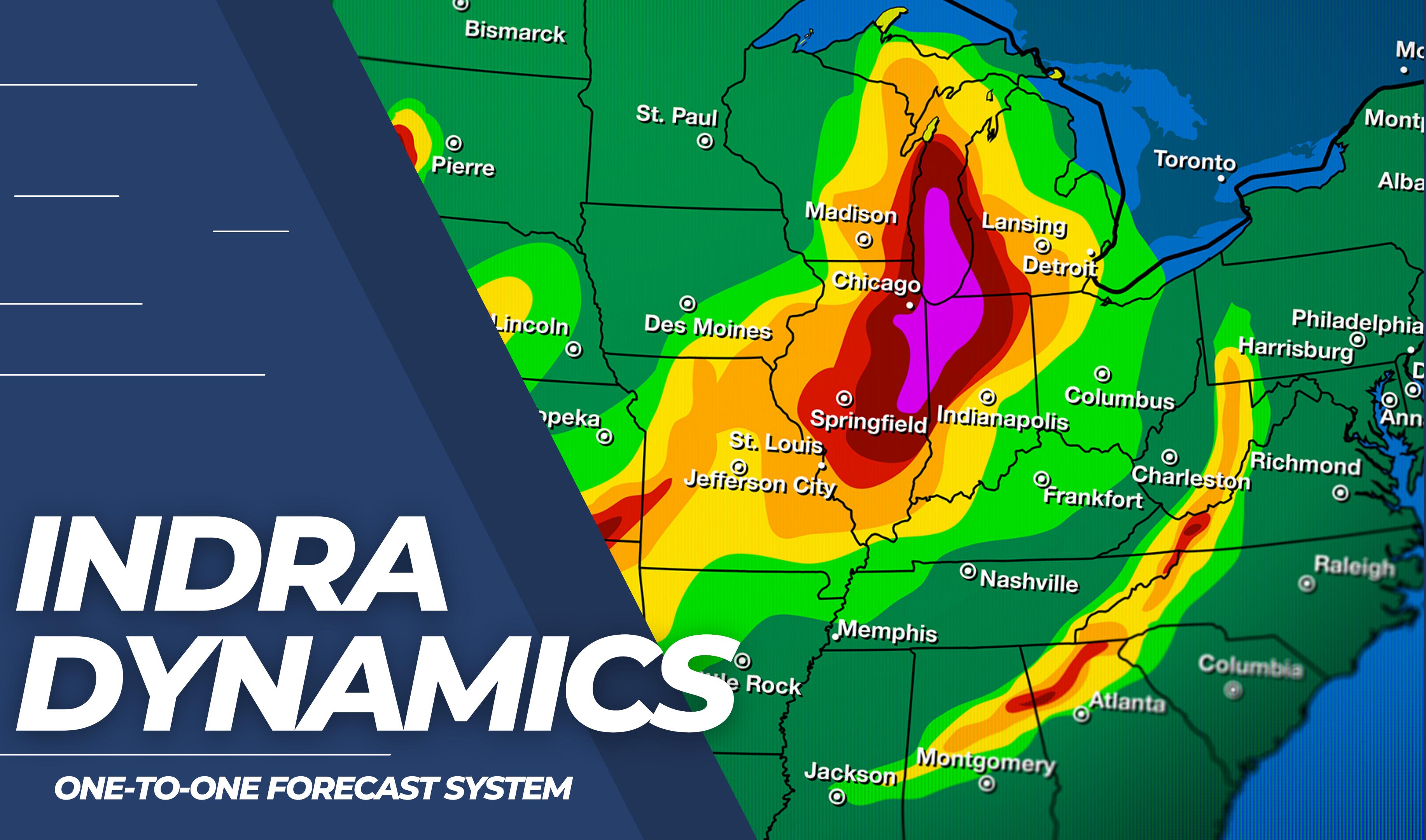


INDRA DYNAMICS®

ONE-TO-ONE FORECAST SYSTEM



OVERVIEW

01

Our Team

02

User Persona

03

Our Mission

04

Project
Overview

05

Subsystems

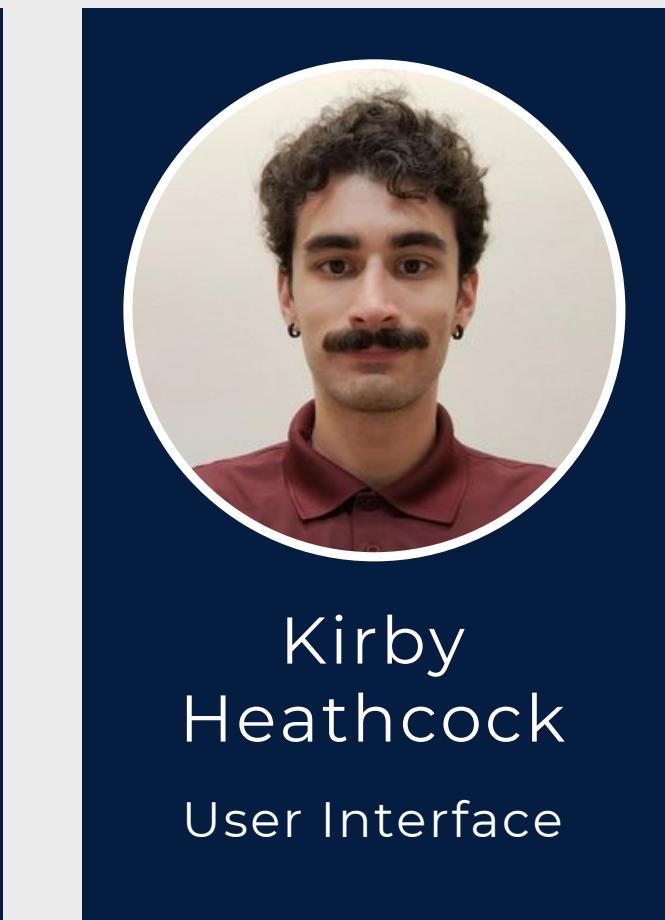
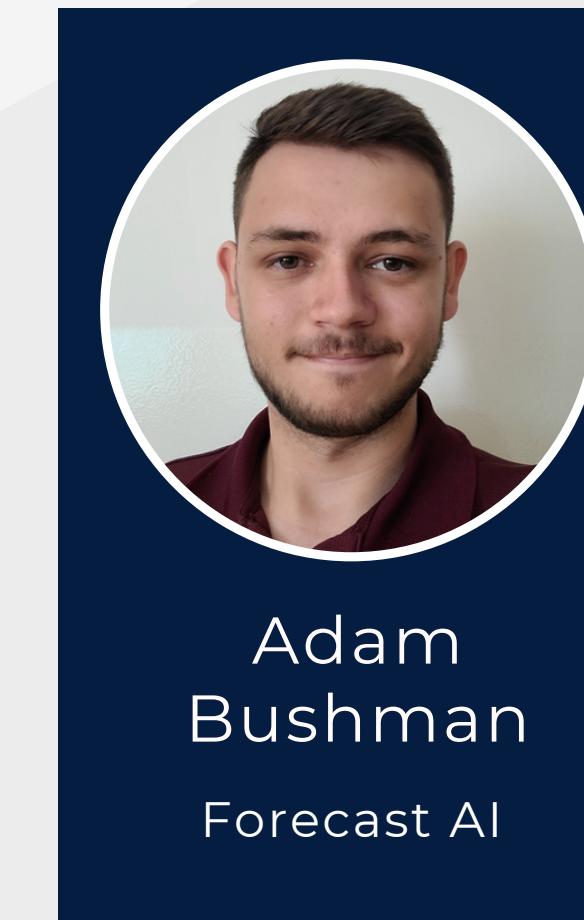
06

