# INVEST SMART

*A*

*Mini Project Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

# BACHELOR OF ENGINEERING

*IN*

# INFORMATION TECHNOLOGY

By

# P.KARTHIK<1602-23-737-092>

**V.SRAVYA<1602-23-737-118 >**

# M.JITHENDRA SAI<1602-23-737-091 >



**Department of Information Technology Vasavi College of Engineering (Autonomous)**

# ACCREDITED BY NAAC WITH 'A++' GRADE

**(Affiliated to Osmania University and Approved by AICTE) Ibrahimbagh, Hyderabad-31**

# 2024

**Vasavi College of Engineering (Autonomous)**

***ACCREDITED BY NAAC WITH 'A++' GRADE***

# (Affiliated to Osmania University and Approved by AICTE) Hyderabad-500 031

**Department of Information Technology**



# DECLARATION BY THE CANDIDATE

We, **P.KARTHIK, V.SRAVYA ,** and **M.JITHENDRA SAI**

bearing hall ticket numbers, **1602-23-737-092, 1602-23-737-118** and **1602-23-737-091**, hereby declare that the project report entitled **”INVEST SMART”** is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report havenot been submitted to any other university or institute for the award of any other degree or diploma.

# P.KARTHIK

**1602-23-737-092**

# V.SRAVYA

**1602-23-737-118**

# M.JITHENDRA SAI

**1602-23-737-091**

(Faculty In-Charge) (External Examiner) (Head,Dept of IT)

# ACKNOWLEDGMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering, for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators

DIVYA LINGINENI and Mr.Krushi Raj Tula for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable input and suggestion

**ABSTRACT**

**Financial Overview and Retirement Calculation Tool**

The "Financial Overview and Retirement Calculation Tool" is a user-friendly application designed to assist individuals in planning their financial future. It allows users to compute their projected retirement corpus and monthly income after retirement, based on their current savings and investment strategies. This tool combines simplicity with powerful features such as scenario analysis, graphical representations, and data management, making retirement planning an accessible task for everyone. Application is divided into two subparts Calculation and Graphical Visualization

#### ****1. Calculation****

* Computes total retirement savings and monthly income over 20 years.
* Analyzes investments using predefined return rates for options like Mutual Funds and Fixed Deposits.
* Evaluates the impact of increased contributions on savings.
* Ensures accuracy with robust input validation.

#### ****2. Graphical Visualization****

* Displays income and savings breakdown using clear bar graphs.
* Shows the effect of contribution changes on total savings.
* User-friendly graphs with intuitive labels and color-coded categories.

**Table of Contents**

1. **Cover Sheet** ............................................................................ Pg 1

1. **Declaration** ............................................................................. Pg 2
2. **Acknowledgements** ..................................................................Pg 3
3. **Abstract** ....................................................................................Pg 4
4. **Table of Contents** ....................................................................Pg 5
5. **Abstract and Introduction** .....................................................Pg 6
   * 6.1 General Information about the Domain
   * 6.2 Features Prioritized in the Project
6. **Technology Requirements** ......................................................Pg 8
   * 7.1 Software and Hardware Requirements
7. **Proposed Work** .......................................................................Pg 9
   * 8.1 Design
     + 8.1.1 High-Level Design Diagram
     + 8.1.2 Workflow Diagram
     + 8.1.3 Diagramatic Representation
   * 8.2 Implementation
     + 8.2.1 Module-Wise Code Snippets
     + 8.2.8 Folder Structure
   * 8.3 Github link
     + 8.3.1 Git Link
8. **Results** ......................................................................................Pg 22
   * 9.1 Screenshots of Test Cases and Outputs
9. **Additional Knowledge Gained** ................................................Pg 28
10. **Conclusion and Future Work** ..................................................Pg 29
11. **References** ................................................................................. Pg 30

**ABSTRACT&INTRODUCTION**

**ABSTRACT**

**Financial Overview and Retirement Calculation Tool**

The "Financial Overview and Retirement Calculation Tool" is a user-friendly application designed to assist individuals in planning their financial future. It allows users to compute their projected retirement corpus and monthly income after retirement, based on their current savings and investment strategies. This tool combines simplicity with powerful features such as scenario analysis, graphical representations, and data management, making retirement planning an accessible task for everyone. Application is divided into two subparts Calculation and Graphical Visualization

#### ****1. Calculation****

* Computes total retirement savings and monthly income over 20 years.
* Analyzes investments using predefined return rates for options like Mutual Funds,Gold and Fixed Deposits.
* Evaluates the impact of increased contributions on savings.
* Ensures accuracy with robust input validation.

#### ****2. Graphical Visualization****

* Displays income and savings breakdown using clear bar graphs.
* Shows the effect of contribution changes on total savings.
* User-friendly graphs with intuitive labels and color-coded categories.

**INTRODUCTION**

**a. Information About the Project Domain**

The project falls under the domain of **personal finance management** with a specific focus on **retirement planning**. As financial literacy becomes increasingly important, individuals are seeking tools that help them prepare for their future while navigating diverse investment opportunities.

**b. Features Specific to the Project**

1. **Retirement Savings Projection:** Calculates total retirement corpus and expected monthly income for 20 years post-retirement.

2. **Graphical Visualizations:** Provides clear bar graphs for income breakdown and savings comparisons.

3. **Scenario Analysis:** Explores how increasing monthly contributions affects retirement savings.

4. **Data Handling:** Save user inputs and retrieve them for continued financial planning.

