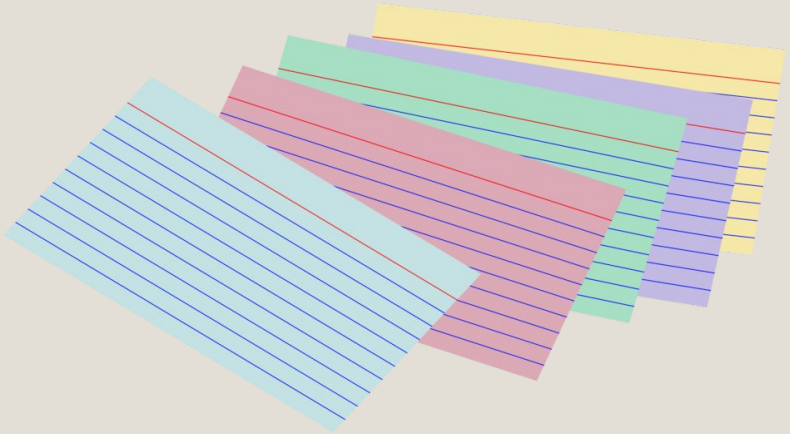
Python Final Project

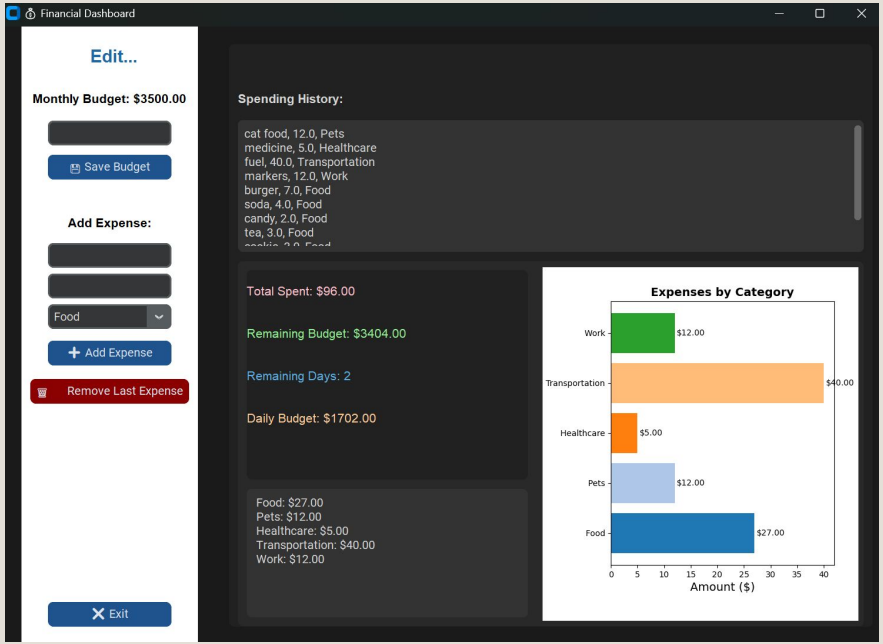
Personal Expense Dashboard

Index



Approach

Simple, user-friendly desktop app that helps users track daily expenses, manage a monthly budget and view categorized spending



SIDEBAR

Manipulates data

MAIN

Displays data

FUNCTIONALITIES

1 SET MONTHLY BUDGET

User inputs budget

Budget saved and displayed

2 ADD EXPENSE

User enters expense name, amount and category

Entry immediately appended

3 VIEW SPENDING HISTORY

Displays 10 most recent expenses

FUNCTIONALITIES

4 SUMMARIZE SPENDING

Displays total spent, remaining budget, remaining month days and calculated daily allowance

Shows categorized expenses in bar chart

5 REMOVE EXPENSE

Deletes most recently added expense

DESIGN ELEMENTS

1 OOP

```
class Finance:
    def __init__(self, name, category, amount) -> None:
        self.name = name
        self.category = category
        self.amount = amount

    def __repr__(self):
        return f"<Expense: {self.name}, {self.category}, ${float(self.amount):.2f}>"
```

