

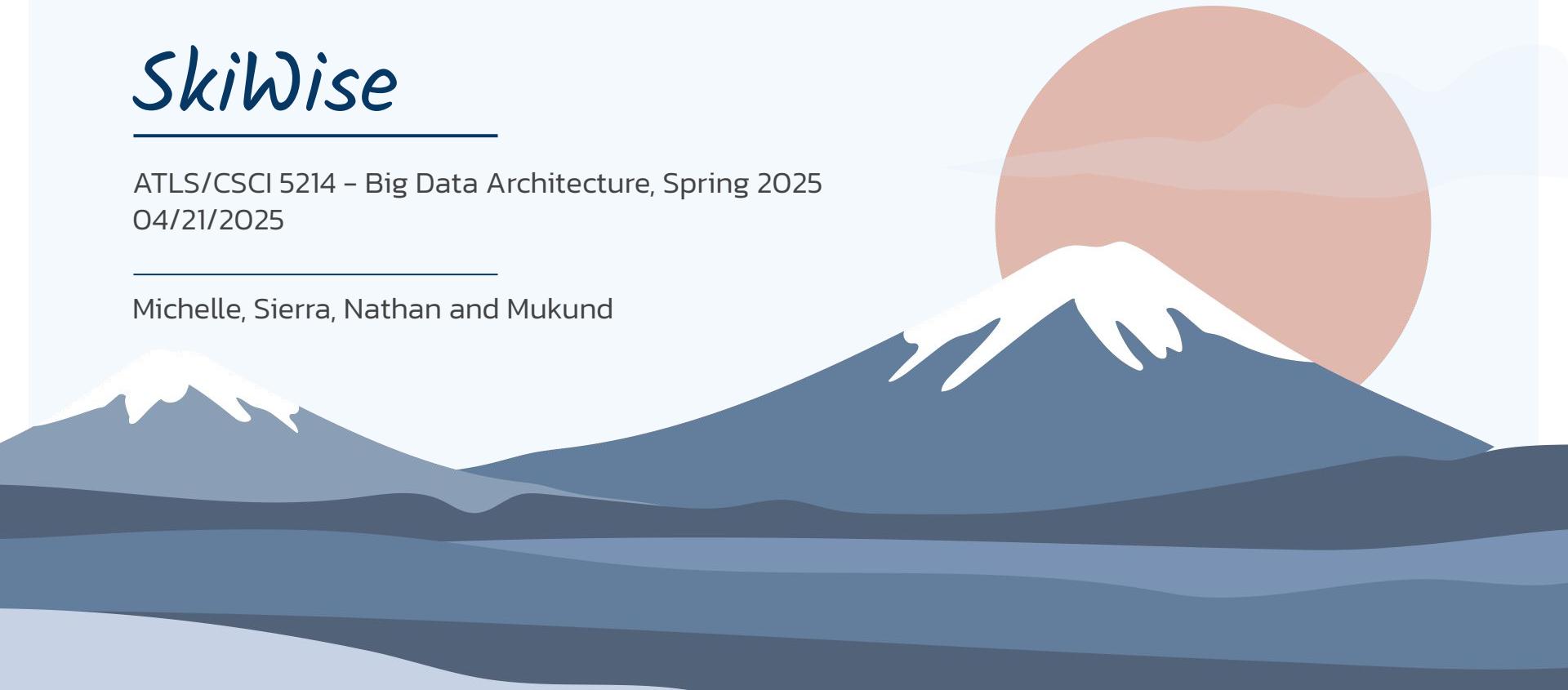
# SkiWise

---

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



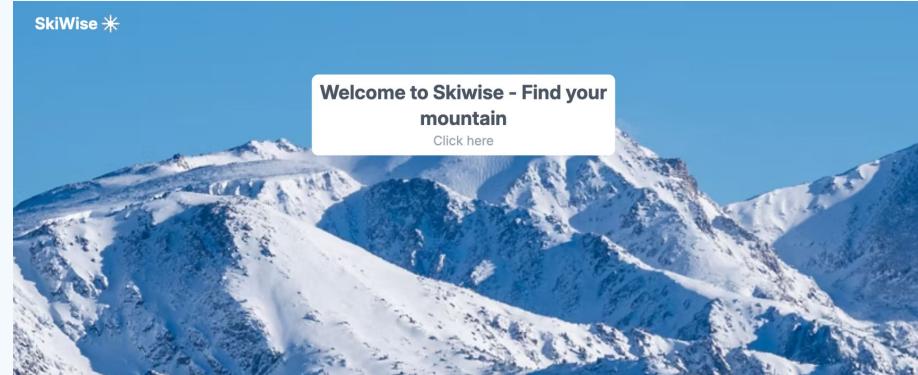
*Nathan Varghese*

Optimization and Testing  
MS AE – Robotics