5. **Investment Options:** Supports multiple investment types with pre-configured annual return rates (e.g., Mutual Funds, Fixed Deposits).

6. **Error Handling:** Validates user inputs to ensure accurate calculations and guides users with clear error messages.

7. **Help and About Sections:** Offers user instructions and application details.

**TECHNOLOGY**

1. **Software Requirements**

1. **Programming Language:** Python 3.7 or above.

2. **Libraries:** tkinter, matplotlib.

3. **Operating System:** Windows/Linux/MacOS with GUI support.

4. **Additional Tools:** Python IDE (e.g., PyCharm, VSCode) for development.

1. **Hardware Requirements**

1. **Operating System:** Windows, macOS, or Linux.

2. **Internet Access:** Required for initial setup or updates (optional for offline use).

**3. Processor:** Dual-core CPU or higher.

**4.RAM:** Minimum 4 GB (8 GB recommended for smoother performance).

**5.Storage:** At least 500 MB of free disk space.

**6.Display:** Resolution of 1366 x 768 pixels or higher.

**PROPOSED WORK**

**A.Design**

**I . High-Level Design Diagram**

1. **User Interface (UI)**

* Built using **Tkinter**.
* Collects inputs (income, age, contributions, investment type).
* Includes buttons for calculations and graphs, with a menu bar for file operations and help.

2. **Processing Logic**

* Validates inputs.
* Performs financial calculations:
  + Total retirement corpus (compound interest).
  + Monthly retirement income (fixed withdrawal period).

3. **Visualization**

* Generates graphs using **Matplotlib**:
  + Financial overview (expenses, retirement income, contributions).
  + Impact of additional savings.

4. **File Management**

* Saves user data to a text file.
* Loads data from a text file into input fields.

5.**Help and About**

* Provides usage instructions and tool details via message boxes.

6. **Error Handling**

* Ensures input correctness and handles file operation errors gracefully with clear error messages.

**Workflow Diagram**

1. **User Input**

* Collect income, age, contributions, investment type.

2. **Validate Input**

* If invalid → Show error.
* Else → Proceed to calculations.

3. **Perform Calculations**

* Calculate retirement corpus and income.

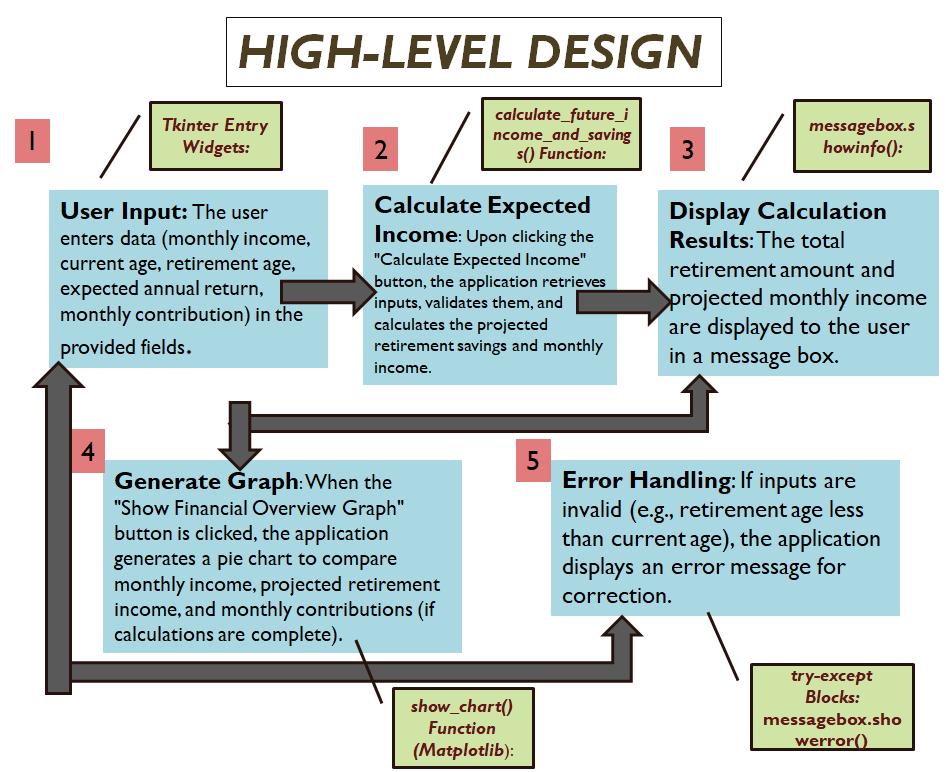
4. **Show Results**

* Display results in message boxes.
* Generate visual graphs.

5. **Additional Features**

* Save data → File operations.
* Load data → File operations.
* Help/Documentation → Display instructions.

