Stage 1 – Library Management

This stage will mainly include getting the basic GUI layout ready and creating the library section so that the user can see what music they have stored on the computer

Design

Algorithms	
Algorithm	Explanation

Play/Pause button:
Button = “Paused”
Checked = False
If button is clicked:
  If checked = True
  Button = “Playing”
  Else
  Button = “Paused”
 	
  Checked = False	

Data Dictionary		
Variable	Type	Explanation
Class Diagrams		

Develop

Screenshot	Explanation/Justification

<table>
<tr>
<td>

```python
from PySide6.QtWidgets import QApplication, QPushButton, QMainWindow, QWidget, QHBoxLayout, QVBoxLayout, QTabWidget

1 usage  ± BHASVIC-willmoore24
class MainWindow(QMainWindow):
    ± BHASVIC-willmoore24
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Music Player")

        layout = QHBoxLayout()

        self.play_button = QPushButton("Paused")
        self.play_button.setCheckable(True)
        self.play_button.clicked.connect(self.play_pause)

        layout.addWidget(self.play_button)

        widget = QWidget()
        widget.setLayout(layout)
        self.setCentralWidget(widget)

    1 usage  ± BHASVIC-willmoore24
    def play_pause(self, checked):
        if checked == True:
            self.play_button.setText("Playing")
        else:
            self.play_button.setText("Paused")


app = QApplication([])

window = MainWindow()
window.show()

app.exec()
```

</td>
<td>MainWindow class will have other widgets included in it.</td>
</tr>
<tr>
<td>

```
◆ main.py
class LeftPane(QMainWindow):
    def __init__(self):
        super().__init__()
        layout = QVBoxLayout()

        self.tab_bar = QTabWidget()
        self.tab_bar.addTab(Library(), "Library")
        self.tab_bar.addTab(NowPlaying(), "Now Playing")

        layout.addWidget(self.tab_bar)

        widget = QWidget()
        widget.setLayout(layout)
        self.setCentralWidget(widget)


class Library(QMainWindow):
    def __init__(self):
        super().__init__()


class NowPlaying(QMainWindow):
    def __init__(self):
        super().__init__()
        layout = QVBoxLayout()

        self.play_button = QPushButton("Paused")
        self.play_button.setCheckable(True)  # button can be toggled
        self.play_button.clicked.connect(self.play_pause)

        layout.addWidget(self.play_button)

        widget = QWidget()
        widget.setLayout(layout)
        self.setCentralWidget(widget)

    def play_pause(self, checked):
        if checked:
            self.play_button.setText("Playing")
        else:
            self.play_button.setText("Paused")


app = QApplication([])

N  main.py                                   ◆ python  43|79  1|32
```

</td>
<td>Decided to use OOP classes for each of the main widgets as this allows me to separate out the variables they use (encapsulation) and makes it easy to add new widgets in the future, by just creating a new class.</td>
</tr>
</table>

```python
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Music Player")

        main_layout = QHBoxLayout()  # main layout w
        left_pane = LeftPane()
        right_pane = NowPlaying()

        main_layout.addWidget(left_pane)
        main_layout.addWidget(right_pane)

        widget = QWidget()
        widget.setLayout(main_layout)
        self.setCentralWidget(widget)
```

Using a horizontal layout (QHBoxLayout) allows for the two panes, where each individual pane will contain nested widgets.

Test

Test Description		Evidence

Review