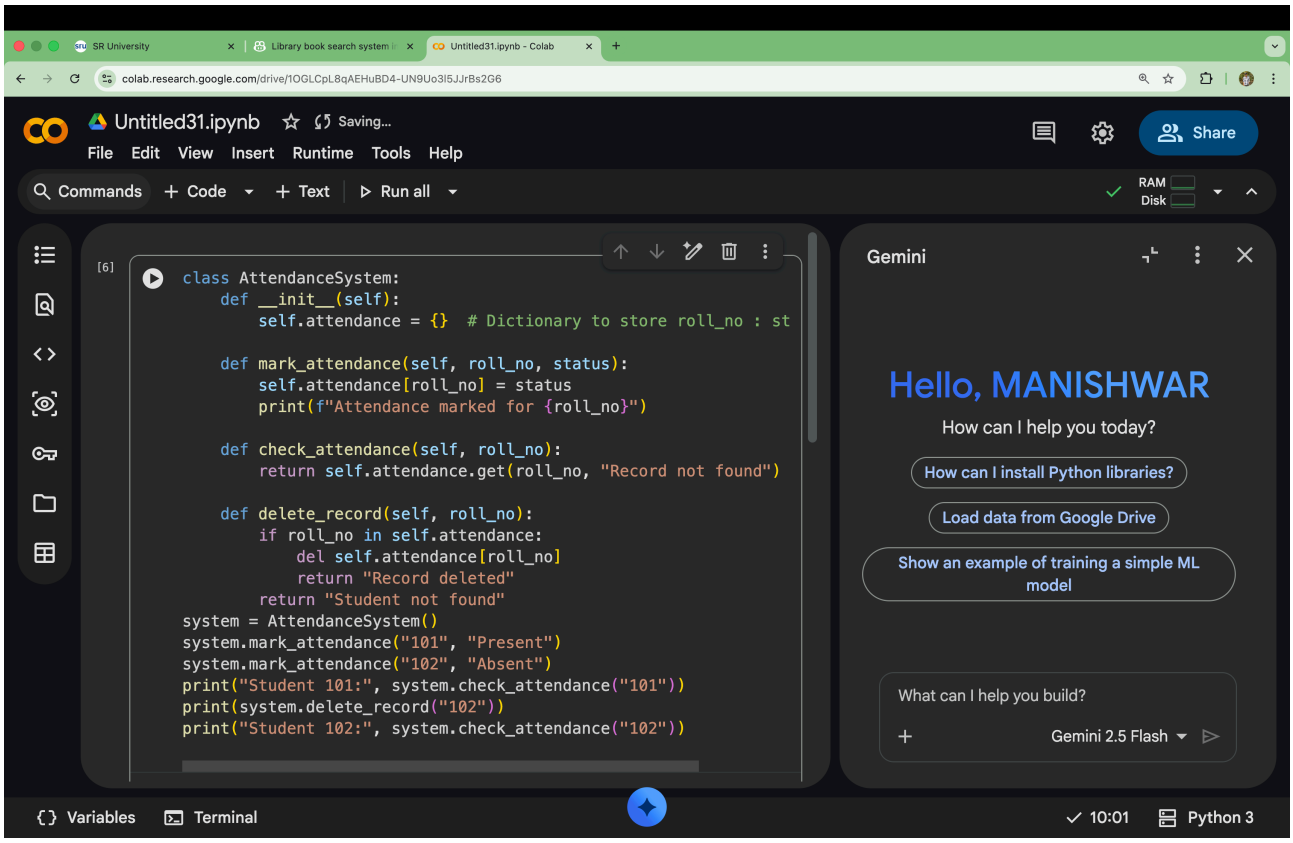
5) The delete() method removes the key-value pair from the bucket, ensuring proper collision handling.

