# CSU11013 – 2024/25: PROGRAMMING PROJECT
# JUNIOR FRESHMAN – GROUP 24 PROJECT REPORT

**FINN MITCHELL – CHLOE DI BLASSIO – HONOR MIERS- TANMAYEE REDDY GOLI – LARISA KUTREVAC**

## ABSTRACT

The aim of our project was to build a user-friendly graphical interface that allows users to explore and analyse U.S. domestic flight data. Using Processing, we created an interactive tool that reads data from a CSV file and turns it into visual displays, helping users identify trends and patterns without needing to look through raw data.

The application includes three main features: a bar chart that shows frequency, distance, or lateness across selected categories; a map that displays flight paths across the U.S. using real coordinates and zoom/pan controls; and a table/list view that presents detailed flight information in a clean, scrollable format.

Users can filter and sort the data through dropdown menus and interact with buttons and textboxes, making the interface dynamic and adaptable. The code is organized using separate classes like Screen, DropDownMenu, BarChart, and List, following object-oriented principles to keep things modular and easier to update.

## PROJECT DETAILS

The program was designed using a modular, object-oriented approach to keep the code organized, readable and scalable. It follows a clear separation between data handling and display logic. All sorting, filtering and data calculations are handled separately from how the data is visualised on screen.

We used a two-screen setup, with the Screen class having two instances: a welcome screen that contains one widget, an entry button, and a data screen that houses all the visual components like bar charts, maps, and lists, along with the more complex widgets for user interaction.

The Display class serves as an abstract superclass for all visual components. From it we created three major visualization types:

- BarChart – shows flight frequency, lateness or distance by state, airport or airline
- Map – plots flight routes across U.S. using the Web Mercator projection
- List – a scrollable table-like view showing individual flight records with colour- coded delay status

Similarly, we made an abstract Widget class with subclasses like

- Button – basic navigation and actions
- DropDownMenu – selecting sort and filter options
- FilterDropdown – interactive filtering

The core logic for sorting and filtering the data set is handled by a dedicated SortAndFilter class, which passes relevant slices of the flight data  to each display based on user selection. We also made use of Processing's built-in Table class to manage our dataset. It allowed easy access to columns and rows, which simplified a lot of our data operations.

To handle interaction across all parts of the interface, we implemented an EventHandler class. It manages all user input, including clicks, dropdown selection and keyboard entries and ensures the correct updates are reflected in the displays.

**GROUP MEMBER DETAILS**

| MEMBER NAME | FEATURES IMPLEMENTED |
|---|---|
| Finn Mitchell | • Designed and implemented the BarChart class, including variable sorting, dropdown integration, and text input feature allowing users to customize the number of bars<br>• Added x-axis label rotation, enhancing readability when many bars are displayed<br>• Implemented a gradient colour scheme for bars<br>• Developed the SortByDate functionality in the List display for chronological analysis<br>• Built the navigation button system for switching between screens |
| Chloe Di Blassio | • Refined BarChart UI and functionality<br>• Refined overall UI, including Welcome Screen and Widget layout<br>• Improved the map's interactivity by refining zoom/pan logic; incorporated bounding box for map display and interaction<br>• Helped adjust the Web Mercator projection for better accuracy of state/airport locations on the map<br>• Refactored and integrated the SortAndFilter class, ensuring data processing logic was clean and functional across all display types |
| Honor Miers | • Integrated the map into the full application, ensuring it worked with other classes and UI components<br>• Wrote the EventHandler class and connected map features to it<br>• Created and wrote the Screen class that manages all UI components, displays objects and handles rendering.<br>• Created the abstract Display class and ensured all subclasses (Map, BarChart and List) stayed in sync with user input and filtering |
| Tanmayee Reddy Goli | • Handled initial data loading, cleaning and preprocessing, including lateness calculation and cancelled flights<br>• Developed the original data queries that the SortAndFilter class is built upon<br>• Suggested using the ControlP5 library to manage dropdown menus and UI interactions, but we challenged ourselves and built it from scratch using our custom Widget subclasses |
| Larisa Kutrevac | • Developed Map class and drawing logic for flight paths and airport plotting |

| | |
|---|---|
| | • Processed and mapped airport location data from the dataset into screen coordinated.<br>• Implemented zoom and pan functionality<br>• Built logic to associate flights with airports and states for accurate visual representation<br>• Provided the base for all map interactivity, including visualising routes selecting airports and overlaying real-world flight paths |

## EXECUTION AND RESULTS

When the program is launched, users first see a welcome screen. Clicking "Enter Flight Program" transitions to the data screen, which holds all three major displays: BarChart, Map, and List. Each display is interactive and updates in real time based on user input.

- The BarChart display reflects metrics such as frequency, average distance, and lateness. Users can select how many bars to show using a custom textbox. The chart updates dynamically when dropdown options change, with gradient-coloured bars to improve clarity.
- The Map display shows flight paths across the continental U.S., with zoom and pan controls. Clicking on an airport or state reveals detailed statistics including delays, cancellations, and frequent routes.
- The List display presents flight records in a scrollable format, with sortable columns and color-coded delay statuses. It adjusts the number of visible rows based on screen size.

