Programming Project – Group 18

Outline of Design

For our program, we wanted to make a simple and intuitive UI that anybody could pick up and use. We wanted to create a fast and responsive GUI where the user did not feel lost or confused at any given moment. This meant there could be no hangs or freezes for database pulls, and every button must be clear on what it does, as well as the UI being as simplistic as possible, and avoiding clutter, while still delivering the necessary information and functionality.

Our program aimed to represent the given statistical data in three ways, a Search Bar, An Overview screen and an interactive map. These screens were to be navigated through the Home screen.

Work Distribution

We decided to communicate over discord and meet up once a week to discuss progress and deal with any issues.

- **Darryl** - Base Widget/Screen functionality, configuration of the SQLite database, Repo management (solving merge conflicts, etc), interactive map functionality & screen, menu screen, loading functionality, debugging, UI design, various Widget extensions.
- **Ted** - Searchbar Screen, early database functionality (before SQLite implementation), UI design, testing.
- **Tiernan** - PieChart, Overview screen, UI design, testing.
- **Michael** - TextWidget, radio button, filter slider, UI design, testing.

Problems Encountered

Charts - There is no library available on processing for pie charts, which were the main charts we wanted to use to display the data. This meant that we needed to draw one using various arcs which was quite time consuming to figure out. The first problem encountered when doing this was that out get functions returned integers, and we were using an array of integers for the data of the pie chart. This led to the pie chart components either overlapping each other or a gap being present when testing, this obviously was an easy fix in the end, but i feel like the problem should be mentioned due to the amount of time that it took to figure out.

Another problem we encountered was with bar charts, we found a library that can create bar charts but it required a PApplet to be put into the argument. And if I'm being honest (Tiernan writing here) I didn't know how to get that with the way the screens were set up. I did have a go at creating a manual bar chart but it was too late. I couldn't get it done in time and therefore we had no bar chart.

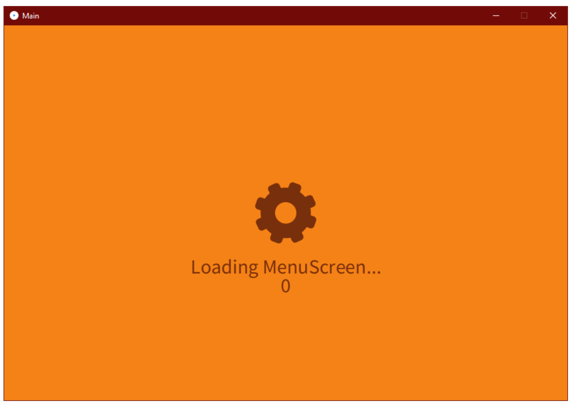
Scope Creep - It was clear to us that we would not be able to implement everything in a timely manner. For example, we originally planned to use sliders and implement date ranges in the search bar. As a result, we were left with an unused slider class than can be seen as a waste of time and resources. More planning could have been used here.

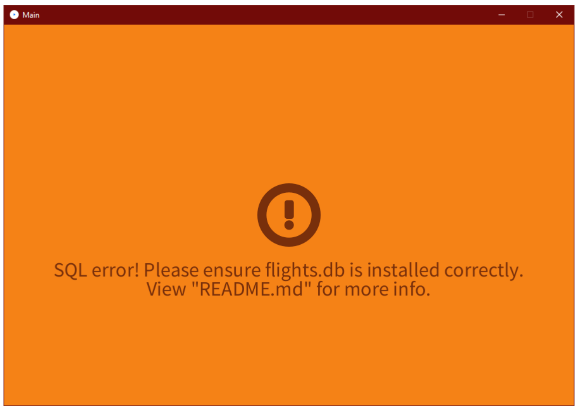
Database Speed/Choice - Initially, we used a generic Java file-loading method as a temporary way to load files and get basic features down while we seeked out a better one. Nobody in the group had ever tackled a problem like this before, so discovering a new database system and porting to it would be a challenge. Eventually, we settled on SQLite, due to its ease to implement in a end-user program, as opposed to other SQL variations that are more suited for servers.

Structure - Making a program from scratch, you don't have a lot of the things that other things like engines offer. We had to decide on a set structure on how to do things, which can be seen in the Widget implementation. This can be hard, as later on you can realise that some things could be done or named in a better, more efficient, way, but it becomes difficult as the functions are already used so commonly in the program.

Features Implemented

Loading Screen:

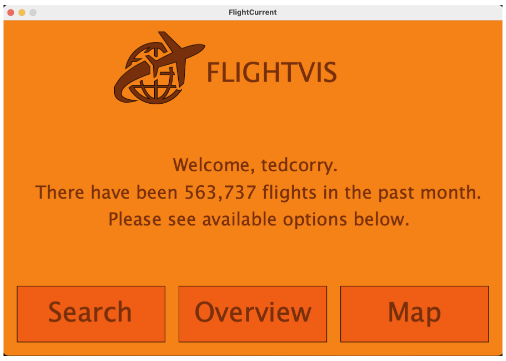




SQL error! Please ensure flights.db is installed correctly.
View "README.md" for more info.

This is the first Screen the User is presented with, it will show loading progress with a spinning gear and current time elapsed. If the incorrect database is being used, this screen will tell the user. This avoids a large hang upon loading the program, and keeps the user informed.

