# Record Keeper Project

Table of Contents

[Record Keeper Project 1](#_Toc120385601)

[Introduction 1](#_Toc120385602)

[GUI Details 1](#_Toc120385603)

[Important Classes 1](#_Toc120385604)

[Program Execution 2](#_Toc120385605)

[Comments 2](#_Toc120385606)

[Action and Form Validation 2](#_Toc120385607)

[Testing Strategy 2](#_Toc120385608)

[Verification and Validation Testing 3](#_Toc120385609)

[White box testing 4](#_Toc120385610)

[References: 8](#_Toc120385611)

# Introduction

This assignment adds a graphical user interface to the work of part 1, which only contained the implementation of a command line interface.

In part 2, to provide GUI components, the Tkinter library was used which is part of the python standard library (Grayson,2000).

A new GUI class was created. The basic idea was to keep the graphical user interface separate from the business logic to make it easier to make changes, scale and to manage the code with ease. The logic behind insertion, searching, deletion and sorting has already been discussed in part 1 (Nowak,2008).

# GUI Details

Tkinter library is part of the standard python libraries and is included in all the new python distributions after python 3.7. It is the simplest GUI library and contains all the basic GUI widgets to create the application (Lundh,1999).

Firstly, a tkinter root is created to create the application, and then a frame or window is created on top of it by passing the root application. Various GUI widgets were used as given below:

**List:** A list widget is part of the tkinter library, and it was used to display the records list in the program.

**Button:** A button widget was used to show various buttons including the main window buttons which allow users to add, delete, and view records

**Table:** A table view widget is not part of the Tkinter, however, a table view was required to show the records with columns. This was emulated using the grid layout, and labels were used as headers, additionally, two buttons were provided which could soar the grid records in ascending and descending order. A table component was written from scratch.

**Entry:** A tkinter entry widget is like a line input component and can take input from the user. It has been used to take record details such as name, phone, and address and to search for records using an input field.

**Frame/Window:** Both frame and window components are containers for other widgets. They can hold other widgets and widgets can be arranged using the layouts.

To fully understand the GUI part of the project, an understanding of Tkinter layouts is important. Two layouts were used in the program

**Pack layout:** This layout is the simpler one and can display objects horizontally or vertically. It was used to display forms such as “delete a record”, and “search a record”.

**Grid layout:** This arranges the available space in a grid of rows and columns and can be used to display components arranged in a matrix. The grid layout was used in the “add a record” and “view records”.

# 

# Important Classes

**RecordKeeper:** This class contains the core record keeping and management methods such as add new record, delete record, search record etc. It only contains the business logic.

**GUI:** This is the class which contain only methods related to the creation and management of user interface, it also contains methods to create widgets like list, buttons, and input boxes. It has functions to show add, delete, view, search forms which use the GUI widgets. This class internally uses the RecordKeeper to manage and store records

**Main Methods:**

**showAddForm/showSearchForm/showViewForm/showDeleteForm:** These methods provide a GUI based functionality of the RecordKeeper class. For example, the showAddForm executes when the user clicks the add button on the main windows. This method creates the input and button components on the form and internally uses the recordkeeper add a method to add the record. These methods also make use of tkinter message box to confirm the action of adding new records and for asking users to confirm their option. For example, in the case of a “showDeleteForm”, messagebox is used to show the record details and confirm from the user with two options namely ok and cancel to confirm their choice of deletion. If the user presses cancel and no deletion is performed.

# Program Execution

The whole program is written in python without any external library, so we can simply execute it by first unzipping the program file and then executing the program using python 3 (Python3, 2022).

Step 1:

Firstly, unzip the file, it should be extracted into an rcgui folder.

Step 2:

cd to the rcgui folder

cd ./rcgui

Step 3:

Execute the python code using python 3

python3 gui.py

### Comments

In GUI, Main, Recordkeeper and TestRecordKeeper classes, class and method comments have been added to make it easier to understand the implementation details.

# Action and Form Validation

In all the forms, messagebox has been used to confirm the user action. For example, if the user selected a record for deletion, a confirmation box is displayed before the record is deleted. Also, in the add form, user phone number is validated for numerical input before the record is added to the list, in case it is not numeric, an error message is displayed.

