In [ ]:
```python
import os
import random
import math
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QLabel, QButtonGroup, QComboBox, QHBoxLayout
from PyQt5 import QtGui
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure

class BabyNameApp(QWidget):
    def __init__(self):
        super(BabyNameApp, self).__init__()
        self.initUI()
        self.rated_boys = self.load_names('boy')
        self.rated_girls = self.load_names('girl')

    def initUI(self):
        self.setWindowTitle('Baby Name Rater')
        self.layout = QVBoxLayout()

        self.topLayout = QHBoxLayout()

        self.label = QLabel('Press Start to rate baby names')
        self.label.setAlignment(Qt.AlignCenter)  # Change this line to align the text to center
        self.layout.addWidget(self.label)  # Add label to main layout directly

        self.sparkline_label = QLabel()
        self.sparkline_label.setAlignment(Qt.AlignCenter)
        self.layout.addWidget(self.sparkline_label)
        self.sparkline_label.hide()
        self.sparkline_label.setScaledContents(True)  # Allow QLabel to resize QPixmap

        self.skip_btn = QPushButton('Skip')
        self.layout.addWidget(self.skip_btn)
        self.skip_btn.clicked.connect(self.skip_name)
        self.skip_btn.hide()

        self.gender_selector = QComboBox(self)
        self.gender_selector.addItem("Boys")
        self.gender_selector.addItem("Girls")
        self.layout.addWidget(self.gender_selector)

        self.button_group = QButtonGroup(self)
        for i in range(1, 11):
            btn = QPushButton(str(i))
            btn.setCheckable(True)
            self.button_group.addButton(btn, i)
            self.layout.addWidget(btn)
            btn.clicked.connect(self.rate_name)
            btn.hide()

        self.start_btn = QPushButton('Start')
        self.layout.addWidget(self.start_btn)
        self.start_btn.clicked.connect(self.start_rating)

        self.setLayout(self.layout)
        self.show()

    def start_rating(self):
        gender_option = self.gender_selector.currentText()
        self.name, self.sex, self.year, self.count, self.rank, self.total_names, self.most_popular_year, self.most_popular_

        self.skip_btn.show()
        self.gender_selector.hide()

        most_popular_year_index = self.most_popular_year - min(range(1880, 2022))  # assuming 'years' is still defined as r
        sparkline = self.generate_sparkline(self.yearly_counts, self.sex, most_popular_year_index)

        self.label.setText(
            f"<h1>{self.name}</h1>"
            f"<i><b>Most Popular Year</b></i><br>"
            f"<br>"
            f"<b>Year:</b> {self.most_popular_year} | <b>Count:</b> {self.most_popular_count} | <b>Rank:</b> {self.most_pop
            f"<br>"
            f"<b>Total Count:</b> {self.total_count}<br>"
        )

        for button in self.button_group.buttons():
            button.setChecked(False)
            button.setStyleSheet("")
            button.show()

        if self.sex == 'M':
            self.setStyleSheet("background-color: lightblue;")
        else:
            self.setStyleSheet("background-color: pink;")

        self.start_btn.setText('Exit')
        self.start_btn.disconnect()
        self.start_btn.clicked.connect(self.close)

    def generate_sparkline(self, data, gender, most_popular_year_index):
        color = 'black'
        highlight_color = 'red'
        fig = Figure(figsize=(.7, 0.3), dpi=400)
        fig.patch.set_facecolor("none")  # Make background transparent
        canvas = FigureCanvas(fig)
        ax = fig.add_subplot(111)
        ax.plot(data, color=color, linewidth=0.2)  # Use color and set linewidth to 0.5

        # Add a scatter point at the most popular year index
        ax.scatter([most_popular_year_index], [data[most_popular_year_index]], color=highlight_color, s=.125)

        ax.axis('off')
        fig.patch.set_visible(False)
        ax.axis('off')

        canvas.draw()
        width, height = canvas.get_width_height()

        img = QtGui.QImage(canvas.buffer_rgba(), width, height, QtGui.QImage.Format_RGBA8888)
        pix = QtGui.QPixmap.fromImage(img)

        self.sparkline_label.setPixmap(pix)
        self.sparkline_label.show()

    def rate_name(self):
        rating = self.button_group.checkedId()
        if rating != -1:
            self.rated_names = self.rated_boys if self.sex == 'M' else self.rated_girls
            self.rated_names.add(self.name)
            self.append_name_to_file(self.name, rating, self.sex)
            self.start_rating()

    def skip_name(self):
        self.start_rating()

    def load_names(self, gender):
        filename = f"data/baby_name_{gender}.txt"
        rated_names = set()
        if os.path.exists(filename):
            with open(filename, 'r') as f:
                for line in f:
                    name, rating, _ = line.strip().rsplit(',', 2)
                    rated_names.add(name)
        return rated_names

    def append_name_to_file(self, name, rating, sex):
        gender_mapping = {'M': 'boy', 'F': 'girl'}
        filename = f"data/baby_name_{gender_mapping[sex]}.txt"
        with open(filename, 'a') as f:
            f.write(f"{name},{rating},{sex}\n")

    def get_random_name(self, gender_option):
        years = range(1880, 2022)
        most_popular_year = None
        most_popular_count = 0
        most_popular_rank = 0
        total_count = 0
        yearly_counts = []

        bias_weights = [math.log(year - 1879) for year in years]
        while True:
            random_year = random.choices(years, bias_weights)[0]
            random_filename = f"data/Extracted/yob{random_year}.txt"
            names, sexes, counts = [], [], []
            total_names_of_gender = 0

            with open(random_filename, 'r') as f:
                for idx, line in enumerate(f):
                    name, sex, count = line.strip().split(',')
                    if gender_option == sex:
                        names.append(name)
                        sexes.append(sex)
                        counts.append(int(count))
                        total_names_of_gender += 1

            if counts:
                log_counts = [math.log(count + 1) for count in counts]
                chosen_index = random.choices(range(len(names)), log_counts)[0]
                chosen_name, chosen_sex, chosen_count = names[chosen_index], sexes[chosen_index], counts[chosen_index]
                chosen_rank = chosen_index + 1
                rated_names_set = self.rated_boys if chosen_sex == 'M' else self.rated_girls

                if chosen_name not in rated_names_set:
                    for year in years:
                        filename = f"data/Extracted/yob{year}.txt"
                        with open(filename, 'r') as f:
                            for line in f:
                                name, sex, count = line.strip().split(',')
                                if name == chosen_name and sex == chosen_sex:
                                    count = int(count)
                                    yearly_counts.append(count)
                                    total_count += count
                                    if count > most_popular_count:
                                        most_popular_year = year
                                        most_popular_count = count
                                        most_popular_rank = chosen_index + 1
                                    break
                            else:
                                yearly_counts.append(0)
                    return chosen_name, chosen_sex, random_year, chosen_count, chosen_rank, total_names_of_gender, most_pop
```

In [ ]:

```python
if __name__ == '__main__':
    app = QApplication([])
    ex = BabyNameApp()
    app.exec_()
```