Designing a Nowcasting System for Maritime 🚢

Maritime activities are always risky. Weather is one of several factors that cause maritime work and recreation to be risky and dangerous but forecasting the weather can help prevent accidents that lead to shipping and cargo losses, injuries, and even fatalities.

Weather can be difficult to predict, especially on waterways, but good forecasting can help ships and their crews navigate and make decisions that reduce risks. Bad weather can cause ships to capsize, to run aground, or to collide with other ships or objects. Knowing what kind of weather is coming is extremely important in making maritime activities safe.

Team 3 (The Avengers)

Name	NUID
Ankana Asit Baran Samanta	001007431
Sreepad Parigi	001023373
Parth Shah	001006181

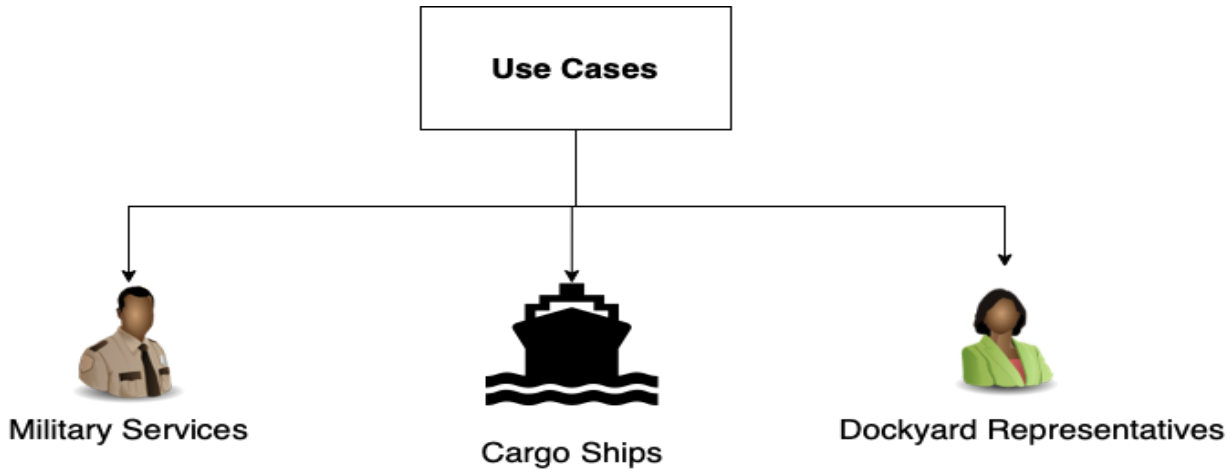
Identifying Use Cases

There are a lot of use cases in the Maritime domain which can be beneficial by using our system. So on this page we try to identify what are some main domains which will be eager to use our system and gain us maximum profits.

1) **Navy Services** : In September 1854, brig Porpoise was lost with at least 62 Sailors somewhere between Formosa and China due to a typhoon. During a powerful storm on 15–16 March 1889 in Apia, Samoa, the Navy lost three ships—Trenton, Vandalia, and Nipsic—by huge destructive waves called tsunamis

2) **Cargo Ships :** Japan's Transportation Safety Board (JTSB) issued an investigation report on the foundering of the cargo ship JIA DE due to heavy weather in Kanagawa Prefecture, which led to fatalities of 8 crew members, and loss of revenue in October 2019.

3) **Offshore facilities:** When it comes to offshore oil and natural gas production, extreme weather is always a risk. Nearly every year, a hurricane hits the Gulf Coast and causes massive damages to energy infrastructure. Unfortunately, 2020 has already had a record-breaking hurricane season, hitting energy companies extremely hard. So a weather intelligence system can really help offshore operations. Weather intelligence can help in decision making to keep the operations team safe, minimize equipment damage. The prediction data can be used to make defensible decisions about when to stop the work or when to evacuate.



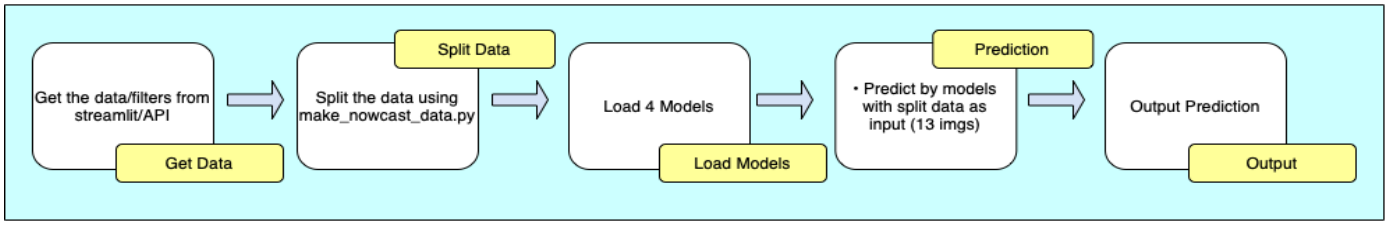
Architecture Breakdown

We have divided the pipeline into the following parts:

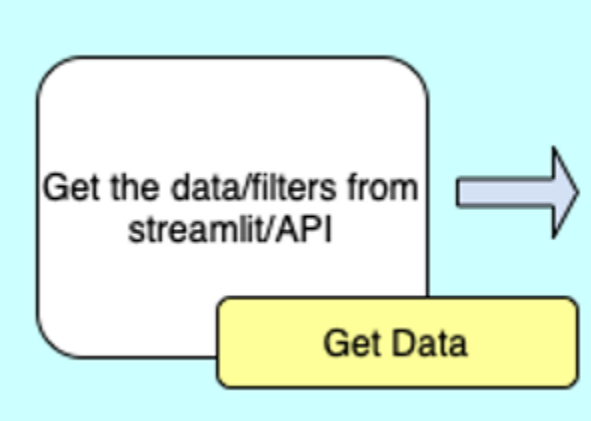
1. Model Pipeline
2. API Pipeline
3. Webapp Pipeline

Model Pipeline

The model pipeline is the process flow of the forecasting model to predict the forecast of the 13 images as input resulting in 12 images as output. Here we are using Nowcasting for prediction of the output images.