DESIGN ELEMENTS

2 FILE HANDLING

```
expense_file_path = "Finances.csv"
budget_file_path = "Budget.txt"
```

```
# --- Functions ---
def load_budget():
    try:
        with open(budget_file_path, "r") as f:
            return float(f.read().strip())
    except (FileNotFoundError, ValueError):
        return 2000.0  # default budget

def save_budget(new_budget):
    if app_closing or not app.winfo_exists():
        return
    with open(budget_file_path, "w") as f:
                                udget))
```

```
def update_spending_history():
    if app_closing or not app.winfo_exists():
        return
    try:
        with open(expense_file_path, "r", encoding="utf-8") as f:
            lines = f.readlines()
    except FileNotFoundError:
        history_box.configure(state="normal")
        history_box.delete("1.0", "end")
        history_box.insert("1.0", "No expenses recorded yet.")
        history_box.configure(state="disabled")
        return
```

DESIGN ELEMENTS

3 EXCEPTION HANDLING

```
try:
    with open(expense_file_path, "r", encoding="utf-8") as f:
        lines = f.readlines()
except FileNotFoundError:
    status_label.configure(text="❌ No expenses found", text_color="red")
    return

expenses = []
for line in lines:
    if line.strip():
        parts = line.strip().split(",")
        if len(parts) == 3:
            name, amount_str, category = parts
            try:
                amount = float(amount_str)
                expenses.append(Finance(name=name, amount=amount, category=category))
            except ValueError:
                continue
```

DESIGN ELEMENTS

4 GUI


```
# Grid configuration
app.grid_rowconfigure(0, weight=1)
app.grid_columnconfigure(0, minsize=375)
app.grid_columnconfigure(1, weight=1)     # Main content grows/fills

frame_sidebar = customtkinter.CTkFrame(app, width=400, corner_radius=0, fg_color="white")
frame_sidebar.grid(row=0, column=0, sticky="ns")
frame_sidebar.grid_rowconfigure(10, weight=1)

frame_main = customtkinter.CTkFrame(app)
frame_main.grid(row=0, column=1, sticky="nsew", padx=10, pady=20)
frame_main.grid_rowconfigure(3, weight=1)
frame_main.grid_columnconfigure(0, weight=1)

# Sidebar Widgets
sidebar_title = customtkinter.CTkLabel(frame_sidebar, text="Edit...", font=("Arial", 20, "bold"),
sidebar_title.pack(pady=20, padx=(20, 10))

budget_label = customtkinter.CTkLabel(frame_sidebar, text=f"Monthly Budget: ${budget:.2f}", font=
budget_label.pack(pady=(0, 5), padx=(10, 10))
```

DESIGN ELEMENTS

5 REGEXES

```
def is_valid_name(name):
    # Allow letters, numbers, spaces, hyphens, apostrophes, and periods
    pattern = r"^[\w\s\-\.'']+$"
    return bool(re.match(pattern, name))

def is_valid_amount(amount_str):
    # Matches the following amount formats:
    #  - .99
    #  - 0.99
    #  - 99
    #  - 99.9
    #  - 99.99
    pattern = r"^(\d+)?(\.\d{1,2})?$"
    return bool(re.match(pattern, amount_str))
```

DESIGN ELEMENTS

6 DATA VISUALIZATION

```
categories = list(data.keys())
amounts = list(data.values())

fig, ax = plt.subplots(figsize=(6, 4))
colors = plt.get_cmap("tab20").colors

bars = ax.barh(categories, amounts, color=colors[:len(categories)])
ax.set_title("Expenses by Category", fontsize=14, weight='bold')
ax.set_xlabel("Amount ($)", fontsize=14)

for bar, amount in zip(bars, amounts):
    ax.text(bar.get_width() + max(amounts)*0.01, bar.get_y() + bar.get_height()/2,
            f"${amount:.2f}", va="center", fontsize=10, color="black")

fig.tight_layout()

canvas = FigureCanvasTkAgg(fig, master=parent_frame)
canvas.draw()
canvas.get_tk_widget().grid(row=0, column=1, rowspan=2, padx=10, pady=10, sticky="nsew")
```

