# COMP1752 Object-Oriented Programming: Jukebox Simulation Report

## Title Page

* Course Title: COMP1752 Object-Oriented Programming
* Assignment: Jukebox Simulation
* Duong Thanh Dat ID
* Submission Date

## Table of Contents

[1. Introduction 1](#_Toc28088)

[2. Design and Development 1](#_Toc22801)

[3. Testing and Faults 7](#_Toc28686)

[4. Conclusions, Further Development, and Reflection 7](#_Toc32430)

[5. Innovations 7](#_Toc31732)

[Appendices 8](#_Toc23774)

[Appendix A: Commented Code (Stage 1) 8](#_Toc29537)

[Appendix B: Test Table and Results 8](#_Toc31171)

[Appendix C: Full Source Code 10](#_Toc26573)

## Introduction

This report include analysis of coursework functionality and development stages of this project. The jukebox simulation provide view tracks functionality which can find and print track information on screen. This project will improve jukebox simulation with few more features such as create tracks playlist and update tracks information. This report follow the chronological order of the development from design, coding, testing, conclusion and innovation.

## Design and Development

* 1. **Design:**

The track player and view track have the exact same layout as original program but wheat colour background, black text colour, dark orange button background and white foreground as well as the other windows.

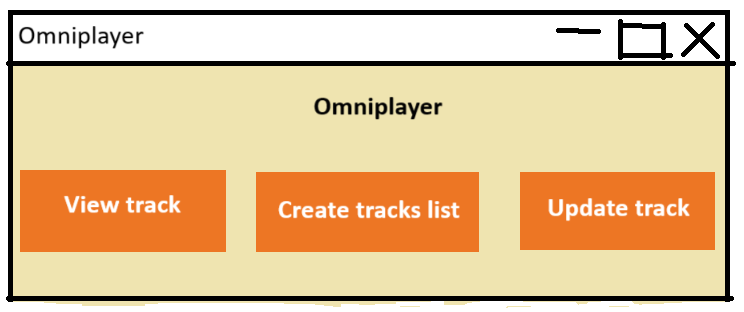


Image 1

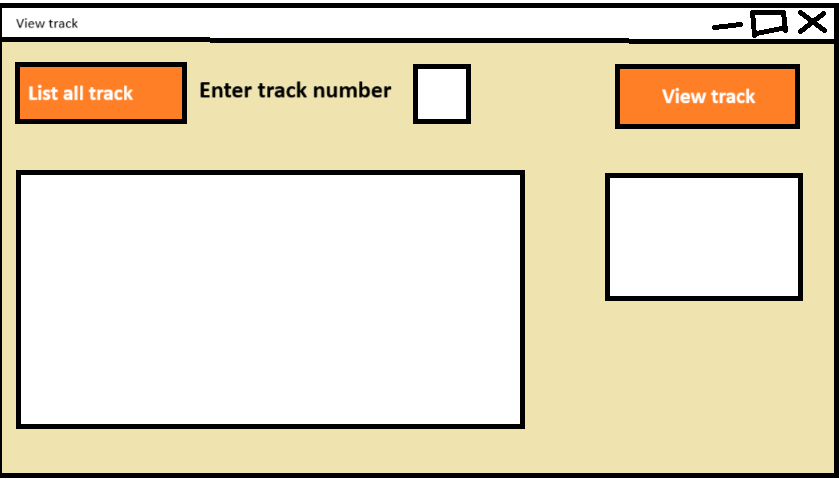


Image 2

The create tracks list layout, which have an input entry to get the track number, one button on the same row as the entry to add the track(if exist) to playlist, another button below to reset list or delete all track from list which have red background, a big scroll text to display all track’s basic information such as name, artist and play count, at the same row, there is a play button to play all track, in this case it only increase play count of track in playlist.