DIAGRAMATIC REPRESENTATION:



**IMPLEMENTATION**

**I . Module-wise Code for the Entire Project**

### ****1. User Interface (UI) Module****

### Handles input collection, buttons, and menus.

from tkinter import \*

from tkinter import filedialog, messagebox, ttk

# Initialize main window

root = Tk()

root.title("Financial Overview")

root.geometry('580x450')

root.configure(bg='#f0f8ff')

# UI labels and input fields

label\_font = ('Helvetica', 12, 'bold')

entry\_font = ('Helvetica', 12)

bg\_color = '#e6f2ff'

fg\_color = '#003366'

labels = ["Monthly Income:", "Current Age:", "Retirement Age:", "Investment Type:", "Monthly Contribution:"]

entries = {}

for i, text in enumerate(labels):

Label(root, text=text, font=label\_font, bg=bg\_color, fg=fg\_color).grid(row=i, column=0, padx=10, pady=10, sticky=W)

if text == "Investment Type:":

investment\_type = StringVar()

entries[text] = ttk.Combobox(root, textvariable=investment\_type, font=entry\_font, state="readonly")

entries[text]['values'] = ["Fixed Deposits (FD)", "Mutual Funds", "PPF", "Gold"]

entries[text].set("Select Investment")

else:

entries[text] = Entry(root, font=entry\_font)

entries[text].grid(row=i, column=1, padx=10, pady=10)

# Buttons

Button(root, text="Calculate", font=label\_font, bg="green", fg="white", command=lambda: calculate(entries)).grid(row=5, column=0, pady=20)

Button(root, text="Show Graph", font=label\_font, bg="blue", fg="white", command=lambda: show\_graph(entries)).grid(row=5, column=1, pady=20)

# Menu

menu\_bar = Menu(root)

root.config(menu=menu\_bar)

file\_menu = Menu(menu\_bar, tearoff=0)

file\_menu.add\_command(label="Save", command=lambda: save\_to\_file(entries))

file\_menu.add\_command(label="Open", command=lambda: open\_file(entries))

file\_menu.add\_command(label="Exit", command=root.quit)

menu\_bar.add\_cascade(label="File", menu=file\_menu)

help\_menu = Menu(menu\_bar, tearoff=0)

help\_menu.add\_command(label="How to Use", command=show\_help)

menu\_bar.add\_cascade(label="Help", menu=help\_menu)

### ****2. Processing Logic Module****

Performs calculations and validation.

def validate\_inputs(entries):

try:

current\_age = int(entries["Current Age:"].get())

retirement\_age = int(entries["Retirement Age:"].get())

if retirement\_age <= current\_age:

raise ValueError("Retirement age must be greater than current age.")

return current\_age, retirement\_age

except ValueError as e:

messagebox.showerror("Validation Error", str(e))

return None

def calculate(entries):

try:

current\_age, retirement\_age = validate\_inputs(entries)

if not current\_age: return

monthly\_contribution = float(entries["Monthly Contribution:"].get())

annual\_return = 0.07 # Example: Fixed Deposits

total\_years = retirement\_age - current\_age

total\_savings = monthly\_contribution \* (((1 + annual\_return / 12) \*\* (total\_years \* 12) - 1) / (annual\_return / 12))

monthly\_income = total\_savings / (20 \* 12) # Assume 20 years of withdrawals

messagebox.showinfo("Results", f"Total Savings: ₹{total\_savings:,.2f}\nMonthly Income: ₹{monthly\_income:,.2f}")

except Exception as e:

messagebox.showerror("Error", str(e))

investment\_options = {

"Fixed Deposits (FD)": 0.07,

"Public Provident Fund (PPF)": 0.071,

"Mutual Funds": 0.11,

"Gold": 0.13,

"National Pension System (NPS)": 0.1,

"Post Office Monthly Income Scheme (POMIS)": 0.074,

"Real Estate": 0.1,

"Corporate Bonds": 0.08,

"Government Bonds": 0.065

}

investment\_type = StringVar()

investment\_dropdown = ttk.Combobox(

root, textvariable=investment\_type, font=("Helvetica", 12),

values=list(investment\_options.keys()), state="readonly"

)

investment\_dropdown.grid(row=4, column=2, padx=15, pady=10)

investment\_dropdown.set("Select Investment Type")

### ****3. Visualization Module****

import matplotlib.pyplot as plt

def show\_graph(entries):

"""Displays a detailed bar graph of monthly financial overview."""

try:

current\_age, retirement\_age = validate\_inputs(entries)

if not current\_age: return

monthly\_income = float(entries["Monthly Income:"].get())

monthly\_contribution = float(entries["Monthly Contribution:"].get())

monthly\_expenses = monthly\_income - monthly\_contribution

categories = ["Monthly Expenses", "Monthly Contributions"]

values = [monthly\_expenses, monthly\_contribution]

colors = ["#ff9999", "#66b3ff"]

plt.figure(figsize=(8, 6))

bars = plt.bar(categories, values, color=colors, edgecolor="black")

for bar in bars:

plt.text(

bar.get\_x() + bar.get\_width() / 2,

bar.get\_height(),

f"₹{bar.get\_height():,.2f}",

ha="center",

va="bottom",

fontsize=10

)

plt.title("Monthly Financial Overview", fontsize=14, fontweight="bold")

plt.ylabel("Amount (₹)", fontsize=12)

plt.tight\_layout()

plt.show()

except Exception as e:

messagebox.showerror("Error", str(e))

def show\_retirement\_projection(entries):

"""Displays a line graph projecting retirement corpus over time."""

