```python
import datetime
import json
import os.path
from datetime import datetime


class User:
    def __init__(self, name, email, password):
        self.name = name
        self.email = email
        self.password = password

    def __str__(self):
        return f"{self.name} - {self.email}"

    def login(self, email, password):
        if not os.path.exists("user.json"):
            return ""
        with open("user.json", "r") as file:
            users = json.load(file)
            for user in users:
                if user["email"] == email and user["password"] == password:
                    return user["email"]
        return ""

    def add_user(self, name, email, password):
        user = User(name, email, password)
        return user

    def delete_user(self, email):
        with open("user.json", "r") as file:
            users = json.load(file)
        if self.email == email:
            del self.email
        else:
            print("Invalid email")

    def view_user(self):
        return self.name, self.email

```

```
41      def to_dict(self):
42          return {
43              "name": self.name,
44              "email": self.email,
45              "password": self.password
46          }
47
48      def append_user_json_file(self, name, email,
    password):
49          users = []
50          if os.path.exists("user.json"):
51              with open("user.json", "r") as file:
52                  users = json.load(file)
53          user = self.add_user(name, email, password)
54          users.append(user.to_dict())
55          with open("user.json", "w") as file:
56              json.dump(users, file, indent=4)
57
58      @staticmethod
59      def print_all_users():
60          with open("user.json", "r") as file:
61              users = json.load(file)
62              for user in users:
63                  print(user)
64
65
66  class Expense:
67      def __init__(self, sno, amount, category,
    description="", date=""):
68          self.sno = sno
69          self.amount = amount
70          self.category = category
71          self.description = description
72          if date == "":
73              from datetime import date
74              self.date = date.today().strftime("%Y-%m
    -%d")
75          else:
76              self.date = date
77          # self.date = date or date.today().strftime
    ("%Y-%m-%d")
```

```
 78
 79     def __str__(self):
 80         return f"{self.sno} - {self.category}: {self
    .amount} - {self.description} on {self.date}"
 81
 82     def to_dict(self):
 83         return {
 84             "sno": self.sno,
 85             "description": self.description,
 86             "category": self.category,
 87             "amount": self.amount,
 88             "date": self.date
 89         }
 90
 91
 92 class ExpenseTracker:
 93     def __init__(self):
 94         self.expenses = {}
 95         self.budgets = {}
 96
 97     def add_expense(self, email, sno, amount,
    category, description="", date=None):
 98         if amount <= 0:
 99             print("Invalid amount")
100             return
101         if not category:
102             print("Category is required")
103             return
104         if not description:
105             print("Description is required")
106             return
107         expense = Expense(sno, amount, category,
    description, date)
108
109         # if not email in self.expenses:
110         #     self.expenses[email] = []
111         # # if not self.expenses[email]:
112         # #     self.expenses[email] = []
113         # self.expenses[email].append(expense)
114         return expense
115
```

```
116     def get_month_expense(self, email, month):
117         if email in self.expenses:
118             return sum(expense["amount"] for expense
     in self.expenses[email] if datetime.strptime(
    expense["date"], "%Y-%m-%d").strftime("%m") == month
    )
119         else:
120             return 0
121 
122     def delete_expense(self, sno, email):
123         if email in self.expenses:
124             expenses_for_email = self.expenses[email
    ]  # Accessing the list using the __getitem__ method
125 
126             for expense in expenses_for_email:
127                 if int(expense["sno"]) == sno:
128                     print(f"Deleting expense {sno}
    for {email}.")
