

Expense.py

```
1 # All Modules
2 import tkinter as tk
3 from tkinter import messagebox
4 from tkinter import ttk
5 import mysql.connector
6 from tkinter import font
7 import random
8
9 # Database Connection
10 def connect_to_db():
11     return mysql.connector.connect(
12         host="localhost",
13         user="Swagatam",
14         password="Swagatam@1509",
15         database="ExpenseTracker"
16     )
17
18 # Add Expense to Database
19 def add_expense():
20     date = entry_date.get()
21     category = combo_category.get()
22     amount = entry_amount.get()
23     description = entry_description.get()
24     while 1:
25         idx=random.randint(1,100)
26         if idx not in l:
27             l.add(idx)
28             break
29
30     if not (date and category and amount):
31         messagebox.showerror("Input Error", "All fields are required except description!")
32         return
33
34     if not entry_description.get():
35         description="No Description Added !"
36
37     try:
38         conn = connect_to_db()
39         cursor = conn.cursor()
40         query = "INSERT INTO expenses (id, date, category, amount, description) VALUES (%s, %s, %s, %s, %s)"
41         cursor.execute(query, (idx, date, category, amount, description))
42         conn.commit()
43         conn.close()
44         messagebox.showinfo("Success", "Expense added successfully!")
45         clear_fields()
46         display_expenses()
47     except Exception as e:
```

```

48         messagebox.showerror("Database Error", str(e))
49
50 # Display Expenses with Optional Filters
51 def display_expenses():
52     for row in tree.get_children():
53         tree.delete(row)
54
55     query = "SELECT * FROM expenses WHERE 1=1"
56     params = []
57
58     # Apply filters
59     if filter_start_date.get():
60         query += " AND date >= %s"
61         params.append(filter_start_date.get())
62     if filter_end_date.get():
63         query += " AND date <= %s"
64         params.append(filter_end_date.get())
65     if filter_category.get() and filter_category.get()!="None":
66         query += " AND category = %s"
67         params.append(filter_category.get())
68     if filter_start_amt.get():
69         query += " AND amount >= %s"
70         params.append(filter_start_amt.get())
71     if filter_end_amt.get():
72         query += " AND amount <= %s"
73         params.append(filter_end_amt.get())
74
75     try:
76         conn = connect_to_db()
77         cursor = conn.cursor()
78         cursor.execute(query, params)
79         rows = cursor.fetchall()
80         conn.close()
81         cnt = 0
82         for row in rows:
83             tree.insert("", tk.END, values=row)
84             cnt+=1
85         tk.Label(root, text="Number of Expenses Shown:
86 {}".format(cnt),font=f2,bg="Yellow",fg="Black",relief="sunken").place(x=700, y=190)
87     except Exception as e:
88         messagebox.showerror("Database Error", str(e))
89
90 # Clear Input Fields
91 def clear_fields():
92     entry_date.delete(0, tk.END)
93     combo_category.set("")
94     entry_amount.delete(0, tk.END)
95     entry_description.delete(0, tk.END)
96
97 # Edit Selected Expense

```

```

97 def edit_expense():
98     selected_item = tree.selection()
99     if not selected_item:
100         messagebox.showerror("Selection Error", "No expense selected!")
101         return
102
103     expense_id = tree.item(selected_item, "values")[0]
104     try:
105         conn = connect_to_db()
106         cursor = conn.cursor()
107         if len(entry_date.get())>0:
108             date = entry_date.get()
109         else:
110             date = tree.item(selected_item, "values")[1]
111         if len(combo_category.get())>0:
112             category = combo_category.get()
113         else:
114             category = tree.item(selected_item, "values")[2]
115         if len(entry_amount.get())>0:
116             amount = entry_amount.get()
117         else:
118             amount = tree.item(selected_item, "values")[3]
119         if len(entry_description.get())>0:
120             description = entry_description.get()
121         else:
122             description = tree.item(selected_item, "values")[4]
123         cursor.execute("UPDATE expenses SET date = %s, category = %s, amount = %s, description
= %s WHERE id = %s", (date,category,amount,description,expense_id,))
124         conn.commit()
125         conn.close()
126         messagebox.showinfo("Success", "Expense Edited successfully!")
127         display_expenses()
128     except Exception as e:
129         messagebox.showerror("Database Error", str(e))
130
131 # Delete Selected Expense
132 def delete_expense():
133     selected_item = tree.selection()
134     if not selected_item:
135         messagebox.showerror("Selection Error", "No expense selected!")
136         return
137
138     expense_id = tree.item(selected_item, "values")[0]
139     try:
140         conn = connect_to_db()
141         cursor = conn.cursor()
142         l.discard(int(expense_id))
143         cursor.execute("DELETE FROM expenses WHERE id = %s", (expense_id,))
144         conn.commit()
145         conn.close()