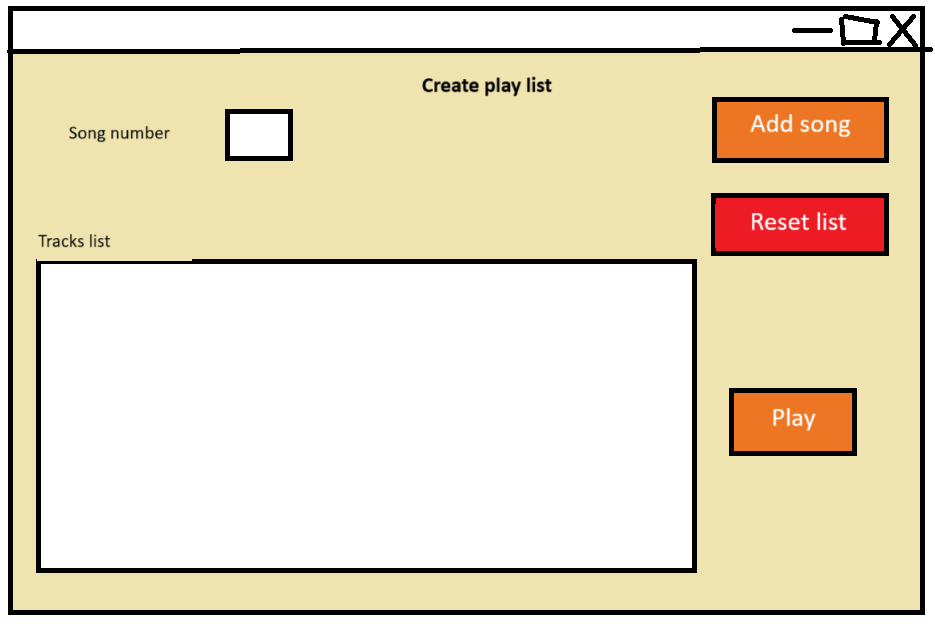


Image 3

Update tracks window, this window have two entries of track number and new rating, update button on the same row to update track rating, change window size display new track information on text box.

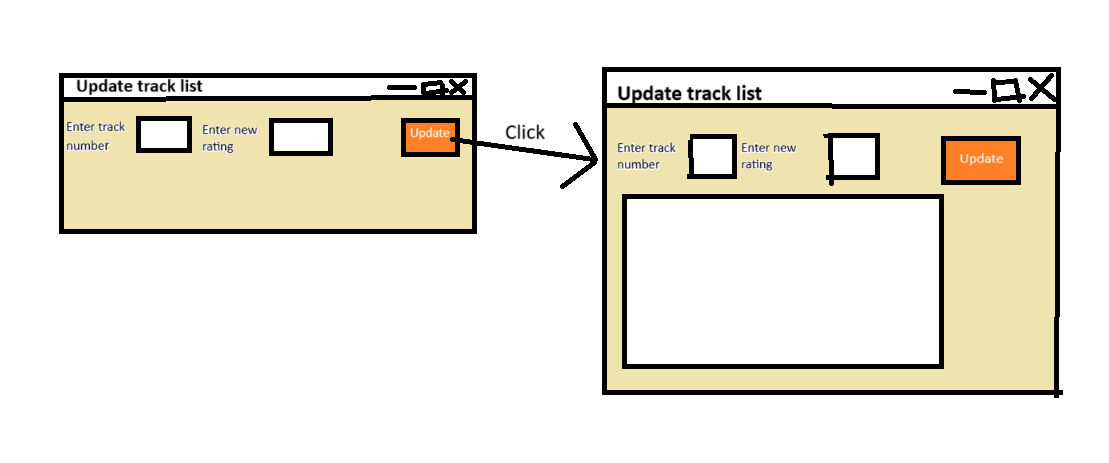


Image 4

* 1. **Development**
* **Planning**

This project based on the original source code and added basic functionality of a jukebox simulation. Including, create tracks playlist and update tracks rating. First step of the process is to make GUIs closest to designs possible. The next step is develop the working version based on the GUIs from previous step and prepare for testing.

