BACSE101 Problem Solving using Python

PROJECT REPORT

on

Budget Tracker

Prepared by

Divyanshu Antil – 25BAI0006
Ustat Kaur – 25BAI0022
Kushal Bagla – 25BAI0044
Saanvi Somani-25BAI0049
Tejash Bhudolia-25BAI0086

Under the supervision of

Dr. Chiranjeevi Chalasani

School of Computer Science and Engineering
Vellore Institute of Technology, Vellore.

November 07, 2025

Table of Contents

Abstract

This project presents a Budget Tracker system using Python and PyQt5. The application helps users record income, expenses, view balance, search transactions, delete entries, and maintain backups. It uses text files to store financial data and offers a graphical interface for easy use. The system focuses on minimal storage requirements by using text-based files, avoiding the need for external databases. With its intuitive UI, users can maintain financial discipline, analyse spending patterns, and ensure better budget planning. The application is lightweight, portable, and suitable for beginners exploring GUI programming in Python.

1. Introduction

The Budget Tracker application is a simple yet efficient financial management tool developed using Python. It uses a GUI created with PyQt5, enabling users to add income, expenses, view budget summaries, and manage data

1.1 Domain Information

- Domain: Finance and Desktop Applications
- Technology: Python, PyQt5 UI

1.2 Software Libraries Used

- PyQt5 – GUI development
- OS – file handling
- Python-docx – documentation

1.3 Contributions by Team Members

- Kushal and Tejash – Preparing the budget tracker functions
- Ustat and Saanvi – Developing the GUI
- Divyanshu – Preparing the document and researching about the topic

1.4 Challenges Faced

- Maintaining file consistency after every operation
- Designing user-friendly UI layout
- Implementing backup and restore without database

2. Problem Statement and Objectives

The objective of this project is to design a user-friendly desktop application to manage personal finances. The system should:

- Record income and expenses
- Display total balance and transaction history
- Allow searching and deletion
- Support backup and restore features

3. Implementation

3.1 Feature

The project contains multiple features such as adding income, expense, search, delete, backup and restore.

