Name: Aryan Raj Singh Rathore

Batch-01

---

1. Event Manager

```
import datetime
import pickle

class event:
  dict1={}
  def _init_(self):
    pass

  def addevent(self):
    ev=input("Enter the name of event:")
    data=input("Enter the date and time of the event in format (yyyy,mm,dd,hour,min):")
    year,month,date,hour,min=map(int,data.split(","))
    waqt=datetime.datetime(year,month,date,hour,min)
    self.dict1[ev]=waqt
    return self.dict1

  def removeevent(self):
    rem=input("Enter the name of event to be removed:")
    if rem not in self.dict1:
      return False
    del self.dict1[rem]
    print("Event successfully removed")

  def display(self):
    a=[]
```

```
        for i in self.dict1.values():
            a.append(i)
        a.sort()
        for i,j in self.dict1.items():
            if j in a:
                print(i,j)

    def save(self):
        with open("event.pkl","wb") as f:
            pickle.dump(self.dict1,f)

    def read(self):
        try:
            with open("event.pkl","rb") as f:
                loaded_data=pickle.load(f)
            print("Data Loaded:")
            print(loaded_data)
        except FileNotFoundError:
            print("FileNotFoundError")

obj=event()
print("1. Add an event","\n2. Remove an event","\n3. Display all events in chronological order","\n4. Save events to a file using the pickle module","\n5. Load events from a file using the pickle module","\n6. Exit")
while True:
    choice=int(input("Enter your choice:"))
    if choice==1:
        obj.addevent()
        print("Saving Your Data")
    elif choice==2:
```

```
        obj.removeevent()
    elif choice==3:
        obj.display()
    elif choice==4:
        obj.save()
    elif choice==5:
        obj.read()
    elif choice==6:
        break
```

---

2. Password Manager

```
import pickle
import os
import datetime
from moduleFOR2 import validate_password, encrypt_password, is_password_old


class PasswrdEntry:
    def __init__(self, site, user, pwd):
        self.site = site
        self.user = user
        self.pwd = pwd
        self.last_updated = datetime.datetime.now()


    def __str__(self):
        return "Site: {} | User: {} | Last Updated: {}".format(self.site, self.user, self.last_updated.date())


class PassManager:
    def __init__(self):
        self.entries = []
```

```
def add_entry(self):
    print("\nAdding a new password entry...")
    try:
        site = input("Enter website: ").strip()
        user = input("Enter username: ").strip()
        pwd = input("Enter password: ").strip()

        if not validate_password(pwd):
            print("Password doesn't meet criteria. Please try again.")
            return

        pwd = encrypt_password(pwd)
        entry = PasswrdEntry(site, user, pwd)
        self.entries.append(entry)
        print("Password entry added successfully.")
    except ValueError as err:
        print("Error: {}".format(err))

def remove_entry(self):
    site = input("Enter website to remove: ").strip()
    for entry in self.entries:
        if entry.site == site:
            self.entries.remove(entry)
            print("Entry removed successfully.")
            return
    print("No entry found for that website.")

def display_entries(self):
    if not self.entries:
        print("No password entries to display.")
        return
```

```
    print("\nYour Password Entries:")
    for entry in self.entries:
        print(entry)

def get_old_entries(self):
    old_entries = []
    for entry in self.entries:
        if is_password_old(entry.last_updated):
            old_entries.append(entry)
    if not old_entries:
        print("No password entries older than required time.")
    return old_entries

def save_entries(self):
    try:
        with open("passwords.pkl", "wb") as f:
            pickle.dump(self.entries, f)
        print("Entries saved successfully.")
    except Exception as err:
        print("Error saving file: {}".format(err))

def load_entries(self):
    if os.path.exists("passwords.pkl"):
        try:
            with open("passwords.pkl", "rb") as f:
                self.entries = pickle.load(f)
            print("Entries loaded successfully.")
        except Exception as err:
            print("Error loading file: {}".format(err))
    else:
        print("No saved entries found.")
```

```
def main():
    manager = PassManager()

    menu = (
        "\n1. Add a password entry\n"
        "2. Remove a password entry\n"
        "3. Display all password entries\n"
        "4. Display old password entries (more than 90 days)\n"
        "5. Save password entries to file\n"
        "6. Load password entries from file\n"
        "7. Exit\n"
    )

    while True:
        print(menu)
        choice = input("Enter your choice: ").strip()

        if choice == "1":
            manager.add_entry()
        elif choice == "2":
            manager.remove_entry()
        elif choice == "3":
            manager.display_entries()
        elif choice == "4":
            old_entries = manager.get_old_entries()
            for entry in old_entries:
                print(entry)
        elif choice == "5":
            manager.save_entries()
        elif choice == "6":
```

```
            manager.load_entries()
        elif choice == "7":
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please try again.")

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

---

3. Itinerary maker