try:

current\_age, retirement\_age = validate\_inputs(entries)

if not current\_age: return

monthly\_contribution = float(entries["Monthly Contribution:"].get())

annual\_return = 0.07 # Example: Fixed Deposits

years = list(range(current\_age, retirement\_age + 1))

total\_corpus = []

for age in years:

total\_years = age - current\_age

corpus = monthly\_contribution \* (((1 + annual\_return / 12) \*\* (total\_years \* 12) - 1) / (annual\_return / 12))

total\_corpus.append(corpus)\

plt.figure(figsize=(10, 6))

plt.plot(years, total\_corpus, marker="o", color="#66b3ff", label="Retirement Corpus")

plt.title("Retirement Corpus Projection", fontsize=14, fontweight="bold")

plt.xlabel("Age", fontsize=12)

plt.ylabel("Corpus (₹)", fontsize=12)

plt.grid(True, linestyle="--", alpha=0.7)

plt.legend(fontsize=10)

plt.tight\_layout()

plt.show()

except Exception as e:

messagebox.showerror("Error", str(e))

def show\_additional\_contribution\_impact(entries):

"""Displays a bar graph to show the impact of increased monthly contributions."""

try:

current\_age, retirement\_age = validate\_inputs(entries)

if not current\_age: return

monthly\_contribution = float(entries["Monthly Contribution:"].get())

annual\_return = 0.07

increments = [0, 10, 20, 30, 40] # Percentage increments

future\_values = []

labels = []

for percent in increments:

additional\_contribution = monthly\_contribution \* (1 + percent / 100)

total\_years = retirement\_age - current\_age

future\_value = additional\_contribution \* (((1 + annual\_return / 12) \*\* (total\_years \* 12) - 1) / (annual\_return / 12))

future\_values.append(future\_value)

labels.append(f"₹{additional\_contribution:,.2f}")

plt.figure(figsize=(10, 6))

bars = plt.bar(labels, future\_values, color=["#66b3ff", "#99ff99", "#ffcc99", "#ff9999", "#c2c2f0"], edgecolor="black")

for bar in bars:

plt.text(

bar.get\_x() + bar.get\_width() / 2,

bar.get\_height(),

f"₹{bar.get\_height():,.2f}",

ha="center",

va="bottom",

fontsize=10

)

plt.title("Impact of Increased Contributions on Retirement Corpus", fontsize=14, fontweight="bold")

plt.xlabel("Monthly Contribution (₹)", fontsize=12)

plt.ylabel("Retirement Corpus (₹)", fontsize=12)

plt.tight\_layout()

plt.show()

except Exception as e:

messagebox.showerror("Error", str(e))

### ****4. File Management Module****

Handles saving and loading data.

def save\_to\_file(entries):

try:

file\_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "\*.txt")])

if not file\_path: return

with open(file\_path, "w") as file:

for label, entry in entries.items():

file.write(f"{label} {entry.get()}\n")

messagebox.showinfo("Success", "Data saved successfully!")

except Exception as e:

messagebox.showerror("Error", str(e))

def open\_file(entries):

try:

file\_path = filedialog.askopenfilename(filetypes=[("Text files", "\*.txt")])

if not file\_path: return

with open(file\_path, "r") as file:

data = file.readlines()

for line, (label, entry) in zip(data, entries.items()):

entry.delete(0, END)

entry.insert(0, line.split(" ", 1)[1].strip())

messagebox.showinfo("Success", "Data loaded successfully!")

except Exception as e:

messagebox.showerror("Error", str(e))

### ****5. Help and Documentation Module****

Provides usage instructions.

def show\_help():

help\_text = (

"1. Enter your monthly income, current age, and retirement age.\n"

"2. Select an investment type and enter monthly savings.\n"

"3. Click 'Calculate' for financial results.\n"

"4. Use 'Show Graph' to visualize your financial data.\n"

"5. Save or load data using the 'File' menu."

)

messagebox.showinfo("How to Use", help\_text)

