In [4]:

```
import tkinter as tk
from tkinter import ttk, messagebox
import csv
import os

# File to store expenses
FILENAME = "expenses.csv"

# Predefined categories
CATEGORIES = ["Food", "Transport", "Entertainment", "Bills", "Shopping", "Other"]

# Initialize the file
def initialize_file():
    if not os.path.exists(FILENAME):
        with open(FILENAME, "w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow(["Date", "Category", "Amount (₹)"])  # Add header row

# Function to add an expense
def add_expense():
    date = date_entry.get()
    category = category_var.get()
    amount = amount_entry.get()

    if not (date and category and amount):
        messagebox.showerror("Input Error", "All fields must be filled.")
        return

    try:
        amount = float(amount)
    except ValueError:
        messagebox.showerror("Input Error", "Amount must be a number.")
        return

    with open(FILENAME, "a", newline="") as file:
        writer = csv.writer(file)
        writer.writerow([date, category, amount])
    messagebox.showinfo("Success", "Expense added successfully!")
    date_entry.delete(0, tk.END)
    category_var.set(CATEGORIES[0])
    amount_entry.delete(0, tk.END)
    view_expenses()  # Refresh the table

# Function to view expenses
def view_expenses():
    # Clear the treeview
    for row in expense_tree.get_children():
        expense_tree.delete(row)

    try:
        with open(FILENAME, "r") as file:
            reader = csv.reader(file)
            next(reader, None)  # Skip header row
            for row in reader:
                expense_tree.insert("", tk.END, values=row)
    except FileNotFoundError:
```

```
        messagebox.showerror("File Error", "Expenses file not found.")

# Function to delete selected expenses
def delete_expense():
    selected_items = expense_tree.selection()
    if not selected_items:
        messagebox.showerror("Error", "No expense selected to delete!")
        return

    # Read all rows from the file
    with open(FILENAME, "r") as file:
        rows = list(csv.reader(file))

    # Remove selected rows from the file
    for item in selected_items:
        values = expense_tree.item(item, "values")
        rows = [row for row in rows if row != list(values)]

    # Write the updated rows back to the file
    with open(FILENAME, "w", newline="") as file:
        writer = csv.writer(file)
        writer.writerows(rows)

    # Update the table
    view_expenses()
    messagebox.showinfo("Success", "Selected expenses deleted!")

# Function to calculate total by category
def total_by_category():
    try:
        with open(FILENAME, "r") as file:
            reader = csv.reader(file)
            next(reader, None)  # Skip header row
            category_totals = {}
            for row in reader:
                category = row[1]
                amount = float(row[2])
                category_totals[category] = category_totals.get(category, 0) + amount

            if not category_totals:
                messagebox.showinfo("Total by Category", "No expenses recorded.")
                return

            result = "\n".join([f"{category}: ₹{total:.2f}" for category, total in category_totals.items()])
            messagebox.showinfo("Total by Category", result)
    except FileNotFoundError:
        messagebox.showerror("File Error", "Expenses file not found.")