# What is SkiWise?

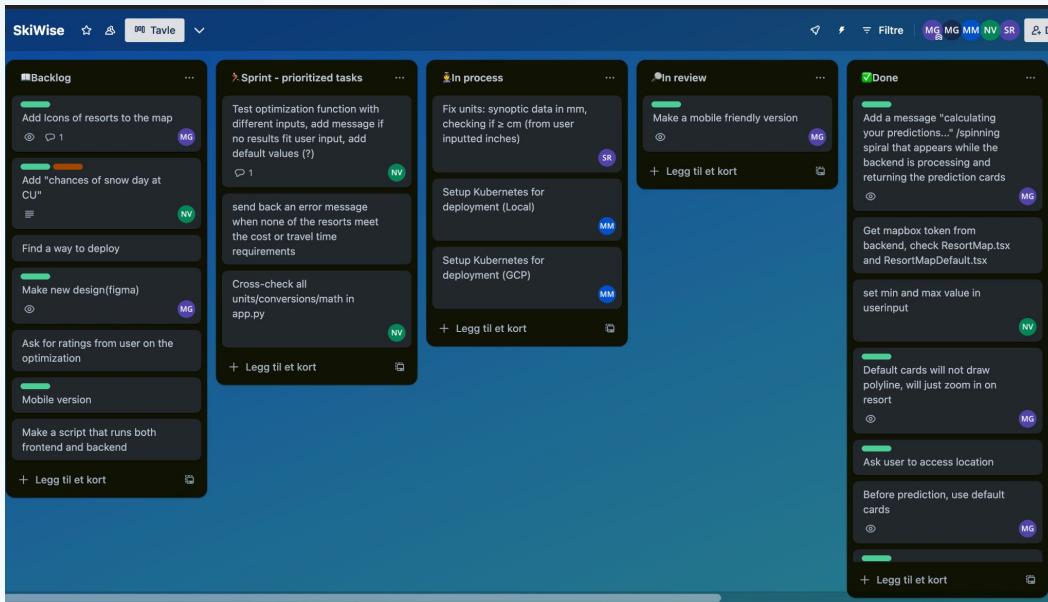
- Where should you go skiing? 🤷
- Finds based on real-time conditions
  - 1 hr and 24 hr snowfall, road conditions, travel time
- Takes into account user preferences