```
import pickle
import os
import datetime
from moduleFOR3 import validate_destination, validate_dates, is_trip_in_next_7_days, is_overlapping


class Itenrary:
    def __init__(self, iten_id, dest, start, end):
        self.iten_id = iten_id
        self.dest = dest
        self.start = start
        self.end = end

    def __str__(self):
        return "ID: " + str(self.iten_id) + " | Destination: " + self.dest + " | " + str(self.start.date()) + " to " + str(self.end.date())


class TripPlanner:
    def __init__(self):
```

```
    self.iten_list = []

def add_iten(self):
    print("\nAdding a new trip...")
    while True:
        iten_id = input("Enter trip ID: ").strip()
        dest = input("Destination: ").strip()

        if not validate_destination(dest):
            print("Invalid destination. Try again.")
            continue

        start_str = input("Start date (yyyy-mm-dd): ").strip()
        end_str = input("End date (yyyy-mm-dd): ").strip()

        try:
            start = datetime.datetime.strptime(start_str, "%Y-%m-%d")
            end = datetime.datetime.strptime(end_str, "%Y-%m-%d")

            if not validate_dates(start, end):
                print("End date must be after start date. Try again.")
                continue

            new_iten = Itenrary(iten_id, dest, start, end)

            if any(is_overlapping(existing, new_iten) for existing in self.iten_list):
                print("This trip overlaps with an existing one. Choose different dates.")
                continue
```

```
            self.iten_list.append(new_iten)
            print("Trip added successfully.")
            break

        except ValueError:
            print("Invalid date format. Use YYYY-MM-DD.")

def show_itens(self):
    if not self.iten_list:
        print("No trips planned yet.")
        return

    print("\nYour Trips:")
    for iten in self.iten_list:
        print(iten)

def show_upcoming(self):
    upcoming = [iten for iten in self.iten_list if is_trip_in_next_7_days(iten.start)]

    if not upcoming:
        print("No trips in the next 7 days.")
        return

    print("\nUpcoming Trips:")
    for iten in upcoming:
        print(iten)

def save_itens(self):
    try:
```

```
            with open("trips.pkl", "wb") as f:
                pickle.dump(self.iten_list, f)
            print("Trips saved successfully.")
        except Exception as e:
            print("Error saving trips: {}".format(e))

    def load_itens(self):
        if not os.path.exists("trips.pkl"):
            print("No saved trips found.")
            return

        try:
            with open("trips.pkl", "rb") as f:
                self.iten_list = pickle.load(f)
            print("Trips loaded successfully.")
        except Exception as e:
            print("Error loading trips: {}".format(e))

def main():
    planner = TripPlanner()

    menu = (
        "\n1. Add a trip\n"
        "2. Show all trips\n"
        "3. Show upcoming trips\n"
        "4. Save trips\n"
        "5. Load trips\n"
        "6. Exit\n"
    )
```

```
    while True:
        print(menu)
        choice = input("Enter choice: ").strip()

        if choice == "1":
            planner.add_iten()
        elif choice == "2":
            planner.show_itens()
        elif choice == "3":
            planner.show_upcoming()
        elif choice == "4":
            planner.save_itens()
        elif choice == "5":
            planner.load_itens()
        elif choice == "6":
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Try again.")

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