Future Work

07

Conclusion

OUR TEAM



ADVISORS



Dr. Phyllis J. Beck
-Advisor-
Computer Science
Artificial Intelligence



Dr. Jamie Dyer
-Consultant-
Meteorology
Climatology



Cameron Lindsey
-Consultant-
First Alert 6
Meteorologist



Jack

- **Age:** 47
- **Occupation:** Farmer
- Uses the system to plan for cutting hay and crop irrigation
- Lives an hour away from the nearest weather station



Jill

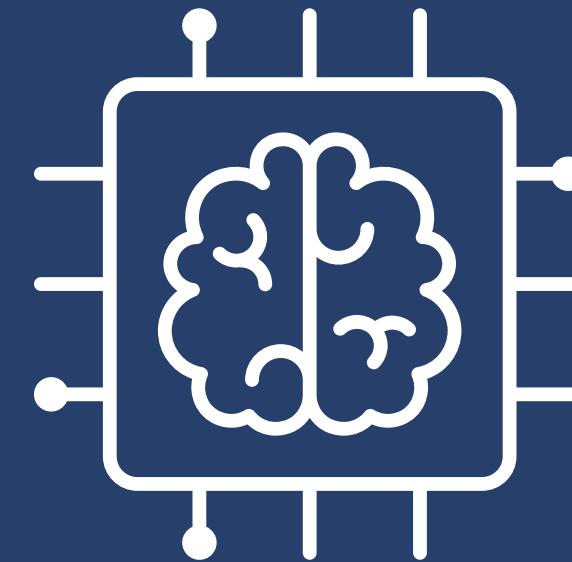
- **Age:** 25
- **Occupation:** Event Planner
- Uses the system to monitor the weather during city events
- Weather apps do not predict immediate weather

THE ADVANTAGE



Common Weather Apps

- Gives out raw forecast data
- No memory of past weather for a location
- Weather percentages suffer from lack of data resolution



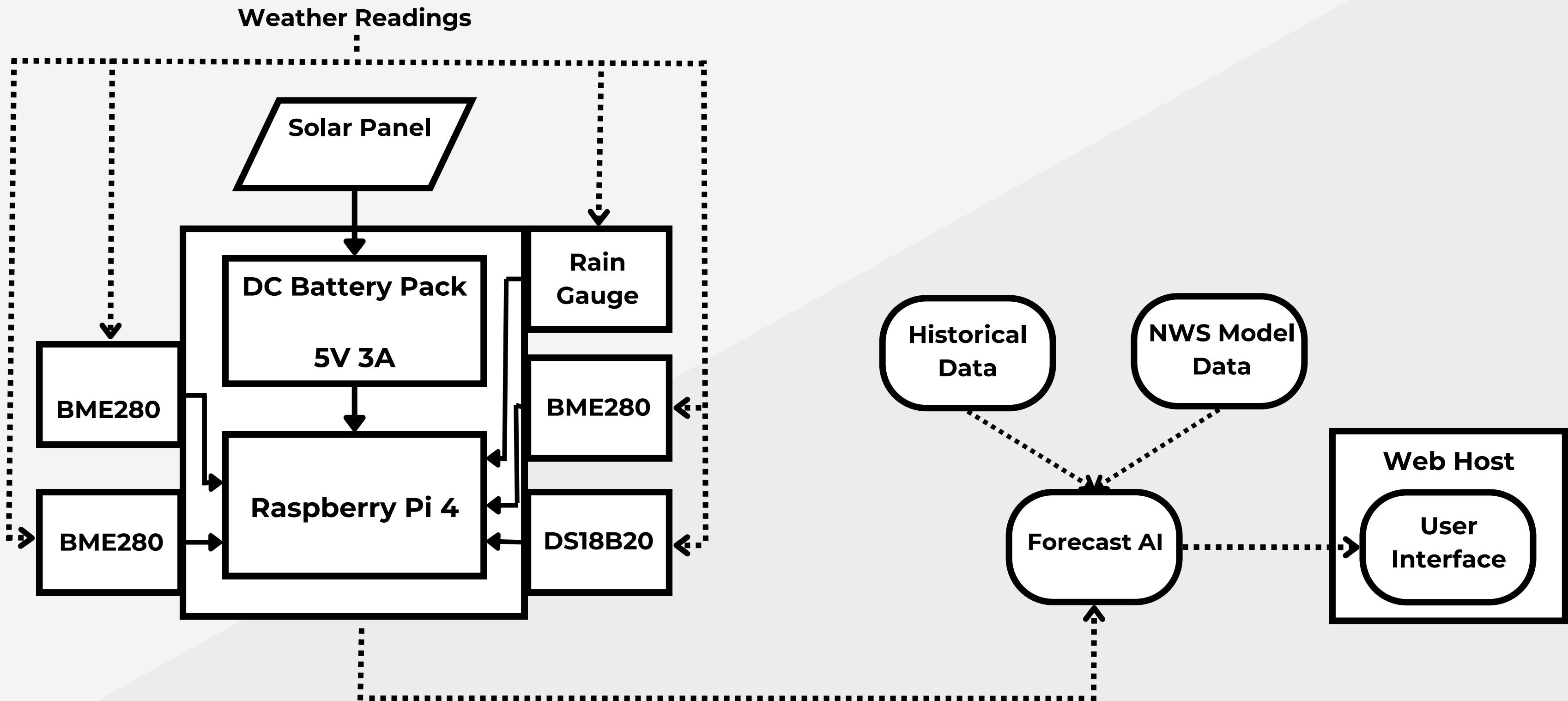
One-to-One Forecast System

- Emulates meteorologist intuition
- Trained by past weather patterns from your local area
- Data resolution enhanced by independent sensors

OBJECTIVE

We are giving consumers a
more accurate and
personalized alternative to
the common weather app