# *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

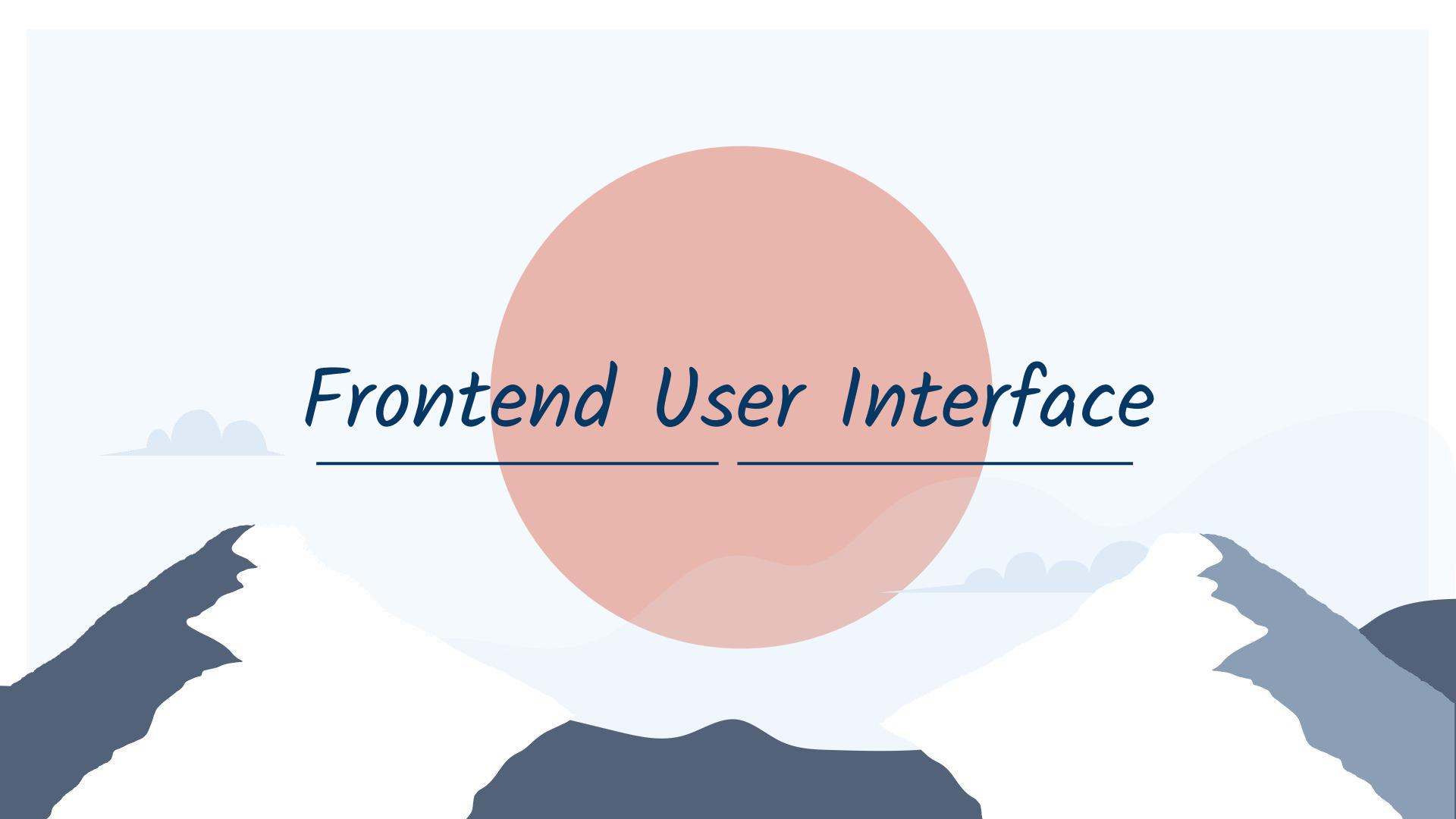
- Fits a small team

2 week long sprints

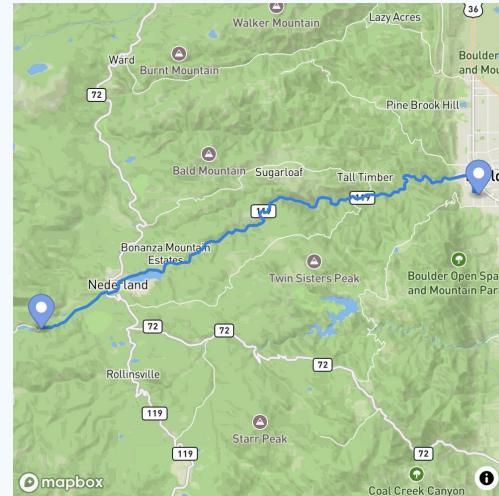
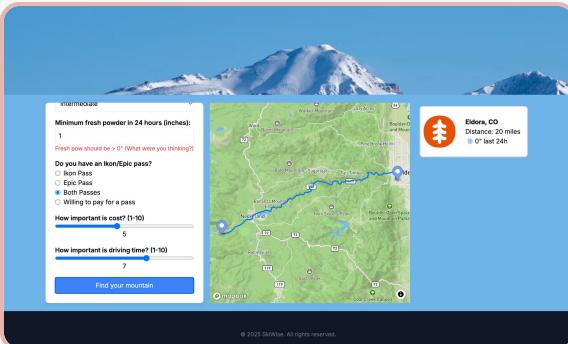
- Planning -> Retrospective

