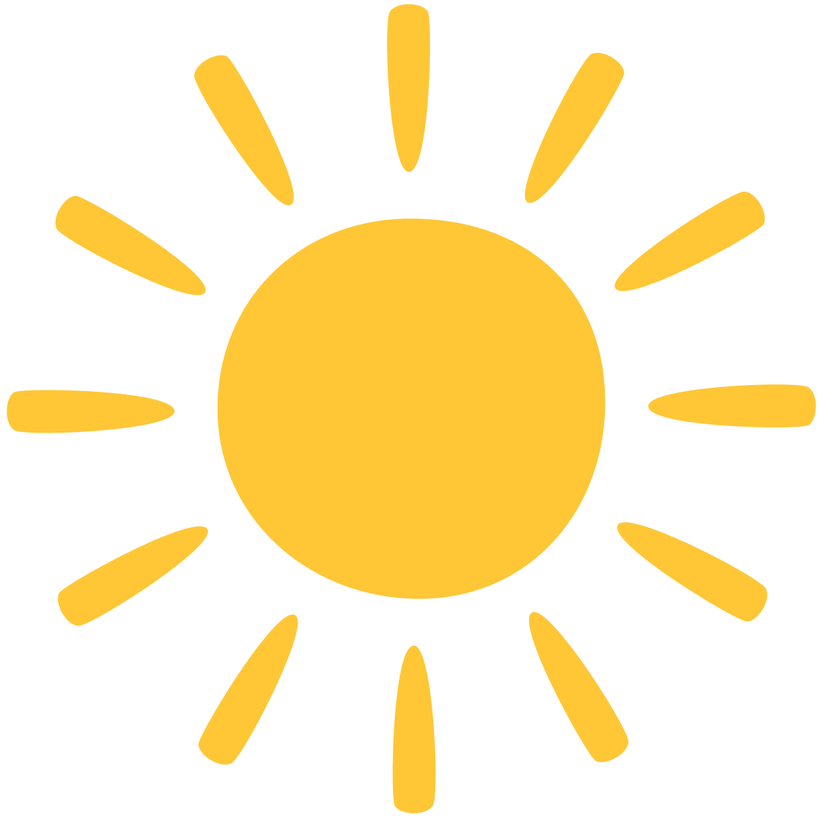
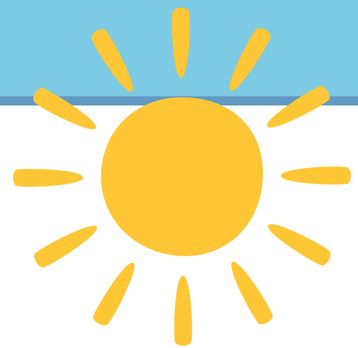
SkyInsight

Kartiki Dindorkar (W1651519)
Anish Dora (W1641666)
Venkat Srikar (W1652687)

Agenda

- Problem Statement
- Assumptions
- Requirement Analysis
- Low Level Requirements
- Use Cases
- OO concepts
- UML Diagrams
- Implementation
- Demo

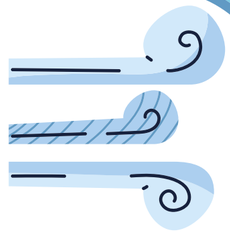




Problem Statement

- Weather Forecasting System using **Object Oriented principles**.
- **Accurate and real-time weather predictions** based on multiple meteorological parameters.
- Automatically fetching the **user's location** and **Search** Option
- **Multiday forecasting** - daily and hourly
- **Severe weather alerts** to notify the user in advance.
- Weather data in **graphical** format
- It **empowers** individuals, businesses, and communities **to proactively adapt to changing weather conditions, enhancing safety** and responsiveness.

Requirement Analysis



- **Weather data extraction:**
  - **Gather real-time** meteorological data from **external weather API**s such as **https://api.weatherapi.com**
  - Collect information on temperature, humidity, wind speed, atmospheric pressure, and precipitation.

- **Data Processing and Analysis:**
  - Utilize statistical methods and machine learning models available through the API for accurate forecasting.

- **Weather Visualization:**
  - Present weather data from the API in **graphical visualizations**.
  - Utilize **charts, graphs** provided by the API for a **user-friendly display**.

Requirement Analysis

- **Location-Based Forecasting:**
  - Implement geolocation features using APIs to **automatically detect** and input the user's current location for personalized weather forecasts. Allow users to **manually input** specific locations for weather forecasts.

- **Multi-Day Forecasting:**
  - Utilize the forecasting capabilities of the API to provide **multi-day weather forecasts**, including **daily and hourly breakdowns** of weather conditions.