Source Code:

```
import os

from PyQt5.QtWidgets import *

from PyQt5.QtGui import *

from PyQt5.QtCore import *


BUDGET_FILE='budget.txt';TRANSACTIONS_FILE='transactions.txt';BACKUP_FILE='budget_backup.txt'


def init_files():

   if not os.path.exists(BUDGET_FILE):

      with open(BUDGET_FILE,'w') as f:f.write('Income: 0\nExpenses: 0\nBalance: 0\n')

   if not os.path.exists(TRANSACTIONS_FILE):

      with open(TRANSACTIONS_FILE,'w') as f:f.write('Type,Amount,Category,Description\n')


def read_budget():

   b={'Income':0,'Expenses':0,'Balance':0}

   try:

      with open(BUDGET_FILE) as f:

         for l in f:

            if l.startswith('Income'):b['Income']=int(l.split(': ')[1])
```

```
            elif l.startswith('Expenses'):b['Expenses']=int(l.split(': ')[1])
            elif l.startswith('Balance'):b['Balance']=int(l.split(': ')[1])
    except:pass
    return b

def write_budget(b):
    with open(BUDGET_FILE,'w') as f:
        f.write(f"Income: {b['Income']}\nExpenses: {b['Expenses']}\nBalance: {b['Balance']}\n")

def add_transaction(t,a,c,d):
    with open(TRANSACTIONS_FILE,'a') as f:f.write(f"{t},{a},{c},{d}\n")
    b=read_budget()
    if t=='Income':b['Income']+=a
    else:b['Expenses']+=a
    b['Balance']=b['Income']-b['Expenses']
    write_budget(b)

def all_transactions():
    if not os.path.exists(TRANSACTIONS_FILE):return []
    with open(TRANSACTIONS_FILE) as f:
        lines=f.readlines()[1:]
    data=[]
    for l in lines:
        parts=l.strip().split(',',3)
        if len(parts)==4:data.append(parts)
    return data
```

```
def recalc_budget():
    b={'Income':0,'Expenses':0,'Balance':0}
    with open(TRANSACTIONS_FILE) as f:
        for l in f.readlines()[1:]:
            t,a,_,_=l.strip().split(',',3);a=int(a)
            if t=='Income':b['Income']+=a
            else:b['Expenses']+=a
    b['Balance']=b['Income']-b['Expenses']
    write_budget(b)


class BudgetApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("💰 Budget Tracker (PyQt5)")
        self.setGeometry(200,100,900,600)
        self.setStyleSheet("""
            QMainWindow{background-color:#121212;color:white;font-family:'Segoe
UI';}
            QPushButton{background-
color:#0077b6;border:none;color:white;padding:8px 12px;border-radius:6px;font-
weight:bold;}
            QPushButton:hover{background-color:#00b4d8;}
            QTableWidget{background-color:#1e1e1e;border:1px solid
#333;color:white;gridline-color:#444;}
            QHeaderView::section{background-color:#00b4d8;color:white;font-
weight:bold;border:none;}
        """)
        self.initUI()
        self.refresh_ui()
```

```
def initUI(self):
    main=QWidget();layout=QVBoxLayout(main)

    title=QLabel("💼 Budget Dashboard");title.setAlignment(Qt.AlignCenter)
    title.setStyleSheet("font-size:22px;color:#00b4d8;font-weight:bold;")
    layout.addWidget(title)

    # Budget summary cards
    card_layout=QHBoxLayout()
    self.income_lbl=self.make_card("Income","lime")
    self.expense_lbl=self.make_card("Expenses","tomato")
    self.balance_lbl=self.make_card("Balance","#00b4d8")
    card_layout.addWidget(self.income_lbl[0])
    card_layout.addWidget(self.expense_lbl[0])
    card_layout.addWidget(self.balance_lbl[0])
    layout.addLayout(card_layout)

    # Buttons
    btn_layout=QHBoxLayout()
    buttons=[
      ("➕ Add Income",self.add_income),
      ("➖ Add Expense",self.add_expense),
      ("🔍 Search",self.search_transactions),
      ("🗑 Delete Last",self.delete_last),
      ("💾 Backup",self.backup),
      ("♻ Restore",self.restore)
    ]
```