DESIGN ELEMENTS

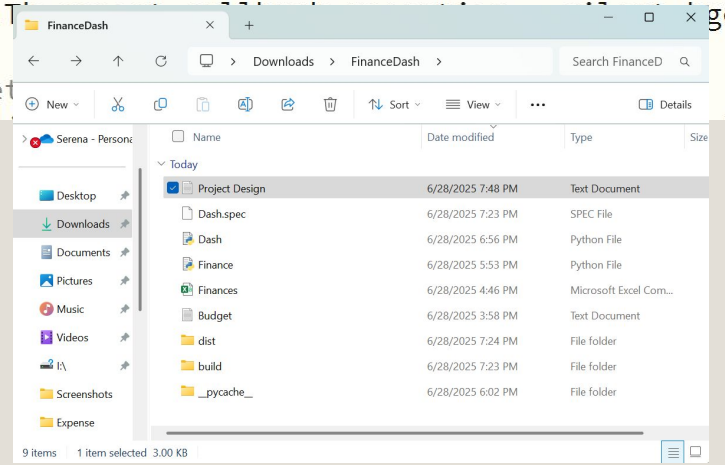
7 DATE AND TIME

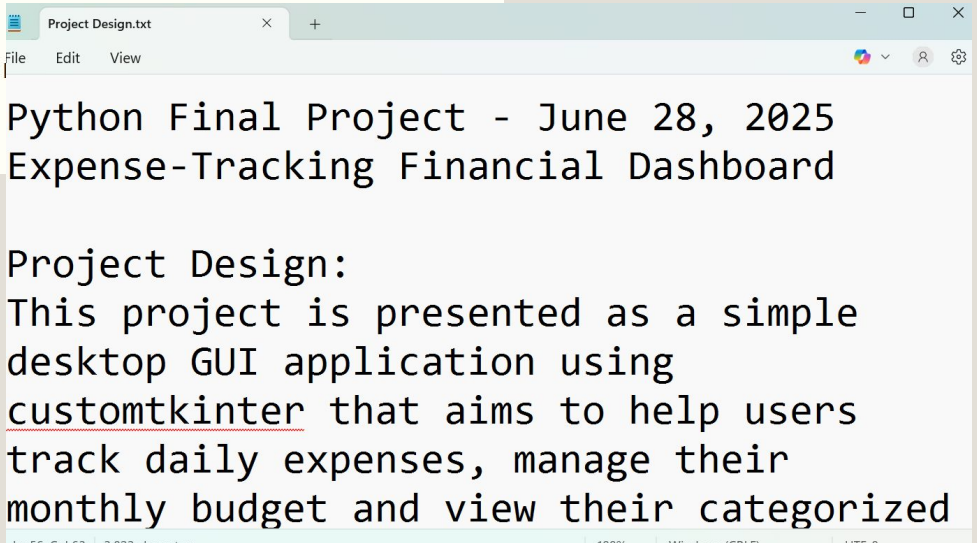
```
total_spent = sum(x.amount for x in expenses)
remaining_budget = budget - total_spent
now = datetime.datetime.now()
days_left = calendar.monthrange(now.year, now.month)[1] - now.day
daily_budget = remaining_budget / days_left if days_left > 0 else 0

# Create the summary text lines
summary_lines = [
    f"Total Spent: ${total_spent:.2f}",
    f"Remaining Budget: ${remaining_budget:.2f}",
    f"Remaining Days: {days_left}",
    f"Daily Budget: ${daily_budget:.2f}",
    "",  # blank line
]
```