Home Screen:

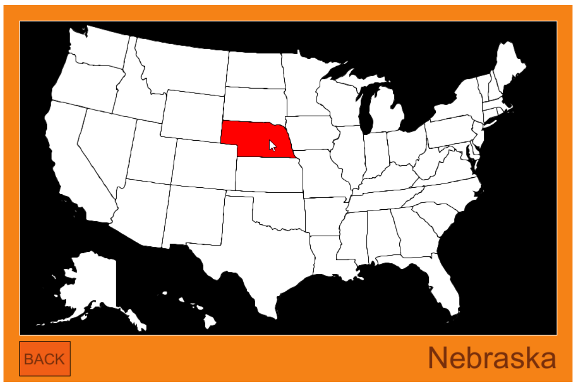


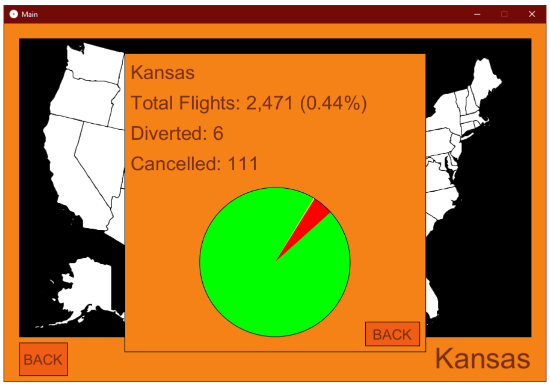
Welcome, tedcorry.
There have been 563,737 flights in the past month.
Please see available options below.

This screen allows the user to navigate to the three functions, it also supplies the user with information about the dataset's contents (total flights.)

As well as this, there are smooth transitions to each of the three pages, which utilises the parenting system within Widget and Screen.

Map Screen:

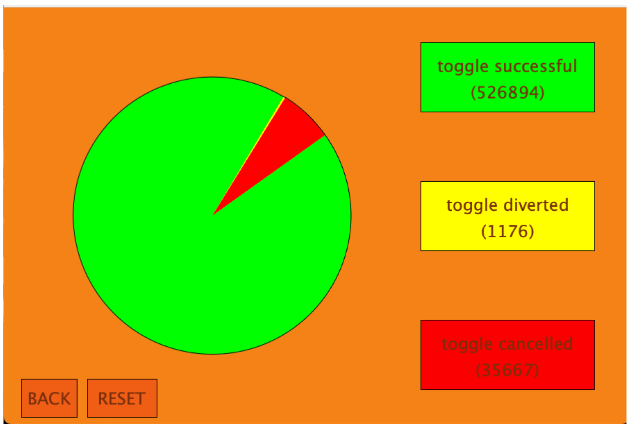




Kansas
Total Flights: 2,471 (0.44%)
Diverted: 6
Cancelled: 111

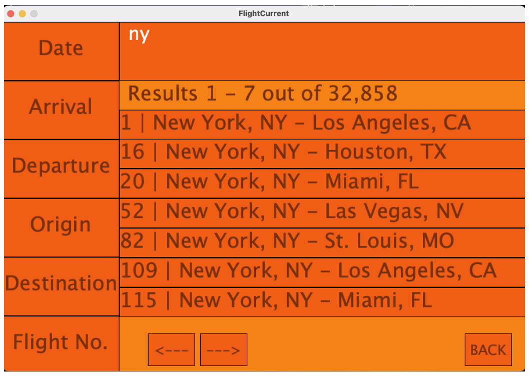
The map screen is an interactive map, which allows the user to zoom and pan the map, as well as select a state and see information about that state. This screen utilises the library Geomerative to avoid some bugs in stock Processing involving scaling vectors, namely involving .contains().

Overview Screen:



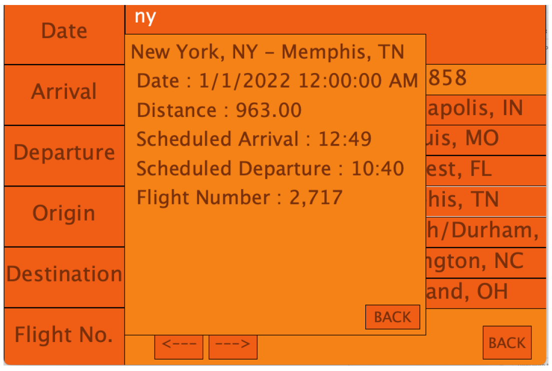
The overview screen shows the user a pie chart with the Diverted, Successful and cancelled flights on it. The user can toggle on/off on each of the three and can reset it back to its original state with the reset button.

Search Screen:



ny

Results 1 – 7 out of 32,858
1 | New York, NY – Los Angeles, CA
16 | New York, NY – Houston, TX
20 | New York, NY – Miami, FL
52 | New York, NY – Las Vegas, NV
82 | New York, NY – St. Louis, MO
109 | New York, NY – Los Angeles, CA
115 | New York, NY – Miami, FL



ny

New York, NY – Memphis, TN
Date : 1/1/2022 12:00:00 AM
Distance : 963.00
Scheduled Arrival : 12:49
Scheduled Departure : 10:40
Flight Number : 2,717

The search screen gets the user to enter a value in relation to the six filters on the right. Once the value is entered in the search bar and the user selects a filter. The results are printed on the screen. The user can navigate through the results using the arrow buttons. They can also select a certain result by pressing on it which will reveal a pop up window containing the relevant information about that flight.