# 

# Testing Strategy

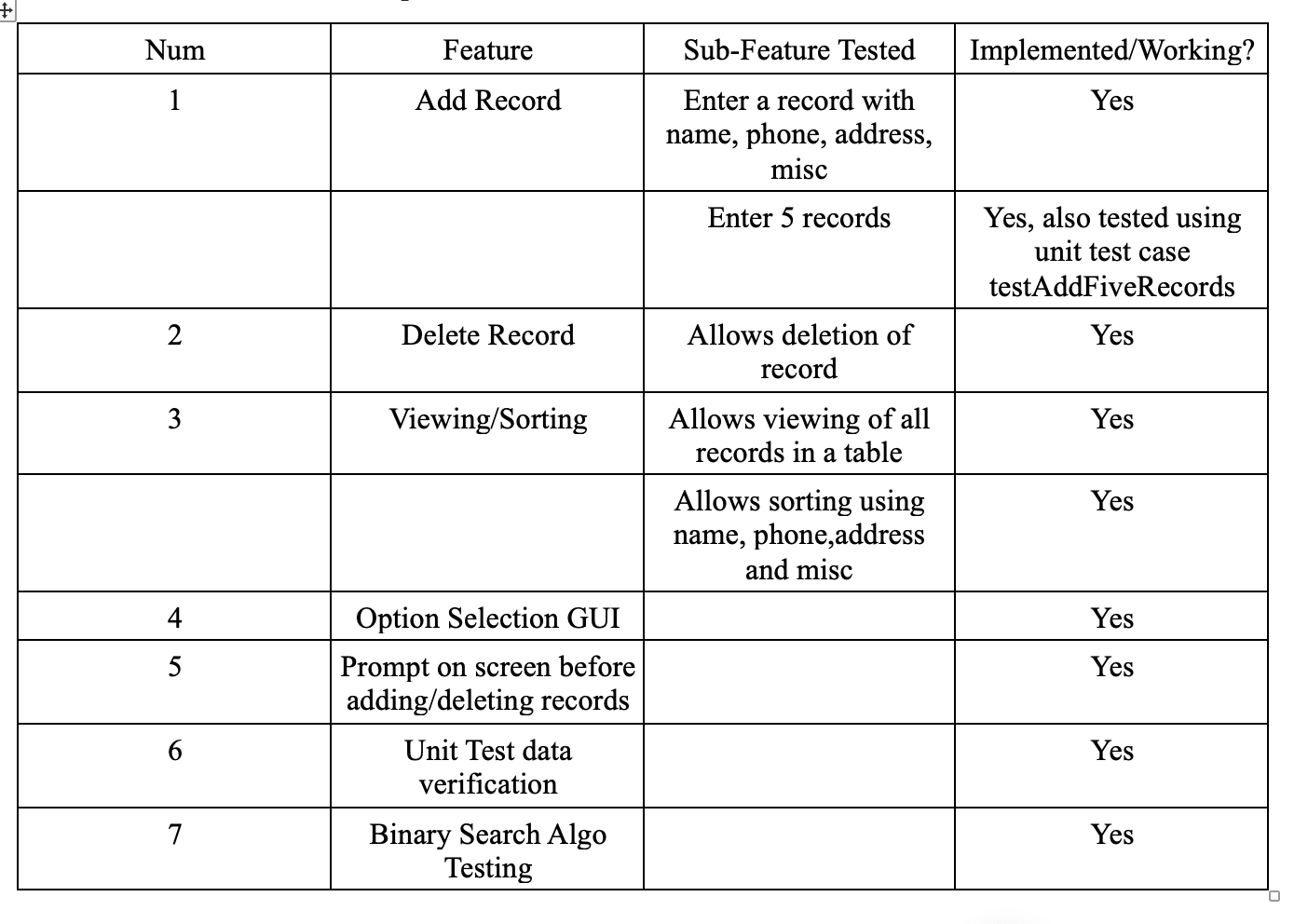
In the first part, test cases were used to test the core functionality of the record keeper project. However, part two includes GUI over the existing core functionality. So, in this part, the testing strategy focuses on testing the GUI functionality testing and execution. GUI testing is also known as user interface (UI) testing. Two types of testing were performed which are given below