* **Development**

As the plan above, the first step is develop the GUIs based on the designs. This step is not difficult since track player and view tracks have no change in layout and only have to configure the colour(Image 5,Image 6). Meanwhile the create track list and update track GUIs was built upon the other class work which use inheritance class of tkinter.TK for the GUIs instead of nomal class like view track.(Image 7,Image 8-Update track GUI, Image 9-Create tracks list GUI)

The next step is develop the functions for update track and create tracks list. Create track list have three function for three button. First function is add song to add a track to the play list. When you type a number in the song number Entry and press Add song button, if the track with that library number exist, the track will be added in the tracks list and track list will be display on scroll text box(Image 10). If track doesn’t exist, the status label will be configure to Song doesn’t exist(Image 11). Second function is reset list, which delete all track from playlist and write list reset on screen (Image 12). Third function is play track, this function will increase play count of tracks in play list(Image 13).

The only function in update track is update track rating, if track number and rating are valid, the track with that key will change the rating to the rating in the rating entry, the window change size, text box appear and track information display on text box(Image 14). If any of the inputs is invalid, the text box appear and write invalid inputs depend on what is invalid.

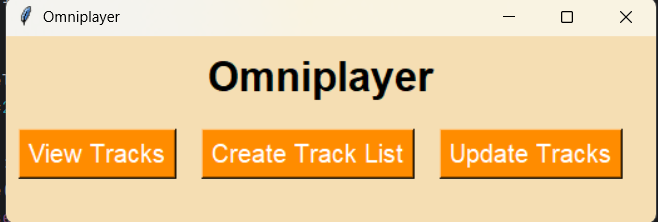


Image 5

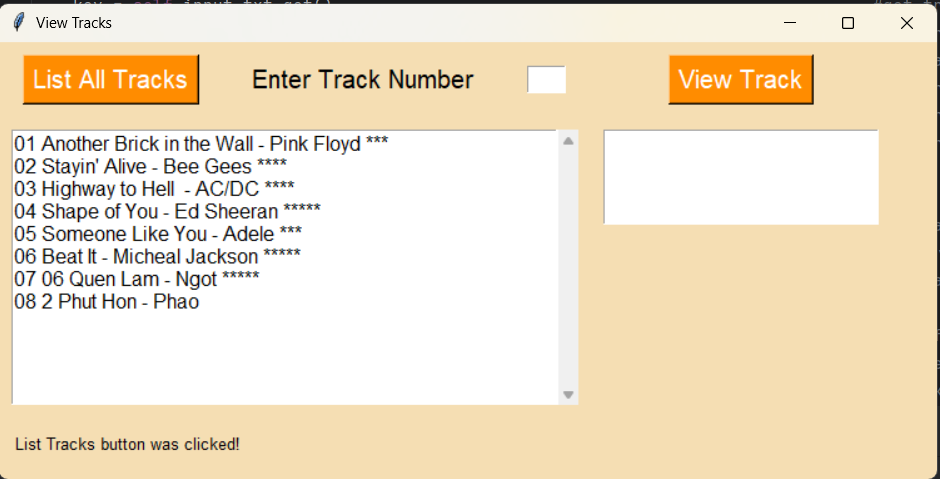


Image 6

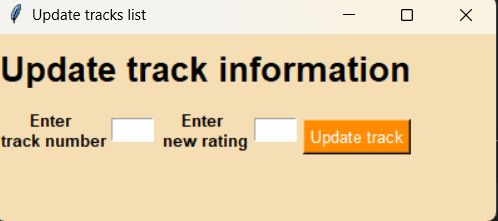


Image 7

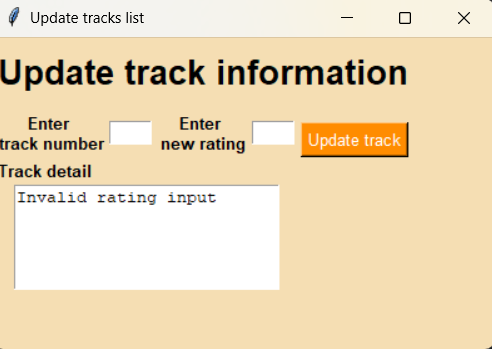


Image 8

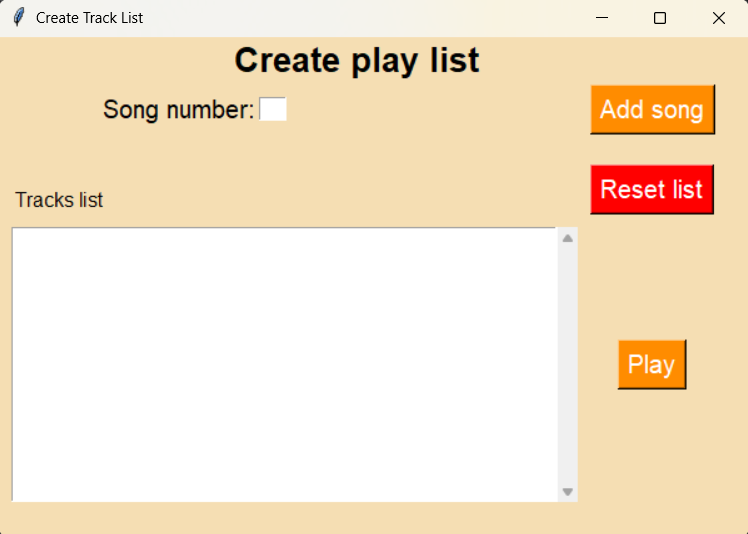


Image 9

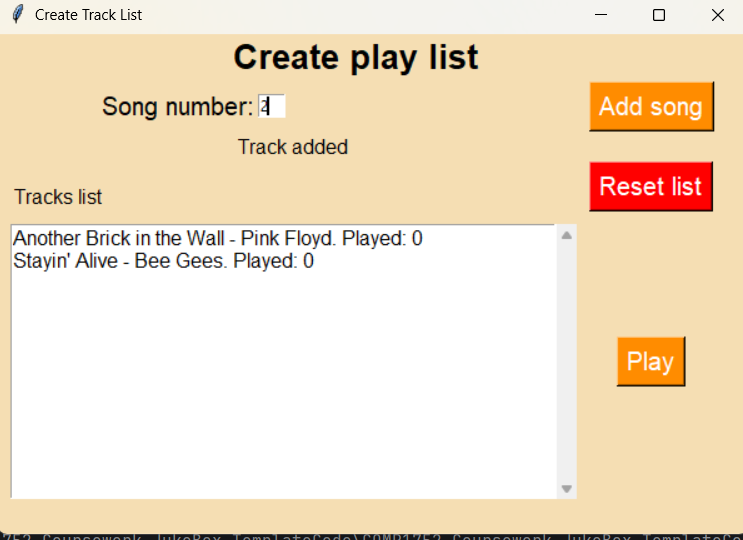
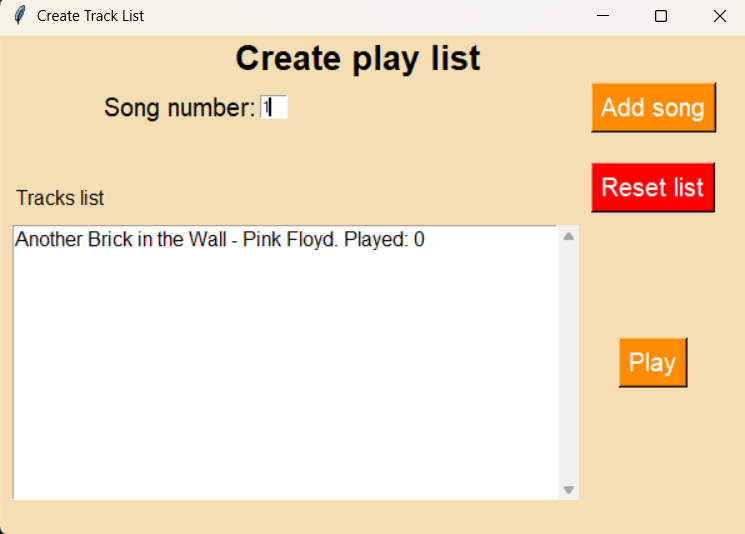