# *Frontend User Interface*

---



# Interactions with the user:



mapbox

The screenshot shows the SkiWise website. At the top, there's a header with the word "SkiWise ⛷" and a "Welcome to SkiWise – Find your mountain" message with a "Click here" button. Below it is a large image of a snowy mountain peak. On the left, there's a smaller map of Colorado with a blue route line and location markers for "Grand Junction" and "Durango". The main content area is titled "Best Results:" and lists three ski resort suggestions with icons and details:

- Silverton, CO**  
Distance: 370 miles  
0.254" last 24h
- Telluride, CO**  
Distance: 375 miles  
0.254" last 24h
- Wolf Creek, CO**  
Distance: 297 miles  
0" last 24h

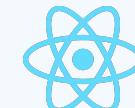
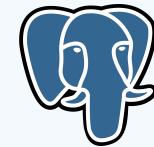
# SYSTEM ARCHITECTURE

---

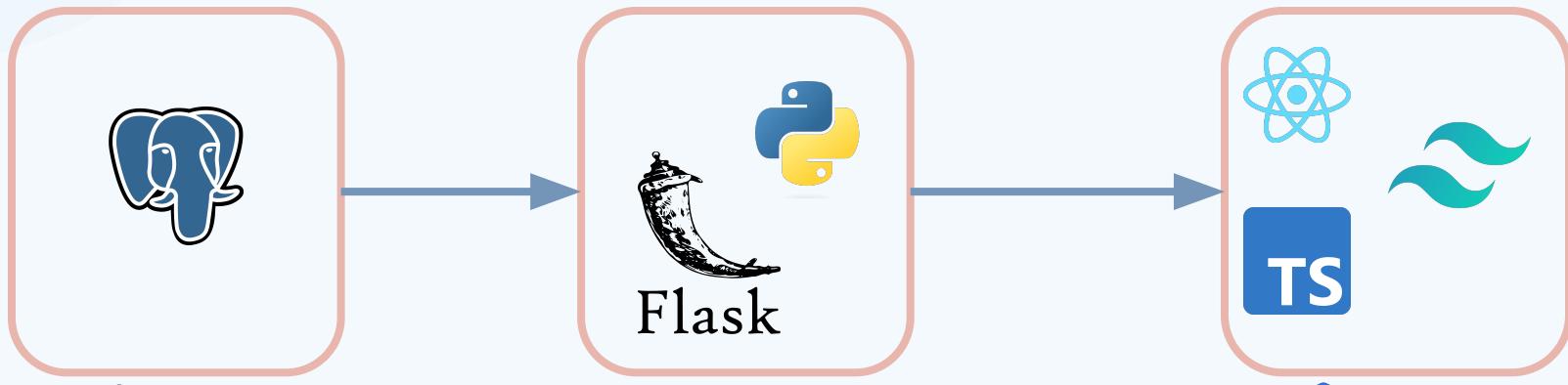
# Trello Technologies Leveraged & Tech Stack