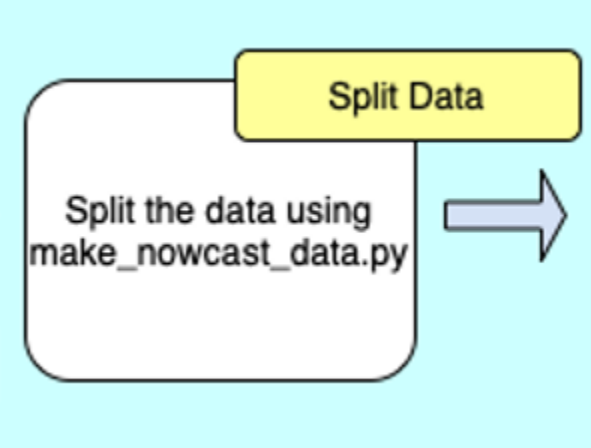


1. Get Filters from Webapp/API/Parameters



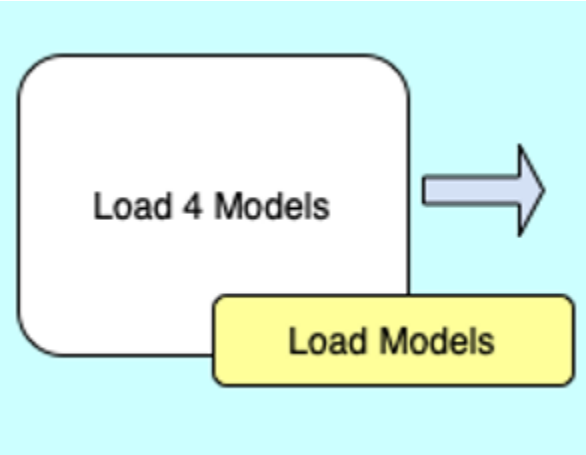
The first step in the Model pipeline will be to get the Data or filters for the specific Location and the month from the webapp Application for weather forecasting. This input will help in getting only those specific H5 files from the SEVIR data.

2. Generate the Test Data



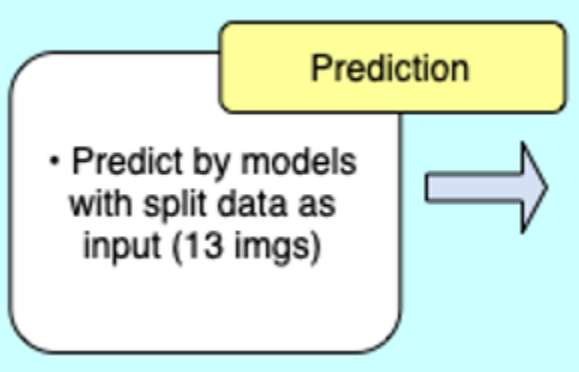
We now need to run the make_nowcast_data.py and make_synrad_data.py file to generate the test data which will be used for running the models. The 13 images will be generated as input here.

3. Load Models



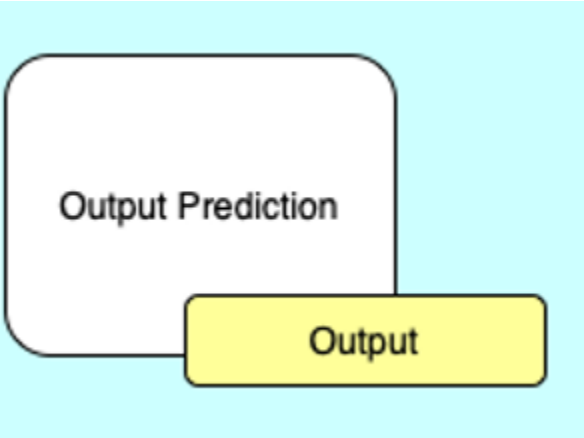
The 4 models will be executed: MSE, Style, MSE and Style and GAN.

4. Predict the forecast



The models will take the input of test data of 13 images and give the forecast as per different models with 12 images

