```python
import re
from tabulate import tabulate

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def add(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node

    def display(self):
        current = self.head
        details = []
        while current:
            details.append(current.data)
            current = current.next
        return details

class Contact:
    def __init__(self, name, address, group):
        self.name = name
        self.address = address
        self.group = group  # New group attribute
        self.details = LinkedList()

    def add_detail(self, detail):
        if self.validate_email(detail) or self.validate_phone(detail):
            self.details.add(detail)
        else:
            print(f"Invalid detail: {detail}")

    def validate_email(self, email):
        # Basic email validation using regex
        pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
        return re.match(pattern, email) is not None

    def validate_phone(self, phone):
```

```
        # Enhanced phone validation for international formats
        pattern = r'^\+?[1-9]\d{1,14}$'  # Matches international phone numbers
        return re.match(pattern, phone) is not None

    def display(self):
        return {
            "name": self.name,
            "address": self.address,
            "group": self.group,  # Include group in display
            "details": self.details.display()
        }

class ContactManagementSystem:
    def __init__(self):
        self.contacts = []

    def add_contact(self, name, address, group):
        contact = Contact(name, address, group)
        self.contacts.append(contact)

    def remove_contact(self, name):
        self.contacts = [c for c in self.contacts if c.name != name]

    def search_contact(self, name):
        for contact in self.contacts:
            if contact.name == name:
                return contact.display()
        return None

    def display_contacts(self):
        table_data = []
        for contact in self.contacts:
            details = contact.details.display()
            details_str = ", ".join(details) if details else "None"
            table_data.append([contact.name, contact.address, contact.group,
details_str])
        print(tabulate(table_data, headers=["Name", "Address", "Group",
"Details"]))

def main():
    cms = ContactManagementSystem()

    while True:
        print("\nContact Management System")
        print("1. Add Contact")
        print("2. Remove Contact")
        print("3. Search Contact")
        print("4. Display All Contacts")
        print("5. Exit")
```

```
        choice = input("Choose an option: ")

        if choice == '1':
            name = input("Enter name: ")
            address = input("Enter address: ")
            group = input("Enter group (e.g., family, friends, colleagues): ")
# New group input
            cms.add_contact(name, address, group)
            while True:
                detail = input("Add phone number or email (or 'done' to finish):
")
                if detail.lower() == 'done':
                    break
                cms.contacts[-1].add_detail(detail)

        elif choice == '2':
            name = input("Enter name of contact to remove: ")
            cms.remove_contact(name)

        elif choice == '3':
            name = input("Enter name to search: ")
            contact = cms.search_contact(name)
            if contact:
                print(contact)
            else:
                print("Contact not found.")

        elif choice == '4':
            cms.display_contacts()

        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```