- **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*

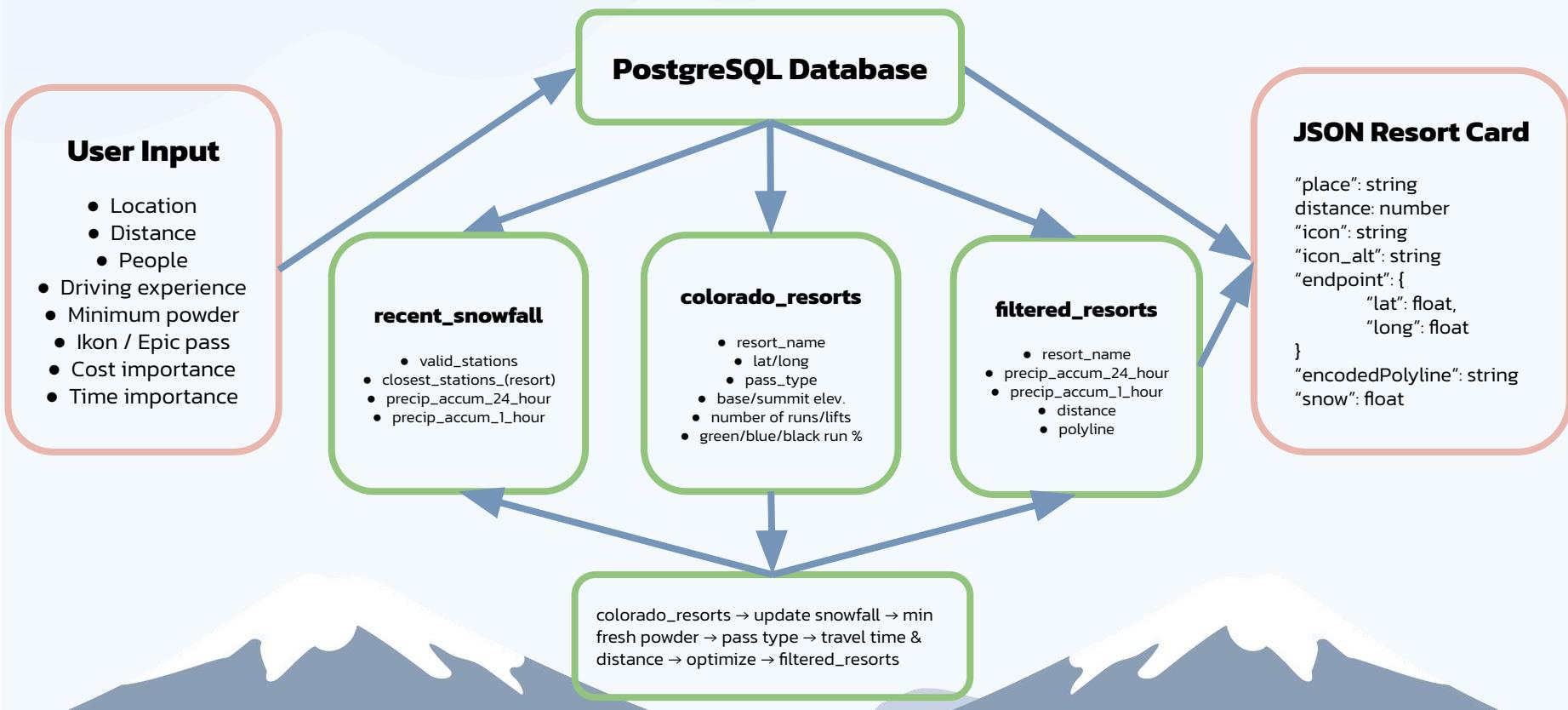


# APIs Utilized

- 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:

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

Subject to:

1. Maximum time
2. Maximum budget

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

Variable	Definition	How we get it
Z	Total score	Calculated
w <sub>1</sub>	Snowfall weight	Predetermined
w <sub>2</sub>	Budget weight	User determined
w <sub>3</sub>	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

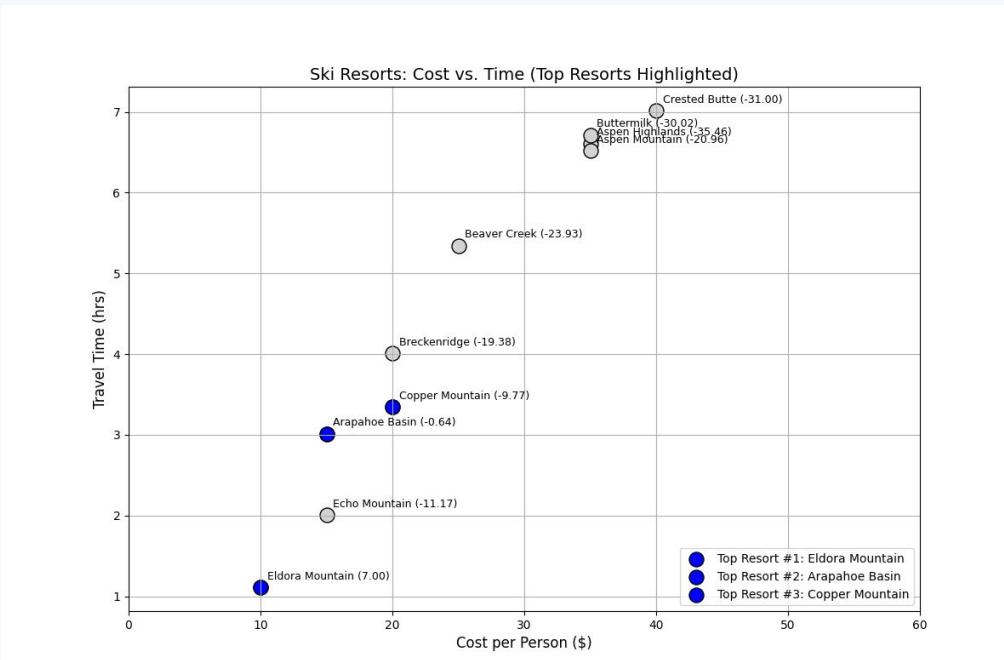
Do you have an Ikon/Epic pass?

