Celem projektu jest umożliwienie pasażerom rejestracji do lotów a także zakup biletów, zmiana obecnego miejsca i klasy oraz generowanie karty pokładowej.

**Główny program zawiera siedem różnych klas:**

* Person

Klasa person słóży do tworzenia obiektów osób na podstawie ich id, które może być rozumiane np. jako pessel oraz name: imienia i nazwiska jako jednego srt. Dodatkowo klasa persona posiada tymczasowy argument fligh\_class, który pozwala przypisać osobę do dobrej klasy lotu i ułatwia testowanie. Oraz słownik który zawiera "miejsca/siedzenia które osoba kupiła" postać słownika jest z powodowana tym, że jedna osoba może mieć kika miejsc w kilku różnych lotach jednocześnie.

* Seat

Classa Seat reprezentuje obiekt siedzenia, zawiera atrybuty: passanger: jest to obiekt pasażera, który obecnie zajmuje to miejsce. Dla różnych lotów nawet jeżeli odbywają się tym samym typem samolotu obiekty siedzeń są różne, obiekt siedzenia istnieje w momęcie kiedy istnieje lot atrybut number: jest to numer siedzenia np (a1) (b3) itp, taki sposób numerowania ogranicza nas do ilości rzędów jaki może mieć samolot. Jeżeli był by to problem i samoloty miały by więcej rzędów w klasie niż liter w alfabecie można by przenieść się nasposób numeracj samymi liczbami np (1:1) odpowiadoało by naszemu (a:1).

* PlaneType

Obiekt ten trzyma typ samolotu a w kosneskwencji wymiary poszczególnych klas biletowych np. eco(4, 10 ) oznacza, że w klasie eco jest 10 rzędów po 4 siedzenia każdy, a bisnes(2, 5) oznacza, że w klasie bisnes jest 5 rzędów po 2 siedzenia każdy itd.

* Pasanngers

Classa Pasangers: jest klasą, którą dodałem do projektu aby umożliwić interfejsowi pamiętanie zmian. Imituje ona działanie bazy danych poprzez wczytywanie obiektów osób, które już zakupiły swoje bilety i mają przydzielone siedzenia, miejsca. Przyjmuje ona jako argumenty: path: ścierzka do pliku json, który imituje działanie bazy danych. plane: obiekt samolotu, który w tym wypadku można utożsamiać bardziej z obiekem samolotu w locie niż samego samolotu jak w rzeczywistości istnieje on po to aby klasa lot nie miała 300 linijek kodu, roździela jej działanie na dwie klasy.

* Plane\_Flight

Jest to klasa samolotu ale bardziej odzwierciedla w realnym życiu sam lot, przyjmuje ona typ samolotu i opcjonalnie ścieżkę z plikiem z osobami, które już zakupiły bilet. Gdybym miał wiecej czasu najpewniej przerobił bym ją aby działała jak prawdziwy samolot, czyli klasa ta miała by jako argumenty typ samoltu oraz liste lotów, jednak wymagało by to przeniesienia całej zawartości tej klasy do klasy fligh, czego nie chciałem tworzyć, ponieważ klasa miała by +300 linijek kodu ze względu na czytelność. Dlatego klasy Plane\_Flight i Flight można traktować jako jedną kalse podzieloną na dwa w celu ułatwienia działania i rozjaśnienia czytelności kodu. Klasa ta zawiera metody słóżące do dodawania nowych osób, które zakupiły bilety ale nie zdecydowały się na dodatkową opcje zakupu miejca, a także metode która tworzy tablice 2D z obiektów siedzeń dla poszczególnych klas.

* Flight

Classa Flight trzyma w sobie podstawowe dane dotyczące lotu takie jak numer lotu, samolot, który ten lot odbywa (w objekcie samolotu już znajdują się pasażerowie), cel, nr bramki odlotu, godzina odlotu, oraz słownik z cenami, który imituje cennik. Dodatkowo funkcja flight wywołuje instancje funkcji DatabasePlane.