```

```

146         messagebox.showinfo("Success", "Expense deleted successfully!")
147         display_expenses()
148     except Exception as e:
149         messagebox.showerror("Database Error", str(e))
150
151 # Clear Filters
152 def clear_filters():
153     filter_start_date.delete(0, tk.END)
154     filter_end_date.delete(0, tk.END)
155     filter_category.set("")
156     filter_start_amt.delete(0, tk.END)
157     filter_end_amt.delete(0, tk.END)
158     display_expenses()
159
160 # Generate Reports
161 def generate_report():
162     query = """
163     SELECT
164         category,
165         COUNT(category) AS expense_count,
166         SUM(amount) AS total_amount
167     FROM expenses
168     GROUP BY category
169     ORDER BY SUM(amount) desc;
170     """
171     try:
172         conn = connect_to_db()
173         cursor = conn.cursor()
174         cursor.execute(query)
175         rows = cursor.fetchall()
176         conn.close()
177
178         # Display report in a new window
179         report_window = tk.Toplevel(root)
180         report_window.title("Expense Report")
181         report_window.geometry("600x400")
182
183         report_columns = ("Category", "Expense Count", "Total Amount")
184         report_tree = ttk.Treeview(report_window, columns=report_columns, show="headings",
height=15)
185         for col in report_columns:
186             report_tree.heading(col, text=col)
187             report_tree.column(col, anchor=tk.W)
188             report_tree.pack(fill=tk.BOTH, expand=True)
189
190         for row in rows:
191             report_tree.insert("", tk.END, values=row)
192
193     except Exception as e:
194         messagebox.showerror("Database Error", str(e))

```

```

195
196 # GUI Setup
197 root = tk.Tk()
198 root.title("Enhanced Expense Tracker")
199 root.geometry("1400x750")
200
201 l = set()
202
203 f1 = font.Font(family="Times New Roman",size=14)
204 f2 = font.Font(family="Times New Roman",size=18)
205
206 for i in range(10,1300,200):
207     for j in range(0,750,50):
208         t1=tk.Label(root, text="My Expense Tracker", font=f1, fg="Dark Grey")
209         t1.place(x=i,y=j)
210
211 img = tk.PhotoImage(file="Expensetrack.png")
212 imglabel = tk.Label(root, image=img)
213 imglabel.place(x=950,y=150)
214
215 # Input Fields
216 tk.Label(root, text="Date (YYYY-MM-DD) --",font=f1).place(x=100, y=20)
217 entry_date = tk.Entry(root,font=f1)
218 entry_date.place(x=300, y=20)
219
220 tk.Label(root, text="Category --",font=f1).place(x=100, y=60)
221 combo_category = ttk.Combobox(root, values=["Food", "Travel", "Shopping", "Medical",
222 "Education", "Bills", "Others"],font=f1)
223 combo_category.place(x=300, y=60)
224
225 tk.Label(root, text="Amount --",font=f1).place(x=100, y=100)
226 entry_amount = tk.Entry(root,font=f1)
227 entry_amount.place(x=300, y=100)
228
229 tk.Label(root, text="Description --",font=f1).place(x=100, y=140)
230 entry_description = tk.Entry(root,font=f1)
231 entry_description.place(x=300, y=140)
232
233 # Buttons
234 btn_add = tk.Button(root, text="Add Expense", bg="Green", fg="White", font=f1,
235 command=add_expense)
236 btn_add.place(x=50, y=180)
237
238 btn_clear = tk.Button(root, text="Clear", bg="Red", fg="White", font=f1, command=clear_fields)
239 btn_clear.place(x=500, y=180)
240
241 btn_delete = tk.Button(root, text="Delete Expense", bg="Blue", fg="White", font=f1,
242 command=delete_expense)
243 btn_delete.place(x=350, y=180)

```

```
242 btn_edit = tk.Button(root, text="Edit Expense", bg="Brown", fg="White", font=f1,
    command=edit_expense)
243 btn_edit.place(x=200, y=180)
244
245 # Filter Fields
246 tk.Label(root, text="Filter Start Date --",font=f1).place(x=600, y=20)
247 filter_start_date = tk.Entry(root,font=f1)
248 filter_start_date.place(x=750, y=20)
249
250 tk.Label(root, text="Filter End Date --",font=f1).place(x=950, y=20)
251 filter_end_date = tk.Entry(root,font=f1)
252 filter_end_date.place(x=1100, y=20)
253
254 tk.Label(root, text="Filter Category --",font=f1).place(x=700, y=100)
255 filter_category = ttk.Combobox(root, values=["None", "Food", "Travel", "Shopping", "Medical",
    "Education", "Bills", "Others"],font=f1)
256 filter_category.place(x=850, y=100)
257
258 tk.Label(root, text="Filter Start Amount --",font=f1).place(x=600, y=60)
259 filter_start_amt = tk.Entry(root,font=f1)
260 filter_start_amt.place(x=770, y=60)
261
262 tk.Label(root, text="Filter End Amount --",font=f1).place(x=970, y=60)
263 filter_end_amt = tk.Entry(root,font=f1)
264 filter_end_amt.place(x=1140, y=60)
265
266 btn_filter = tk.Button(root, text="Apply Filters", bg="Green", fg="White", font=f1,
    command=display_expenses)
267 btn_filter.place(x=700, y=140)
268
269 btn_clear_filters = tk.Button(root, text="Clear Filters", bg="Red", fg="White", font=f1,
    command=clear_filters)
270 btn_clear_filters.place(x=900, y=140)
271
272 btn_report = tk.Button(root, text="Generate Report", bg="Maroon", fg="White", font=f2,
    relief="sunken", command=generate_report)
273 btn_report.place(x=1100,y=150)
274
275 # Expense List (Treeview)
276 columns = ("ID", "Date", "Category", "Amount", "Description")
277 tree = ttk.Treeview(root, columns=columns, show="headings", height=24)
278 for col in columns:
279     tree.heading(col, text=col)
280     tree.column(col, anchor=tk.W)
281 tree.place(x=20, y=230)
282
283 # Populate Expenses
284 display_expenses()
285
286 root.mainloop()
287
```