1) Verification and Validation testing

2) White box testing

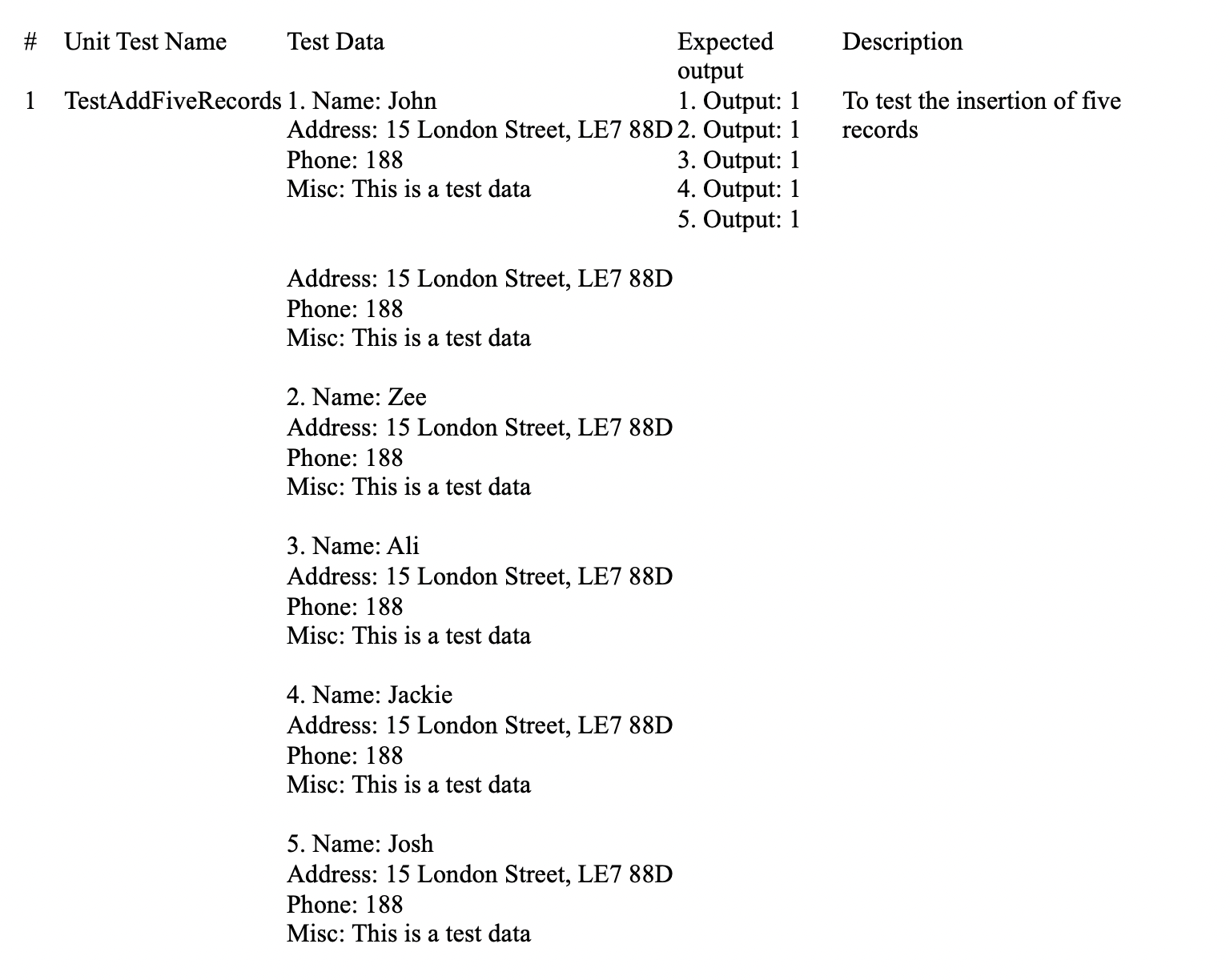
## Verification and Validation Testing

All the required features of the project were tested and implemented in this testing. In the below table, each feature is listed with its implementation status



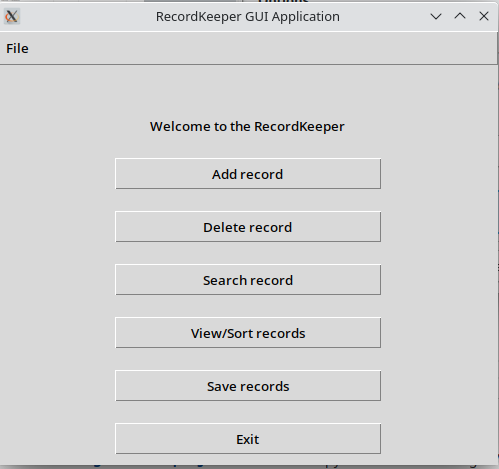
## White box testing

In this testing, individual features were tested using unit tests, most of the unit tests have been written as part of the part 1 Test plan. These tests are re-executed to make sure no accidental changes are made due to part 2 having changed the internal working of the features. All the testing has been passed.