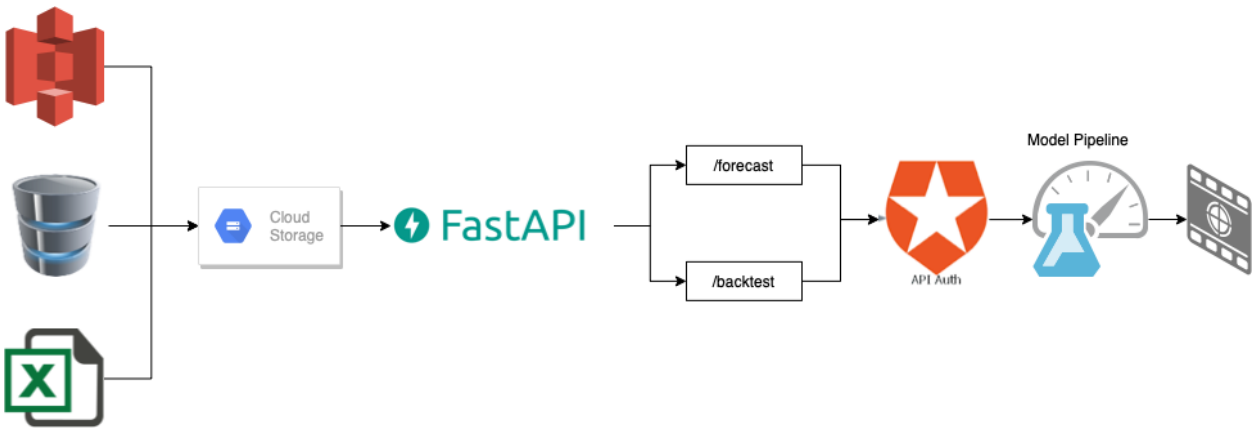
5. **Output**



The model will forecast the 12 images after running the 4 models when it needs to forecast. For Backtesting, the model will generate 12 predicted images for forecast along with actual images from the past data.

API Pipeline

The mechanism of REST API looks like this. Let's say you want to search cat photos on Google. The first step is to send a request to Google by giving them a query like 'cat photos'. Then, the server will send you a response to your computer, which is the compilation of cat photos.



Our API will have 2 Major Endpoints:

1) /Forecast
2) /Backtest

Inside we will have options to run a specific model API

1) /MSE
2) /Style
3) /MSE_Style
4) /GAN

The output of our API will be predicted 12 Images if /forecast endpoint was called or 24 images if /backtest endpoint was called.

Python Package:

The functionalities used in the API and Model Pipeline will be done by our Python package
Python package will have 2 methods

1) Forecast
2) Backtest

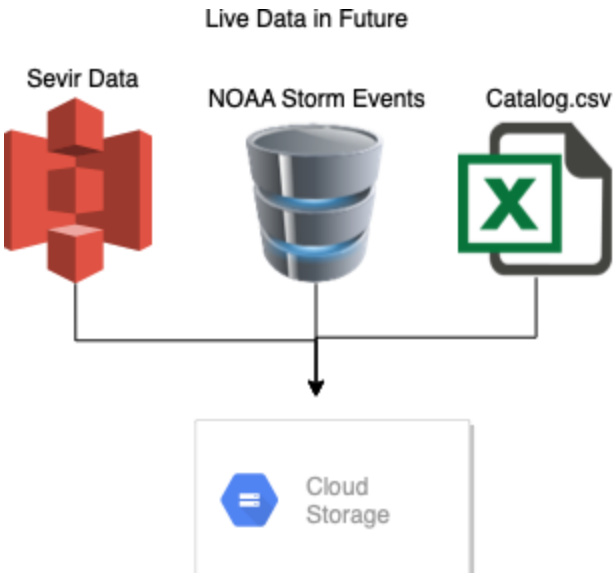
Users

Web App Pipeline

We plan to build a webapp for our users where they have an option to see for themselves the forecast of weather and also backtest few use cases if they want to.

Web App Pipeline contains all the pipelines which we discussed in the above pages

Part 1:



The first part of our API is collecting different data from varied sources.

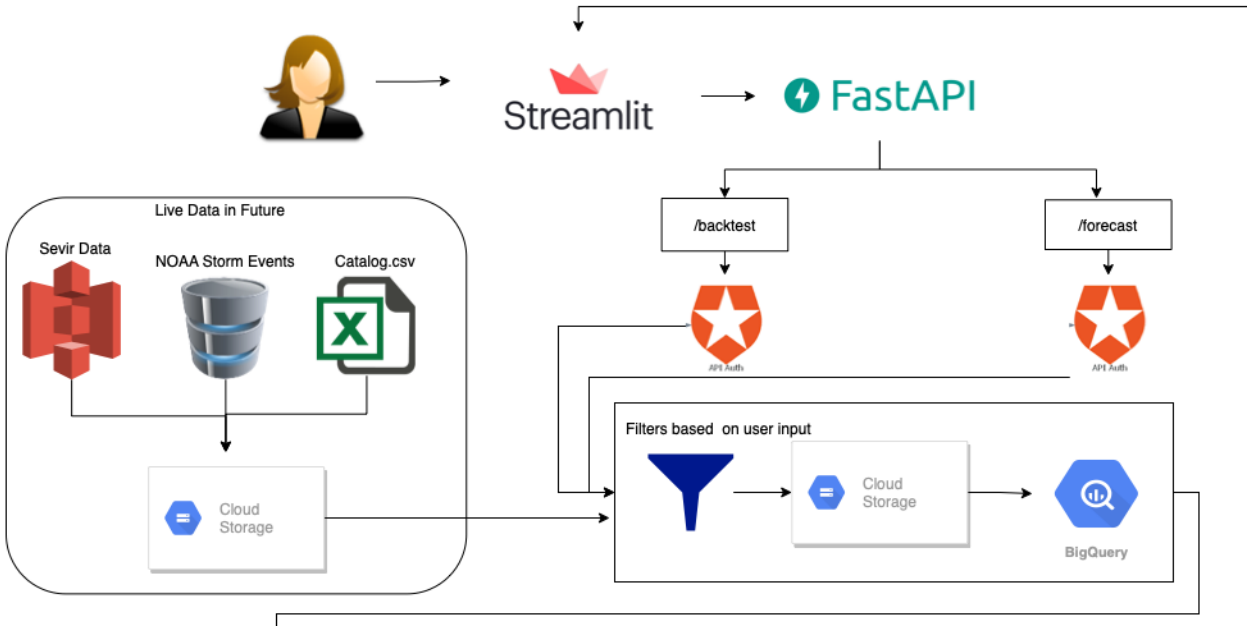
1) SEVIR data from S3 Buckets
2) NOAA Storm Events from government fttp
3) Catalog.csv from github

