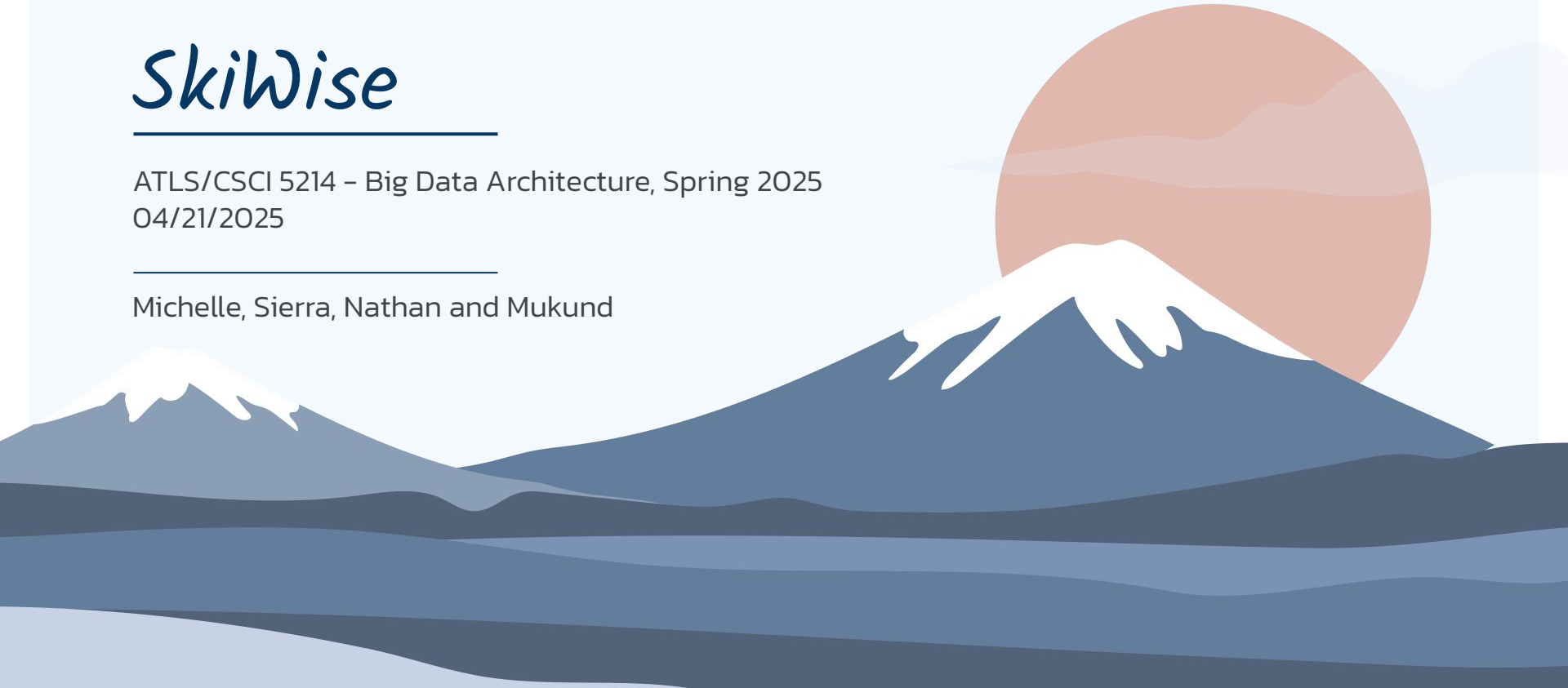
# SkiWise

ATLS/CSCI 5214 – Big Data Architecture, Spring 2025
04/21/2025

Michelle, Sierra, Nathan and Mukund



# Meet the Team

## Michelle Gjolberg

Frontend developer
Exchange student from NTNU
🇳🇴

## Sierra Reschke

Backend and Database
MS CS
🇺🇸

## Mukund Mahesan

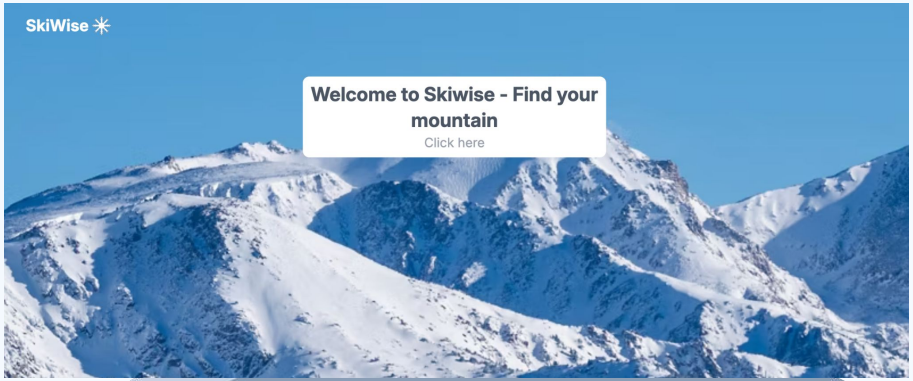
Backend and Deployment
MS CS
🇮🇳

## Nathan Varghese

Optimization and Testing
MS AE – Robotics
🇮🇳

# What is SkiWise?

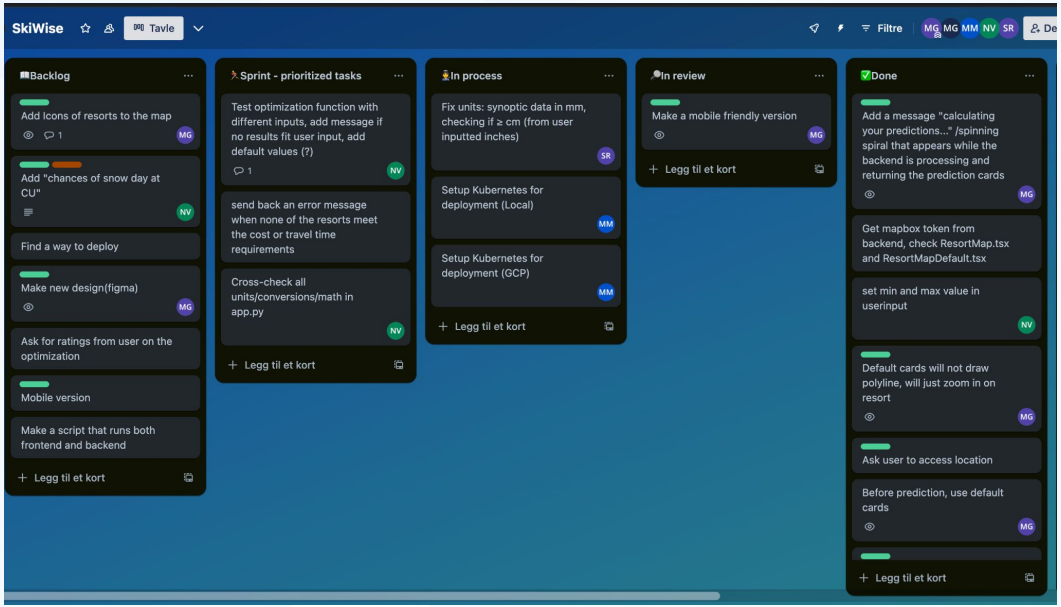
- Where should you go skiing?🫵
- Finds based on real-time conditions
    - 1 hr and 24 hr snowfall, road conditions, travel time
- Takes into account user preferences



# Goals

- Build an Interface Skiers Actually Want to Use
- Make Real-Time Data Actionable
- Optimize for You
- Various technologies and platforms
    - Real-time APIs
    - Google Cloud Platform

# Agile Software Development




Scrum board: Trello

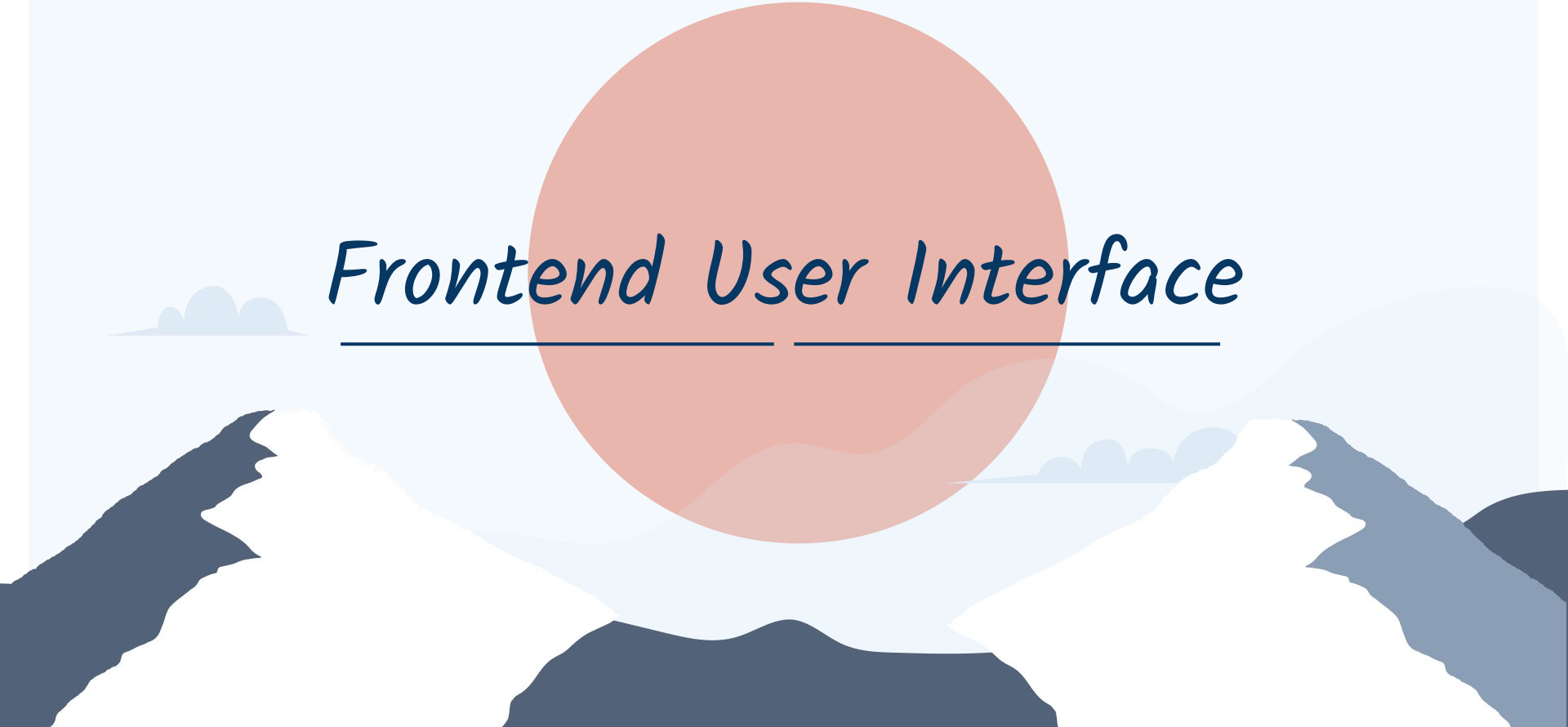
"Daily" standups x2

- Fits a small team

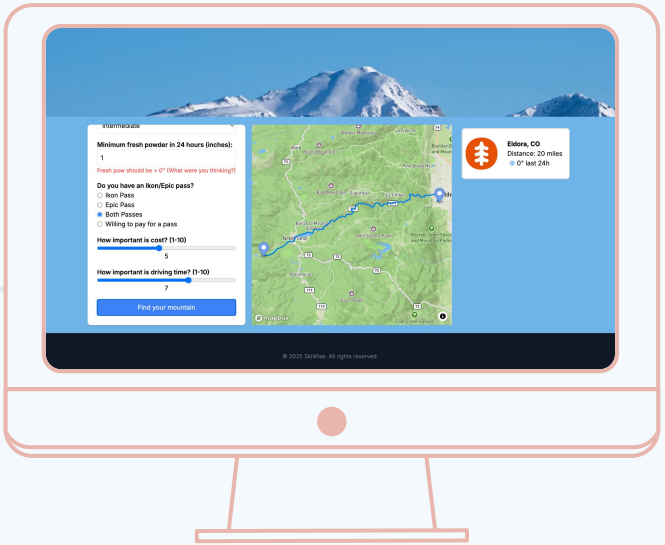
2 week long sprints

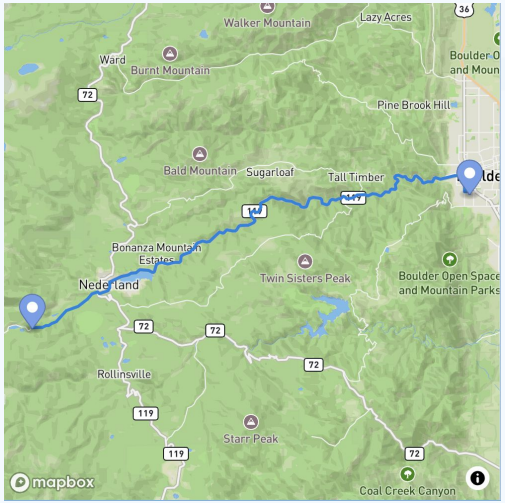
- Planning -> Retrospective

# Frontend User Interface

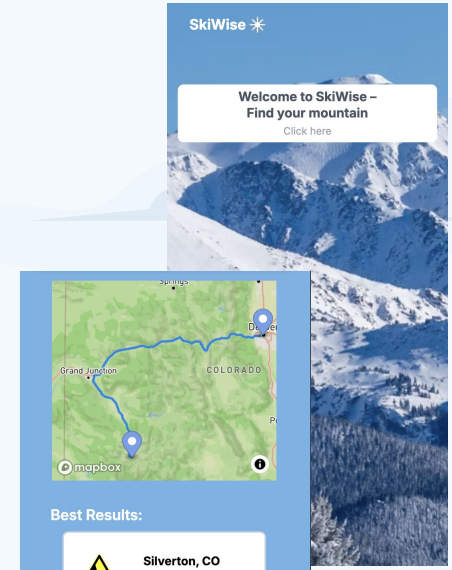


# Interactions with the user:




mapbox

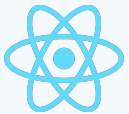


# SYSTEM ARCHITECTURE

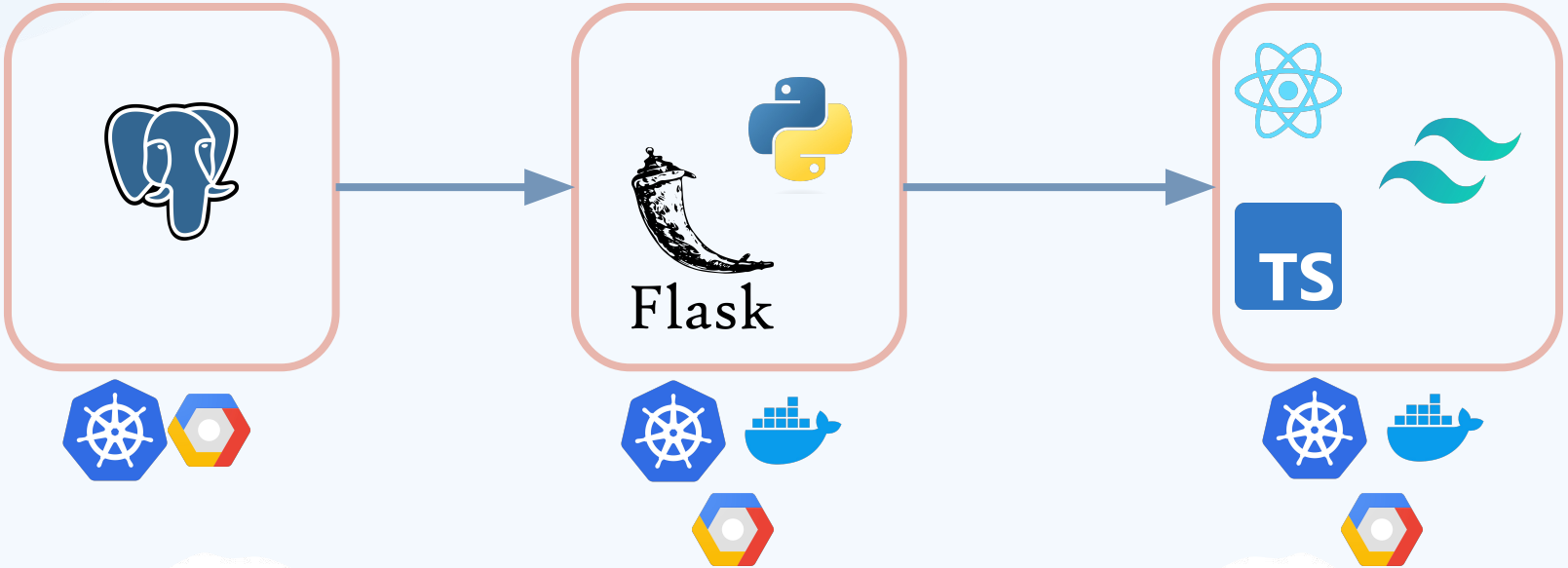
# Trello Technologies Leveraged & Tech Stack

- **Languages:** Python, Typescript
- **Frameworks:** Flask, React
- **Database:** PostgreSQL
- **APIs:** Google Maps API, Weather API (Synoptic API), Traffic API, REST API
- **Frontend:** React, Typescript, Tailwind
- **Backend:** Flask
- **Deployment:** Google Cloud Platform, Docker, Kubernetes
- **Agile tools:** Trello, GitHub
- **Testing tools:** Vitest (frontend)





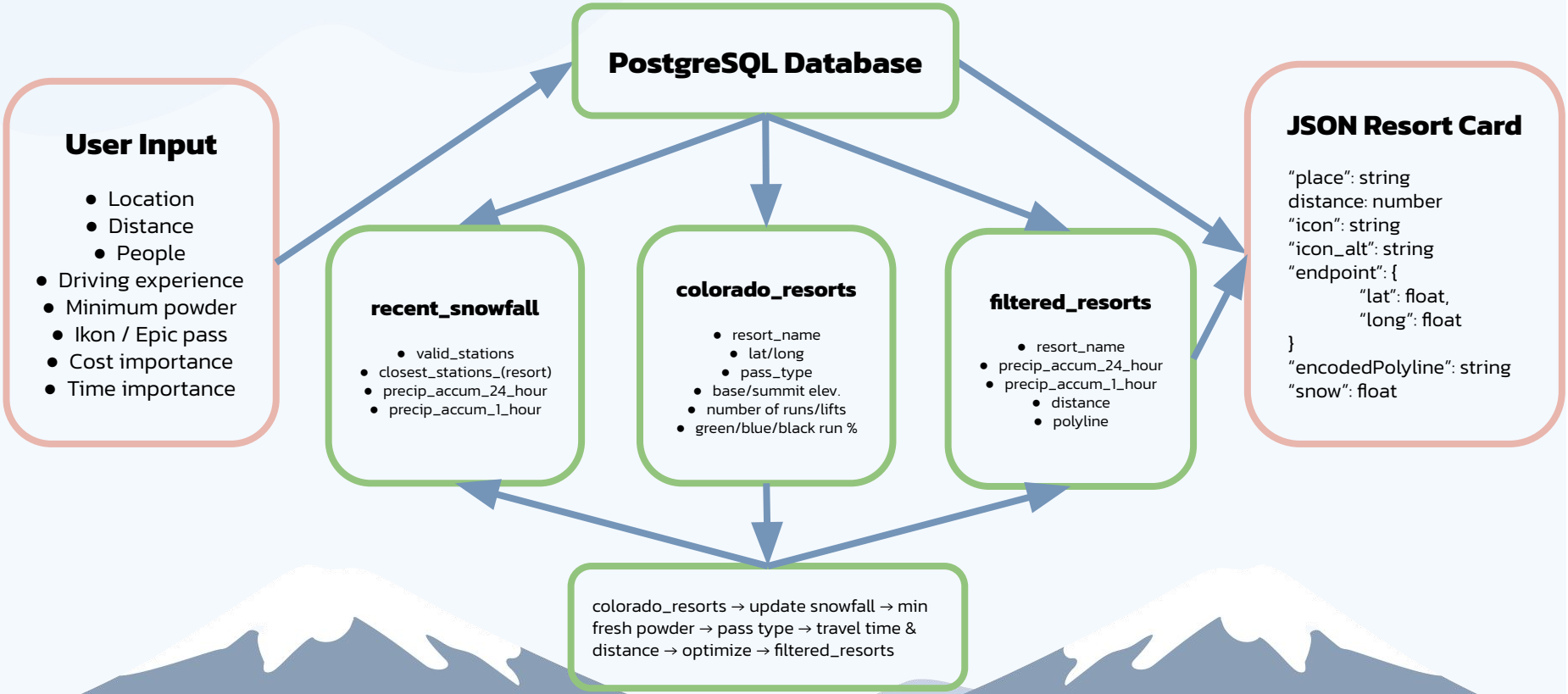
# System Architecture



# APIs Utilized

- Synoptic API (via OpenSnow)
  - Provides weather and snowfall information from many weather stations
  - Variables accessed: precip_accum_one_hour, precip_accum_24_hour
    - Used to filter resorts based on the user's minimum snowfall preferences
    - Used in optimization function for road conditions at start and end location
- Google Maps API
  - Polyline from start location to destination (ski resort) for frontend display (Mapbox)
- Traffic API
  - Travel time from start location to destination (ski resort)
- REST API with Flask as the implementation tool
  - Endpoint examples: /get_mountain, /get_all_resorts

# Data Flow Chart



# Optimization Function

Weighted sum preference-based optimization:

Objective function:

$$\text{maximize} \quad Z = w_1 x + w_2 y + w_3 z$$

Subject to:

1. Maximum time
2. Maximum budget

$$\text{maximize} \quad Z = w_1 x + w_2 y + w_3 z$$

| Variable | Definition | How we get it |
|---|---|---|
| Z | Total score | Calculated |
| $w_1$ | Snowfall weight | Predetermined |
| $w_2$ | Budget weight | User determined |
| $w_3$ | Time weight | User determined |
| x | Snowfall normalized | Snowfall at resort (From API) |
| y | Budget normalized | Function of fuel cost, # people, distance |
| z | Time normalized | Function of traffic data, snowfall at start and end points, driving experience |

# Mock Input: Time Constrained Example

**Your Name:**
Mukund

**Starting Location:**
Boulder, CO

**Driving Time (minutes):**
400

**People on the trip:**
2

**Max Budget ($):**
100

**Driving Experience:**
Intermediate

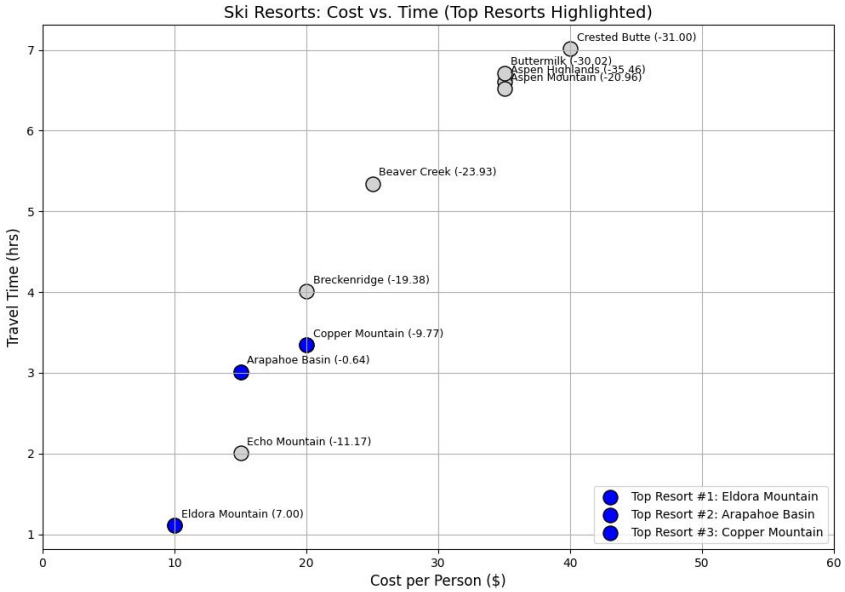
**Minimum fresh powder in 24 hours (inches):**
3

**Do you have an Ikon/Epic pass?**
- Ikon Pass
- Epic Pass
- Both Passes
- Willing to pay for a pass

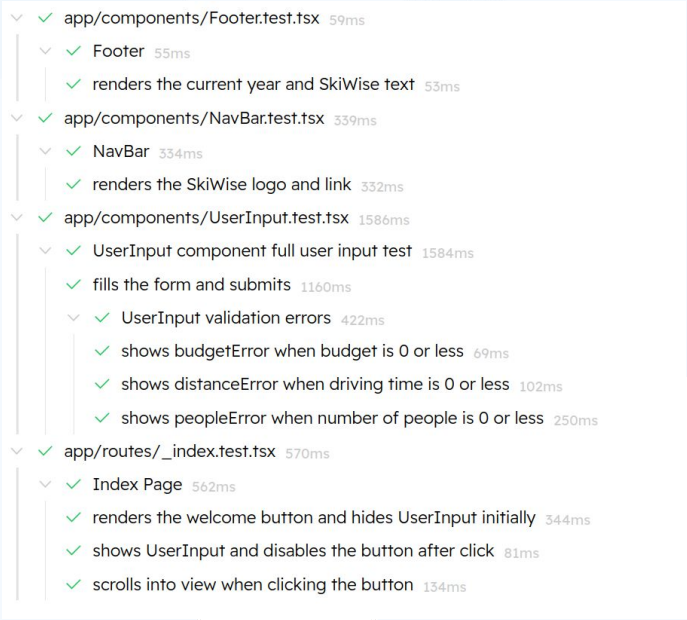
**How important is cost? (1-10)**
2

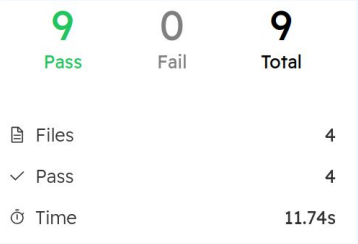
**How important is driving time? (1-10)**
8

Search for resort



# Frontend Testing: Vitest



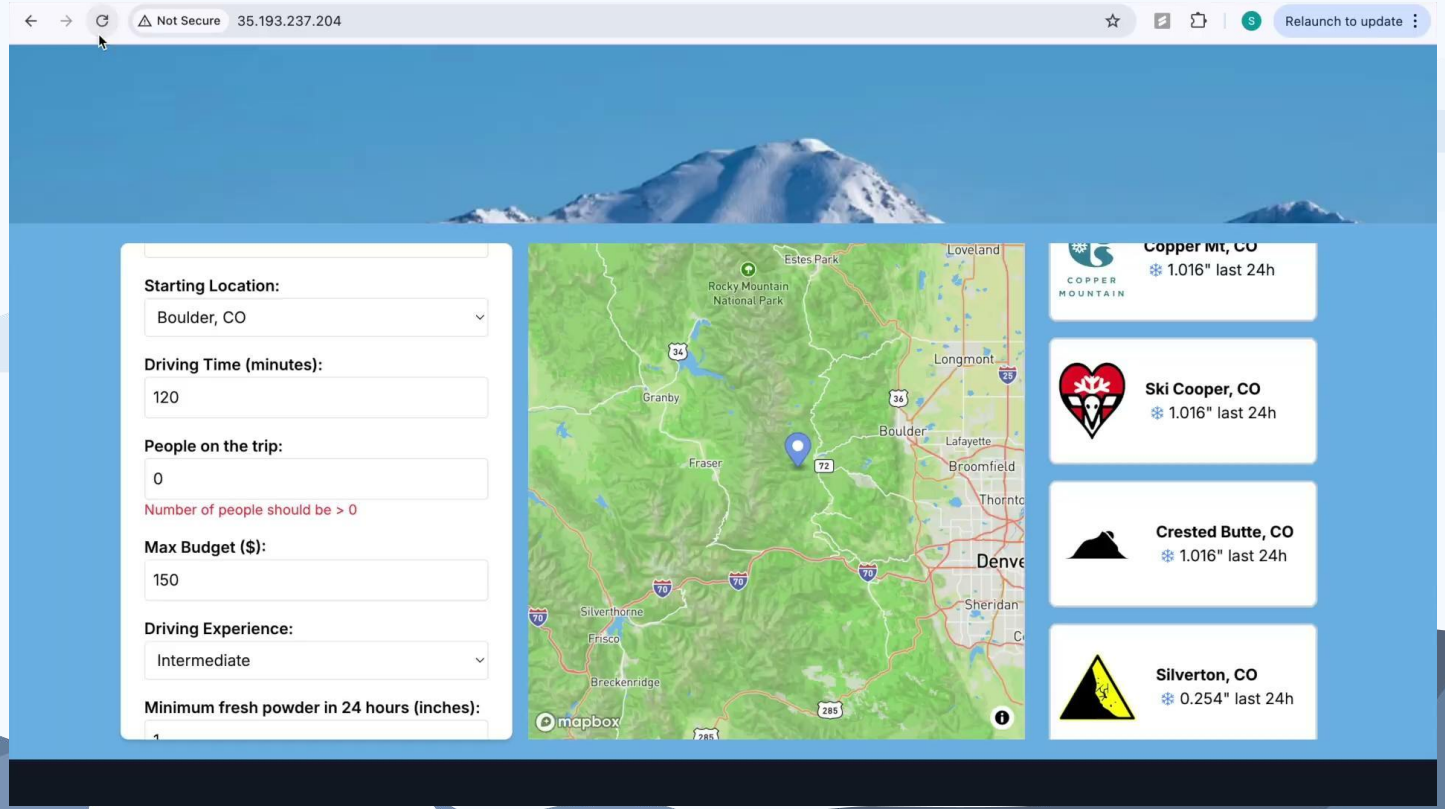




Tested:

1. User input: Input, error msgs, buttons
2. Index Page: render
3. Navigation Bar: render
4. Footer: render


DEMO!

# SkiWise Demo



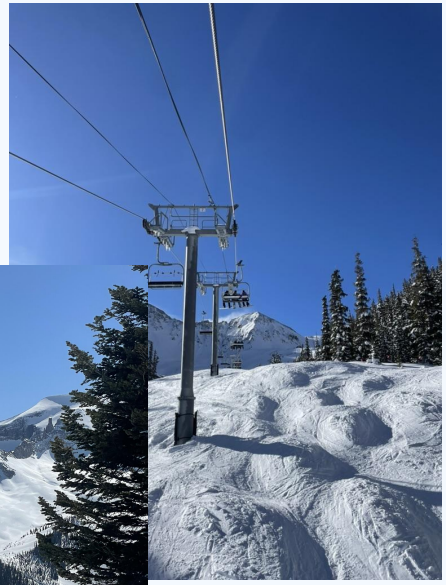
# Challenges

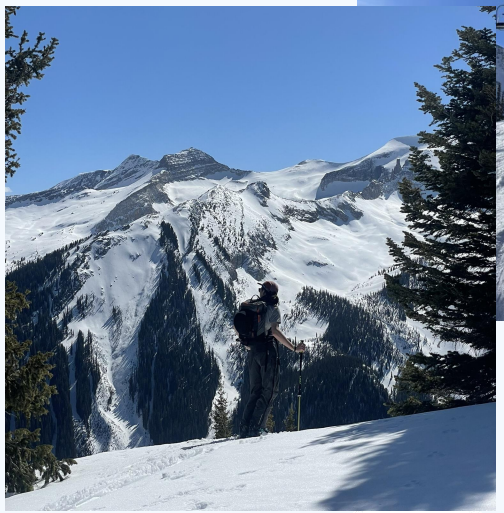
- Optimization of 1-hr and 24-hr snowfall data
    - API discrepancies and non-uniformity (i.e. weather station reporting)
- Error handling when no resorts meet user requirements
- Discrepancies in data formatting frontend vs backend and various sources of truth for data storage
- Privacy of API keys & deployment
- Optimization function:

    No way of capturing seasonal preferences (early, mid and late season)

# Future Work

- Expanding our reach
- Smarter Data & Predictions
    - Tune optimization function
- Learning from users





# CONCLUSION

Make the most of your day on the mountain!

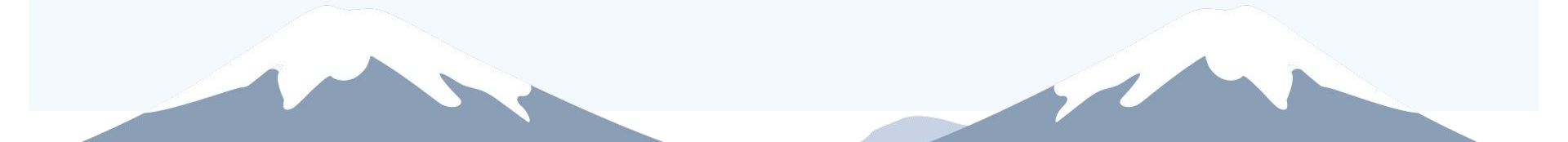
Helping users find the right resort

–> Mission: Effortless, personalized, and even more adventurous

Link: http://35.193.237.204/

# THANK YOU!

# Questions?



# Extras: Budget Function

car avg ↓ Colorado avg ↓ going & coming ↓ 10% ↓

$$y = \text{basefee} + \left\{ \frac{\frac{1}{\text{miles/gal}} \times \text{fuel cost} \times \text{total miles} \times \text{car maintenance}}{\text{people}} \right\}$$

$$y = \text{basefee} + \left( \frac{\frac{1}{20} \times (\$3) \times \text{total miles} \times 1.1}{\text{people}} \right)$$

# Extras: Time Function

$$z = 2 \times \left( \frac{\text{curvat time}}{3600} + \text{DEF}\left( \text{accidents} + \text{SFF}\left( SF_s + SF_e \right) \right) \right)$$

curvat time: API

DEF: UI

accidents: API*

SFF: Hardcoded

- beg − 0.1
- in − 0.05
- ex − 0.02

$SF_s$, $SF_e$: API