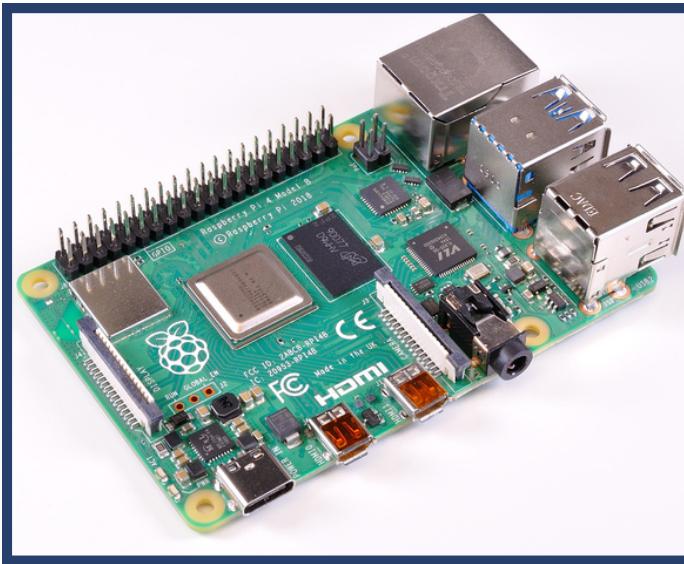
PROJECT OVERVIEW



STRUCTURE



Battery
2 LOVELEDI
15,000 mAh



Processor
Raspberry Pi 4
8 GB RAM

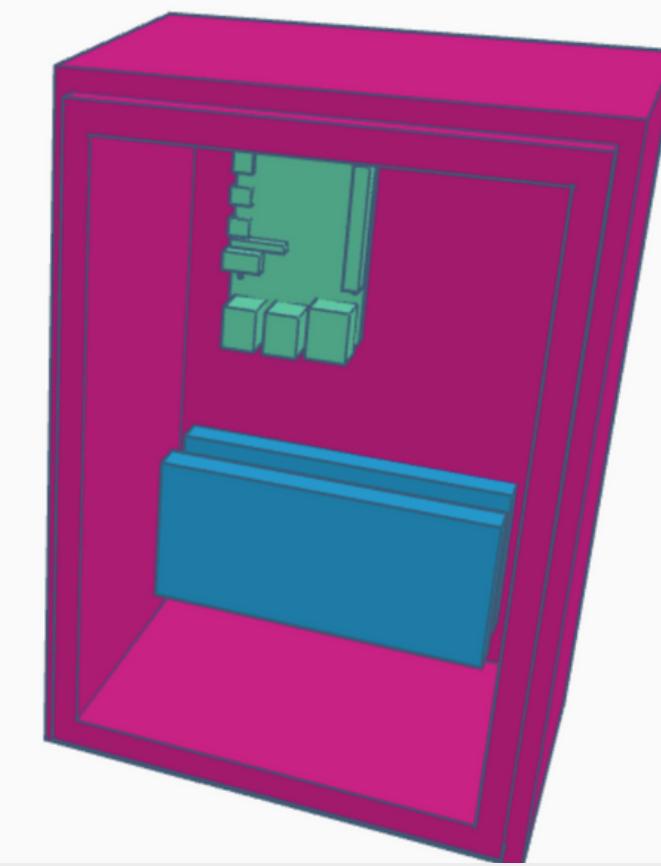
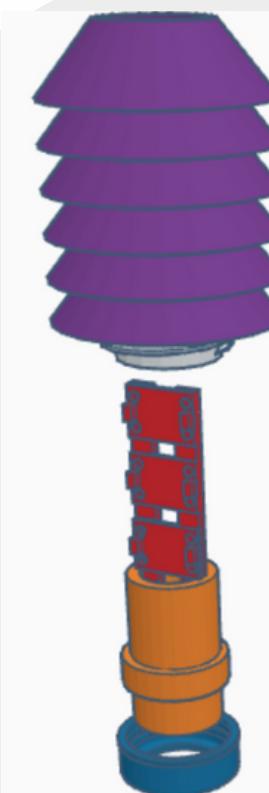