We plan to store all these data in Google cloud buckets

Part 2:

We will be having frontend as a streamlit application with backend as FASTApi

Part 3:

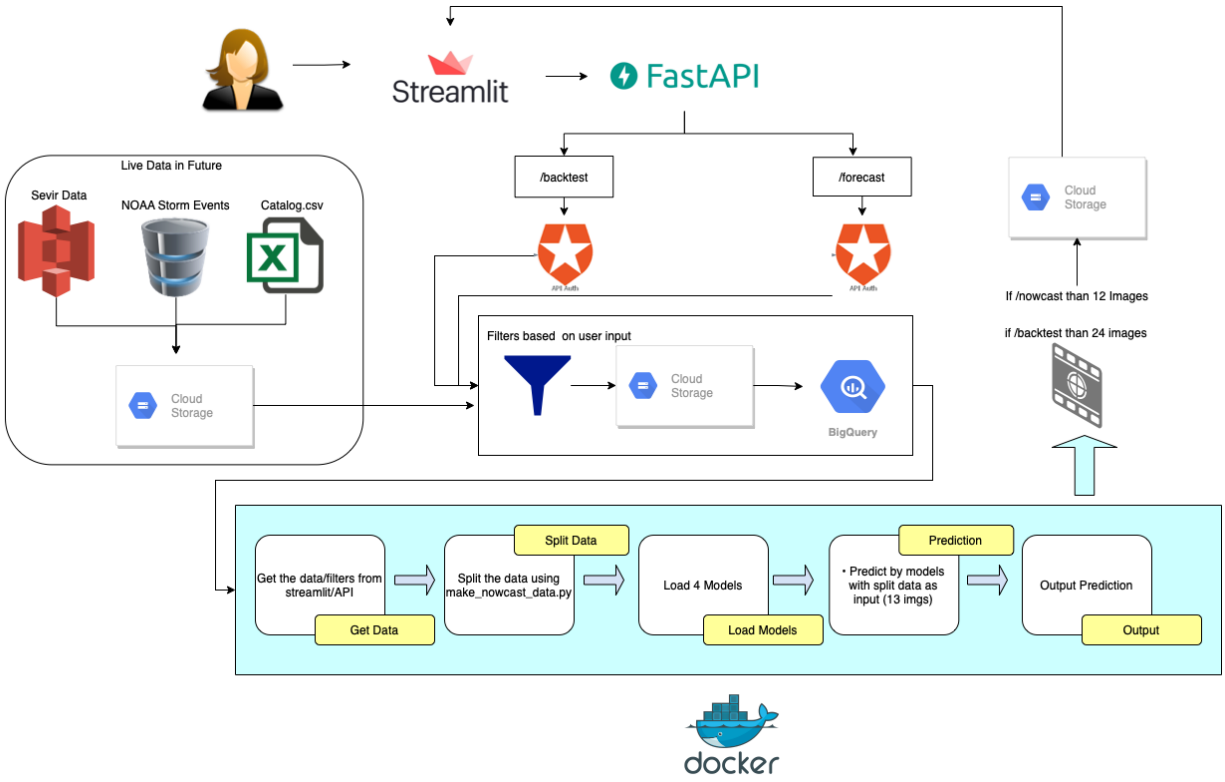


Users in our Application will have 2 options forecast and backtest which will be called using FASTApi endpoints /forecast and /backtest and we saw on previous page.

Once the user inputs its required filters for the inference pipeline we will take that filters and run the filter pipeline where

1) We update the Catalog.csv
2) Take specified SEVIR files from google buckets
3) Store the files in big query

Part 4:

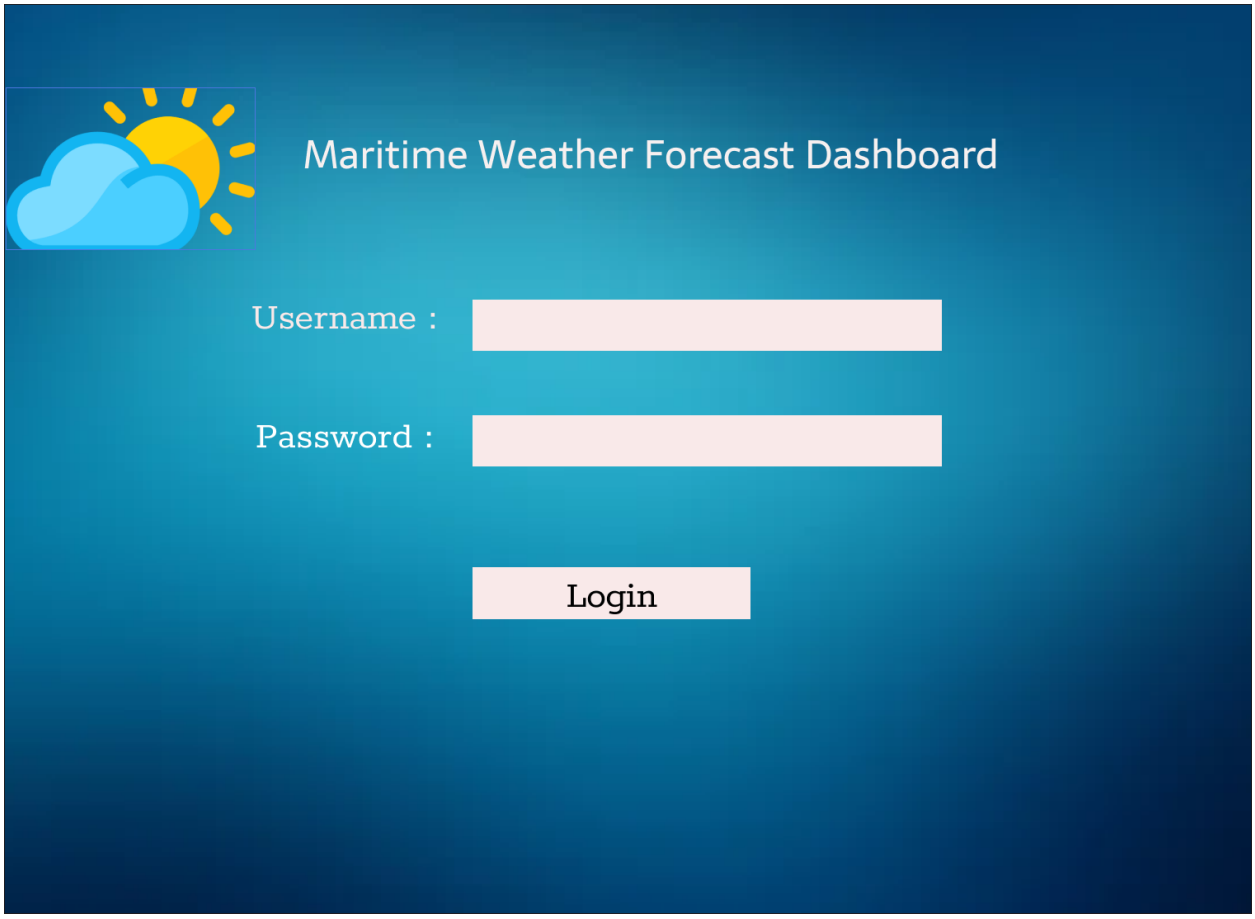


We will have a base docker for model pipeline.

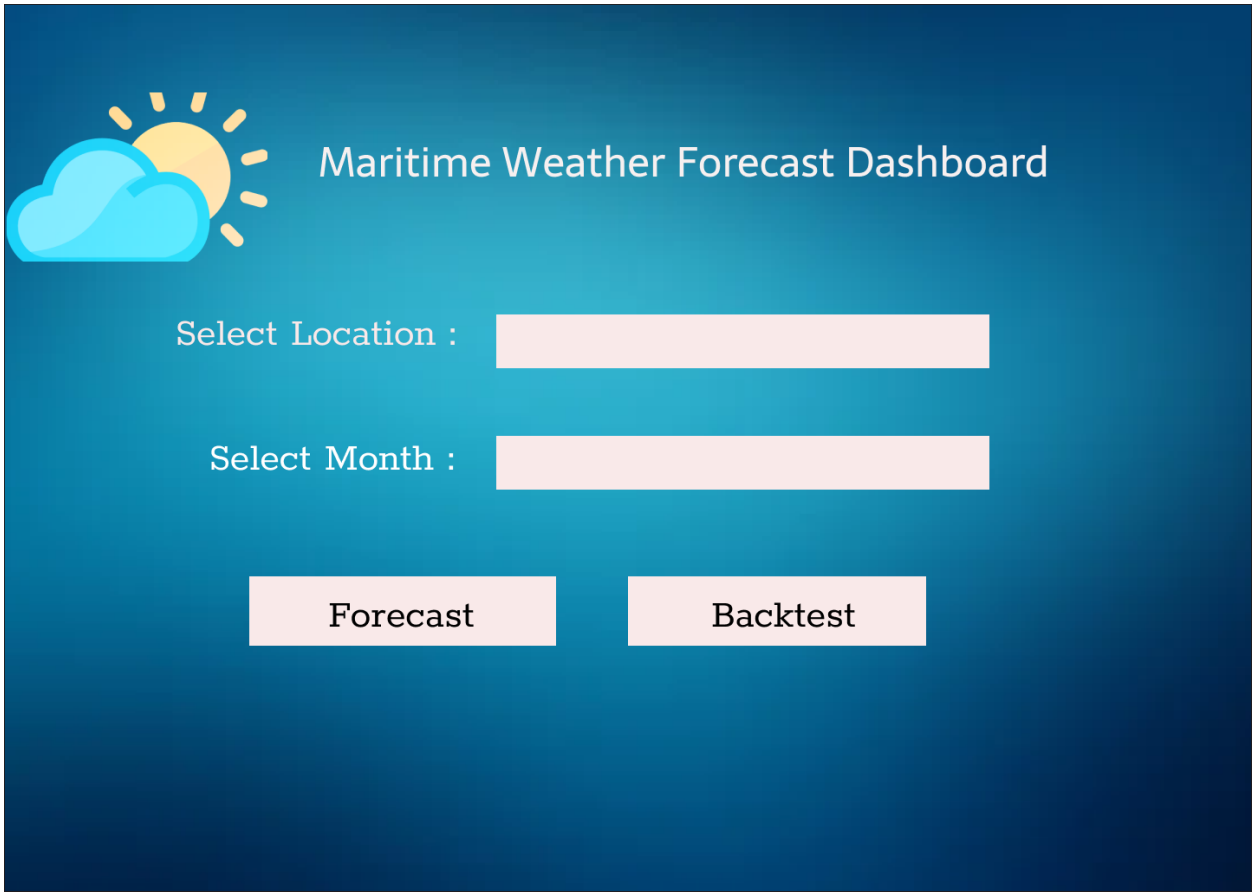
Once the filtered data is ready the data goes through our inference pipeline which is our Model pipeline and outputs required images which are stored in google buckets.