**FOLDER STRUCTURE :**

Financial-Overview-Tool/

├── assets/

│ └── bg\_image.jpg

├── data/

├── src/

│ ├── main.py

│ ├── calculations.py

│ ├── gui.py

│ └── utils.py

├── requirements.txt

└── README.md

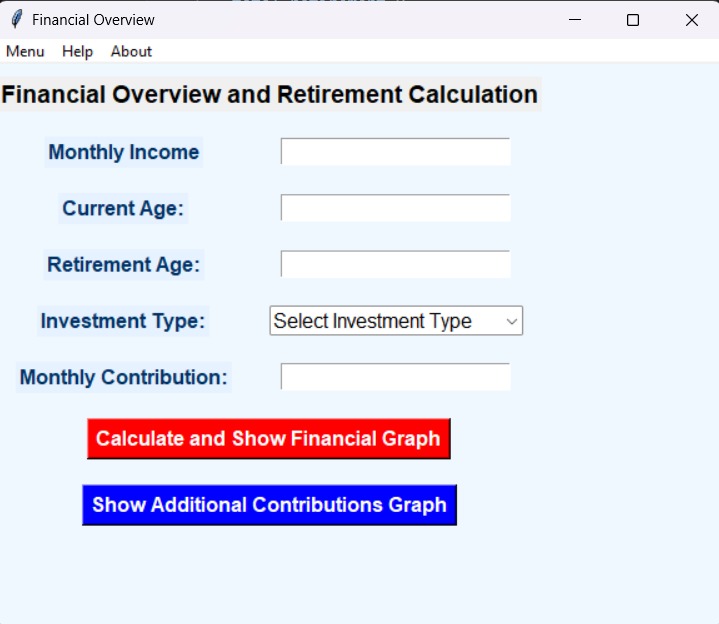
**GITHUB LINK:**

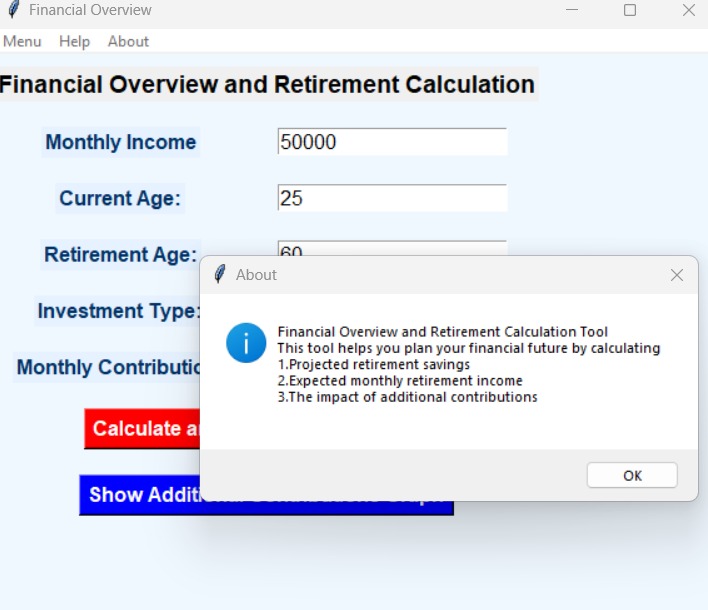
[**https://github.com/Karthik-06-28/mini-project.git**](https://github.com/Karthik-06-28/mini-project.git)

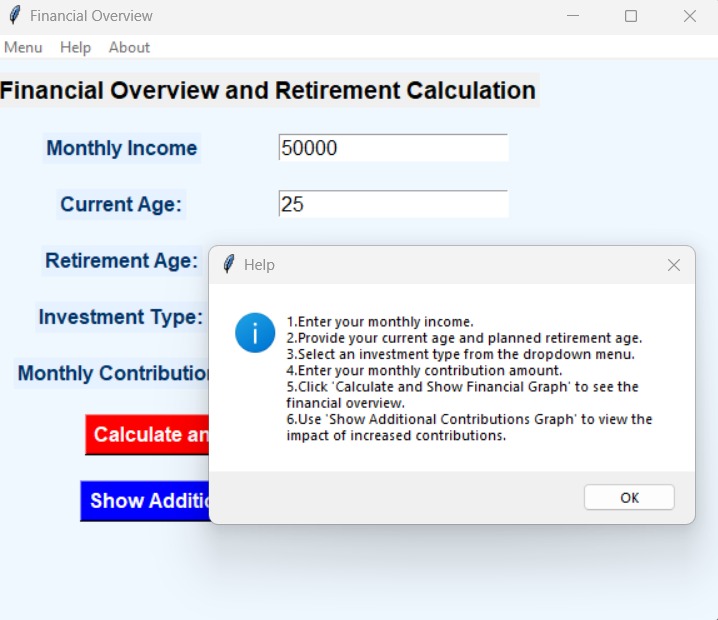
**RESULTS**

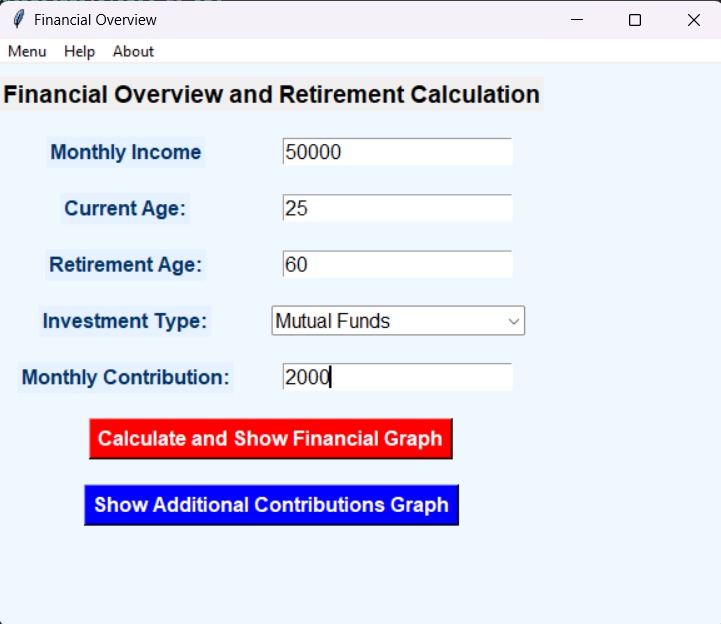
**TC-1**

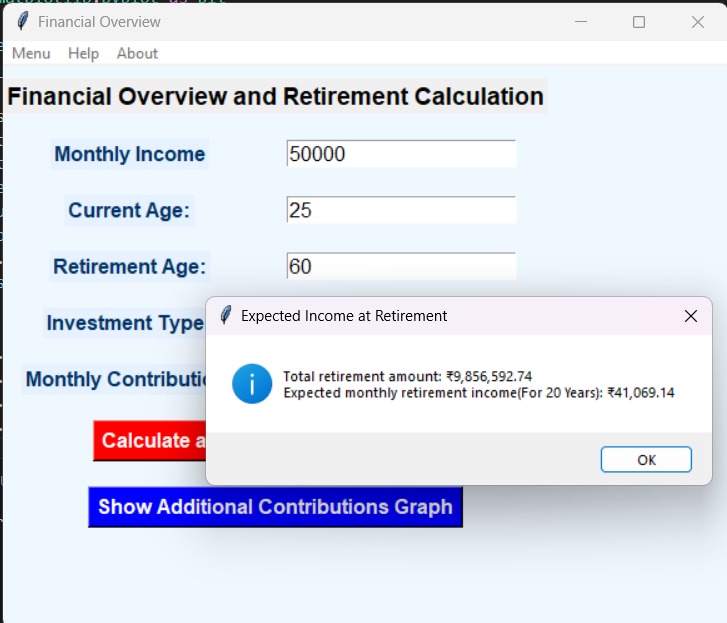
The system correctly calculates the total retirement amount and monthly retirement income, then shows the corresponding financial overview graph