## BEYOND THE BRIEF

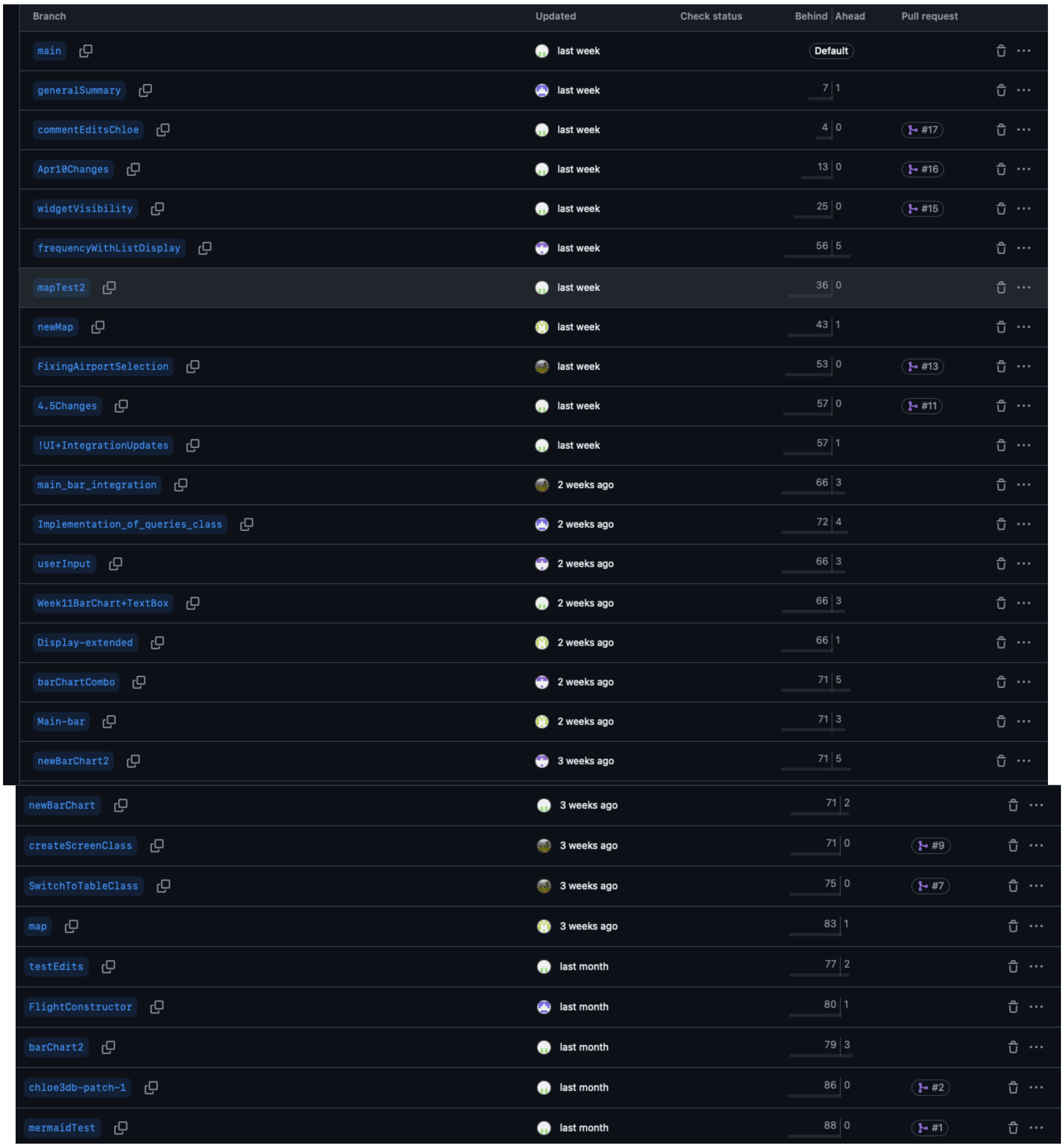
One notable challenge we undertook was building all interface elements entirely from scratch. While we considered using external libraries like ControlP5 for GUI elements, we decided to write our own DropDownMenu, Button, and Textbox classes as Widget subclasses. This gave us full control over their look, behaviour, and integration with the rest of the system, which aligns with object-oriented best practices.

## CONCLUSION

This project demonstrates how large-scale tabular data, such as flight records, can be turned into an engaging and interactive visual tool. With modular components like Widgets, Screens, EventHandler and Display subclasses (BarChart, Map and List), the program is both flexible and easy to maintain. Users can explore flight patterns, delays, and distances through real-time interactions, dynamic sorting, and zoomable maps. The structured design and clean UI make the data accessible and visually informative.

## REFERENCES /LITERATURE REVIEW

- We utilized the Processing reference, examples, and Javadocs. This included full implementation of the built-in Table class for our data processing. We utilized examples such as the scrollbar to guide UI enhancements.
- Javadocs: (HashMap and HashSet)
- Map references:
    - List of US state locations: Wikipedia
    - List of US airport locations: LatLong.net
    - Mercator projection mapping (including formulas): Wikipedia



| Branch | Updated | Check status | Behind | Ahead | Pull request |
|---|---|---|---|---|---|
| main | last week | | Default | | |
| generalSummary | last week | | 7 | 1 | |
| commentEditsChloe | last week | | 4 | 0 | #17 |
| Apr10Changes | last week | | 13 | 0 | #16 |
| widgetVisibility | last week | | 25 | 0 | #15 |
| frequencyWithListDisplay | last week | | 56 | 5 | |
| mapTest2 | last week | | 36 | 0 | |
| newMap | last week | | 43 | 1 | |
| FixingAirportSelection | last week | | 53 | 0 | #13 |
| 4.5Changes | last week | | 57 | 0 | #11 |
| !UI+IntegrationUpdates | last week | | 57 | 1 | |
| main_bar_integration | 2 weeks ago | | 66 | 3 | |
| Implementation_of_queries_class | 2 weeks ago | | 72 | 4 | |
| userInput | 2 weeks ago | | 66 | 3 | |
| Week11BarChart+TextBox | 2 weeks ago | | 66 | 3 | |
| Display-extended | 2 weeks ago | | 66 | 1 | |
| barChartCombo | 2 weeks ago | | 71 | 5 | |
| Main-bar | 2 weeks ago | | 71 | 3 | |
| newBarChart2 | 3 weeks ago | | 71 | 5 | |
| newBarChart | 3 weeks ago | | 71 | 2 | |
| createScreenClass | 3 weeks ago | | 71 | 0 | #9 |
| SwitchToTableClass | 3 weeks ago | | 75 | 0 | #7 |
| map | 3 weeks ago | | 83 | 1 | |
| testEdits | last month | | 77 | 2 | |
| FlightConstructor | last month | | 80 | 1 | |
| barChart2 | last month | | 79 | 3 | |
| chloe3db-patch-1 | last month | | 86 | 0 | #2 |
| mermaidTest | last month | | 88 | 0 | #1 |

*Screenshots of different branches on git and history*

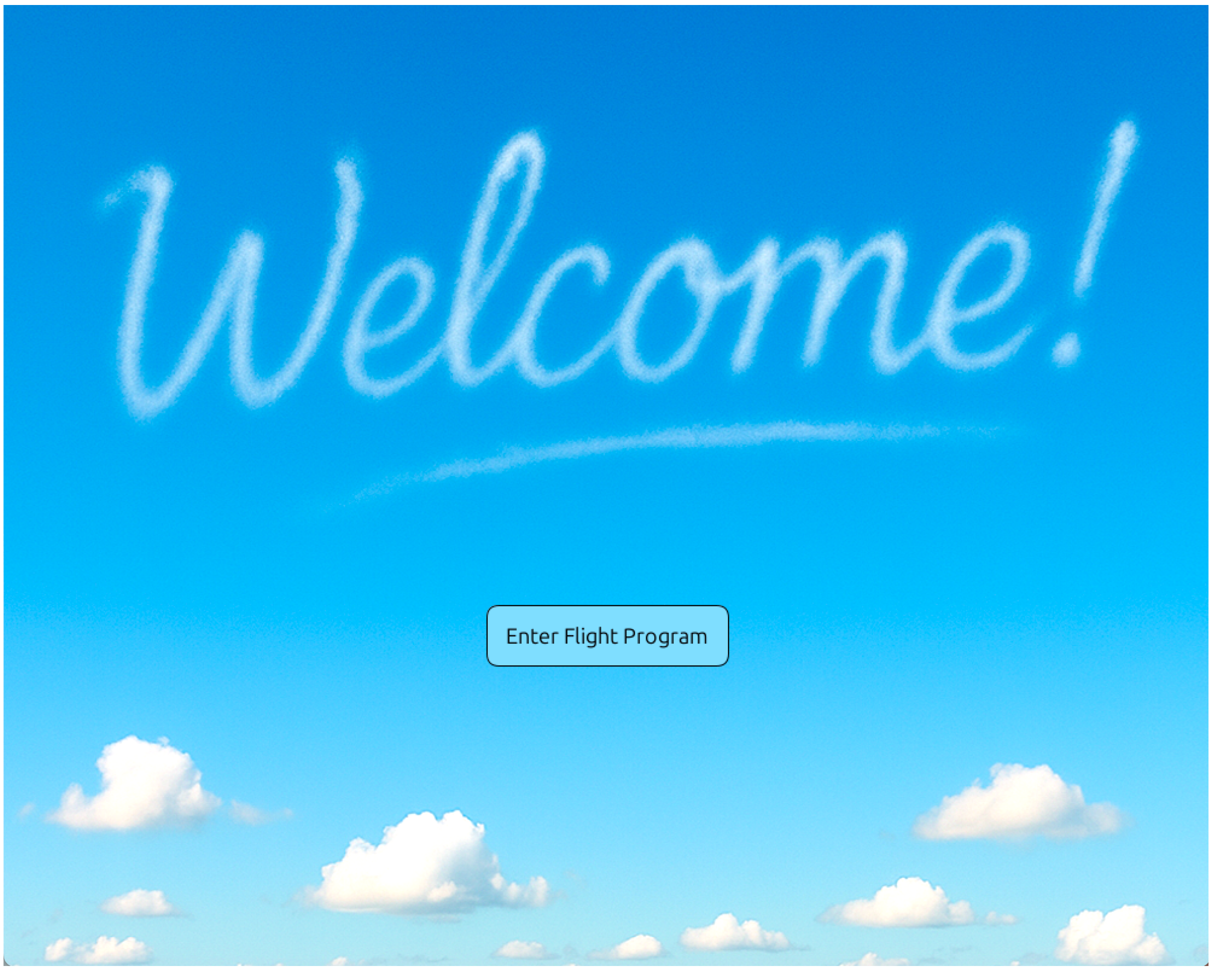


*Figure 1: Welcome screen prompting the user to begin the Flight Data Visualization program*

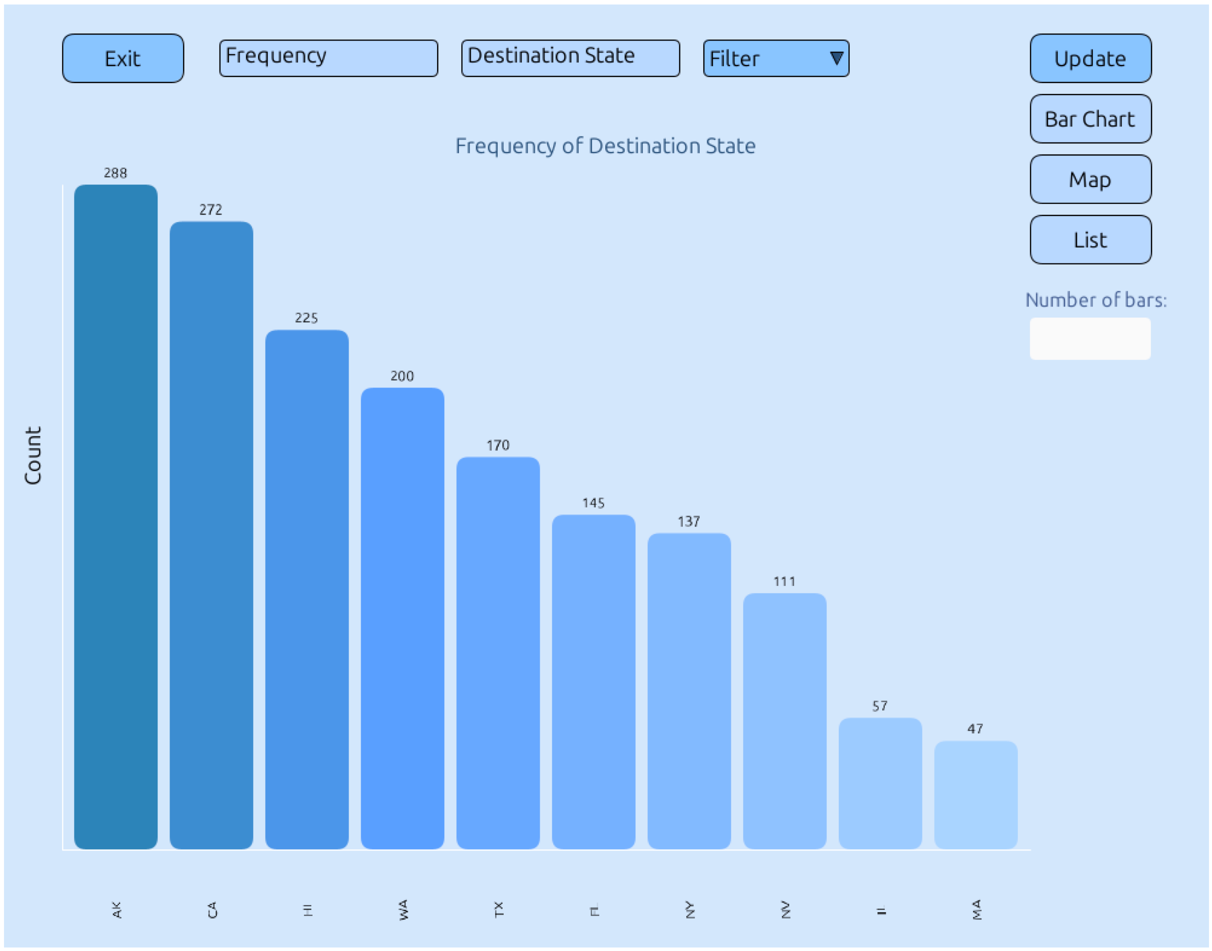


*Figure 2: Data screen layout showing navigation buttons and dropdown menus for selecting visualizations and sorting methods*