* DatabasePlane

Klasa ta zajmuje się tworzeniem słowników z obiektów krzeseł dla poszczególnych lotów. Jako klucz używa numeru siedzenia a jako wartość trzyma jego obiekt. Umożliwia to późniejsze łatwiejsze odwoływanie się właśnie do tych obiektów krzeseł np. w funkcji buy\_ticket, change\_seat, change\_class itp. A także zapisywaniem osób, które kupiły bilety do pliku imitującego działanie bazy danych.

A także metody:

* Buy\_ticket

Metoda pozwalająca na zakup biletów dla wielu osób, w danej klasie dodatkowo za opłatą daje nam możlwiość wyboru miejsca dla poszczególnych osób. Przyjmuje argumenty: people czyli słownik z kluczmi, którymi są obiekty osoby a jako wartości numery siedzeń na których dane osoby chcą siedzieć. Jeżeli wartość nie zostanie podana miejsce zostaje przydzielone danej osobie losowo za pomocą funkcji add\_passangers bez dodatkowej opłaty. Flight jest to lot na, który bilety chcemy zakupić a fligh\_class toklasa siedzeń do, której chcemy kupić bilety.

* Change\_class

Metoda pozwalająca na zmiane obecnej klasy lotu na nową. Przyjmuje ona argumenty: person czyli obiekt osoby dla ,której chcemy zmienić klase lotu wystartczy podać id i name, jeżeli osoba o podanym id i imieniu znajduje się w liście pasażerów danego lotu to jej klasa zostanie zmieniona fligh, czyli lot ,w którym chcemu zmienic klase. New\_class to nowa klasa ,na którą chcemy zmienić naszą obecną klase, new\_seat to opcjonalnie za opłatą możma wybrać sobie w nowej klasie miejsce.

* Change\_seat

Metoda, która pozwala na zmiane miejsca w obrębie danej klasy. Przyjmuje obiekt osoby, lotu oraz numer nowego siedzenia. Jest wykorzystywana przez metoda change\_class.

* Person\_is

Metoda pozwalająca na znalezienie osoby w danym locie za pomocą jego Id i name:

**Interfej:**

**UWAGA zakładamy, że id jest uniklane reprezentuje np. pessel i nie ma osób z tym samym id.**

Plik interface zawier w sobie jedną klase, która odpowiada za działanie interfejsu i wywoływanie takich metod jak Buy\_ticket Change\_seat itp. Komunikuje się on z użytkownikiem za pomocą seri pritnów i inputów. W zależności od wprowadzonych danych wykonuje dane akcje. U góry pliku znajdudją się słowniki imitujące bazy danych zawierające typy samolotów, dostępne samoloty i loty.

W ramach testów interfejsu istnieją dwa loty/samoloty, których plikami wejściowymi i wyjściowymi są fligh997 i 998 .json. Dzięki zastosowaniu tych plików imitujących działanie bazy danych interfejs „pamięta” poprzednie zmiany. Dzięki czemu jeżeli podczas jednego wywołania programu dodamy osobę A do lotu B to przy kolejnym wywołaniu programu ta osoba nadal będzie w tym locie i bedzie zajmować miejsce, które dla niej wybraliśmy. Aby łatwiej testować wyjątki proporcje samolotów zostały lekko zaburzone od rzeczywistości i np. w klasie ekonomicznej pierwszego typu samolotu są tylko 4 miejsca. Aby to zmienić wystarczy zmienić liczby w krotkach w łowniku planetypes gdzie pierwsza liczba reprezentuje ilość miejsc w rzędzie a druga ilość rzędów, a krotki są ustawione kolejno w kolejności eco, bisnes i premium.

Dodatkwo interfejs zawiera metodę pozwalającą nam na zobaczenie w momęcie wykonywania akcji na miejscach i klasach obecnego stanu danej klasy siedzeniowej. Wyświetla on nam tabelkę z numerami miejsc ich stanem.

**Main**

Plik main słóży do uruchomienia interfejsu.

**Boarding\_pass**

Słóży do generowania karty pokładowej za pomocą zbioru etykietek. Jest wywoływany przez jedną z metod klasy Interface.