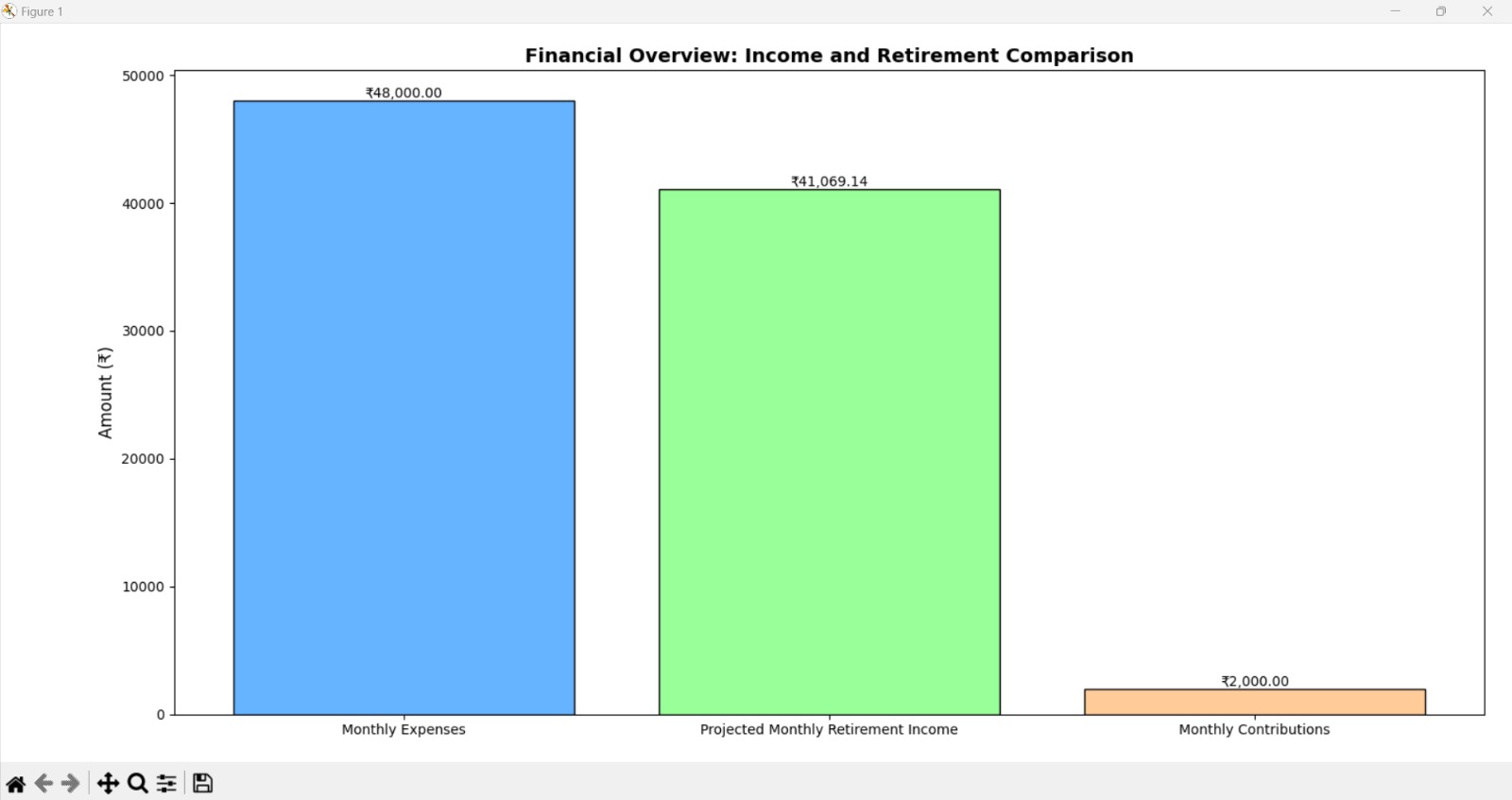


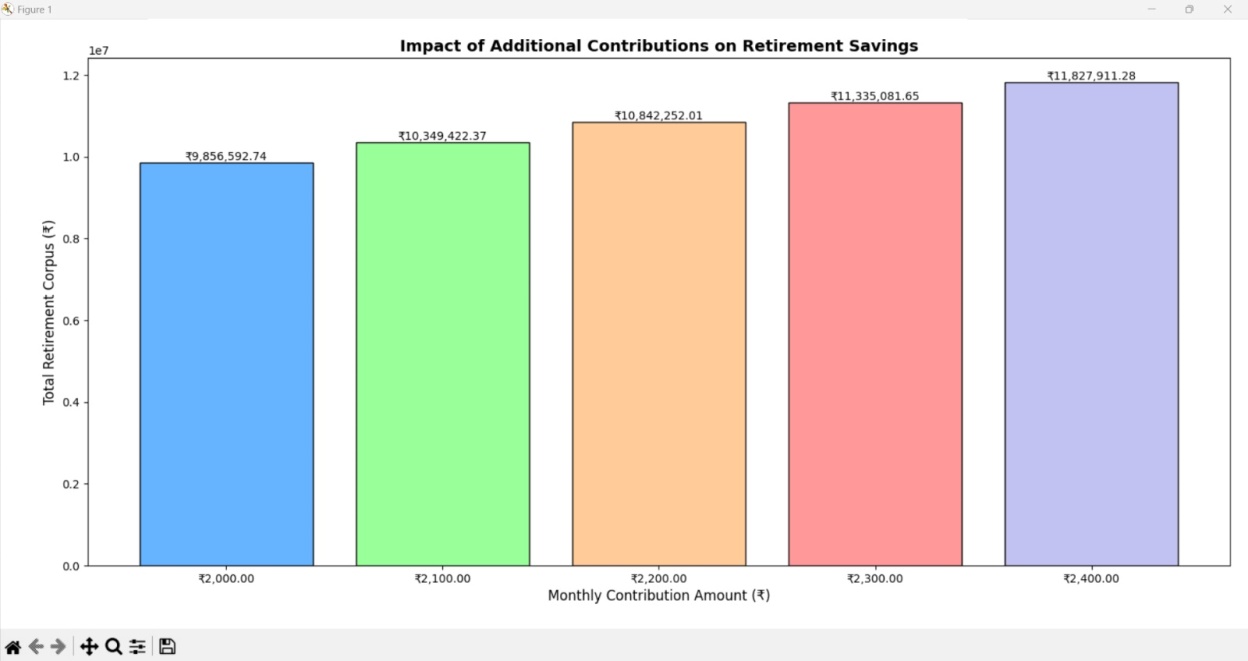


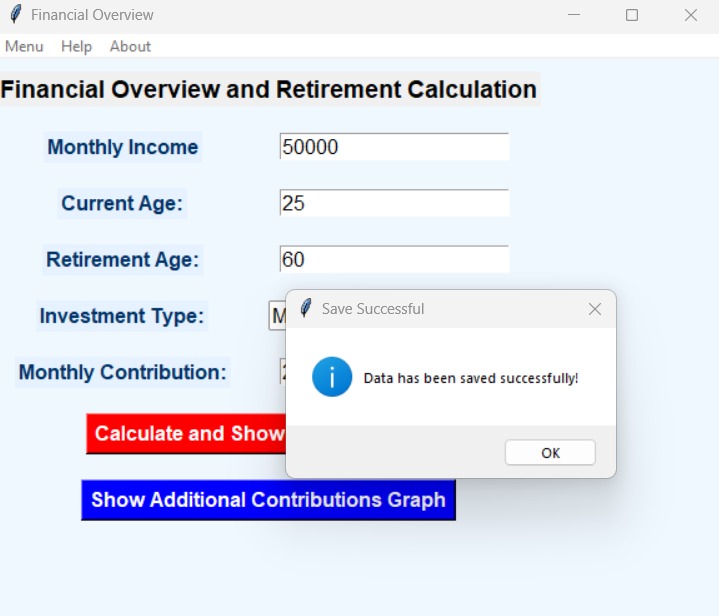






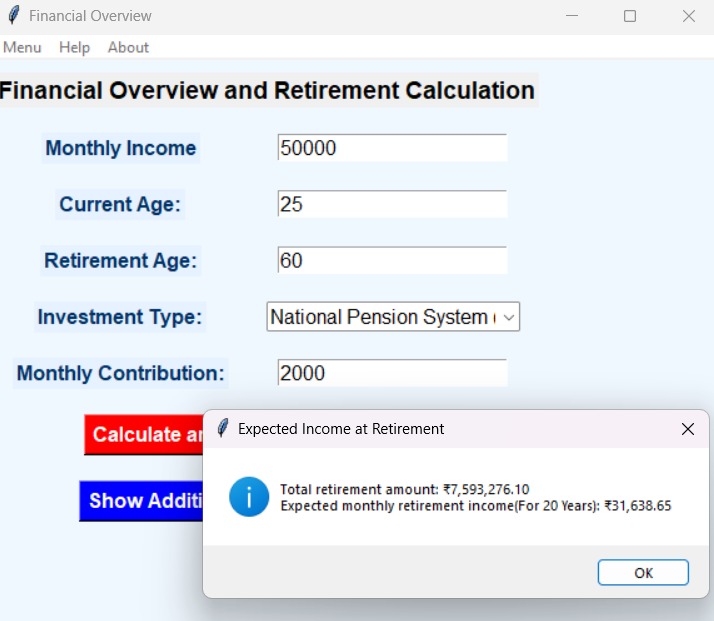


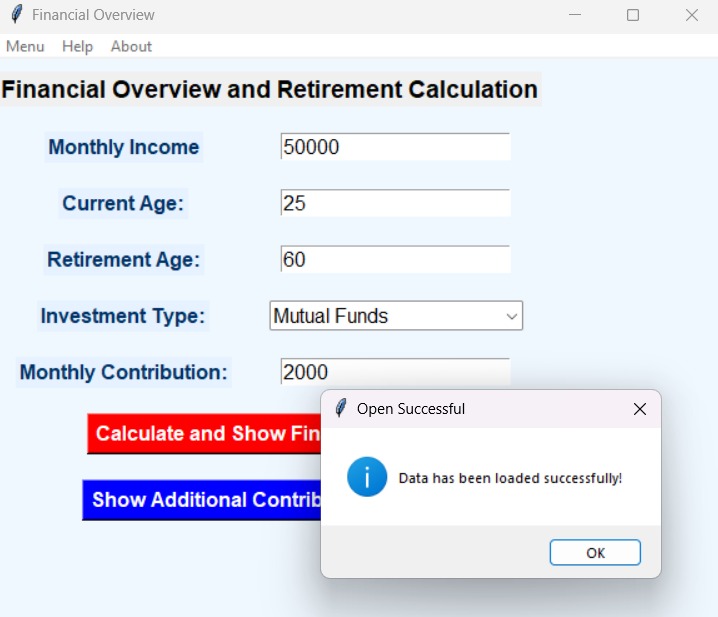




**TC-2**

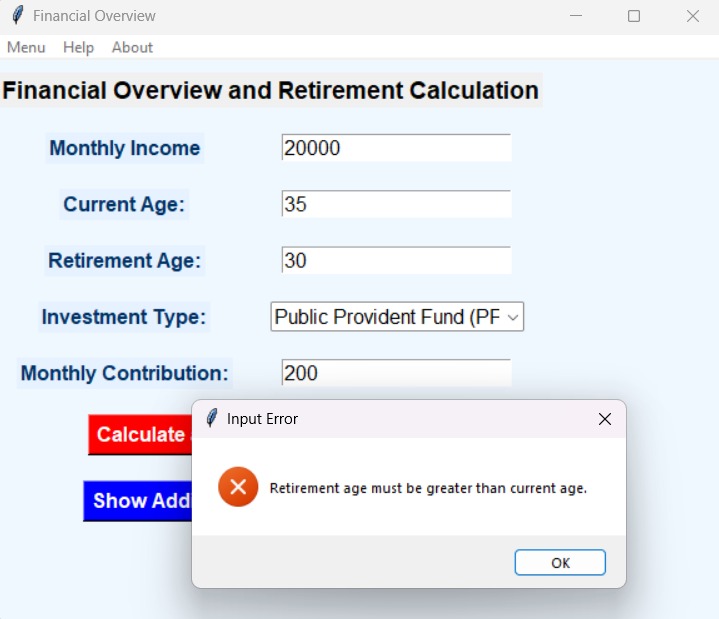
The system should load the financial data from the file and display it correctly in the input fields.





**TC-3**

The system should display an error message indicating that the input for current age or retirement age is invalid.



**Additional Knowledge Gained**

1. **Retirement Planning Concepts:** The implementation of financial calculations such as estimating the future value of monthly contributions, projecting retirement savings, and determining the impact of different investment types (e.g., Mutual Funds, Gold, Fixed Deposits) enhances understanding of long-term financial planning.

2. **Graphical Representation:** The use of matplotlib to visualize financial data such as future savings, monthly retirement income, and the effect of increasing contributions is an effective way to provide users with immediate and clear insights into their financial health.

3. **Tkinter GUI Design:** Working with Tkinter for building a user-friendly GUI interface has reinforced how easy it is to create functional, visually appealing applications. The use of widgets like Entry, Label, and Button for data input and interaction is simple and efficient for basic financial tools.