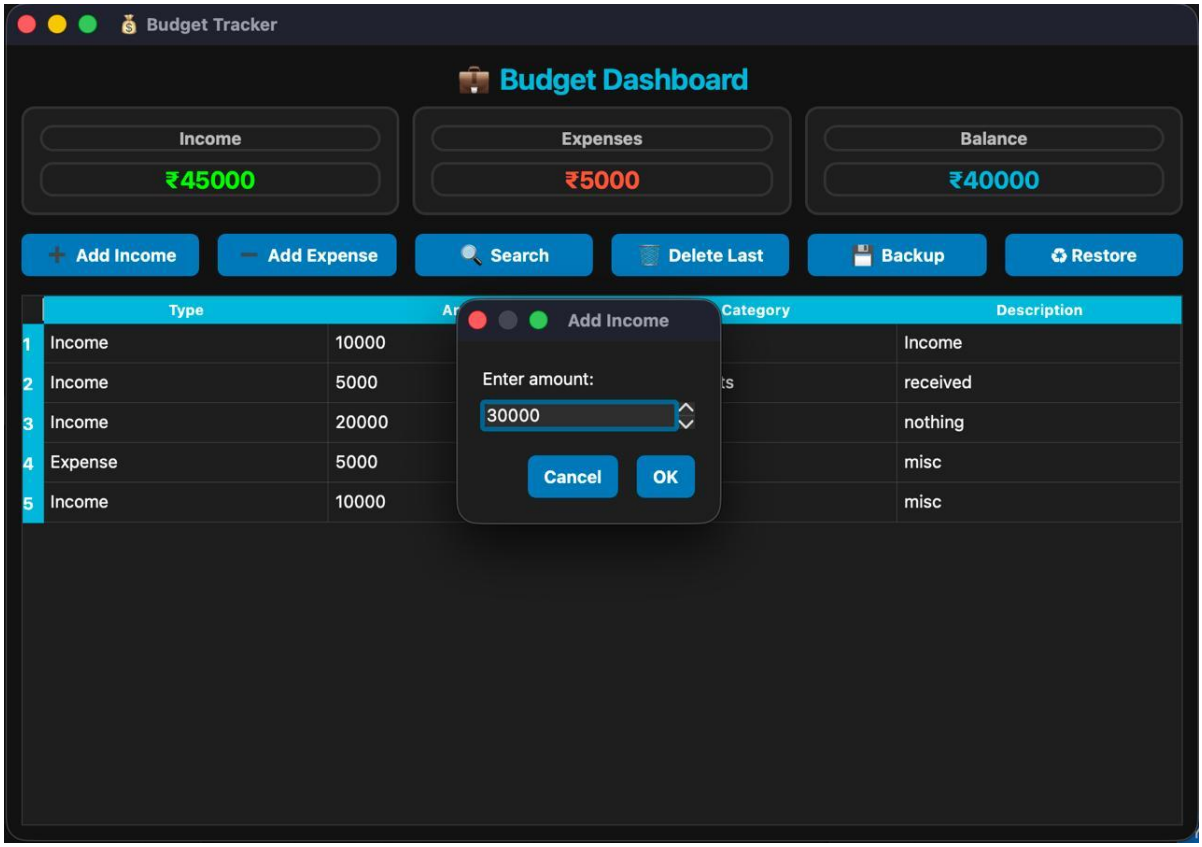
```
        for text,func in buttons:
            b=QPushButton(text);b.clicked.connect(func);btn_layout.addWidget(b)
        layout.addLayout(btn_layout)

        # Table
        self.table=QTableWidget();self.table.setColumnCount(4)
self.table.setHorizontalHeaderLabels(["Type","Amount","Category","Description"]
)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        layout.addWidget(self.table)
        self.setCentralWidget(main)

    def make_card(self,title,color):
        frame=QFrame();frame.setStyleSheet("background-color:#1e1e1e;border:2px
solid #333;border-radius:10px;")
        v=QVBoxLayout(frame)
t=QLabel(title);t.setAlignment(Qt.AlignCenter);t.setStyleSheet("color:#bbb;font-
weight:bold;")
val=QLabel("₹0");val.setAlignment(Qt.AlignCenter);val.setStyleSheet(f"color:{color
};font-size:18px;font-weight:bold;")
        v.addWidget(t);v.addWidget(val)
        return frame,val

    def refresh_ui(self):
        b=read_budget()
        self.income_lbl[1].setText(f"₹{b['Income']}")
        self.expense_lbl[1].setText(f"₹{b['Expenses']}")
        self.balance_lbl[1].setText(f"₹{b['Balance']}")
```