User Interface Using Figma

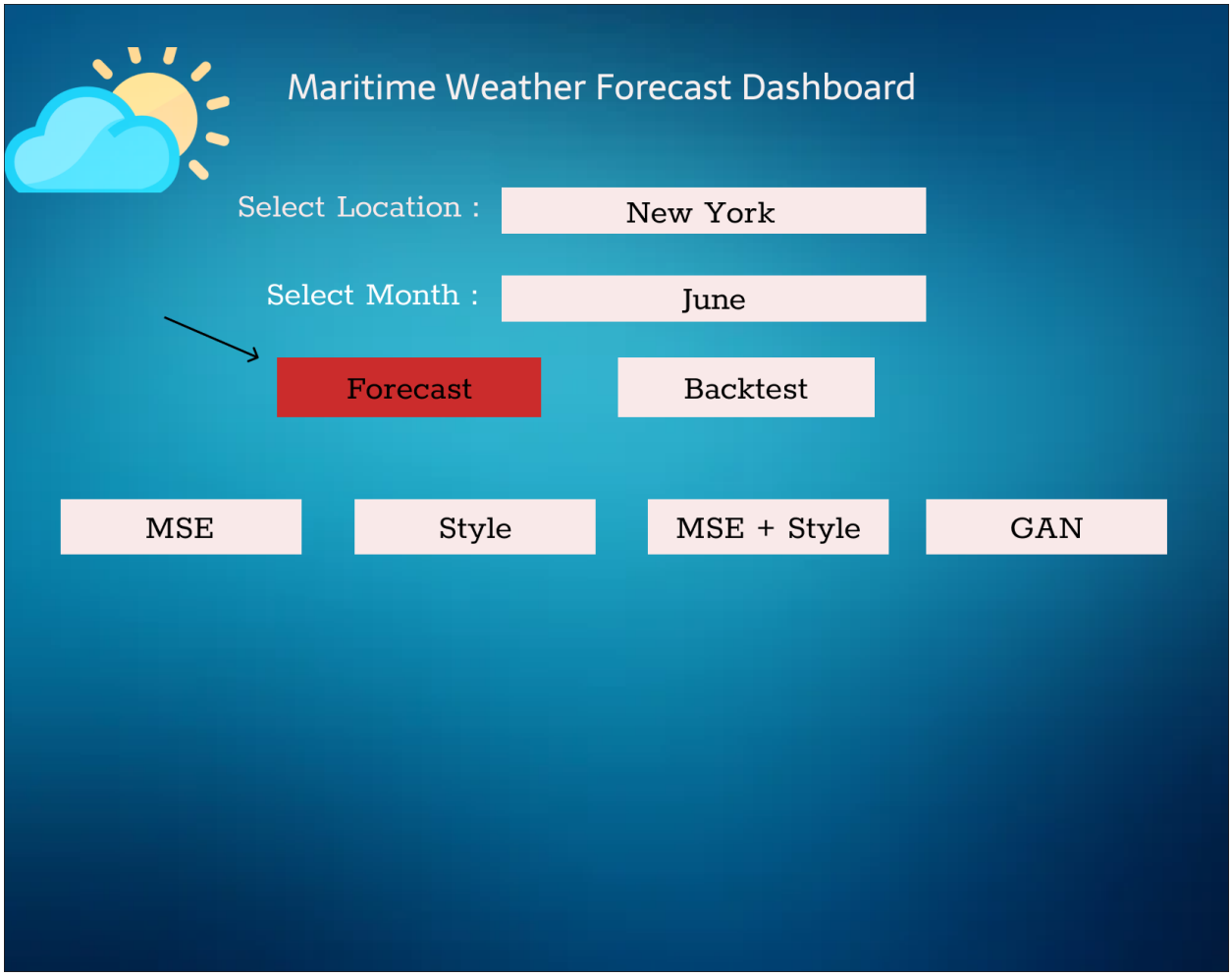
1. Login Page - This Page is a basic login for different use cases including authentication for Military, Cargo and Dockyard representatives