Solar
20W Solar
Panel

STRUCTURE



Enclosure
IP64, IK10
Rated Structure



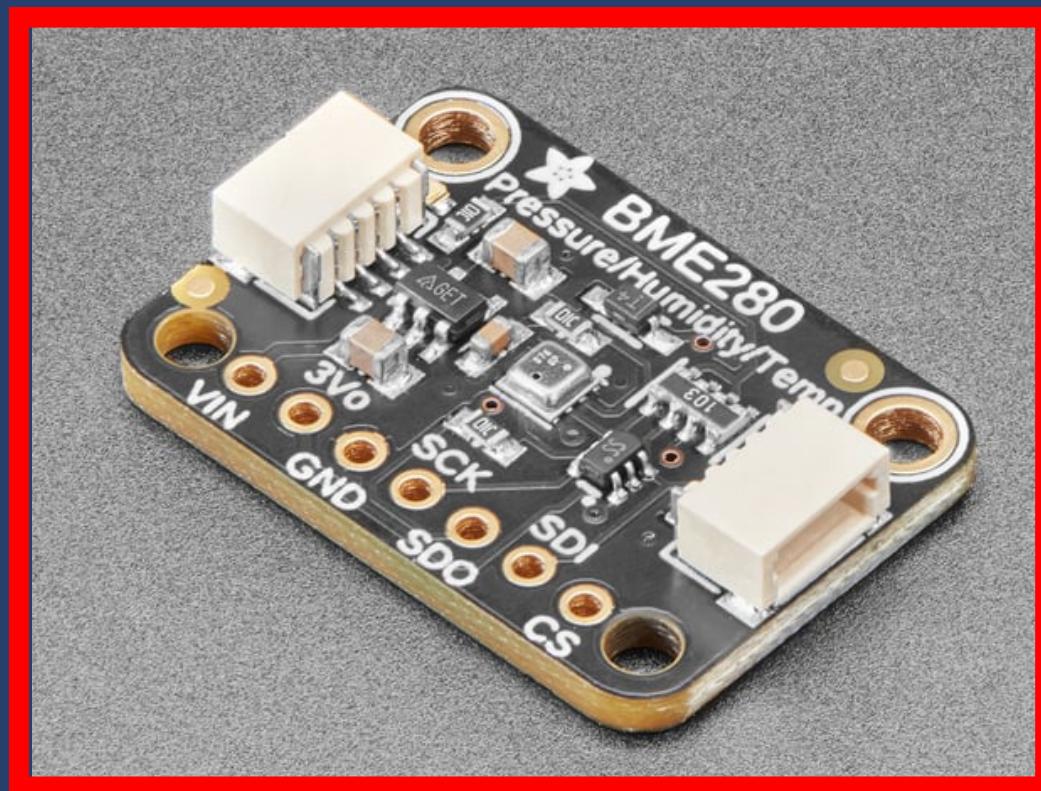
SENSOR NETWORK



Station Interior Sensor

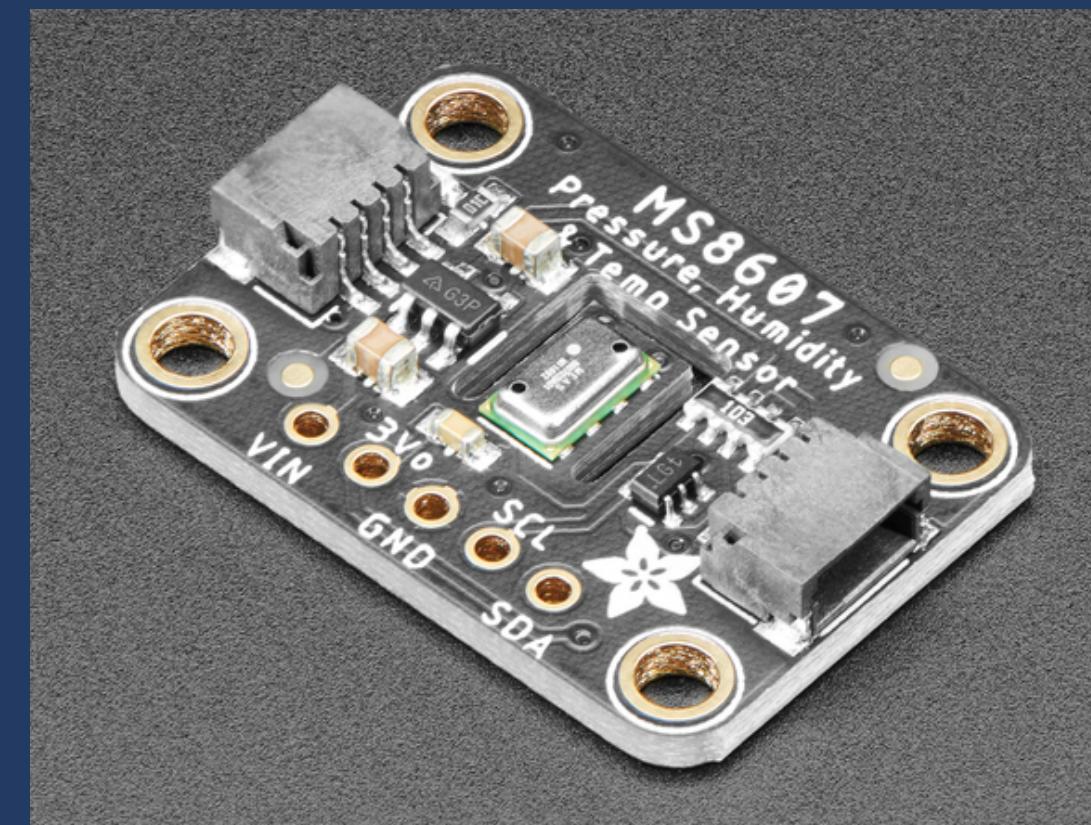
- Safety/ Performance
- DS18B20
- Temperature