129                     # Remove the expense from the
    list of expenses
130                     self.expenses[email] = [t for t
    in expenses_for_email if t["sno"] != sno]
131                     self.save_expenses()
132                     return True
133         return False
134 
135     def set_monthly_budget(self, email, month, year
    , amount):
136         if email not in self.budgets:
137             self.budgets[email] = {}
138         self.budgets[email][f"{year}-{month:02d}"
    ] = amount
139         print(f"Budget for {month}/{year} set to {
    amount}")
140 
141     def get_monthly_budget(self, email, month, year
    ):
142         month = int(month)  # Convert month to
    integer
143         return int(self.budgets.get(email, {}).get(f
    "{year}-{month:02d}", 0))
```

```
144
145     def save_expenses(self):
146         with open("expenses.json", "w") as file:
147             json.dump(self.expenses, file, indent=4)
148
149     def view_expenses(self, email):
150         if os.path.exists("expenses.json"):
151             with open("expenses.json", "r") as file:
152                 try:
153                     existing_expenses = json.load(
    file)
154                     self.expenses =
    existing_expenses
155                 except json.JSONDecodeError:
156                     self.expenses = {}
157
158         else:
159             self.expenses = {}
160
161         # Display the user's expenses
162         if email not in self.expenses:
163             print("No expenses found for this user."
    )
164         else:
165             print("Expenses for:", email)
166             for expense in self.expenses[email]:
167                 print(expense)
168
169     def total_expenses(self, email):
170         if email not in self.expenses:
171             return 0
172         return sum(expense["amount"] for expense in
    self.expenses[email])
173
174     def filter_by_category(self, email, category):
175         if email in self.expenses:
176             return [expense for expense in self.
    expenses[email] if expense["category"] == category]
177         else:
178             return []
179
```

```
180     def monthly_report(self, email, month, year):
181         # Open to debug
182         # if email in self.expenses:
183         #     for expense in self.expenses[email]:
184         #         print(datetime.strptime(expense["
    date"], "%Y-%m-%d").strftime("%m"), datetime.
    strptime(expense["date"], "%Y-%m-%d").strftime("%Y
    "))
185 
186         if email in self.expenses:
187             report = [expense for expense in self.
    expenses[email]
188                       if int(datetime.strptime(
    expense["date"], "%Y-%m-%d").strftime("%m")) ==
    month and
189                       int(datetime.strptime(expense[
    "date"], "%Y-%m-%d").strftime("%Y")) == year]
190             return report
191         else:
192             print(f"No expenses found for \"{email
    }\" user for \"{month}-{year}\".")
193             return []
194 
195     def search_expenses(self, email, keyword):
196         if email not in self.expenses:
197             return []
198         else:
199             return [expense for expense in self.
    expenses[email] if keyword.lower() in expense["
    description"].lower()]
200 
201     def write_expenses_json_file(self, email,
    expense):
202         if os.path.exists("expenses.json"):
203             with open("expenses.json", "r") as file:
204                 try:
205                     existing_expenses = json.load(
    file)
206                 except json.JSONDecodeError:
207                     existing_expenses = {}
208         else:
```

```
209             existing_expenses = {}
210         if not email in existing_expenses:
211             existing_expenses[email] = []
212 
213         # existing_expenses[email].extend([expense.
    to_dict() for expense in self.expenses[email]])
214         existing_expenses[email].extend([expense.
    to_dict()])
215         # existing_expenses[email].append([expense.
    to_dict() for expense in self.expenses[email]])
216         with open("expenses.json", "w") as file:
217             json.dump(existing_expenses, file,
    indent=4)
218 
219     def read_all_expenses_json_file(self, email):
220         with open("expenses.json", "r") as file:
221             expenses = json.load(file)
222             return [Expense(expense["amount"],
    expense["category"], expense["description"],
    datetime.datetime.strptime(expense["date"], "%Y-%m-%
    d").date()) for expense in expenses[email]]
223 
224     def load_expenses(self, email):
225         if os.path.exists("expenses.json"):
226             with open("expenses.json", "r") as file:
227                 try:
228                     existing_expenses = json.load(
    file)
229                     self.expenses =
    existing_expenses
230                 except json.JSONDecodeError:
231                     self.expenses = {}
232         else:
233             self.expenses = {}
234 
235 
236 def expense_tracker_menu():
237     while True:
238         print("\nExpense Tracker Menu:")
239         print("1. Login")
240         print("2. Register")
```

```
241         print("3. View Users")
242         print("4. Exit")
243         choice = input("Enter your choice: ")
244 
245         if choice == "1":
246             email = input("Enter email: ")
247             password = input("Enter password: ")
248             user = User("", email, password)
249             authenticated_username = user.login(
    email, password)
250             if authenticated_username != "":
251                 print(f"Welcome \"{
    authenticated_username}\": Login successful...")