# Function to switch to the home screen
def show_home():
    for widget in root.winfo_children():
        widget.destroy()
    home_screen()

# Function to show the Add Expense screen
def show_add_expense():
    for widget in root.winfo_children():
        widget.destroy()
```

```
    add_expense_screen()

# Function to show the View Expenses screen
def show_view_expenses():
    for widget in root.winfo_children():
        widget.destroy()
    view_expenses_screen()

# Home Screen
def home_screen():
    root.configure(bg="#d1f7ff")
    tk.Label(root, text="Expense Tracker", font=("Arial", 20, "bold"), bg="#d1f7ff", fg="#333").pack(pady=20)

    tk.Button(root, text="Add Expense", command=show_add_expense, font=("Arial", 14), bg="#4caf50", fg="white", width=20).pack(pady=10)
    tk.Button(root, text="View Expenses", command=show_view_expenses, font=("Arial", 14), bg="#2196f3", fg="white", width=20).pack(pady=10)
    tk.Button(root, text="Total by Category", command=total_by_category, font=("Arial", 14), bg="#ff9800", fg="white", width=20).pack(pady=10)
    tk.Button(root, text="Exit", command=root.destroy, font=("Arial", 14), bg="#f44336", fg="white", width=20).pack(pady=10)

# Add Expense Screen
def add_expense_screen():
    root.configure(bg="#ffebcd")
    tk.Label(root, text="Add Expense", font=("Arial", 20, "bold"), bg="#ffebcd", fg="#333").pack(pady=20)

    input_frame = tk.Frame(root, bg="#ffebcd")
    input_frame.pack(pady=10)

    tk.Label(input_frame, text="Date (YYYY-MM-DD):", font=("Arial", 14), bg="#ffebcd").grid(row=0, column=0, padx=5, pady=5)
    global date_entry, category_var, amount_entry
    date_entry = tk.Entry(input_frame, font=("Arial", 14))
    date_entry.grid(row=0, column=1, padx=5, pady=5)

    tk.Label(input_frame, text="Category:", font=("Arial", 14), bg="#ffebcd").grid(row=1, column=0, padx=5, pady=5)
    category_var = tk.StringVar(value=CATEGORIES[0])
    category_menu = ttk.Combobox(input_frame, textvariable=category_var, values=CATEGORIES, state="readonly", font=("Arial", 14))
    category_menu.grid(row=1, column=1, padx=5, pady=5)

    tk.Label(input_frame, text="Amount (₹):", font=("Arial", 14), bg="#ffebcd").grid(row=2, column=0, padx=5, pady=5)
    amount_entry = tk.Entry(input_frame, font=("Arial", 14))
    amount_entry.grid(row=2, column=1, padx=5, pady=5)

    tk.Button(root, text="Add Expense", command=add_expense, font=("Arial", 14), bg="#4caf50", fg="white", width=20).pack(pady=10)
    tk.Button(root, text="Back to Home", command=show_home, font=("Arial", 14), bg="#2196f3", fg="white", width=20).pack(pady=10)

# View Expenses Screen
def view_expenses_screen():
    root.configure(bg="#e6eeff")
    tk.Label(root, text="View Expenses", font=("Arial", 20, "bold"), bg="#e6eeff", fg="#333").pack(pady=20)

    global expense_tree
    expense_frame = tk.Frame(root, bg="#e6eeff")
    expense_frame.pack(pady=10)

    columns = ("Date", "Category", "Amount (₹)")
    expense_tree = ttk.Treeview(expense_frame, columns=columns, show="headings")
    expense_tree.heading("Date", text="Date")
    expense_tree.heading("Category", text="Category")
    expense_tree.heading("Amount (₹)", text="Amount (₹)")
```

```
    expense_tree.pack(fill=tk.BOTH, expand=True)

    tk.Button(root, text="Delete Selected", command=delete_expense, font=("Arial", 14), bg="#f44336", fg="white", width=20).pack(pady=10)
    tk.Button(root, text="Back to Home", command=show_home, font=("Arial", 14), bg="#2196f3", fg="white", width=20).pack(pady=10)

    view_expenses()

# Initialize the file
initialize_file()

# Main Application Window
root = tk.Tk()
root.title("Expense Tracker")
root.geometry("600x500")
home_screen()

root.mainloop()
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\jeeva\anaconda3\Lib\tkinter\__init__.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "C:\Users\jeeva\AppData\Local\Temp\ipykernel_17804\870462463.py", line 42, in add_expense
    view_expenses()  # Refresh the table
    ^^^^^^^^^^^^^^^
  File "C:\Users\jeeva\AppData\Local\Temp\ipykernel_17804\870462463.py", line 47, in view_expenses
    for row in expense_tree.get_children():
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\jeeva\anaconda3\Lib\tkinter\ttk.py", line 1195, in get_children
    self.tk.call(self._w, "children", item or '') or ())
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
_tkinter.TclError: invalid command name ".!frame.!treeview"
```

In [ ]: