Fire Manifesting App

Dawson Nash, Ben Brown
College of Engineering, Computer Science
Professor: Dr. Witmer
Application Client: Bryan Quimby

UAA College of Engineering
University of Alaska Anchorage

Why is it needed?

Figure 1: Swan Lake Fire Sterling, AK, 2019

- Firefighters operate in high-stress, time-sensitive environments where miscalculation can lead to delays in needed resources as well as safety risks.
- The 'Fire Manifesting App' streamlines and automates the process for generating helicopter manifests.
- The app reduces potential human-error and optimizes operational efficiency.

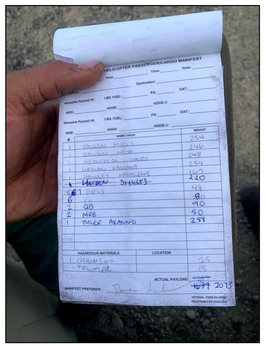

Figure 2: Handwritten Manifest

Manifests are currently Handwritten. Which has several Downsides:

- Time Consuming
- Error Prone
- Mentally Tiring for crewmembers
- Vulnerable to last-minute helicopter requirements changes

Try it Yourself!

Android:

Step 1: Settings > Install Unkown Apps

SCAN ME

IOS

Step 1
Get TestFlight on the App Store

Step 2
https://testflight.apple.com/join/9QxrwrQX

Step 3
View & Download Fire Manifesting App Beta

How is it used?

Figure 3: Home Screen

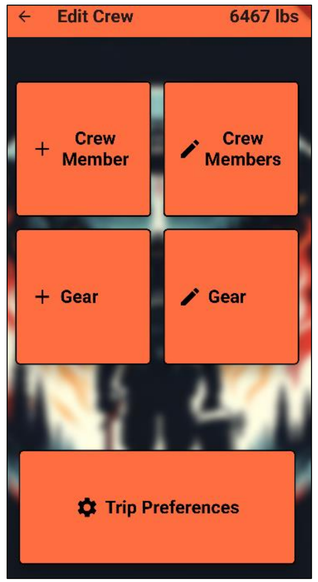

Figure 4: Add/Edit Gear, Crewmembers, Preferences

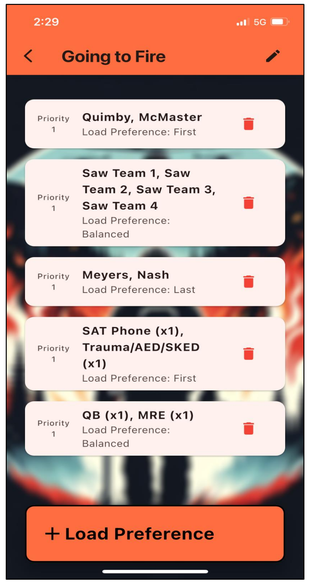

Figure 5: Add Trip Preferences

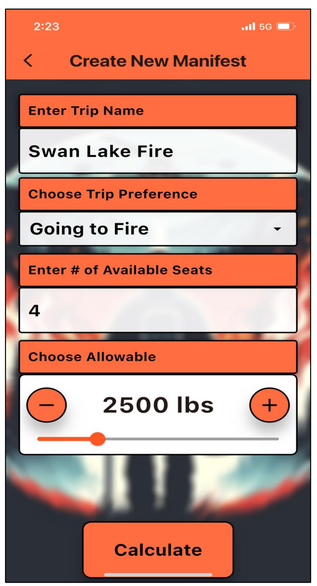

Figure 6: Create Manifest Screen

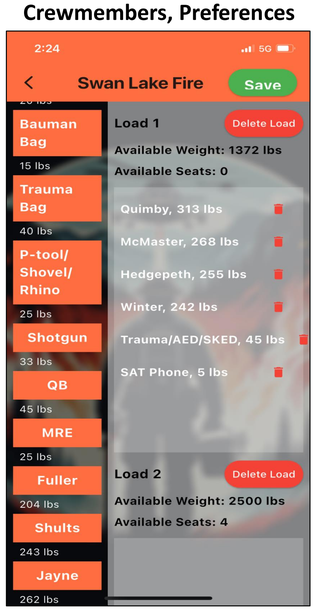

Figure 7: Build Your Own Manifest Screen

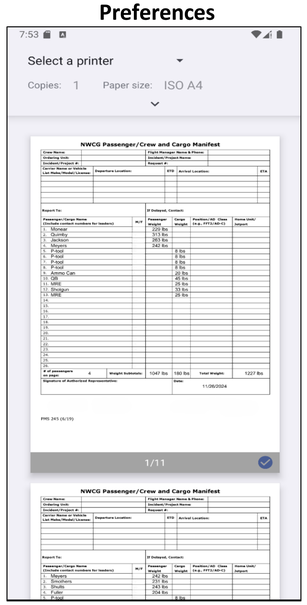

Figure 8: Exported Manifest to PDF

How Is it Built?

- The app was built using Flutter, an open-source UI toolkit by Google.
- It allows for 'multi-platform' app development, letting us write code once and deploy it on both iOS and Android.
- Flutter uses Dart, an object-oriented language that compiles to various formats based on the platform.
- In Flutter, everything on the screen is a widget or a combination of widgets.
- We used Hive, a native NoSQL database for Flutter, to store user input data.

How Does it Sort?

- Users Inputs:
  - crew information, including crewmembers, gear, and their weights.
  - Helicopter requirements, including the number of seats and max weight limit.
  - Sorting preferences, dictating conditions for the sorting algorithm.
- Preferences include prioritizing crewmembers or gear for first, last, or balanced loads.
- The algorithm creates a trip object with several load objects, sorting items based on preferences.
- It then 'smartly' sorts any remaining items not bound by a preference.
- The sorting problem resembles the Knapsack problem, solved using a greedy approach, which is not always optimal.

Future Work:

- The App is currently in an alpha stage with planned future work:
  - Additional Algorithm testing
  - Additional UI testing with different devices
  - Additional Crew input and sorting options
  - Several User QoL features