252                 break
253             else:
254                 print("Login failed. Please try
    again.\n")
255 
256         if choice == "2":
257             name = input("Enter name: ")
258             email = input("Enter email: ")
259             password = input("Enter password: ")
260             user = User(name, email, password)
261             user.append_user_json_file(name, email,
    password)
262 
263         if choice == "3":
264             User.print_all_users()
265 
266         if choice == "4":
267             break
268 
269     tracker = ExpenseTracker()
270     tracker.load_expenses(email)
271 
272     while True:
273         print("\nExpense Tracker Menu:")
274         print("1. Add Expense")
275         print("2. Set Monthly Budget")
276         print("3. View Expenses")
277         print("4. Filter by Category")
```

```
278         print("5. Monthly Report")
279         print("6. Search Expenses")
280         print("7. Total Expenses")
281         print("8. Exit\n")
282         choice = input("Enter your choice: ")
283 
284         if choice == "1":
285             def get_max_sno_json_file(email):
286                 if not os.path.exists("expenses.json
    "):
287                     return 0
288                 with open("expenses.json", "r") as
    file:
289                     expenses = json.load(file)
290                     if not email in expenses:
291                         return 0
292                     return max(expense["sno"] for
    expense in expenses[email])
293 
294             sno = get_max_sno_json_file(email) + 1
295             amount = float(input("Enter amount: "))
296 
297             category = input("Enter category: ")
298             description = input("Enter description
    : ")
299             date_input = input("Enter date (YYYY-MM-
    DD) or leave blank for today: ")
300             if date_input == "":
301                 date_input = datetime.today().
    strftime("%Y-%m-%d")
302 
303             # Get month from date
304             month = date_input.split("-")[1]
305             year = date_input.split("-")[0]
306             monthly_budget = tracker.
    get_monthly_budget(email, month, year)
307             curr_budget = tracker.get_month_expense(
    email, month)
308             if monthly_budget == 0:
309                 print(f"No budget set for the
    current month {month}.\n")
```

```
310             else:
311                 print(f"Current utilized budget for
    month {month} is: {curr_budget}\n")
312 
313             if amount > 0:
314                 if amount + curr_budget >
    monthly_budget:
315                     print(f"[Caution]: Amount
    exceeds Total Monthly budget {monthly_budget}.\n")
316 
317             # date = date_input if date_input else
    None
318             expense = tracker.add_expense(email, sno
    , amount, category, description, date_input)
319             tracker.write_expenses_json_file(email,
    expense)
320 
321         elif choice == "2":
322             # tracker.view_expenses(email)
323             month = int(input("Enter month (1-12): "
    ))
324             year = int(input("Enter year: "))
325             amount = float(input("Enter the total
    amount you want to budget for the month: "))
326             tracker.set_monthly_budget(email, month
    , year, amount)
327 
328         elif choice == "3":
329             tracker.view_expenses(email)
330 
331         elif choice == "4":
332             tracker.view_expenses(email)
333             category = input("\n\nEnter category to
    filter by: ")
334             filtered = tracker.filter_by_category(
    email, category)
335             if not filtered:
336                 print("No expenses found for the
    given category")
337             else:
338                 for expense in filtered:
```

```
339                     print(expense)
340
341         elif choice == "5":
342             month = int(input("Enter month (1-12): "
    ))
343             year = int(input("Enter year: "))
344             report = tracker.monthly_report(email,
    month, year)
345             for expense in report:
346                 print(expense)
347
348         elif choice == "6":
349             keyword = input("Enter keyword to search
     expenses: ")
350             searched = tracker.search_expenses(email
    , keyword)
351             if not searched:
352                 print("No expenses found for the
    given keyword")
353             else:
354                 for expense in searched:
355                     print(expense)
356
357         elif choice == "7":
358             print(f"Total expenses for \"{email}\"
    user: {tracker.total_expenses(email)}")
359
360         elif choice == "8":
361             break
362
363         else:
364             print("Invalid choice. Please try again
    .")
365
366
367 if __name__ == "__main__":
368     expense_tracker_menu()
369
```