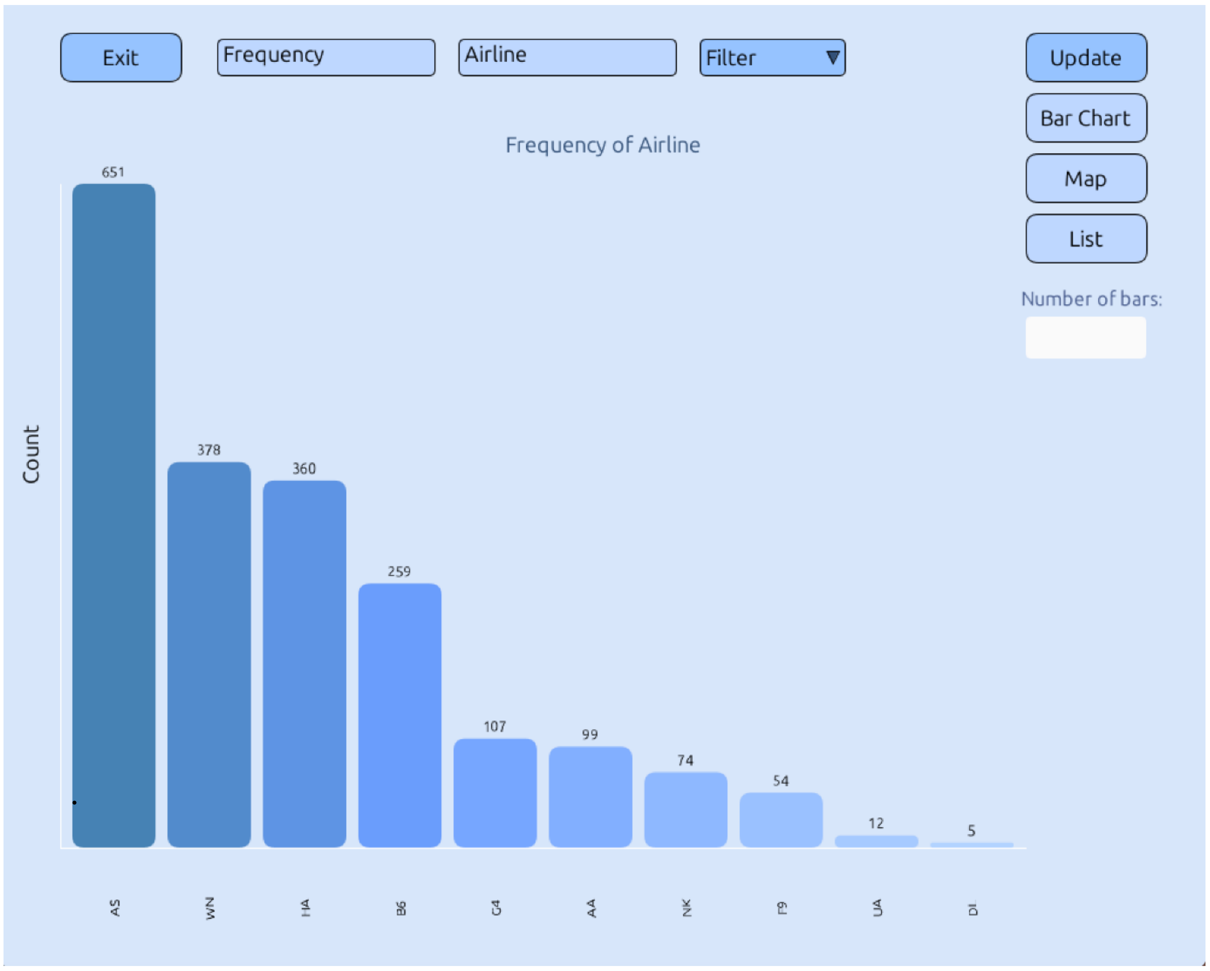


*Figure 3: Bar Chart displaying flight frequency by airline*

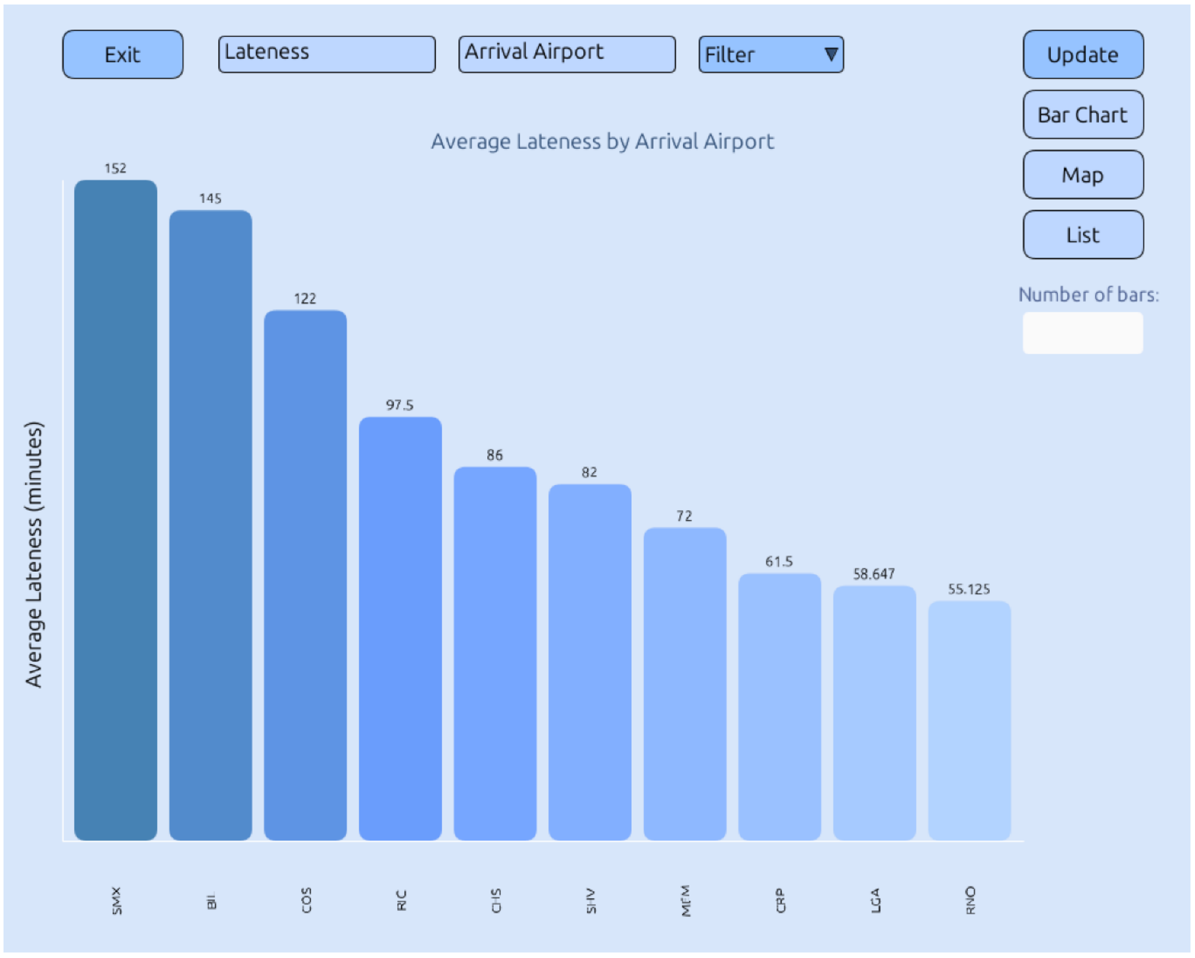