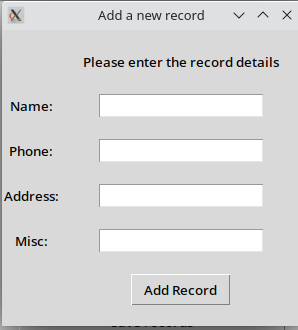
****

**Testing Screenshots:**

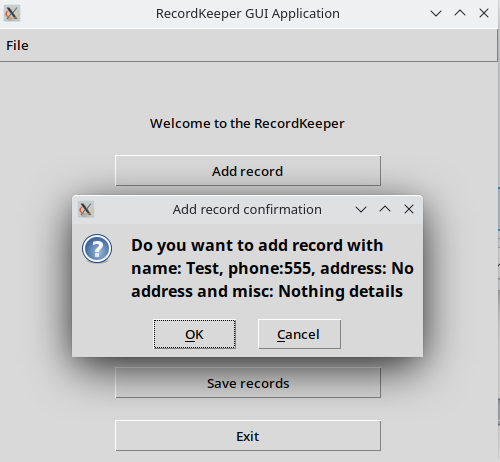
**Main GUI Application**

****

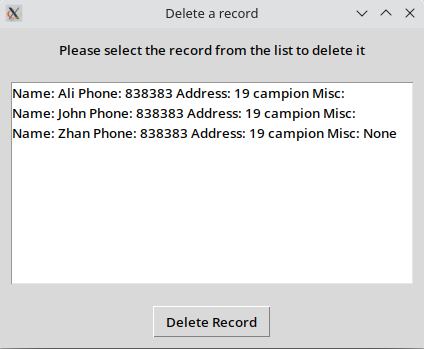
**Add a new record Frame**

****

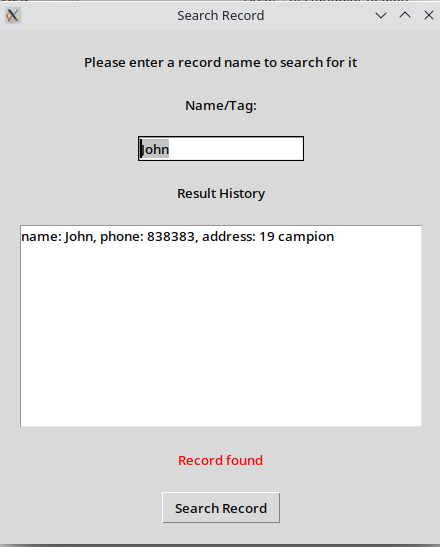
**Record Adding Action Validation**

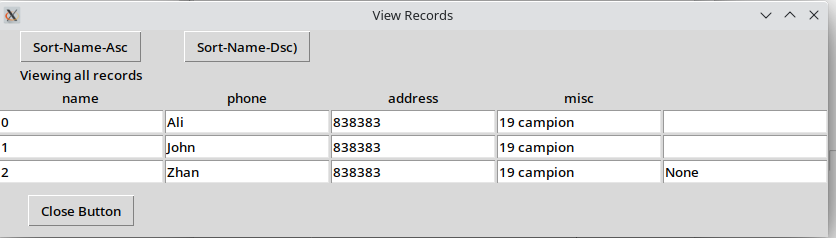
****

**Delete a record**

****

**Search for a record**

****

**View/Sort Records**

# References:

* 1. Python3 Python.org. Available at: https://www.python.org/downloads/ (Accessed: November 13, 2022).
  2. Hash table (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Hash\_table (Accessed: November 13, 2022).
  3. Python pickle – serialize and de-serialize objects (no date) Python Pickle – Serialize and De-serialize Obunjects - Python Examples. Available at: https://pythonexamples.org/python-pickle-tutorial/ (Accessed: November 13, 2022).
  4. Nowak, Robert. "Generalized binary search." 2008 46th Annual Allerton Conference on Communication, Control, and Computing. IEEE, 2008.
  5. Lundh, Fredrik. "An introduction to tkinter." URL: www. pythonware. com/library/tkinter/introduction/index. htm (1999).
  6. Grayson, John E. Python and Tkinter programming. Manning Publications Co. Greenwich, 2000.