- Ikon Pass
- Epic Pass
- Both Passes
- Willing to pay for a pass

How important is cost? (1-10)

How important is driving time? (1-10)

Search for resort



# Frontend Testing: Vitest

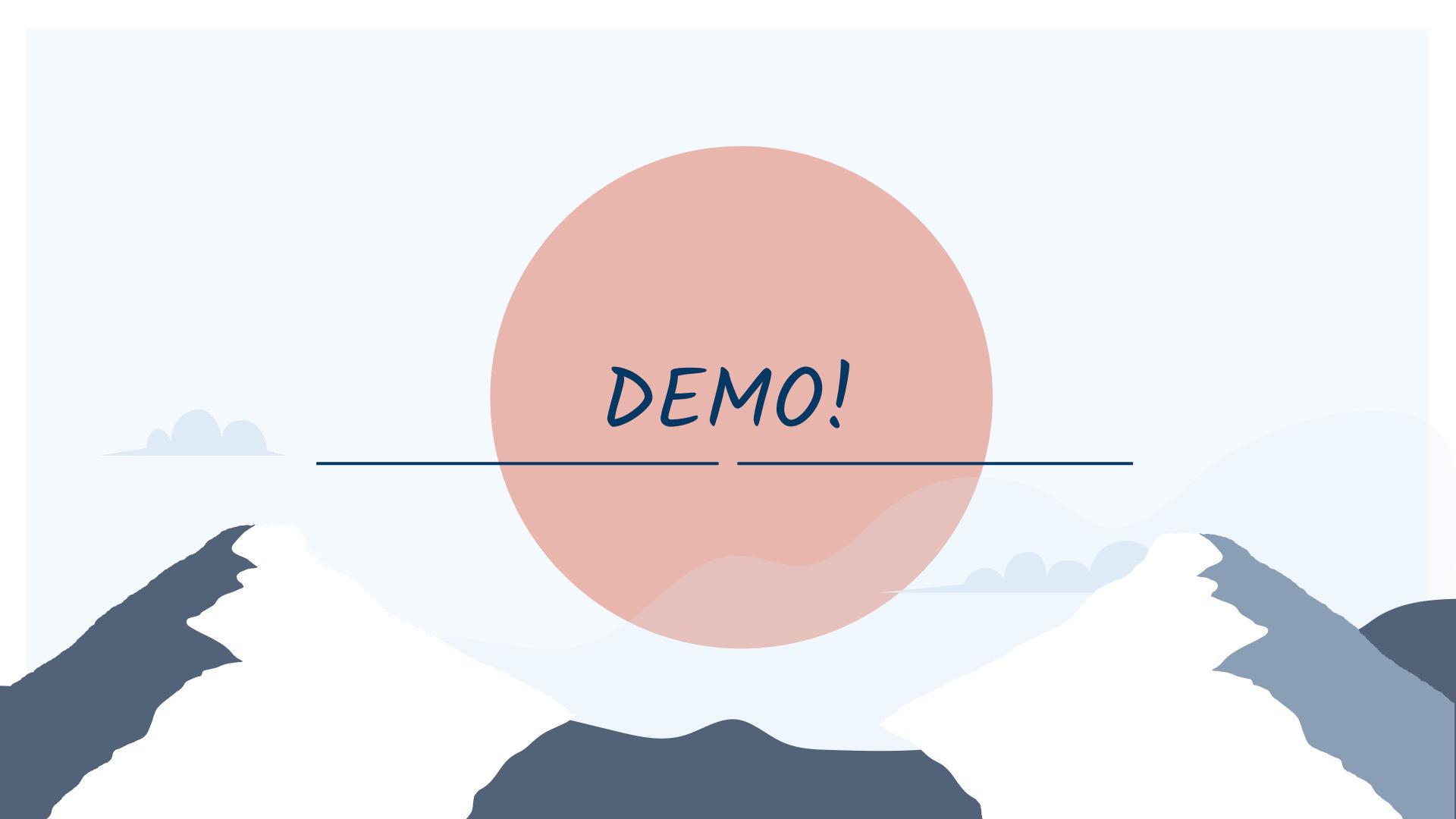
```
✓ app/components/Footer.test.tsx 59ms
  ✓ Footer 55ms
    ✓ renders the current year and SkiWise text 53ms
✓ app/components/NavBar.test.tsx 339ms
  ✓ NavBar 334ms
    ✓ renders the SkiWise logo and link 332ms
✓ app/components/UserInput.test.tsx 1586ms
  ✓ UserInput component full user input test 1584ms
  ✓ fills the form and submits 1160ms
  ✓ UserInput validation errors 422ms
    ✓ shows budgetError when budget is 0 or less 69ms
    ✓ shows distanceError when driving time is 0 or less 102ms
    ✓ shows peopleError when number of people is 0 or less 250ms
✓ app/routes/_index.test.tsx 570ms
  ✓ Index Page 562ms
    ✓ renders the welcome button and hides UserInput initially 344ms
    ✓ shows UserInput and disables the button after click 81ms
    ✓ scrolls into view when clicking the button 134ms
```