TASK 5:

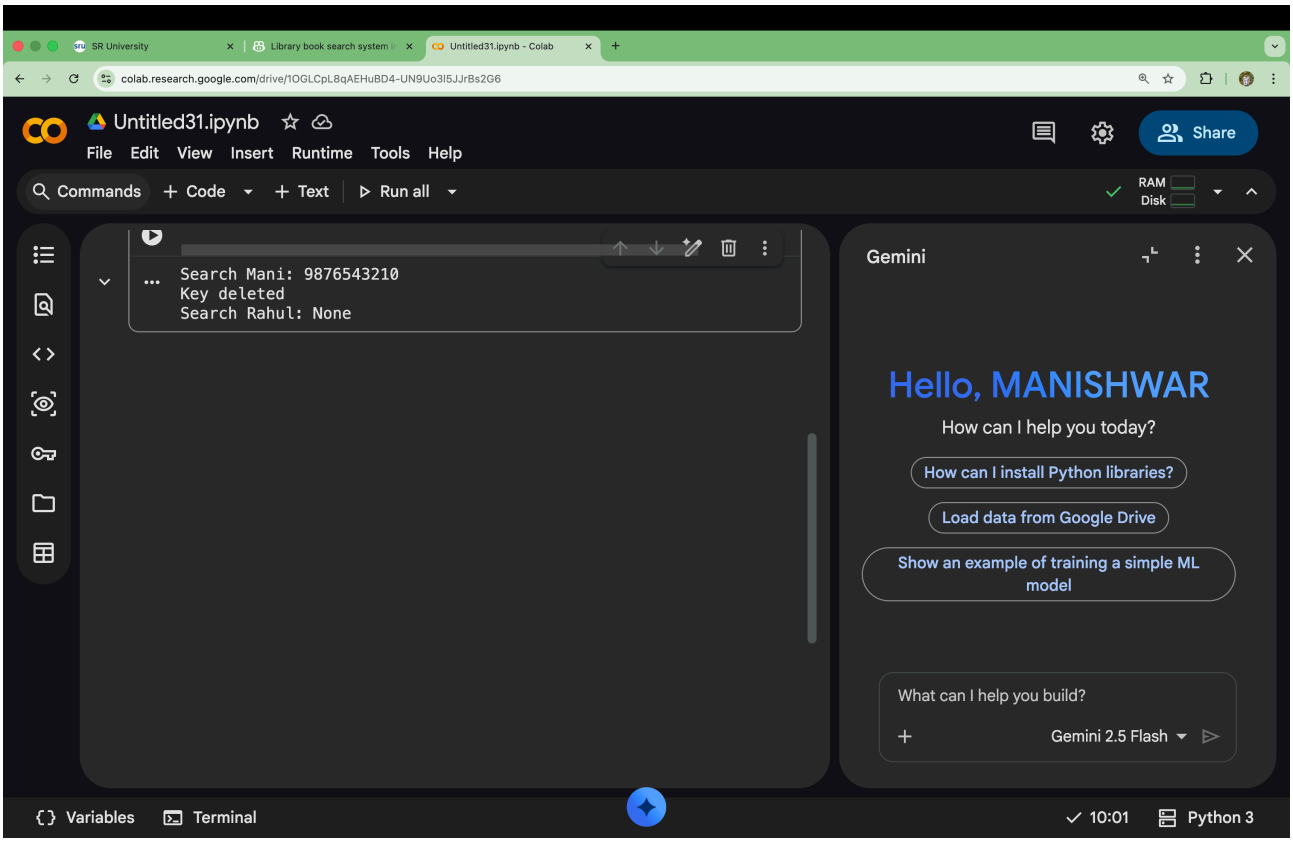
PROMPT:

Design a Campus Resource Management System and suggest appropriate data structures for each feature with justification.Generate Python code to implement one selected feature with proper methods and test cases.

CODE:



OUTPUT:



EXPLANATION:

1) The system uses a dictionary where roll numbers are keys and attendance status is values.

2) The mark_attendance() method inserts or updates attendance in O(1) time.

3) The check_attendance() method quickly retrieves attendance using the key lookup feature of dictionaries.

4) The delete_record() method removes a student record safely after checking existence.

5) Dictionary is efficient because it provides fast insertion, deletion, and searching operations.