SENSOR NETWORK

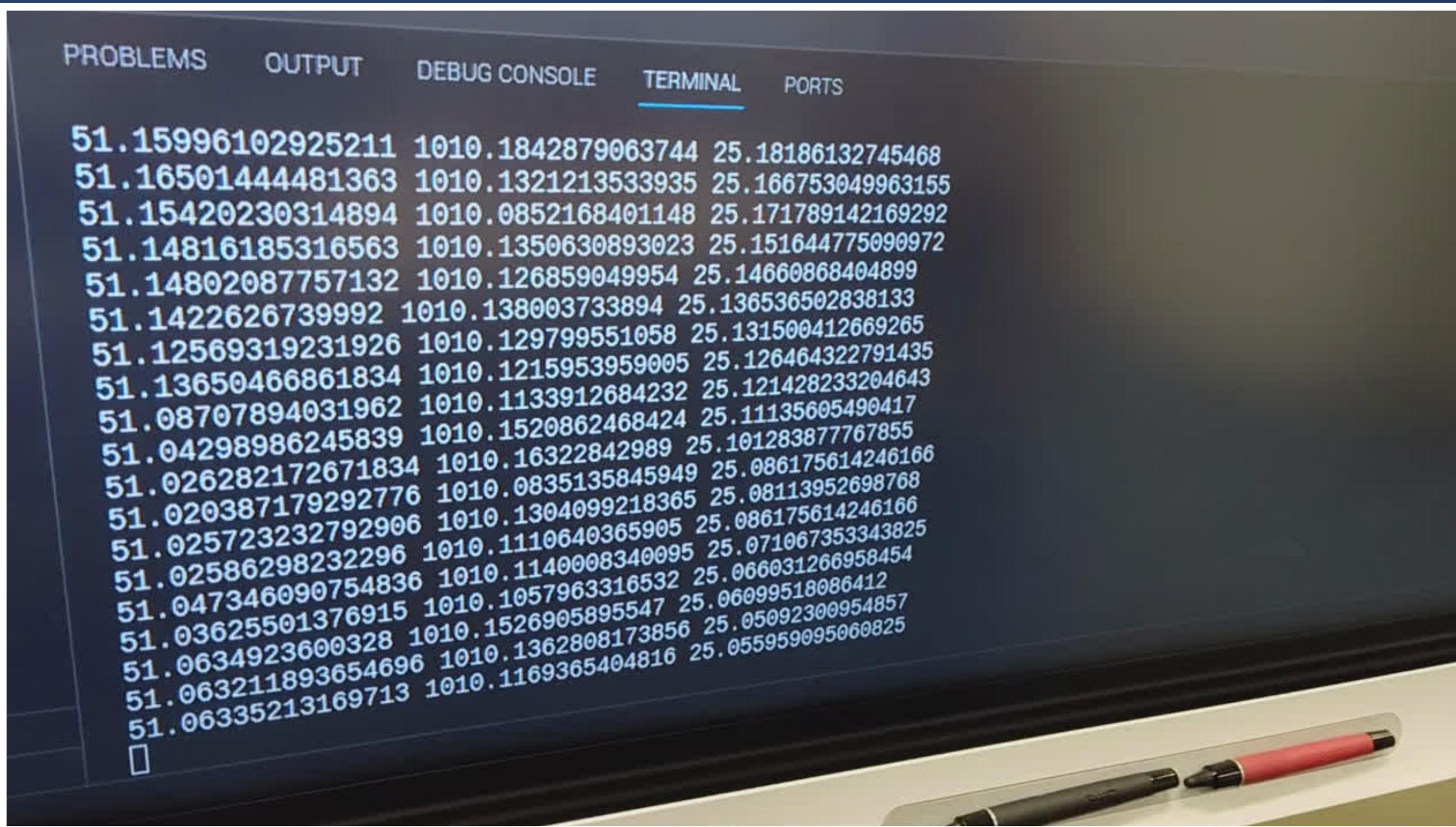


Atmospheric Sensors

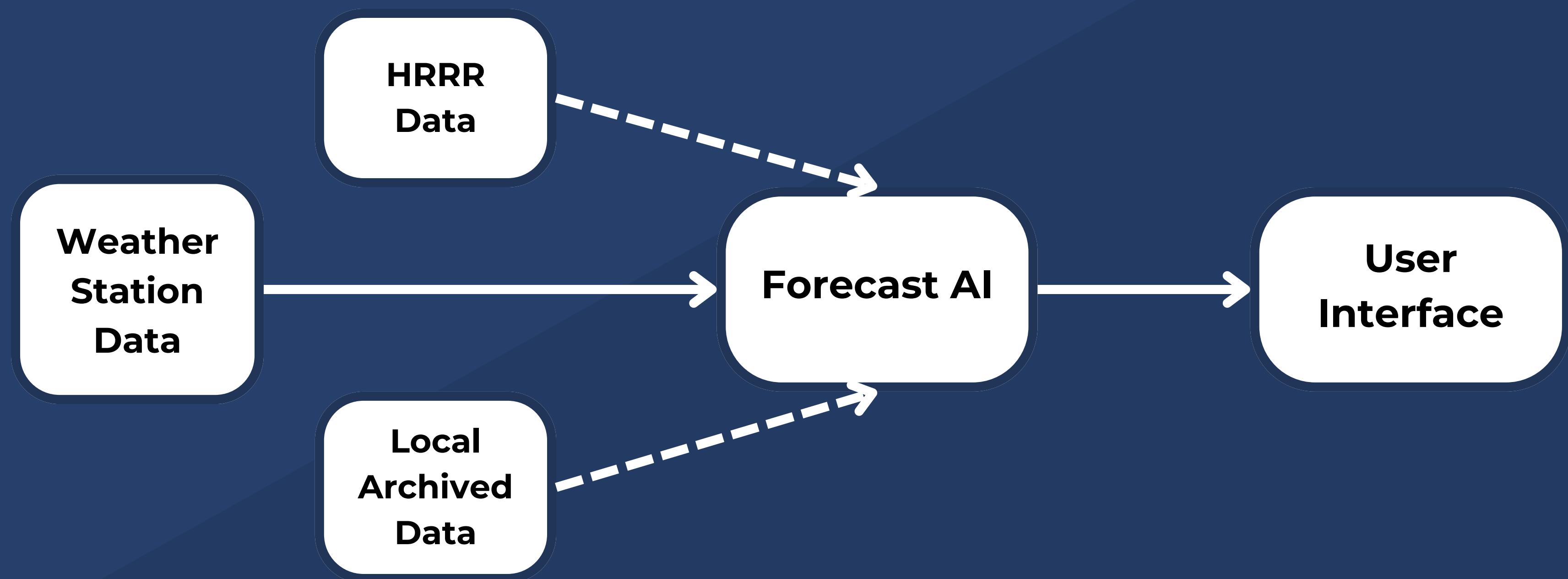
- Redundant Sensors
 - BME280
 - MS8607
- Temperature
- Pressure
- Humidity