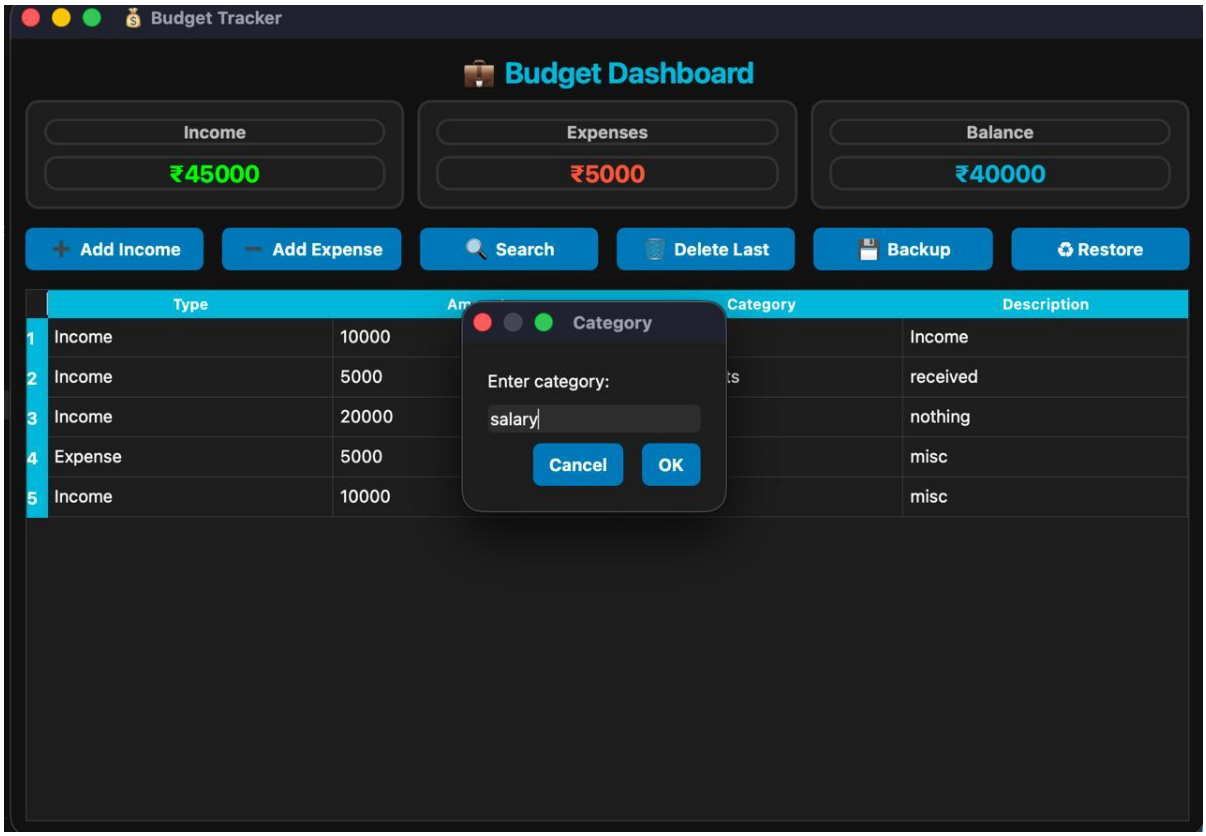
```
        data=all_transactions()
        self.table.setRowCount(0)
        for r,row in enumerate(data):
            self.table.insertRow(r)
            for c,val in enumerate(row):
                self.table.setItem(r,c,QTableWidgetItem(val))

    def add_income(self):
        a,ok=QInputDialog.getInt(self,"Add Income","Enter amount:")
        if not ok:return
        c,ok=QInputDialog.getText(self,"Category","Enter category:")
        if not ok:return
        d,ok=QInputDialog.getText(self,"Description","Enter description:")
        if not ok:return
        add_transaction('Income',a,c or 'General',d or '')
        self.refresh_ui()

    def add_expense(self):
        a,ok=QInputDialog.getInt(self,"Add Expense","Enter amount:")
        if not ok:return
        c,ok=QInputDialog.getText(self,"Category","Enter category:")
        if not ok:return
        d,ok=QInputDialog.getText(self,"Description","Enter description:")
        if not ok:return
        add_transaction('Expense',a,c or 'General',d or '')
        self.refresh_ui()

    def search_transactions(self):
```