Image 10

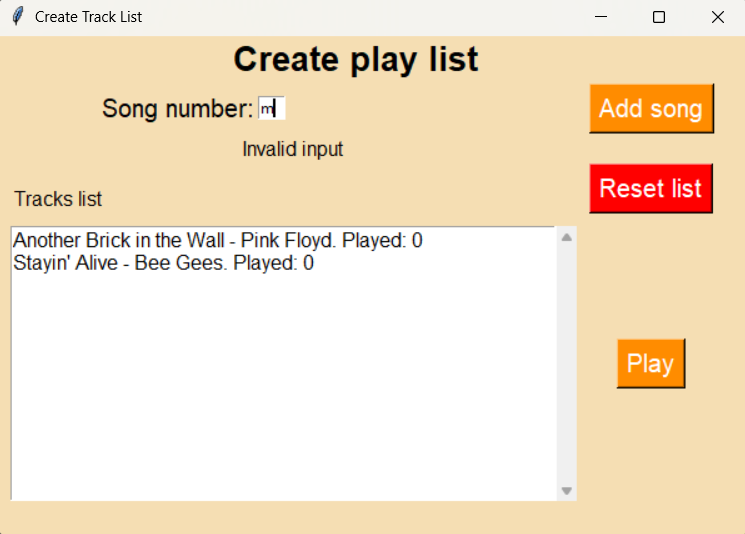


Image 11

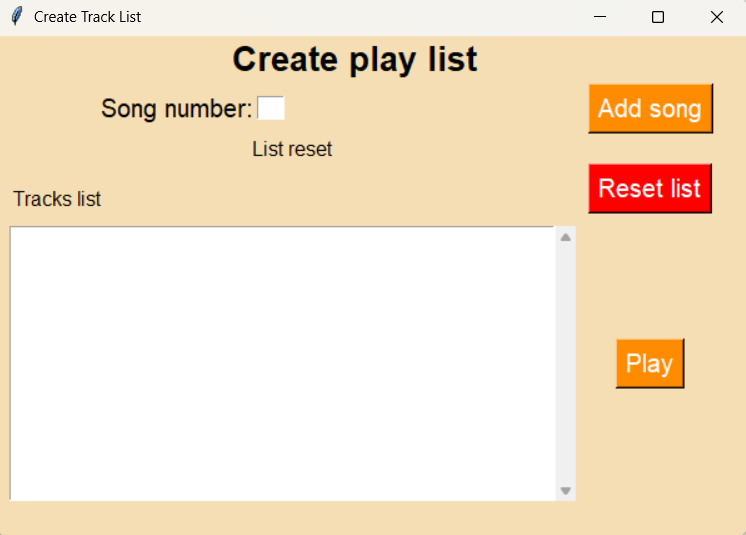


Image 12

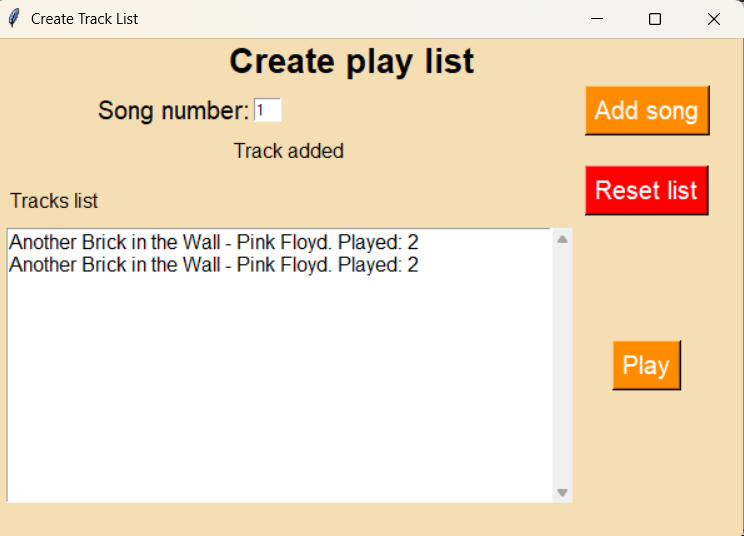


Image 13

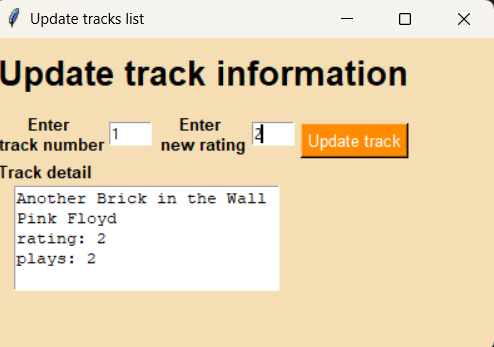


Image 14

## Testing and Faults

For testing,

* Summary of test results
* Discussion of encountered issues
  + Resolved faults and corrections made
  + Unresolved issues and limitations
  + Validation implementation
* Unit testing approach and results

## 4. Conclusions, Further Development, and Reflection

* Summary of the program achievements
* Evaluation against original requirements
* Further development possibilities (what you would do with additional time)
* Reflection (choose one):
  + Option A: Achievements, difficulties faced and why, straightforward aspects and why
  + Option B: Personal development, knowledge and skills gained, long-term value

## 5. Innovations

* Innovation 1:Save the playlist in an external file
  + Description and rationale
  + Implementation details
  + Benefits and limitations
* Innovation 2: Save the tracks library in a csv file
  + Description and rationale
  + Implementation details
  + Benefits and limitations

## Appendices

### Appendix A: Commented Code (Stage 1)

* Complete commented version of view\_tracks.py

### Appendix B: Test Table and Results

* Input values