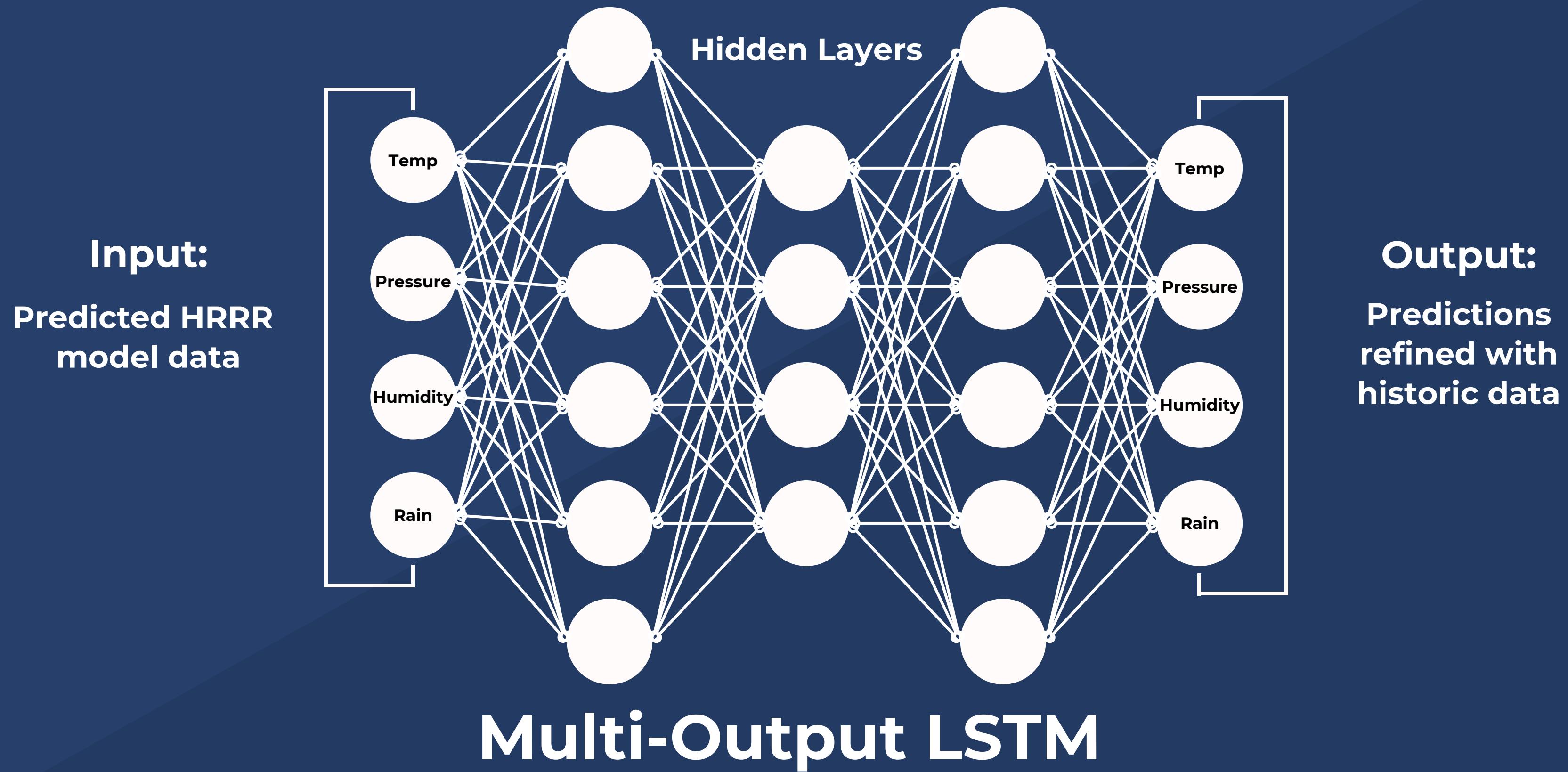
Sensor Network



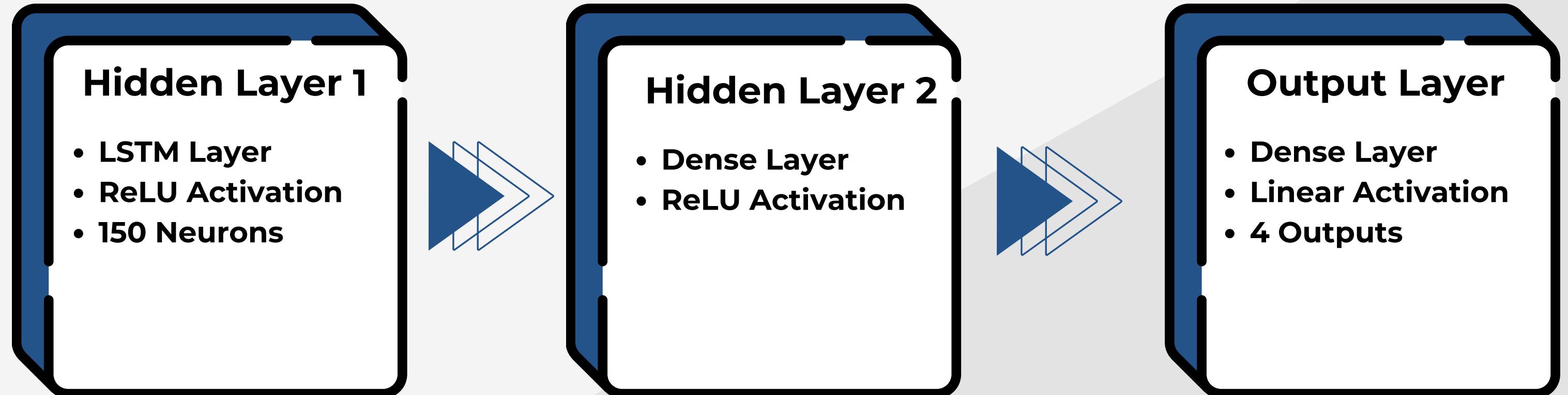
COMMUNICATIONS



FORECAST AI



FORECAST AI



Loss Function
Mean squared error

Data Normalization
Min-Max Scaler

Optimization
Adam optimizer

FORECAST AI

Training

- 80/20 training split
- 50 epochs

Loss:
0.00468

Validation Loss:
0.011

```
# neuralnetwork/presentation.ipynb - O: train
+ Code + Markdown | ⌂ Interrupt ⌂ Restart ⌂ Clear All Outputs ⌂ Go To ⌂ Variables ⌂ Outline ...
Cell ✓ 0s.

# Define the LSTM layers
lstm_ff = LSTM(150, return_sequences=True)(input_ff)
lstm_ff = LSTM(150)(lstm_ff)

Cell ✓ 0s.

output_ff = Dense(X_ff_train_normalized.shape[1], name='output_ff')(lstm_ff)

Cell ✓ 0s.

model = Model(inputs=input_ff, outputs=output_ff)

Cell ✓ 0s.

model.compile(optimizer='adam', loss='mean_squared_error')

Cell ✓ 0s.

# Train
model.fit(x_ff_train_reshaped, y_ff_train_reshaped, epochs=50, batch_size=32, validation_split=0.1)

Cell 0s.

Epoch 1/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0128
Epoch 2/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0159
Epoch 3/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0156
Epoch 4/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0137
Epoch 5/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0152
Epoch 6/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0135
Epoch 7/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0122
Epoch 8/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0111
Epoch 9/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0117
Epoch 10/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0112
Epoch 11/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0144
Epoch 12/50
395/395 [=====] - 0s 3ms/step - loss: 0.0061 - val_loss: 0.0118
Epoch 13/50
...
395/395 [=====] - 0s 3ms/step - loss: 0.0064 - val_loss: 0.0116
Epoch 38/50
395/395 [=====] - 0s 3ms/step - loss: 0.0064 - val_loss: 0.0119
Epoch 49/50
262/395 [=====] - ETA: 0s - loss: 0.0068
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output options.

Cell 0s.

x_ff_test_reshaped = X_ff_test_normalized.reshape((X_ff_test_normalized.shape[0], 1, X_ff_test_normalized.shape[1]))
y_ff_pred_normalized = model.predict(x_ff_test_reshaped)
y_ff_pred = y_ff_pred_normalized.reshape((y_ff_pred_normalized.shape[0], y_ff_pred_normalized.shape[1]))
y_ff_pred_inverse = scaler_y_ff.inverse_transform(y_ff_pred)

Cell 0s.

test_loss = model.evaluate(x_ff_test_reshaped, y_ff_test_reshaped)
print("Test Loss:", test_loss)
Cell ✓ 0s.

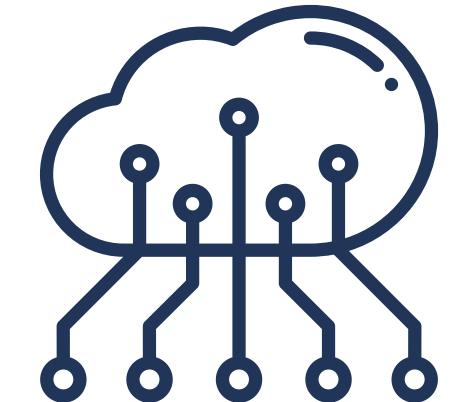
118/118 [=====] - 0s 936us/step
118/118 [=====] - 0s 1ms/step - loss: 0.0055
Test Loss: 0.005538416965484619
```