DESIGN ELEMENTS

8 DOCUMENTATION

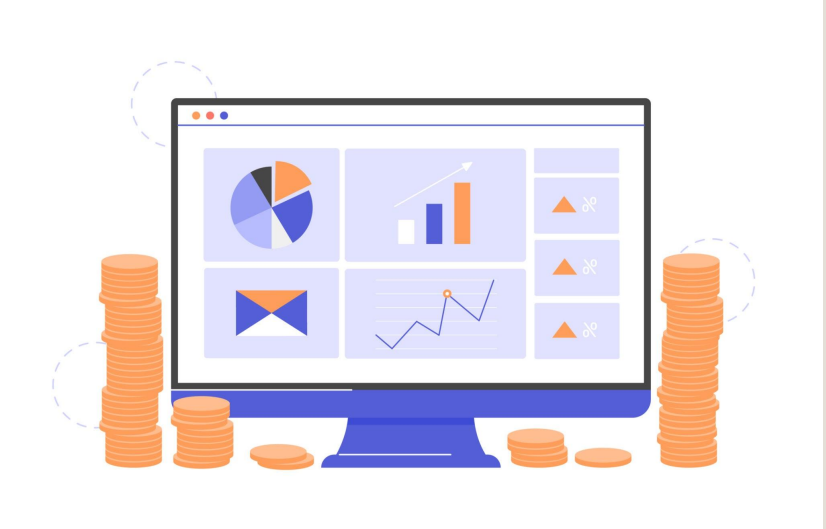
```
# --- Global Flag ---
app_closing = False

# --- Suppress Tkinter bgerror to avoid some errors after app is destroyed --
def silent_bgerror(*args, **kwargs):
    pass
tkinter.T                                                    ge
# --- Set
```



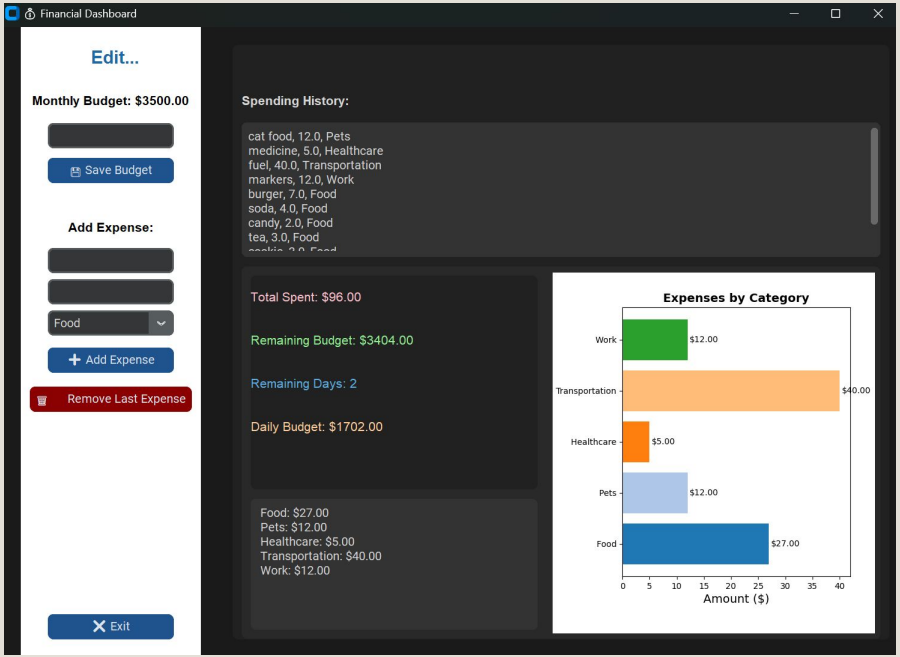


NEXT STEPS

- Additional charts, graphs and plots
- View data by month
- Search by expense name, amount, etc.
- Excel integration
- Track income, deposits and assets
- Display net cash flow



Thank You



SERENA REESE

06.28.2025

CS1314 – D10

Summer I – 2025

Instructor Jason Watson