* Actions performed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testing subject | Input | Expected output | Actual output | Test result |
| Track player GUI | None | GUI looks like Image 1 | GUI in Image 5 | Pass |
| Track viewer GUI | None | GUI looks like Image 2 | GUI in Image 6 | Pass |
| Create tracks list GUI | None | GUI looks like Image 3 | GUI in Image 9 | Pass |
| Update track GUI | None | GUI look like Image 4 | GUI in Image 7 and Image 8 | Pass |
| Update track’s update button | Any input or None in Track number input  Non-number or none in Rating input | Update track GUI With invalid rating input on text box | GUI in Image 15 | Pass |
| Update track’s update button | Non-number or None in Track number input  Number from 1 to 5 in rating input | Update track GUI with invalid number input on text box | GUI in Image 16 | Pass |
| Update track’s update button | 1 in Track number input  2 in rating input | Update track GUI with  ” Another Brick on the Wall Pink Floid Rating:2 Play:0” on text box | GUI in Image 17 | Pass |
| Create track list’ Add track button | Non-number input | Create track list GUI with invalid input label below entry | GUI in Image 18 | Pass |
| Create track list’ Add track button | Number lesser than 1 or greater than 8, in this case , 9 | Create track list GUI with Song doesn’t exist label below entry | GUI in Image 19 | Pass |
| Create track list’ Add track button |  |  |  |  |
|  |  |  |  |  |