**Testy**

W testach zazywczaj poza ostatnim nie podaje ścieżki do odczytu z pasażerami aby uniknąć generowania niepotrzebnych plików zamiast tego podaje do metody add\_passenger liste osób już z ustawionymi klasami co imituje działanie funkcji kupowania biletu bez wyboru siedzenia, a dopiero potem wykonuje opercaje takie jak zmiana klasay czy siedzenia. Do testów potrzebny jest plik flight\_test.json jako plik z pasażerami dla jednego testu i plik flight999.json w normalnym działaniu programu te dwa pliki pownny być jednym, ale tutaj aby uniknąć dodawania w nieskończoność tych samych osób do pliku podajemy dwa różne.

***Instrukcja użytkownika***

Aby uruchomić interfejs trzeba przez interpreter uruchomić plik main. A następnie za pomocą klawiatury podawać terminalowi wymagane dane.

1). Zaraz po uruchomieniu terminalu wyświetli nam się lista możliwych akcji.

0) List people

1) Add person

2) Remove person

3) Buy tickets

4) Change class

5) Change seat

6) Generate boarding pass

9) Exit

Aby rozpocząć działanie trzeba najpierw dodać osbę/osoby ,na których będziemy chcieli wykonywać dane akcje. Słóży do tego opcja Add person ,do której przejdziemy po wciśnięciu przycisku 1. Wtedy zostaniemy poproszeni o podanie wpierw ID: np. 123 a potem imienia i nazwisa np. Jan Kowalski. W ten sposób dodajemy osoby tyle ile potrzebujemy. Aby zobaczyć jakie osoby dodaliśmy do programu trzeba wcisnąć przycisk 0. Wyświetli nam się wtedy lista osób ,które już dodaliśmy. Aby usunąć osobęz tej listy wciskamy przysik 2 i analogicznie jak przy dodawaniu osoby wpisujemy jej id i imie i nazwisko.

2). Kupowanie biletów. Funkcja kupowania biltów wywołuje się dla wszystkich osób w liście pasażerów dlatego musimy być pewni ,że w tej liście są tylko te osoby dla ,których ten bilet chcemy kupić. Jeżeli znajduje się tam niepożądana osoba to należy ją z tej listy wpierw usnąć a dopiero potem przejść do kupowania biletów. W czasie kupowania biletów postępujemy wedle intrukcji wyświetlających się na ekranie. Wybieramy lot ,na który chcemy kupić bilety poprzez podanie jego numeru. Następnie wybieramy klase w jakiej chcemy siedzieć. Wyświetlona zostanie nam podobna tabelka z X oznaczającymi miejsca zajęte a O miejsca wolne sekwencja liter i cyfr przy X i O to numer siedzenia.

(a1 X) (a2 O) (a3 X) (a4 O)

(b1 O) (b2 O) (b3 O) (b4 O)

A na końcu opcjonalnie za opłatą 50 zł można wybrać miejsce ,w którym chcemy siedzieć z tabeli powyżej. Jeżeli chcemy żeby przydzielono nam miejsce automatycznie to wystarczy nic nie wpisywać i kliknąć enter. W tym momencie na ekranie pojawi się kwota do zapłaty. Jeżeli chcemy zakupić kolejne bilety na ten sam lot dla innych osób to trzeba obecne osoby usunąć i dodac nowe albo wyjść z interfejsu i włączyć go jeszcze raz. Ponieważ **dozowlone jest kupno tylko jednego biletu na dany lot na osobe**. Nie jest problemem zakup kilku biletów przez jedną osobę tak długo jak każdy z nich jest na inny lot.