USER INTERFACE

**Information
Assessment**



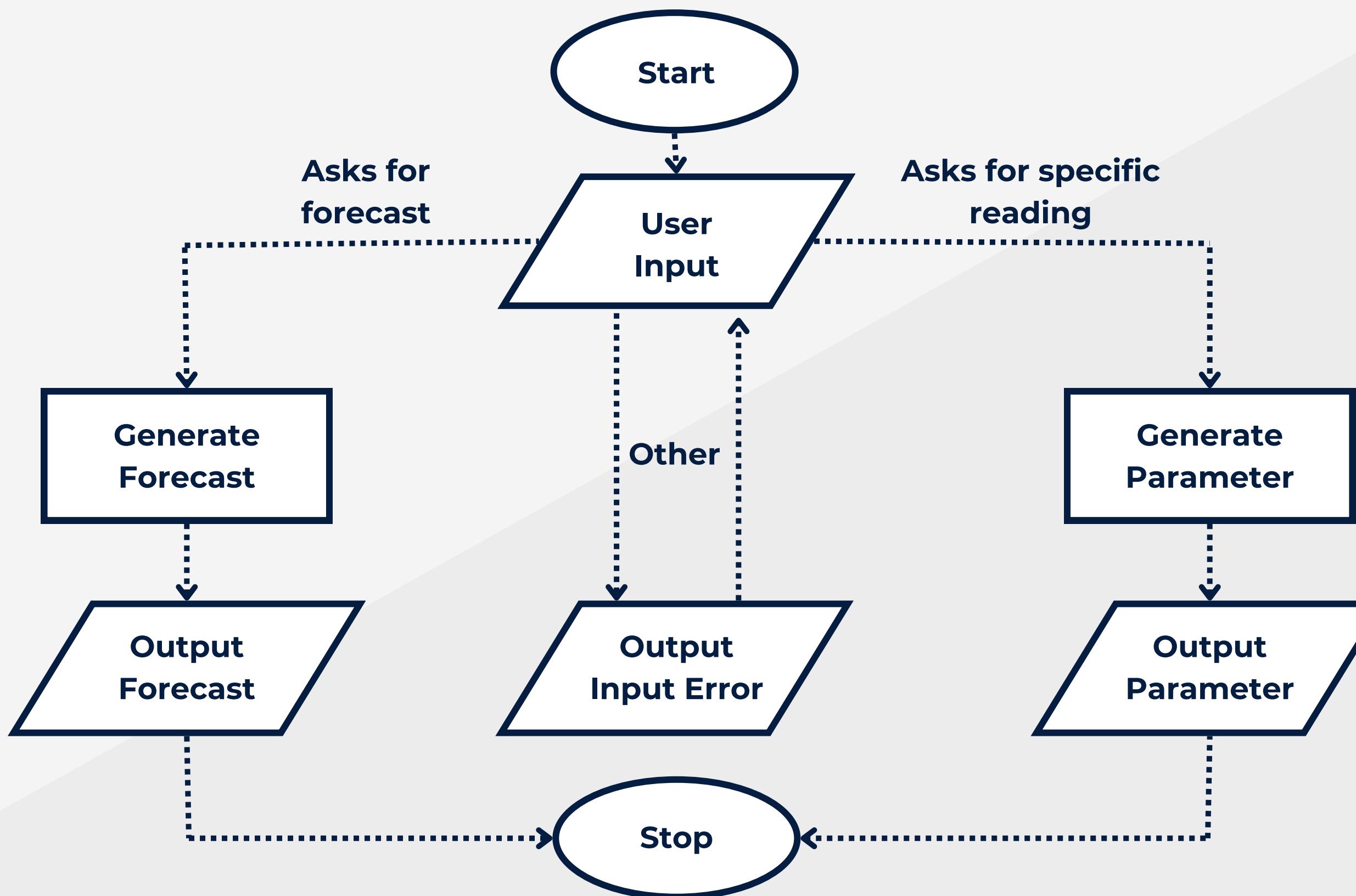
**Process and
Generation**



**Content
Delivery**



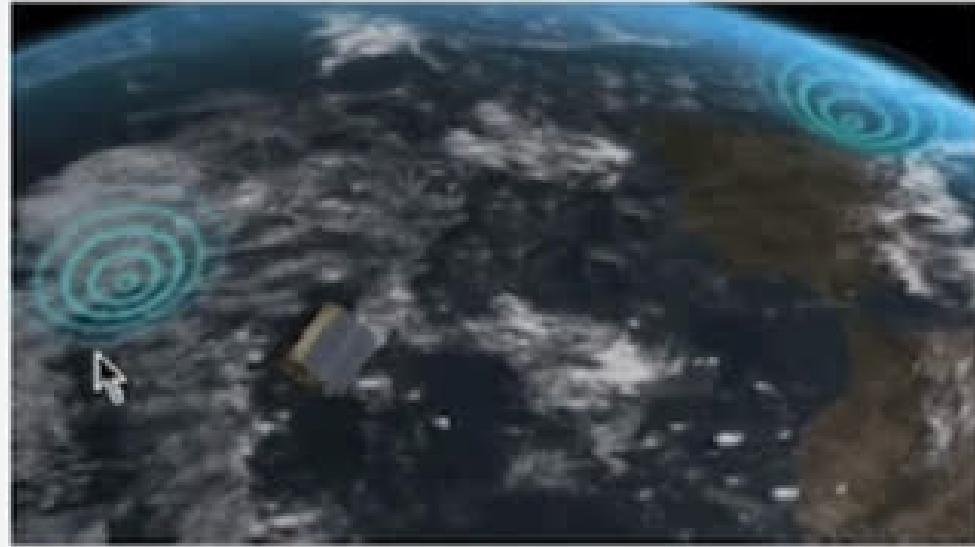
USER INTERFACE



USER INTERFACE



Nov 15, 9:43 AM



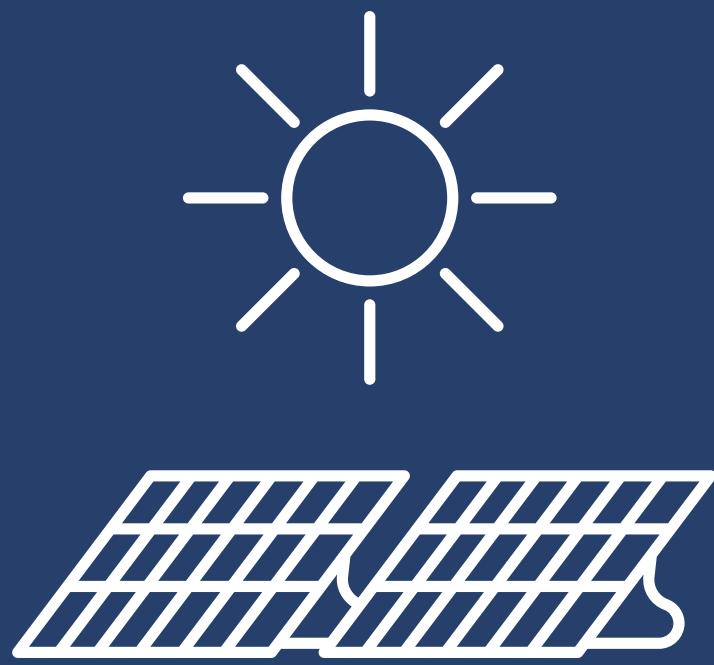
Hey there! I'm Dee, your friendly weather reporter. How can I help you today? ☀️ 🌧️



