

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 our 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.

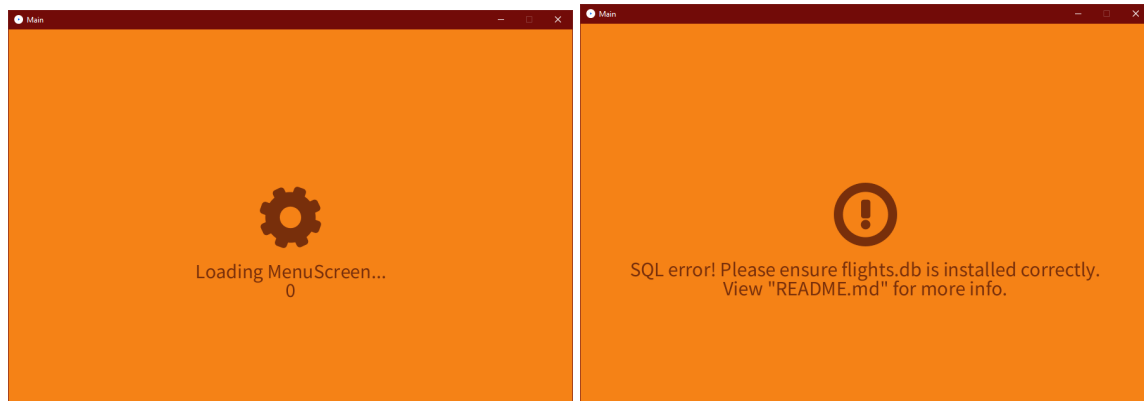
Scope Creep - It was clear to us that we would not be able to implement everything in a timely manner. For example, we originally planned to use sliders and implement date ranges in the search bar. As a result, we were left with an unused slider class than can be seen as a waste of time and resources.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

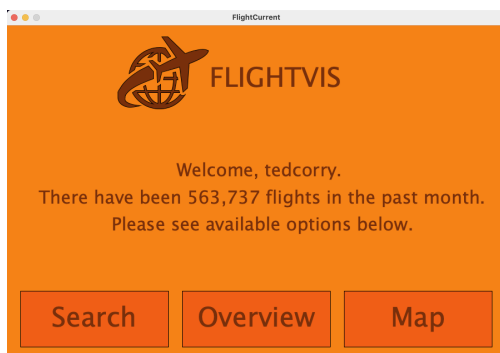
Features Implemented

Loading Screen:



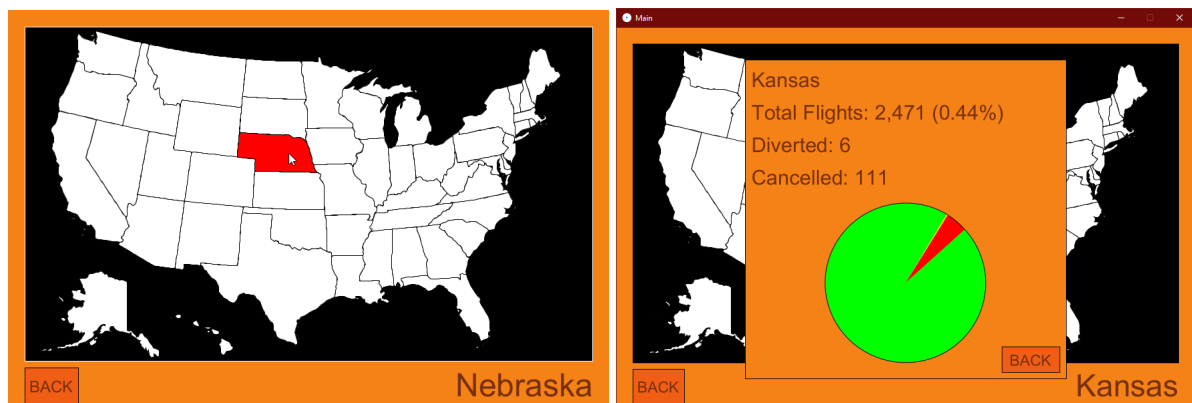
This is the first Screen the User is presented with, it will show loading progress with a spinning gear and current time elapsed. If the incorrect database is being used, this screen will tell the user. This avoids a large hang upon loading the program, and keeps the user informed.

Home Screen:



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.)

Map Screen:



The map screen is an interactive map, which allows the user to zoom in and out of the map as well as select a state and see information about that state. This screen uses the library Geomorative.

A pie chart illustrating the distribution of toggle actions. The chart is divided into three segments: a large blue segment representing 'toggle successful' (526894), a small red segment representing 'toggle diverted' (1176), and a medium green segment representing 'toggle cancelled' (35667). The segments are labeled with their respective counts and descriptions. Below the chart, there are two buttons: 'BACK' and 'RESET'.

Action	Count
toggle successful	526894
toggle diverted	1176
toggle cancelled	35667

Search Screen:

FlightCurrent		ny	
Date		New York, NY – Memphis, TN	
Arrival	Results 1 – 7 out of 32,858	Date : 1/1/2022 12:00:00 AM	858
	1 New York, NY – Los Angeles, CA	Distance : 963.00	apolis, IN
Departure	16 New York, NY – Houston, TX	Scheduled Arrival : 12:49	is, MO
	20 New York, NY – Miami, FL	Scheduled Departure : 10:40	est, FL
Origin	52 New York, NY – Las Vegas, NV	Flight Number : 2,717	his, TN
	82 New York, NY – St. Louis, MO		h/Durham
Destination	109 New York, NY – Los Angeles, CA		ngton, NC
	115 New York, NY – Miami, FL		and, OH
Flight No.	<input type="text" value="<--"/> <input type="text" value="-->"/> <input type="button" value="BACK"/>	<input type="text" value="<--"/> <input type="text" value="-->"/> <input type="button" value="BACK"/>	

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.