* Expected outputs
* Actual outputs

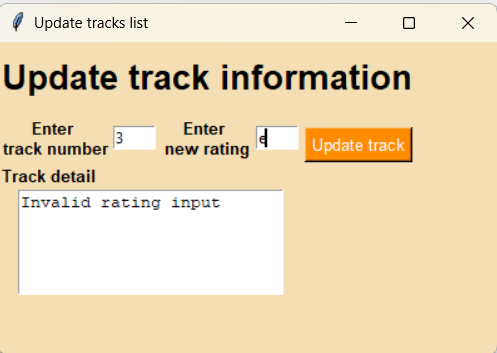


Image 15

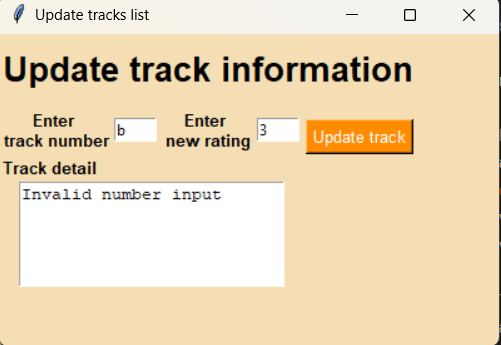


Image 16

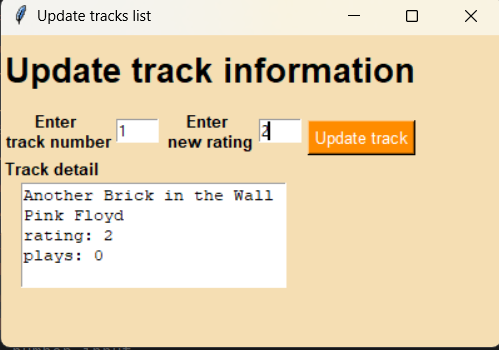


Image 17

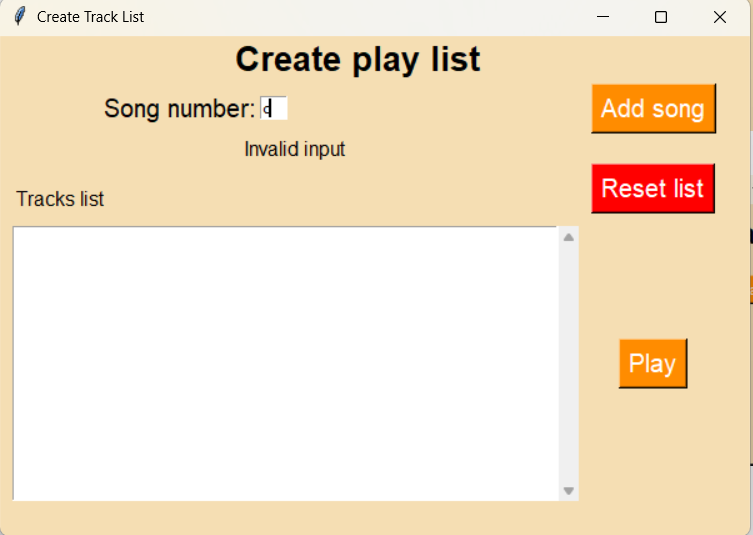


Image 18

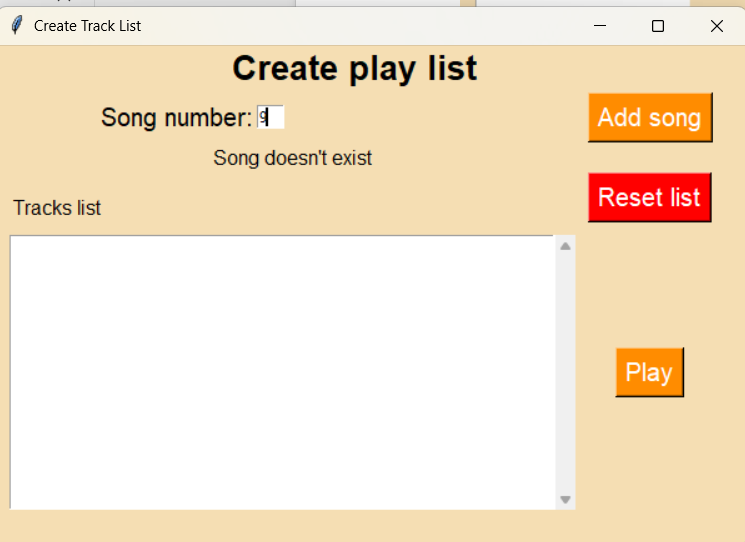


Image 19

* Pass/fail status

### Appendix C: Full Source Code

* All Python files with brief descriptions