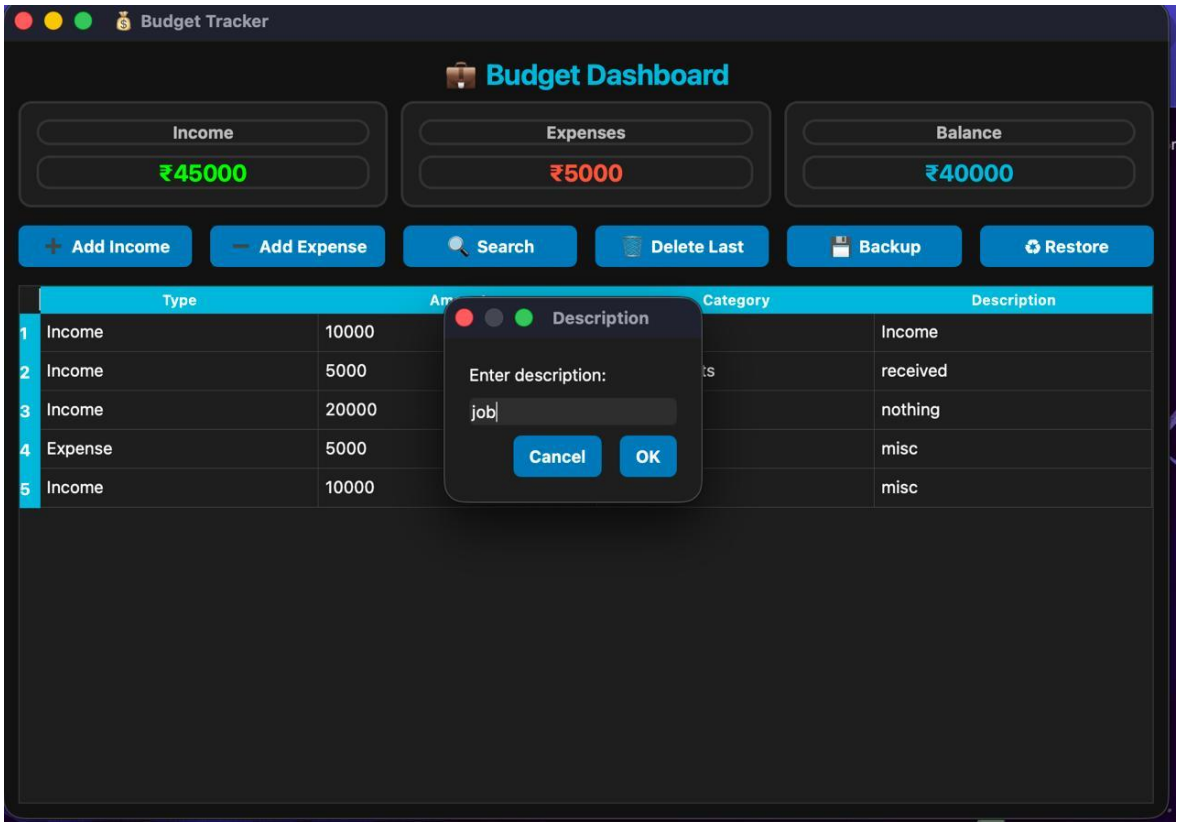
```
        kw,ok=QInputDialog.getText(self,"Search","Enter keyword:")
        if not ok:return
        data=[row for row in all_transactions() if kw.lower() in ','.join(row).lower()]
        self.table.setRowCount(0)
        for r,row in enumerate(data):
            self.table.insertRow(r)
            for c,val in enumerate(row):
                self.table.setItem(r,c,QTableWidgetItem(val))

    def delete_last(self):
        with open(TRANSACTIONS_FILE) as f:lines=f.readlines()
        if len(lines)<=1:return QMessageBox.information(self,"Info","No transactions to delete.")
        with open(TRANSACTIONS_FILE,'w') as f:f.writelines(lines[:-1])
        recalc_budget();self.refresh_ui()
        QMessageBox.information(self,"Deleted","Last transaction deleted.")

    def backup(self):
        with open(BACKUP_FILE,'w') as b,open(BUDGET_FILE) as bf,open(TRANSACTIONS_FILE) as tf:
            b.write(bf.read()+"\n--- Transactions ---\n"+tf.read())
        QMessageBox.information(self,"Backup","Backup created successfully.")

    def restore(self):
        if not os.path.exists(BACKUP_FILE):return QMessageBox.warning(self,"Error","No backup found.")
        with open(BACKUP_FILE) as b:lines=b.readlines()
        bd,td=lines[:3],lines[4:]
        with open(BUDGET_FILE,'w') as f:f.writelines(bd)
```