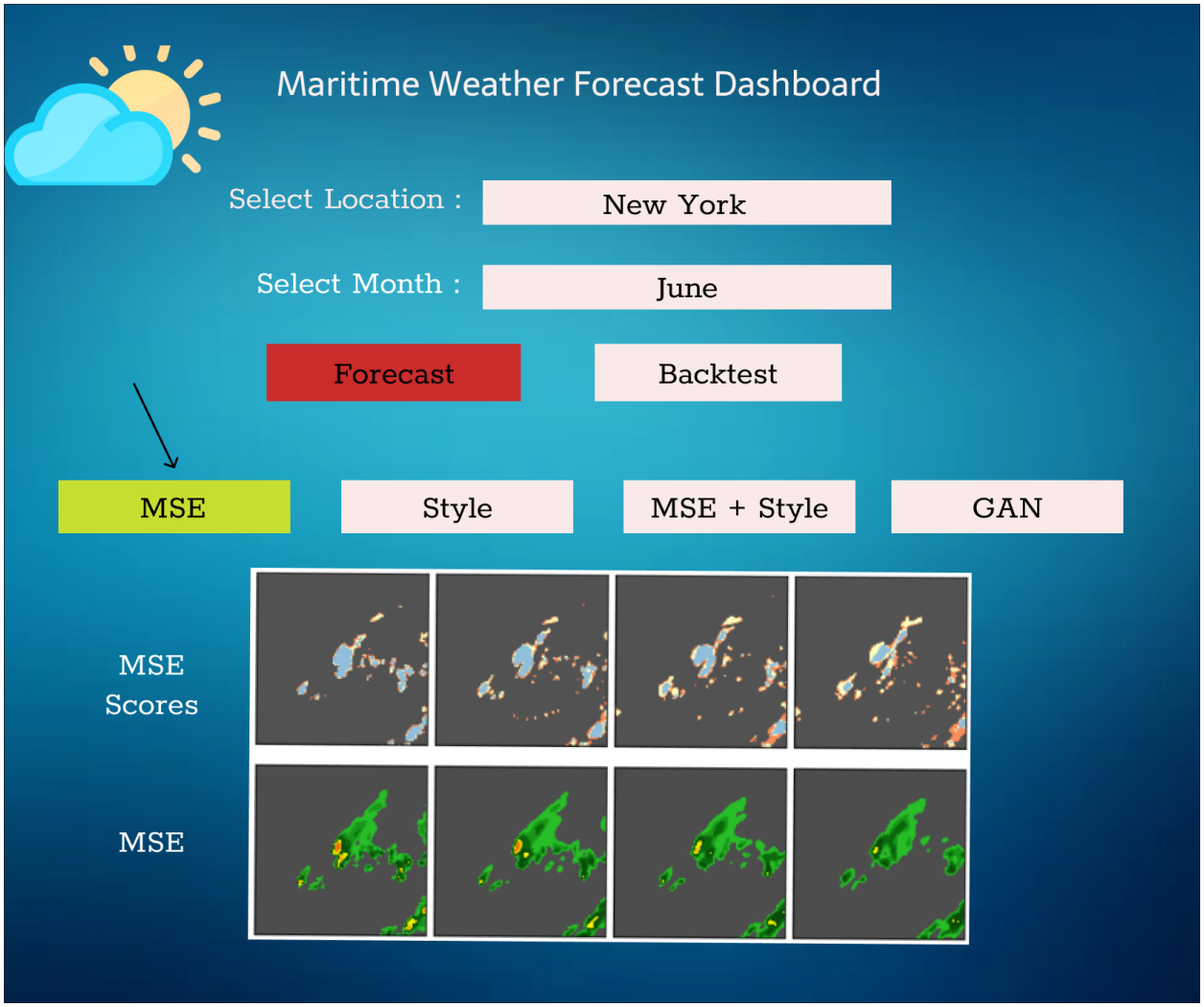
2. After the user is logged in, it will ask for the location and the month so the model can find those events id belonging to the specific location and month. After entering these details the user will now have two options to select from, **Forecast** or **Backtest**

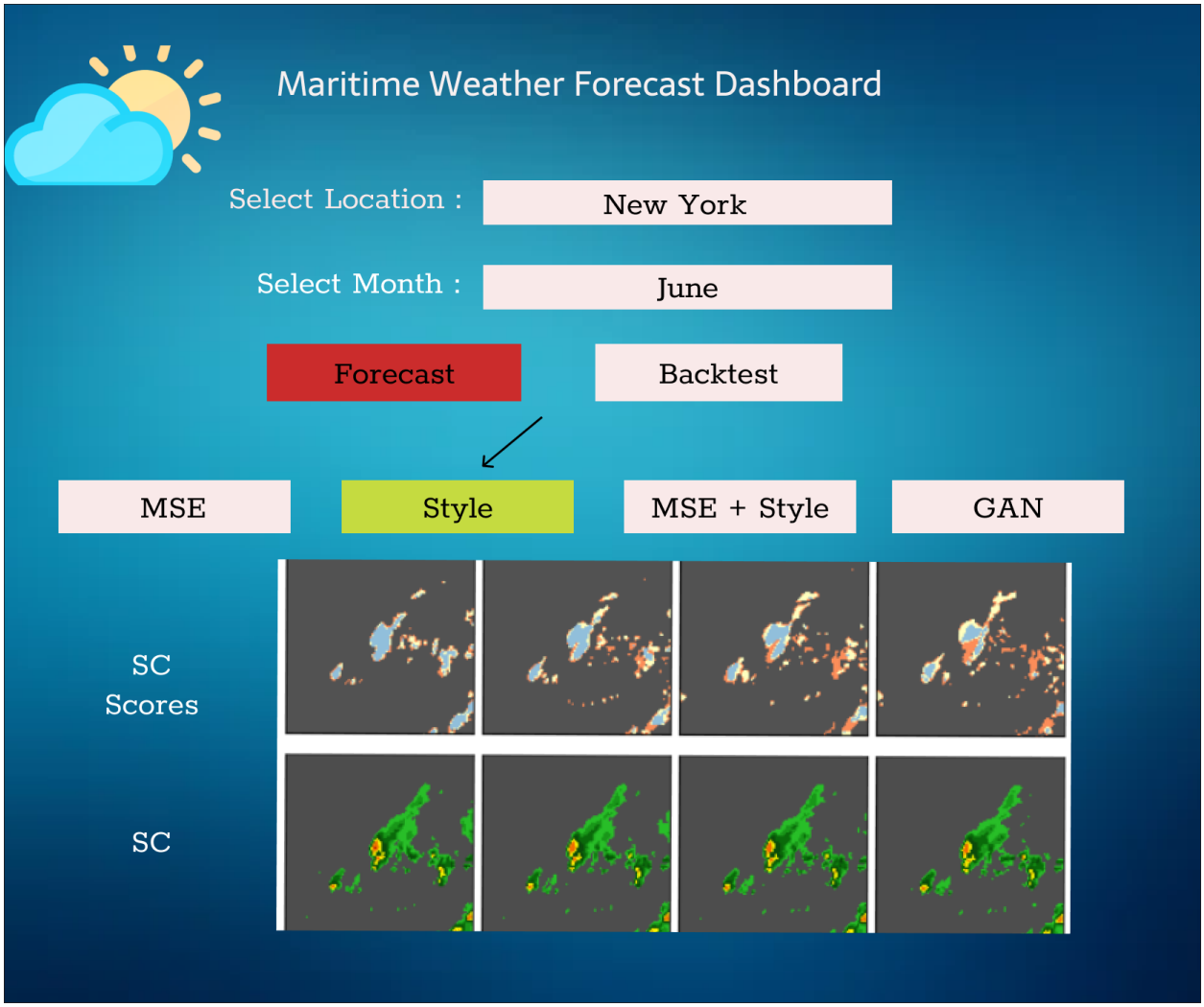


3. When the user clicks on the Forecast button the interface will give options for 4 different models to choose from. MSE, Style, MSE and Style and GAN.