Tested:

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

9  
Pass  
0  
Fail  
9  
Total

Files	4
Pass	4
Time	11.74s



DEMO!

# SkiWise Demo

Not Secure 35.193.237.204 Relaunch to update

Starting Location:  
Boulder, CO

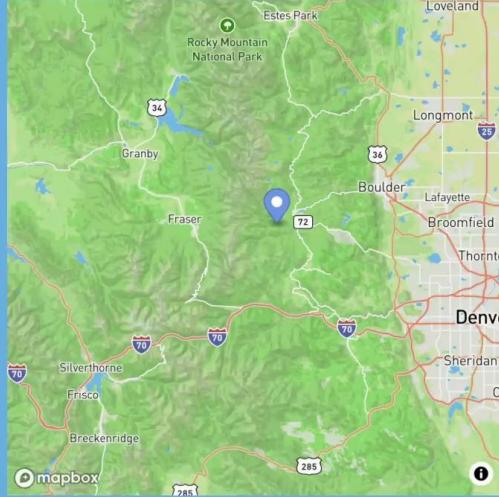
Driving Time (minutes):  
120

People on the trip:  
0  
Number of people should be > 0

Max Budget (\$):  
150

Driving Experience:  
Intermediate

Minimum fresh powder in 24 hours (inches):  
1



COPPER MOUNTAIN

Copper Mt, CO  
1.016" last 24h

Ski Cooper, CO  
1.016" last 24h

Crested Butte, CO  
1.016" last 24h

Silverton, CO  
0.254" last 24h

# *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

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

Annotations above the equation:

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

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



YUBULLYME

## Extras: Time Function

$$z = 2 \times \left( \frac{\text{current time}}{360^0} + \text{DEF} (\text{accidents} + \text{SFF} (\text{SF}_s + \text{SF}_e)) \right)$$

Annotations:

- API: Points to the first term  $\frac{\text{current time}}{360^0}$ .
- UI: Points to the second term  $\text{DEF} (\text{accidents} + \text{SFF} (\text{SF}_s + \text{SF}_e))$ .
- API\*: Points to the  $\text{accidents}$  term.
- Handcoded: Points to the  $\text{SFF} (\text{SF}_s + \text{SF}_e)$  term.
- beg - 0.1: Points to the value 0.1 below the  $\text{accidents}$  term.
- in - 0.05: Points to the value 0.05 below the  $\text{accidents}$  term.
- ex - 0.02: Points to the value 0.02 below the  $\text{accidents}$  term.
- API: Points to the  $\text{SFF}$  term.