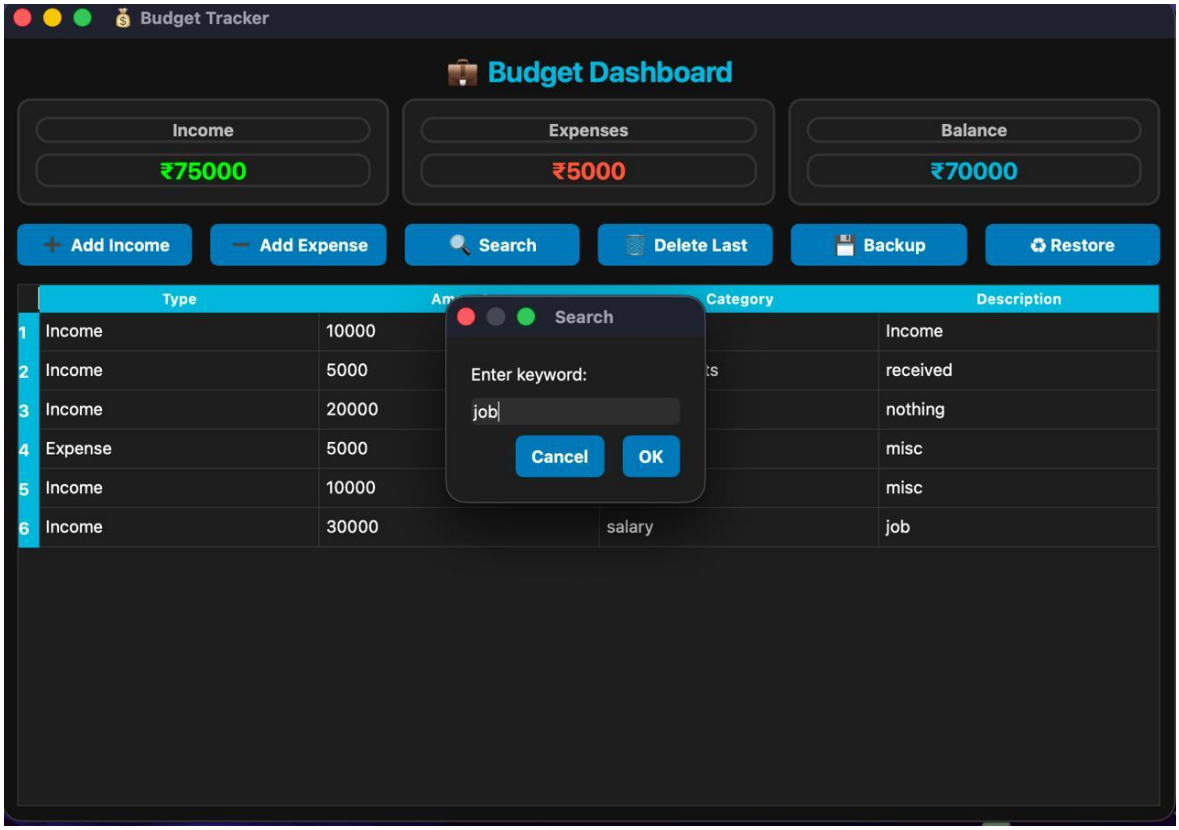
```
        with open(TRANSACTIONS_FILE,'w') as f:f.writelines(td)
        self.refresh_ui()
        QMessageBox.information(self,"Restored","Data restored from backup.")

if __name__=="__main__":
    import sys
    init_files()
    app=QApplication(sys.argv)
    win=BudgetApp()
    win.show()
    sys.exit(app.exec_())
```