*Figure 4: Bar Chart displaying average lateness by arrival airport*

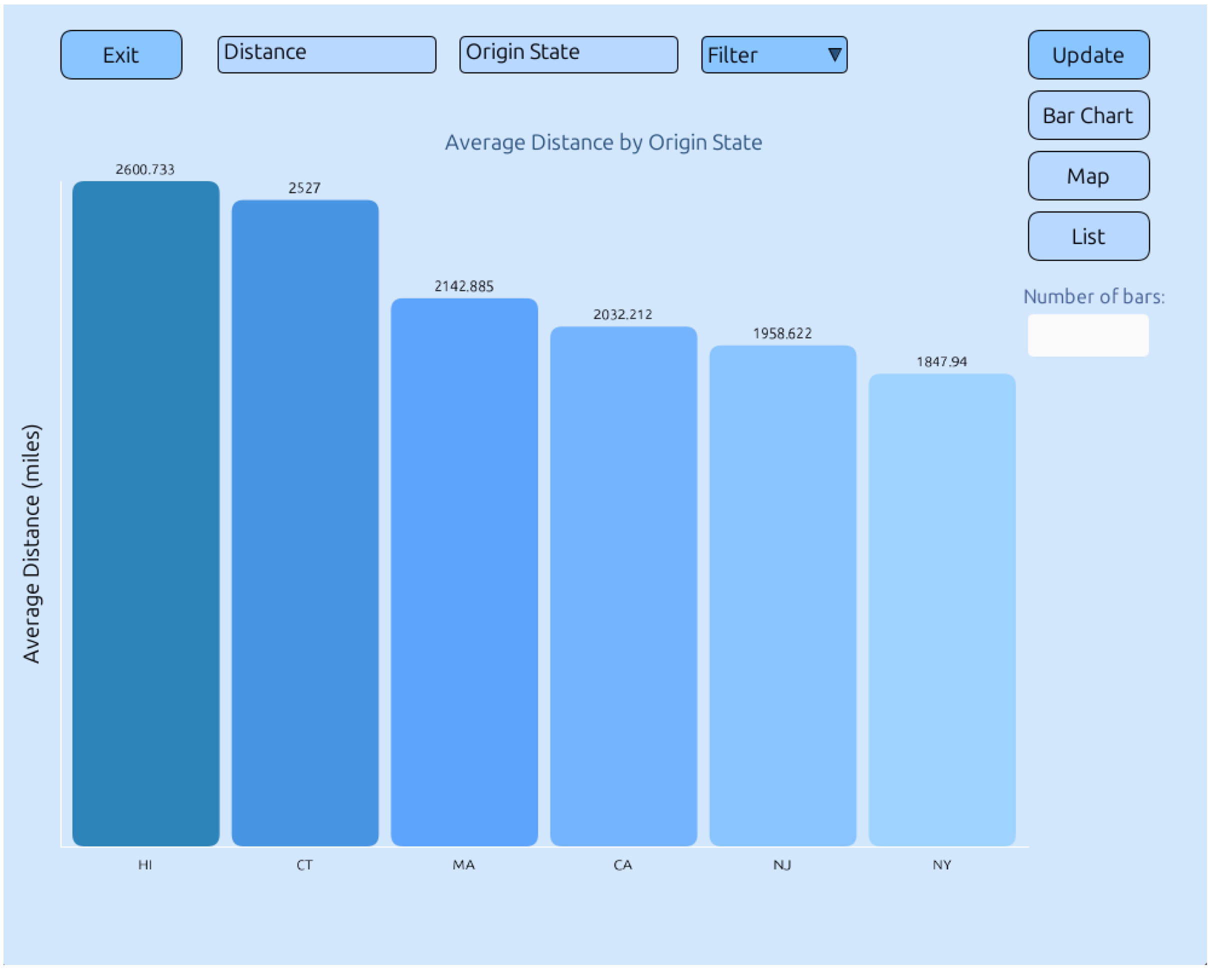
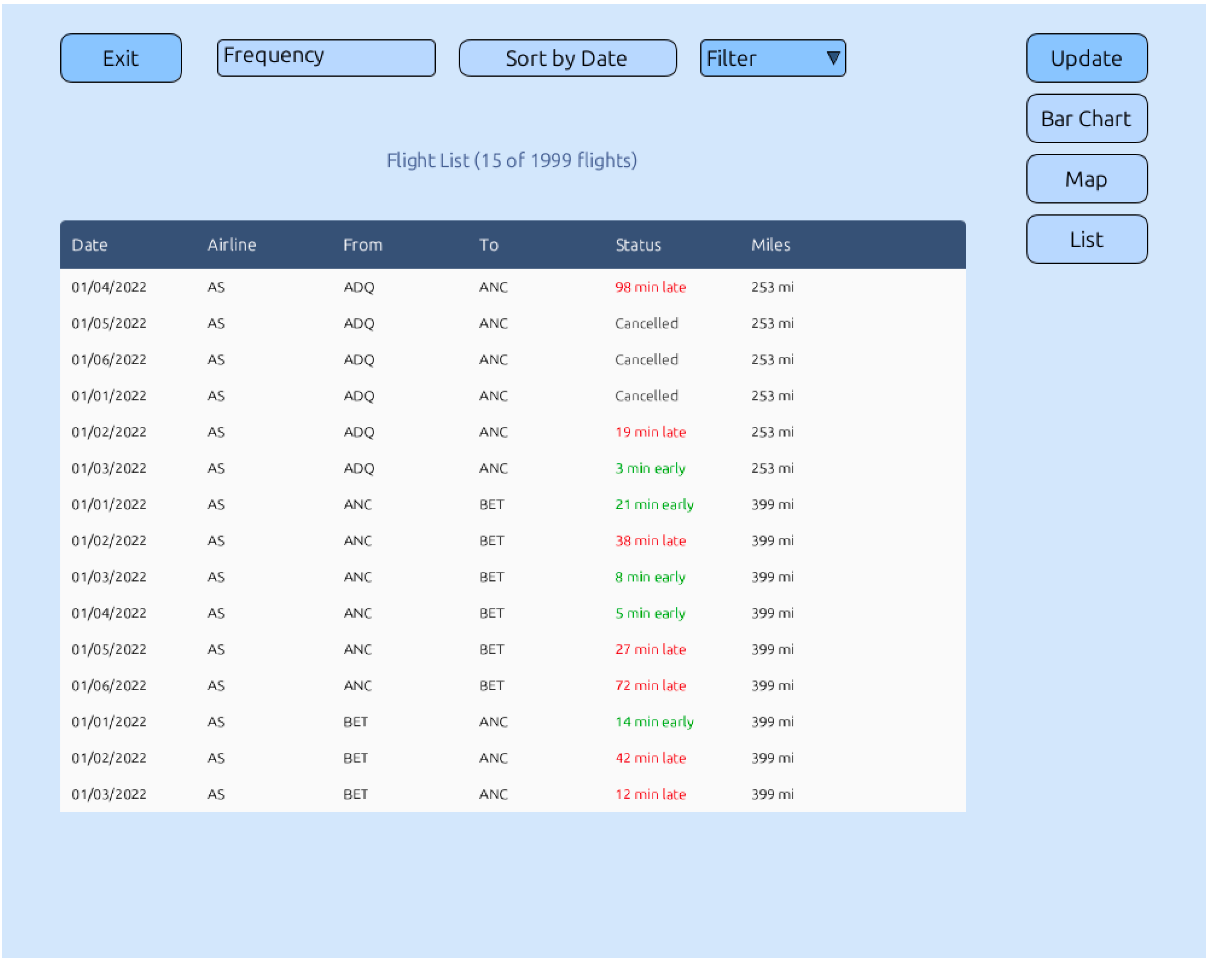


