

```

1 import datetime
2 import json
3 import os.path
4 from datetime import datetime
5
6
7 class User:
8     def __init__(self, name, email, password):
9         self.name = name
10        self.email = email
11        self.password = password
12
13    def __str__(self):
14        return f"{self.name} - {self.email}"
15
16    def login(self, email, password):
17        if not os.path.exists("user.json"):
18            return ""
19        with open("user.json", "r") as file:
20            users = json.load(file)
21            for user in users:
22                if user["email"] == email and user["
password"] == password:
23                    return user["email"]
24        return ""
25
26    def add_user(self, name, email, password):
27        user = User(name, email, password)
28        return user
29
30    def delete_user(self, email):
31        with open("user.json", "r") as file:
32            users = json.load(file)
33            if self.email == email:
34                del self.email
35            else:
36                print("Invalid email")
37
38    def view_user(self):
39        return self.name, self.email
40

```

```

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
119                 in self.expenses[email] if datetime.strptime(
120                     expense["date"], "%Y-%m-%d").strftime("%m") == month
121                 )
122         else:
123             return 0
124
125     def delete_expense(self, sno, email):
126         if email in self.expenses:
127             expenses_for_email = self.expenses[email]
128             ] # Accessing the list using the __getitem__ method
129
130             for expense in expenses_for_email:
131                 if int(expense["sno"]) == sno:
132                     print(f"Deleting expense {sno}
133                     for {email}.")
134
135                     # Remove the expense from the
136                     list of expenses
137
138                     self.expenses[email] = [t for t
139                     in expenses_for_email if t["sno"] != sno]
140                     self.save_expenses()
141                     return True
142
143                     return False
144
145     def set_monthly_budget(self, email, month, year
146         , amount):
147         if email not in self.budgets:
148             self.budgets[email] = {}
149             self.budgets[email][f"{year}-{month:02d}"]
150             ] = amount
151             print(f"Budget for {month}/{year} set to {
152             amount}")
153
154     def get_monthly_budget(self, email, month, year
155         ):
156         month = int(month) # Convert month to
157         integer
158         return int(self.budgets.get(email, {}).get(f
159         "{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(
154                         file)
155                     self.expenses =
156                     existing_expenses
157                 except json.JSONDecodeError:
158                     self.expenses = {}
159
160         else:
161             self.expenses = {}
162
163         # Display the user's expenses
164         if email not in self.expenses:
165             print("No expenses found for this user."
166             )
167         else:
168             print("Expenses for:", email)
169             for expense in self.expenses[email]:
170                 print(expense)
171
172     def total_expenses(self, email):
173         if email not in self.expenses:
174             return 0
175         return sum(expense["amount"] for expense in
176             self.expenses[email])
177
178     def filter_by_category(self, email, category):
179         if email in self.expenses:
180             return [expense for expense in self.
181                 expenses[email] if expense["category"] == category]
182         else:
183             return []
184
185

```

```

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                 ):
288                     return 0
289                 with open("expenses.json", "r") as
290                 file:
291                     expenses = json.load(file)
292                     if not email in expenses:
293                         return 0
294                     return max(expense["sno"] for
295                     expense in expenses[email])
296
297                 sno = get_max_sno_json_file(email) + 1
298                 amount = float(input("Enter amount: "))
299
300                 category = input("Enter category: ")
301                 description = input("Enter description
302                 : ")
303                 date_input = input("Enter date (YYYY-MM-
304                 DD) or leave blank for today: ")
305                 if date_input == "":
306                     date_input = datetime.today().
307                     strftime("%Y-%m-%d")
308
309                 # Get month from date
310                 month = date_input.split("-")[1]
311                 year = date_input.split("-")[0]
312                 monthly_budget = tracker.
313                 get_monthly_budget(email, month, year)
314                 curr_budget = tracker.get_month_expense(
315                 email, month)
316                 if monthly_budget == 0:
317                     print(f"No budget set for the
318                     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,
345             month, year)
346             for expense in report:
347                 print(expense)
348
349         elif choice == "6":
350             keyword = input("Enter keyword to search
351             expenses: ")
352             searched = tracker.search_expenses(email
353             , keyword)
354             if not searched:
355                 print("No expenses found for the
356                 given keyword")
357             else:
358                 for expense in searched:
359                     print(expense)
360
361         elif choice == "7":
362             print(f"Total expenses for \"{email}\"
363             user: {tracker.total_expenses(email)}")
364
365         elif choice == "8":
366             break
367
368         else:
369             print("Invalid choice. Please try again
370             .")
371
372 if __name__ == "__main__":
373     expense_tracker_menu()
374
```