3). Zmiana klasy i miejsca możliwa jest tylko dla jednej osoby naraz, ale nadal ta osoba musi być wcześniej dodana za pomocą metody Add person postępujemy analogicznie jak w przypadku zakupu biletów. Wybieramy wtedy daną opcje za pomocą przycisku 4 lub 5 podajemy dane osoby ,której chcemy zmienić siedzenie/klase a potem wprowadzamy dane ,o które poprosi nas interfejs takie jak nowe siedzenie czy nowa klasa. W przypadku zmiany klasy wybór siedzenia jest opcjonalny i ponosi za sobą koszt 50zł. Zmiana klasy i siedzenia jest tylko możliwa jeżeli już znajduje się dana osoba w liście pasażerów tego lotu. Dodatkowo zmiana klasy z wyższej na niższą premium => bisnes => eco nie skutkuje zwrotem pieniędzy. Analogicznie jeżeli zapłaciliśmy za wybór miejsca w klasie A i zmiamy klase na klase B to jeżeli chcemy wybrać sobie siedzenie to także musimy dopłacić 50 zł opłata z poprzedniej klasy nie przenosi się.

4) **Uwaga do wygenerowania karty pokładowej potrzebny jest włączony Xserver i posiadanie biblioteki PySide2.QtWidgets** Generowanie karty pokładowej, postepujemy analogicznie jak przy zmianie miejsca lub klasy. Wybieramy opcje 6 podajemy swoje dane i nasza karta pokładowa zostanie nam wygenerowana i wyświetlona na ekranie.

5) Aby zakończyć działanie programu należy wcisnąć przycisk 9.

***Część refleksyjna***

Program działa tak jak na początku zostało zamierzone z małymi modyfikacjami. Wraz z powstawaniem projektu pojawiały się nowe pomyły na implementacje ,które starałem się w miare mozliwości implementować tak jak np. wczytywanie listy pasażerów z pliku co pozwala na zapamiętywanie czynności kupna biletu przez interfejs. Pojawiały się taże nowe pomysły na struktóry klas jedne z nich wykorzystałem inne nie, ze względu na ograniczony czas i czytelność kodu tak jak pomysł ze złączeniem klasy Plane\_Flight i Flight w jedną dużą klase i stworzenie klasy Plane ,która trzymała by tylko typ samolotu i liste lotów. Tym nad czym można się zastanowić to numercja siedzeń obecnie rzędy są określane za pomocą kolejych liter a siedzenia za pomocą kolejnych liczb, można by tą proporcje odwrócić co spowodowało by ,że system ten mógł by obsługiwać nieskończenie długi samolot, ponieważ zakładamy jednak ,że nie ma samolotów z ponad 25 rzędami. Ewentualnie całkowicie zastąpić litery za pomocą cyfr w formacie (1:1) (1:2) (1:3) itd. Jednak uważam ,że raczej dla polskich samolotów nie spotykamy się z taką sytuacją żeby ilosć rzędów pojedyńczej klasy przekraczała ilość liter w alfabecie a dodatkowow system a1 a2 a3 jest bardziej przyjazny dla pasażera. W programie nie obsługiwaliśmy wyjątków ,które raczej nie miały wpływu z samą tematyką programu a więc trzeba wziąć to po uwagę podczas jego użytkowania. To czego nie udało się wykonać a raczej z braku czasu niż umiejętności był interfejs całkowicie zrobiony w QT zamiast wyświetlania komunikatów w terminalu. Były by one wyświetlane w oknie i tam użytkownik by je obsługiwał np. wybrór miejsca zamiast wyświetlania tabeli z miejscami jak pokazane poniżej:

(a1 X) (a2 O) (a3 X) (a4 O)

(b1 O) (b2 O) (b3 O) (b4 O)

Należało by wyświetlić tabele przyciksów np. czerwonych zajętych a niebieskich wolnych reprezentujacych miejca i wtedy klikając na poszczególny przycisk urzytkownik wybierałby swoje miejsce. Można by także umożliwić kupowanie jednej osobie wielu biletów na ten samo lot ale wymagało by to przetrzymywania słownika list a nie słownika obiektów i uznałem ,że zaciemniało by to tylko kod.

Uważam ,że wykonałem to o co byłem proszony w treści projektu ,a także dodałem kilka rzeczy od siebie ,które sprawiają ż,e jego użytkowanie jest bardziej intuicyjne. Dodatkowo kod starałem się dzielić na jak najwięcej klas dzięki czemu owiele łatwiej jest w nim wprowadzać modyfikacje i dodawać nowe rozwiązania. Dlatego uważam ,że zasługuje na wysoką ocenę. :D