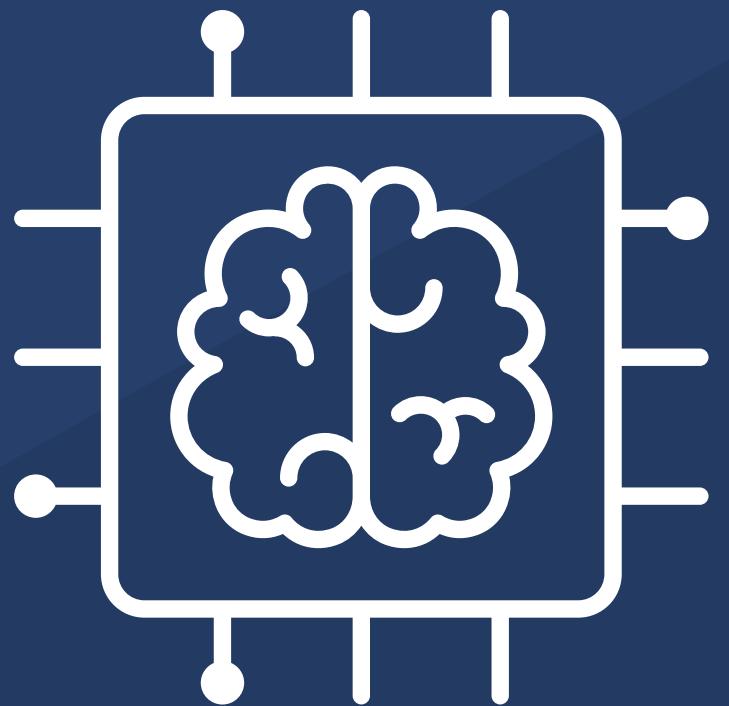
Hi|



FUTURE WORK



Solar Power

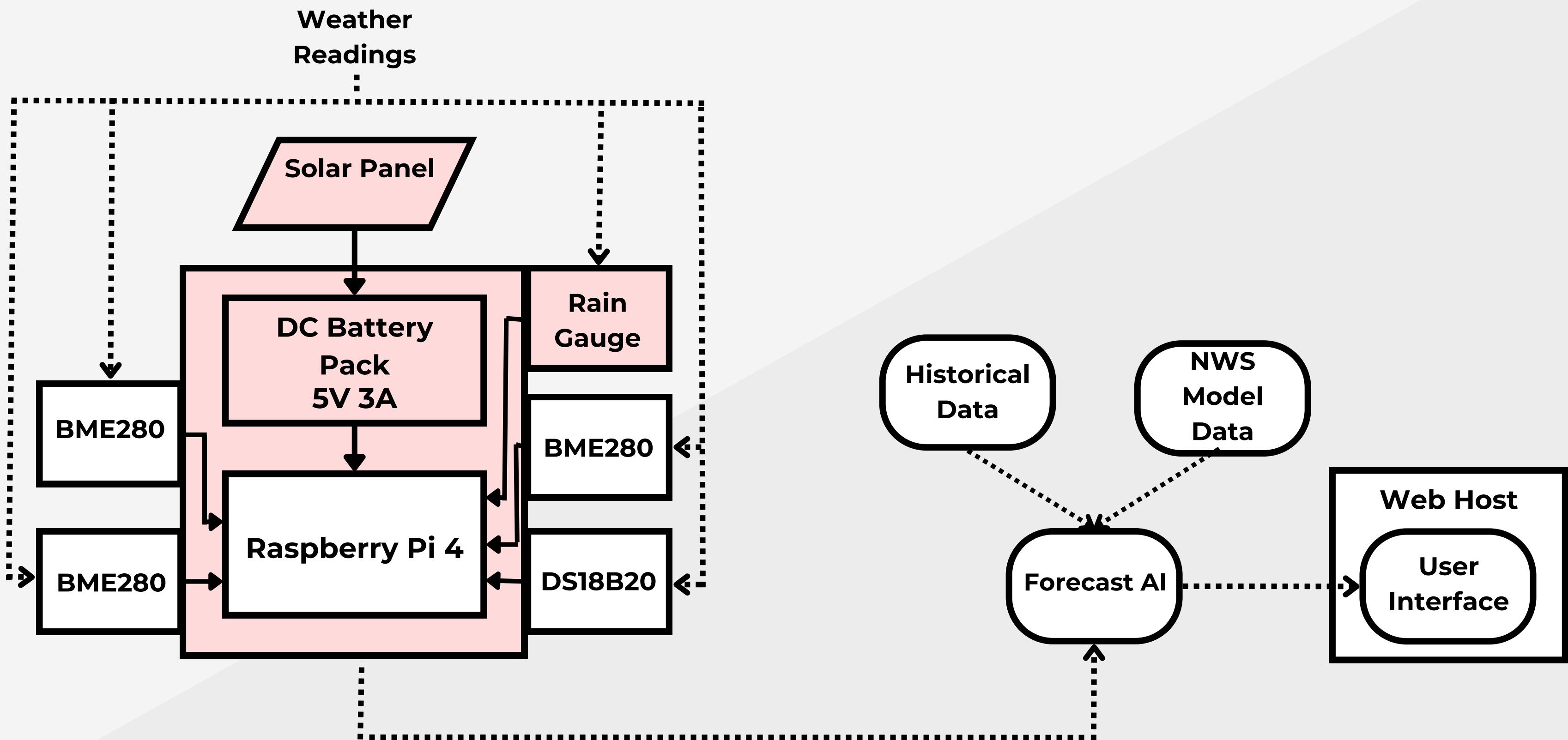


Onboard AI



Rain Gauge

FUTURE WORK



CHALLENGES

- Battery Amperage
 - Sensor Control Timing
 - Format of Weather Data
 - UI Data Reading Problems
-

CONCLUSION

**The One-to-One Forecast system is
making weather accuracy more
accessible and consistent.**

REFERENCES

- [1] Maxim, “DS18B20 Programmable Resolution 1-Wire Digital Thermometer,” DS18B20 datasheet, Mar. 2007 [Revised Apr. 2008].
- [2] Bosch, “BME 280 Digital Humidity, Pressure, and Temperature Sensor,” BME280 datasheet, Nov. 2014 [Revised Jan. 2022].
- [3] TE Connectivity, “MS8607-02BA01 PHT Combination Sensor,” MS8607 datasheet, Sep. 2015 [Revised Jun. 2017]

INDRA DYNAMICS

ONE-TO-ONE FORECAST SYSTEM