*Figure 5: Customizable bar count allows users to control how many categories are visualized.*



Flight List (15 of 1999 flights)

| Date | Airline | From | To | Status | Miles |
|---|---|---|---|---|---|
| 01/04/2022 | AS | ADQ | ANC | 98 min late | 253 mi |
| 01/05/2022 | AS | ADQ | ANC | Cancelled | 253 mi |
| 01/06/2022 | AS | ADQ | ANC | Cancelled | 253 mi |
| 01/01/2022 | AS | ADQ | ANC | Cancelled | 253 mi |
| 01/02/2022 | AS | ADQ | ANC | 19 min late | 253 mi |
| 01/03/2022 | AS | ADQ | ANC | 3 min early | 253 mi |
| 01/01/2022 | AS | ANC | BET | 21 min early | 399 mi |
| 01/02/2022 | AS | ANC | BET | 38 min late | 399 mi |
| 01/03/2022 | AS | ANC | BET | 8 min early | 399 mi |
| 01/04/2022 | AS | ANC | BET | 5 min early | 399 mi |
| 01/05/2022 | AS | ANC | BET | 27 min late | 399 mi |
| 01/06/2022 | AS | ANC | BET | 72 min late | 399 mi |
| 01/01/2022 | AS | BET | ANC | 14 min early | 399 mi |
| 01/02/2022 | AS | BET | ANC | 42 min late | 399 mi |
| 01/03/2022 | AS | BET | ANC | 12 min late | 399 mi |

*Figure 6: Scrollable table of flight data with color-coded delay statuses and sortable columns.*