4. **File I/O in Python:** The ability to save and open financial data using text files (filedialog for saving and loading data) was useful in improving data management and allows users to retain their settings, which can be especially important for long-term planning tools.

5. **Investment Strategies:** Understanding how various investment types affect retirement savings and how to model these effects using formulas for future value and monthly withdrawals is crucial for users who want to maximize their savings potential

**Conclusion and Future Work**

**Conclusion**

The project provides a simple yet effective tool for retirement planning, enabling users to visualize their financial future based on different savings and investment strategies. It highlights the importance of disciplined saving and informed investment choices, making financial planning accessible to everyone.

**Future Work**

1. **Enhanced Investment Option Analysis:** Future improvements could include adding more detailed investment types (e.g., stocks, bonds, REITs) and providing more granular information such as historical returns and risk factors for each investment type.

2. **Advanced User Interface:** The GUI can be enhanced by incorporating more interactive elements, such as sliders to input values dynamically, or graphs that update in real-time based on user input. A more modern design could make the tool even more accessible and visually appealing.

3. **Real-Time Data Integration:** Incorporating real-time financial data (such as stock or bond performance) could make the tool even more robust. APIs that provide live investment returns could be integrated to adjust projections based on current market trends.

4. **Security Features:** As financial data is sensitive, implementing secure login features and encrypting saved files or data would be an important addition for privacy and security.

5. **Mobile App Development:** The tool can be developed into a mobile application, making it more accessible to users on-the-go. This could involve using platforms such as Kivy or React Native for cross-platform development.

**REFERENCES**

1. **Retirement Planning Basics**

* <https://www.investopedia.com/terms/r/retirement-planning.asp>
* <https://www.nerdwallet.com/article/investing/retirement-savings-plan>

2. **Python Tkinter Documentation**

* <https://docs.python.org/3/library/tkinter.html>
* <https://realpython.com/python-gui-tkinter/>

3. **Matplotlib for Data Visualization**

* <https://matplotlib.org/>
* <https://www.geeksforgeeks.org/python-matplotlib-an-overview/>

4. **Future Value and Investment Calculations**

* <https://www.bankrate.com/calculators/savings/savings-calculator.aspx>
* <https://www.wallstreetmojo.com/future-value-formula/>