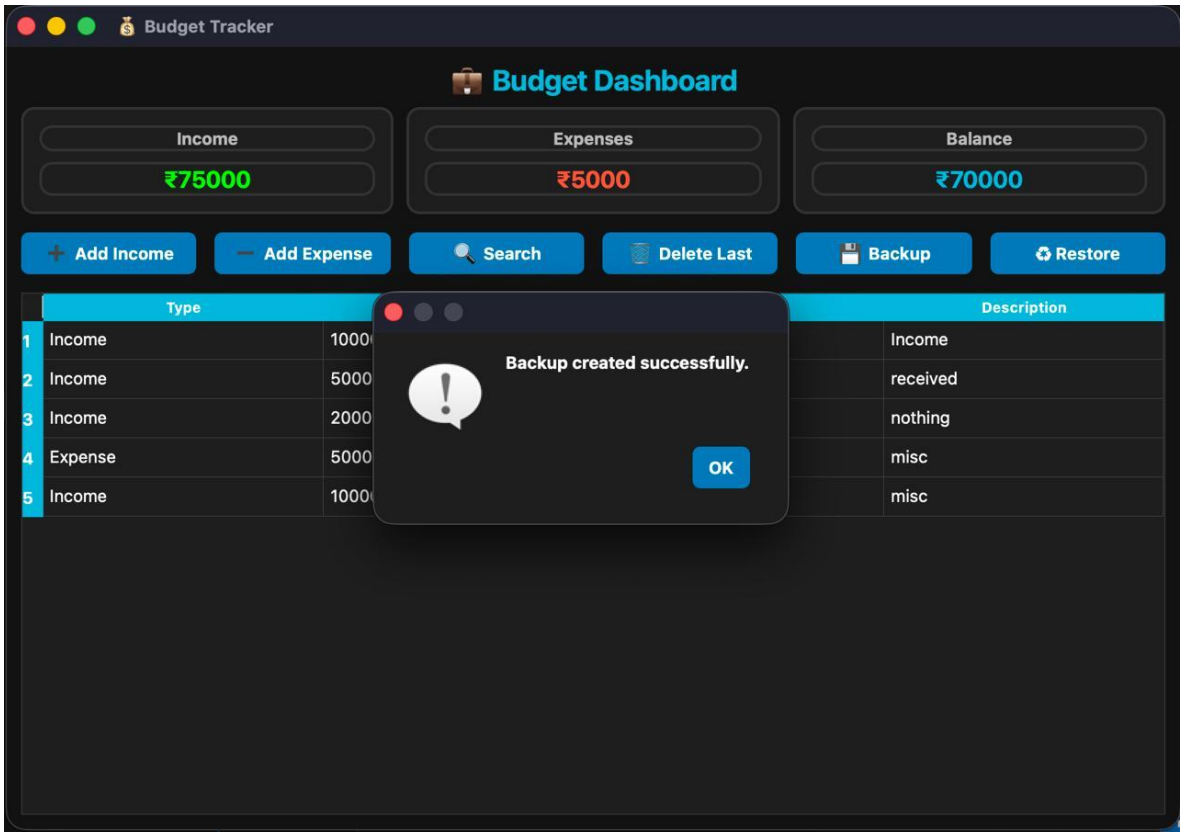
4. Demo Screenshots

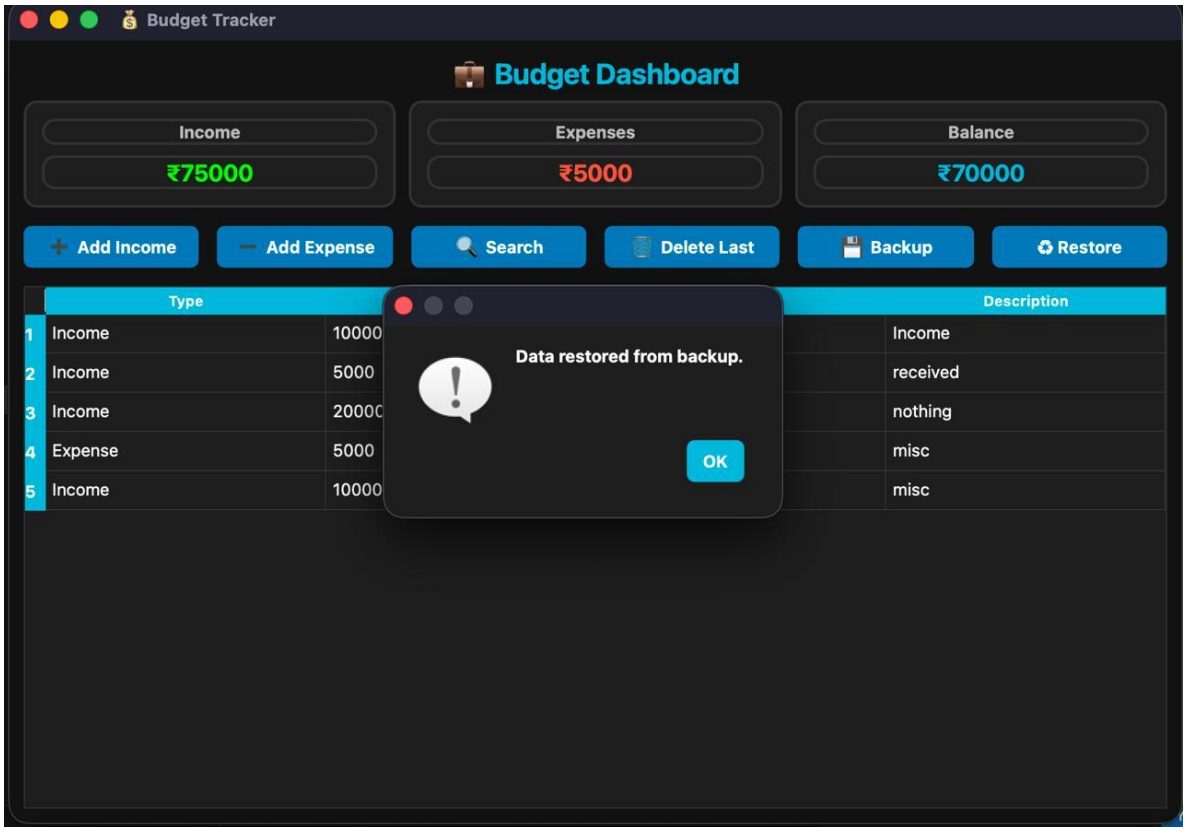





Budget Tracker
Budget Dashboard
Income
₹45000
Expenses
₹5000
Balance
₹40000
Add Income
Add Expense
Search
Delete Last
Backup
Restore
Type
Category
Description
1 Income 10000 Income
2 Income 5000 received
3 Income 20000 nothing
4 Expense 5000 misc
5 Income 10000 misc
Description
Enter description:
job
Cancel
OK


Budget Tracker
Budget Dashboard
Income
₹75000
Expenses
₹5000
Balance
₹70000
Add Income
Add Expense
Search
Delete Last
Backup
Restore
Type
Category
Description
1 Income 10000 Income
2 Income 5000 received
3 Income 20000 nothing
4 Expense 5000 misc
5 Income 10000 misc
6 Income 30000 salary job
Search
Enter keyword:
job
Cancel
OK


Budget Tracker
Budget Dashboard
Income
₹75000
Expenses
₹5000
Balance
₹70000
Add Income
Add Expense
Search
Delete Last
Backup
Restore
Type
Description
1 Income 1000
2 Income 5000
3 Income 2000
4 Expense 5000
5 Income 1000
Income
received
nothing
misc
misc
Backup created successfully.
OK


Budget Tracker
Budget Dashboard
Income
₹75000
Expenses
₹5000
Balance
₹70000
Add Income
Add Expense
Search
Delete Last
Backup
Restore
Type
Description
1 Income 10000
2 Income 5000
3 Income 20000
4 Expense 5000
5 Income 10000
Income
received
nothing
misc
misc
Data restored from backup.
OK

5. Conclusion

The Budget Tracker application successfully provides a personal finance management solution with an interactive interface. It demonstrates GUI programming, file handling, and data management concepts in Python. The application succeeds in simplifying budget tracking through a clean and manageable interface. The use of PyQt5 ensures that users interact with the system seamlessly while the backend file storage keeps data lightweight and accessible. The project effectively meets all objectives and demonstrates practical implementation of Python concepts.