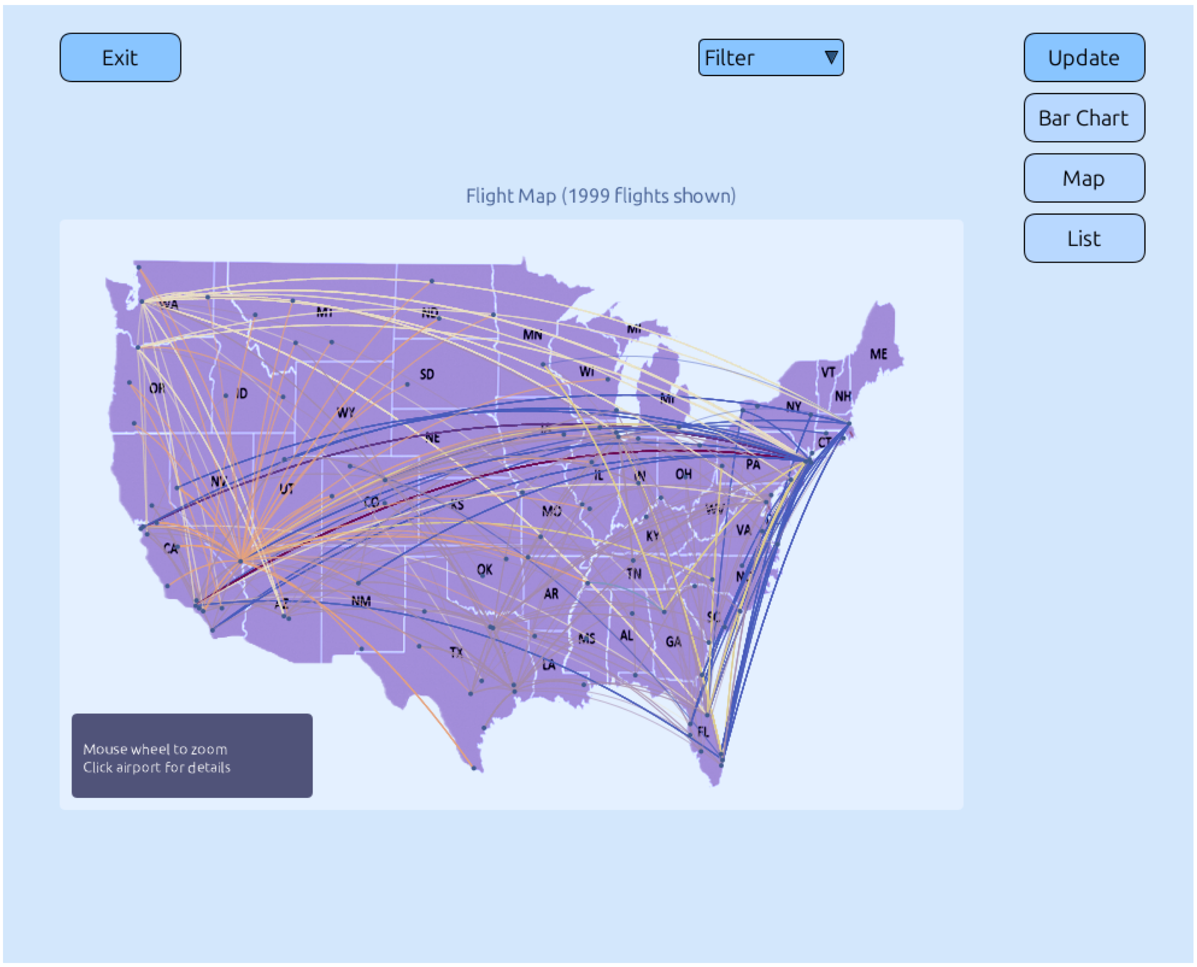
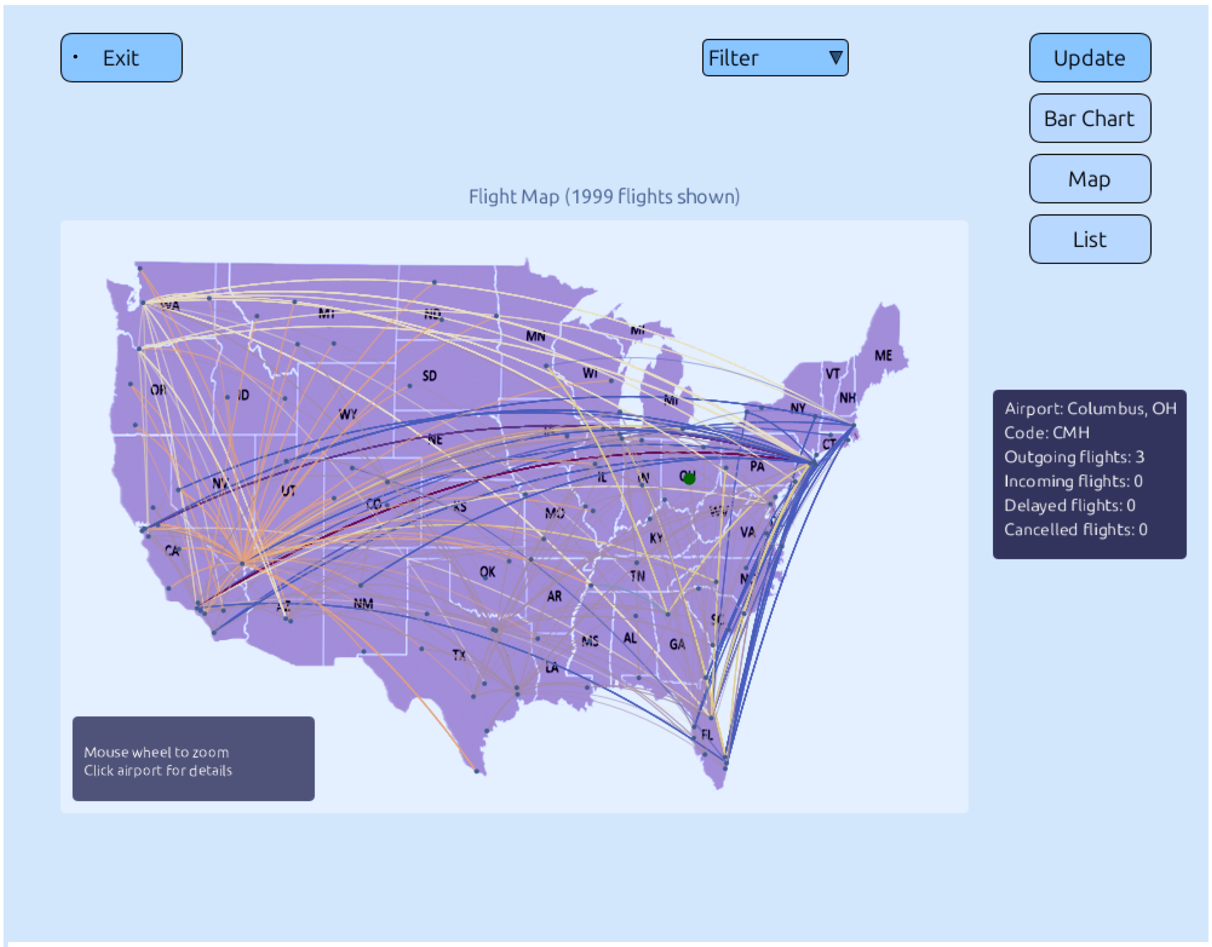


*Figure 7: Interactive U.S. map plotting domestic flight paths and airport locations.*



Airport: Columbus, OH
Code: CMH
Outgoing flights: 3
Incoming flights: 0
Delayed flights: 0
Cancelled flights: 0

*Figure 8: Hovering over an airport displays city name; clicking reveals detailed flight statistics.*