```
import json
import datetime

# Class to represent an expense entry
class Expense:
    def __init__(self, amount, date, category, notes=None):
        self.amount = amount
        self.date = date
        self.category = category
        self.notes = notes

    def to_dict(self):
        return {
            'amount': self.amount,
            'date': self.date,
            'category': self.category,
            'notes': self.notes
        }

# Class to manage expenses and monthly budget
class ExpenseTracker:
    def __init__(self):
        self.expenses = []

    def add_expense(self, expense):
        self.expenses.append(expense)

    def delete_expenses_by_month(self, month):
        self.expenses = [expense for expense in self.expenses if not
expense.date.startswith(month)]

    def get_total_expenses(self, month=None):
        if month:
            total_expenses = sum(expense.amount for expense in
self.expenses if expense.date.startswith(month))
        else:
            total_expenses = sum(expense.amount for expense in
self.expenses)
        return total_expenses

    def get_expenses_by_category(self, month=None):
        if month:
            expenses_by_category = {}
            for expense in self.expenses:
                if expense.date.startswith(month):
                    expenses_by_category[expense.category] =
expenses_by_category.get(expense.category, 0) + expense.amount
            return expenses_by_category
```

```
        else:
            expenses_by_category = {}
            for expense in self.expenses:
                expenses_by_category[expense.category] =
expenses_by_category.get(expense.category, 0) + expense.amount
            return expenses_by_category

    def is_within_budget(self, month_budget, month=None):
        total_expenses = self.get_total_expenses(month)
        return total_expenses <= month_budget

    def load_expenses_from_file(self, filename):
        try:
            with open(filename, 'r') as file:
                data = json.load(file)
                self.expenses = [Expense(expense['amount'],
expense['date'], expense['category'], expense.get('notes', None)) for
expense in data]
            print("Expenses loaded successfully from file.")
        except FileNotFoundError:
            print("No expense data found in the file. Starting with an
empty expense list.")
        except json.JSONDecodeError:
            print("Error decoding JSON. The file may be empty or
corrupted. Starting with an empty expense list.")

    def save_expenses_to_file(self, filename):
        with open(filename, 'w') as file:
            json.dump([expense.to_dict() for expense in self.expenses],
file, indent=4)

# Function to prompt user for expense details
def get_expense_details():
    while True:
        try:
            amount = float(input("Enter the amount you spent: "))
            break  # If input is valid, exit the loop
        except ValueError:
            print("Invalid input. Please enter a valid amount.")
    while True:
        date = input("Enter the date of the expense (YYYY-MM-DD): ")
        try:
            datetime.datetime.strptime(date, '%Y-%m-%d')
            break  # If input is valid, exit the loop
        except ValueError:
            print("Invalid date format. Please enter the date in YYYY-
MM-DD format.")
    category = input("Enter the category of the expense: ")
```

```
    notes = input("Enter any notes for the expense (optional): ")
    return Expense(amount, date, category, notes)


# Main function to interact with the user
def main():
    print("Welcome to the Expense Tracker App!")
    tracker = ExpenseTracker()
    filename = "expenses.json"  # File to store expense data
    tracker.load_expenses_from_file(filename)  # Load existing expense
data
    monthly_budget = float(input("Enter your monthly budget: $"))
    while True:
        print("\n1. Add Expense\n2. View Total Monthly Expenses\n3.
View Monthly Budget Status\n4. Delete Expenses by Month\n5. Save
Expenses to File\n6. Exit")
        choice = input("Enter your choice: ")
        if choice == '1':
            expense = get_expense_details()
            tracker.add_expense(expense)
            print("Expense added successfully!")
        elif choice == '2':
            month = input("Enter month to view total expenses (YYYY-
MM): ")
            total_expenses = tracker.get_total_expenses(month)
            print(f"Total expenses for {month}: ${total_expenses:.2f}")
            expenses_by_category =
tracker.get_expenses_by_category(month)
            print("Expenses by category:")
            for category, total_amount in expenses_by_category.items():
                print(f"{category}: ${total_amount:.2f}")
        elif choice == '3':
            month = input("Enter month to check budget status (YYYY-
MM): ")
            if tracker.is_within_budget(monthly_budget, month):
                print("You are within the monthly budget.")
            else:
                print("You have exceeded the monthly budget.")
        elif choice == '4':
            month_to_delete = input("Enter month to delete expenses
(YYYY-MM): ")
            tracker.delete_expenses_by_month(month_to_delete)
            print(f"Expenses for {month_to_delete} deleted
successfully.")
        elif choice == '5':
            tracker.save_expenses_to_file(filename)
            print("Expenses saved to file.")
        elif choice == '6':
            print("Exiting...")
```

```
            break
        else:
            print("Invalid choice. Please try again.")

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