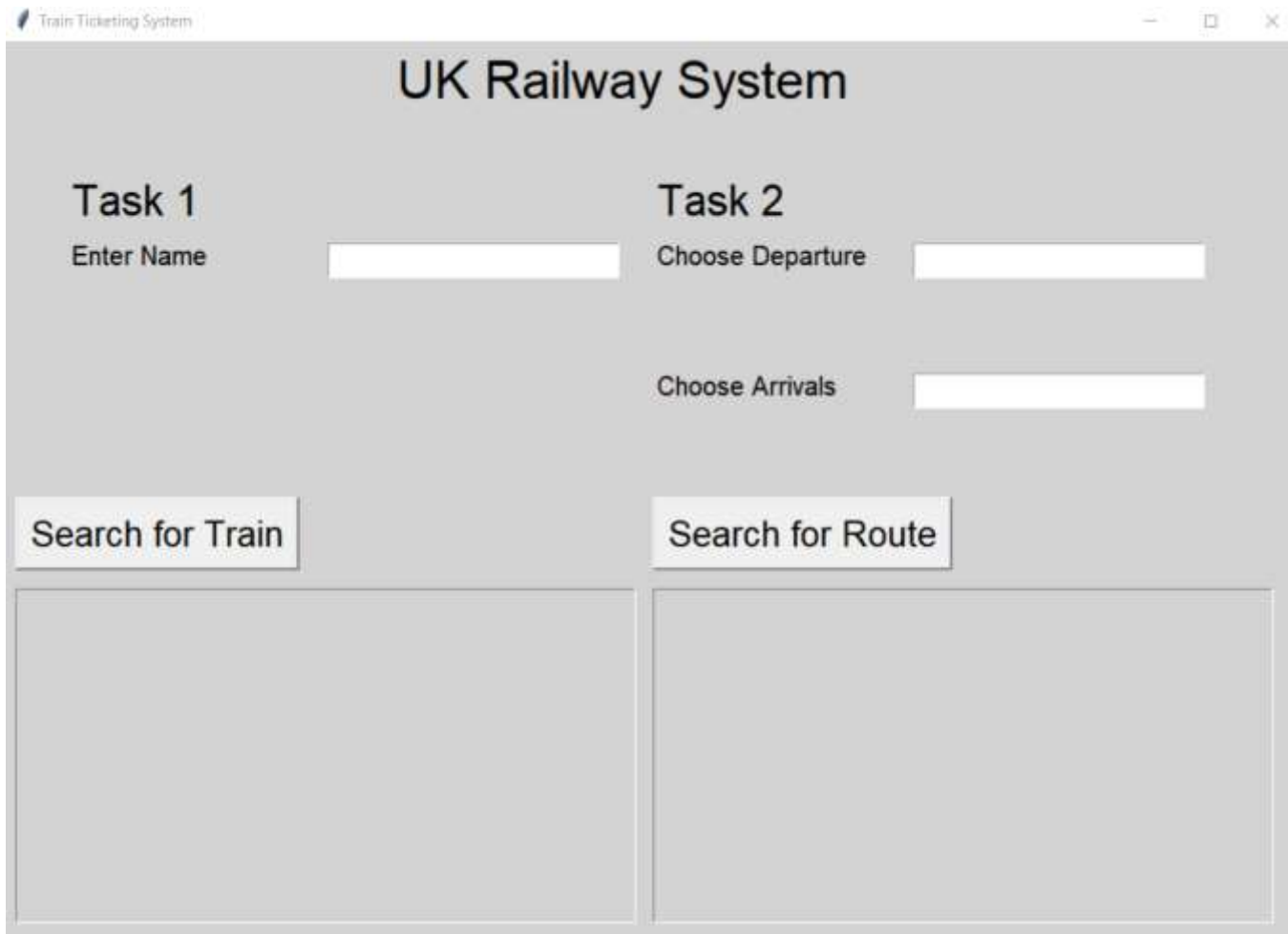


# Task 1: Design

Both Task 1 & 2 use a single unified interface which is show below:



The screenshot shows a web application window titled "Train Ticketing System". The main heading is "UK Railway System". Below this, there are two columns. The left column is labeled "Task 1" and contains the text "Enter Name" followed by a text input field. The right column is labeled "Task 2" and contains the text "Choose Departure" followed by a text input field, and "Choose Arrivals" followed by another text input field. Below the input fields, there are two buttons: "Search for Train" on the left and "Search for Route" on the right. At the bottom of each button column is a large, empty rectangular box, likely for displaying search results.

As we know that Task 1 is related to efficient search of Train Stations, a class *RailwayNetwork* is implemented for this purpose. The class uses a dictionary object *railway\_stations* to store the information about railway stations. The different functions in this class relevant to Task 1 are explained below:

## 1. load\_railway\_stations\_data

This function accepts a csv filename as parameter and loads the file contents into the dictionary object *railway\_stations*. The dictionary object is stored in the form that for each railway station, the code of the railway station becomes the key and the name of the station becomes the value. e.g.  
*railway\_stations['BLP'] = "Belper"*

## 2. search\_railway\_station

This function is responsible for performing the main functionality required in task 1. It implements a search algorithm for the loaded *railway\_stations* object in with following conditions:

1. The searched items is first searched in the codes of railway stations.
2. If not found in the codes of railway stations, it is searched in the names of the railway stations.
3. If not found in the names of the railway stations, it is searched as a part of the name of the stations.
4. Lastly, If the required item is not found anywhere in the dictionary object, we calculate the matching strings in the codes and names of the railway stations and return a list of possible matches. The match values are calculated using thefuzz library which is explained in the function below

## 3. \_\_get\_similarity\_index

This function calculated the similarity index between two strings for index-based searching and checks how many words actually have the prefix. It uses a library called *thefuzz*. From *thefuzz* library, *fuzz.partial\_token\_sort\_ratio function* is used to calculate the similarity between two strings.

## OUTPUT

The screenshot displays a web application titled "UK Railway System" with a search interface. It is divided into two main sections: "Task 1" and "Task 2".

**Task 1:** Labeled "Enter Name", it contains a text input field with the value "PTA" and a button labeled "Search for Train".

**Task 2:** Labeled "Choose Departure" and "Choose Arrivals", it contains two empty text input fields and a button labeled "Search for Route".

**Search Results:** Below the "Search for Train" button, a box displays the "Search Result: PTA, Port Talbot Parkway". The "Search for Route" button has an empty result box below it.