**SYSTEM ANALYSIS**

2.1 Existing system

• The current weather detection systems used on ships are limited in their ability to provide accurate and real-time data. Most systems are manually operated, require extensive maintenance, and are not reliable in harsh weather conditions. Additionally, the data they provide is often limited, which makes it difficult for ship captains to make informed decisions.

• One existing system based on weather detecting app for ships is the Weather Routing Inc. software, which provides weather forecasting and routing services to the maritime industry.

• The system uses a combination of satellite data, numerical weather models, and historical weather patterns to create custom weather forecasts for a particular ship's route. The system takes into account a ship's speed, size, and draft, as well as the cargo it is carrying, to calculate the optimal route and provide weather alerts and warnings along the way.

• The Weather Routing Inc. software also includes a real-time monitoring system that tracks a ship's location, speed, and course, and adjusts the route as needed to avoid dangerous weather conditions. The system can also provide data on wave height and direction, sea surface temperature, and ocean currents to help ships navigate safely and efficiently.

• In addition to the software, Weather Routing Inc. also provides 24/7 support and consulting services to help ships manage weather-related risks and make informed decisions about their routes. This includes advice on how to modify speed or adjust course to avoid bad weather, as well as emergency response planning in case of severe weather events.

• Overall, the Weather Routing Inc. system provides critical weather information and routing services to ships of all sizes and types, helping to increase safety, efficiency, and profitability in the maritime industry.

Proposed system

● The proposed system for a weather API with Streamlit consists of a web application that allows users to access real-time weather data for a specific location. The system utilizes an API to fetch weather data and displays it in an interactive and user-friendly way using the Streamlit framework.

● The system is designed to provide users with a range of weather metrics, including temperature, humidity, wind speed, precipitation, and more. Users can search for weather data by entering a city, state, or zip code into a search bar. The app then queries the weather API and returns the current weather data for the specified location.

● The system is also designed to be responsive, allowing users to access the app from laptop This ensures that users can access the weather data they need no matter where they are. Overall, the proposed system for a weather API with Streamlit is a powerful tool for accessing weather data. By utilizing the Streamlit framework and a weather API, developers can build a user-friendly and visually appealing weather application that provides users with the information they need to make informed decisions based on the weather conditions in their area.

**SYSTEM REQUIREMENT**

The system requirements for a weather forecasting web application can vary depending on the specific features and functionalities you want to include. However, here are some general system requirements to consider:

Web Server: You will need a web server to host your application. Popular choices include Apache, Nginx, and Microsoft IIS.

Programming Languages: Depending on your preference and the technologies you want to use, you may need knowledge of HTML, CSS, and JavaScript for the front-end, as well as a back-end language like Python, PHP, or Node.js.

Database: If you plan to store and retrieve weather data or user information, you may need a database system. Popular choices include MySQL PostgreSQL, and MongoDB.

APIs: Weather forecasting data is typically obtained through APIs provided by weather services. You will need to integrate with a reliable weather API, such as OpenWeatherMap, Weather Underground, or the National Weather Service API.

Data Storage: Depending on your requirements, you may need to store historical weather data. In such cases, you might require additional storage capacity or cloud-based solutions like Amazon S3 or Google Cloud Storage.

User Authentication: If you want to offer personalized features or user-specific settings, you may need to implement a user authentication system using technologies like OAuth, JWT, or session management.

Responsive Design: Ensure your application is designed responsively to provide a consistent user experience across different devices and screen sizes. This may involve using CSS frameworks like Bootstrap or Tailwind CSS.

Performance and Scalability: Consider optimizing your application's performance by implementing caching mechanisms, load balancing, and scalable infrastructure to handle increased traffic.

Security: Implement security best practices to protect user data and prevent unauthorized access. This includes using encryption, secure communication protocols (e.g., HTTPS), input validation, and protection against common web vulnerabilities.

Testing and Deployment: Set up a testing environment to verify the application's functionality and ensure proper deployment processes to production environments.

It's important to note that the above requirements are general guidelines, and the specific needs of your weather forecasting web application may vary based on its scope and intended functionality.

12.1 **SOFTWARE REQUIREMENT**

Operating System : Windows 7,8,10 (64 bit)

Software : Python

Tools : Anaconda (Jupyter notebook IDE)

12.2 HARDWARE REQUIREMENT

Hard Disk : 500GB and above

RAM : 4GB and above

Processor : I5 and above