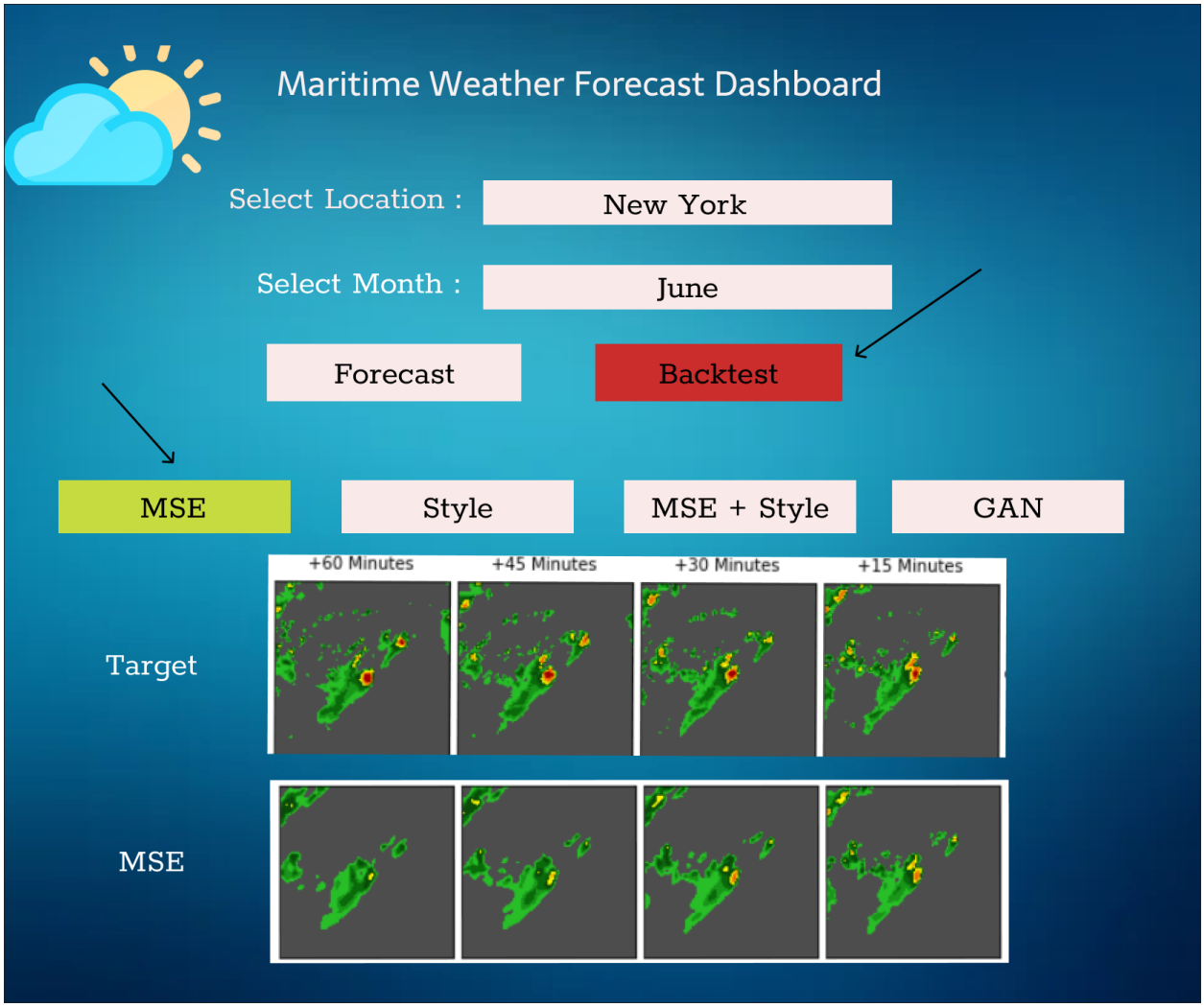


4. Here when the user selects MSE, it will display the forecasts for MSE models


Maritime Weather Forecast Dashboard
Select Location :
New York
Select Month :
June
Forecast
Backtest
MSE
Style
MSE + Style
GAN
MSE Scores
MSE



5. When the user clicks on Backtest it will give the predicted images of the model alongside the expected output to depict how well the model is performing.



Distribution

1. **Documentation:** We will be using codelabs for our documentation.

2. **Distributing the package:** We will be delivering our application package as Docker containers. The deliverable will be a docker image. Once the Docker image is built, it can be run on any machine that has Docker installed. The image is uploaded in the Docker hub and then it can be downloaded from there.

3. **Advertising the API:** We will be advertising our API by listing it in online API directories like ProgrammableWeb, RapidAPI and also advertising on social media.

The document for this codelab can be found at

https://docs.google.com/document/d/1vDU4PUry6cB-0SMbEo7oPp-YSTL1wquu2rJqpBVut9o/edit?usp=sharing