- **Severe Weather Alerts:**
  - Integrate with alert APIs to implement an **alert system for severe weather conditions**. Receive notifications or warnings from external sources to notify users in advance of severe weather events

Assumptions

- The system assumes that the **external weather APIs provide accurate and reliable** weather information, including temperature, humidity, wind speed, atmospheric pressure, and precipitation.
- User has agreed to access and save **user's location**.
- Users of the system can access weather forecasts for both their current location (via geolocation features) and specific manually input locations.
- The system assumes that **users** have basic knowledge and **understanding** of **weather terminology and symbols** used in weather forecasts.

Assumptions

- The Weather Forecast System will follow **Imperial Unit System**.
- **Severe weather alerts** are provided **based on** notifications or warnings received from **external sources**, and the system assumes that these alerts are **timely and accurate.**
- The system design acknowledges that **real-time updates** may be subject to delays inherent in **data retrieval processes from external APIs** and aims to provide the most up-to-date information available.

Low Level Requirements

SR. No.	Requirement ID	Requirements	Details
1	Req_1	Register a new user	Get User email ID, password and location (city name or zip) and save the encrypted password in to the database
	Req_2	Check if User Already Existed	Based on the email ID, if user is already present show the pop up: "A user with this email already exists."
2	Req_3	Login	Get User email ID, password and authenticate the user with the user- and encrypted password entry in the data base
	Req_4		If failed to authenticate, send error message
3	Req_5	Logout	Show a button "Logout" to logout from the system and show the login page.
4	Req_6	Extraction of location	On Login, System should fetch the user's saved location from database
	Req_7		Show a button "Current". On pressed should extract the current user location
	Req_8		Give a text box and "Search" button for user to search for the location. Once Search button is pressed get the user entered location.
5	Req_9	Show Weather Data	On login, always show current date, time and the temperature in a panel. Temperature should be shown in Farenheit.
	Req_10		System should fetch user's saved location from database and show weather data for that location
	Req_11		Temperature should be shown in Farenheit.
6	Req_12	Weather Visualization	Show graphs for : 1. Hourly Temperature for today in Farenheit 2. Hourly Cloud percentage for today
7	Req_13	Multi-day Forecasting	Give an option for user to see 1 Week's Weather Forecast data in one click
	Req_15		Weather data should show following parameters hour-wise: 1. Condition of data (Sunny, Cloudy, Rainy) 2. Temperature (Farenheit) 3. Wind Speed (Mph) 4. Humidity (%) 5. Atmospheric Pressure (In) 6. Precipitation (mm)
8	Req_16	Severe Weather Warnings	Based on user location feched from Req_6 / Req_7 / Req_8, show alerts if any severe weather warning is present.
	Req_17		Show following details for a warning : 1. Category of the warning (wind advisory, flood advisory, etc) 2. Effective from (date and time) 3. Expires on (date and time) 4. Description of the warning

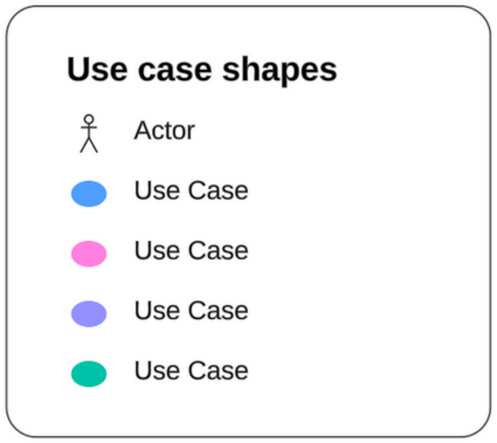
Use Cases

- **Use Case Actors**
  - **User:** The primary actor who **interacts with the Weather Forecasting System** to access weather information and forecasts.
  - **System:** To show **Weather Forecast** for the user's location and provide **location search** and **multiday forecast** option to the user. As well as alert the user for **severe weather**. Give a **visual representation** of weather data

- **Top Use Cases**
  - **Register User:** To register the user. The user has to provide an email, password, and location(Optional) to register.
  - **Login:** To authenticate the user with the provided email ID and password.
  - **Logout:** To log the user from the system.
  - **Search for location:** To let the user search for different locations to see that location's weather information.
  - **Weather Forecast:** To fetch the hourly weather information from weather APIs and display various weather parameters to the user.
  - **Multiday Forecast:** To let users see the weather forecast details for the next 7 days.
  - **Weather Visualization:** Display Weather data using multiple graphs for better understanding.
  - **Severe Weather Warning:** To alert the user if there is any upcoming severe weather conditions.

UML Diagrams

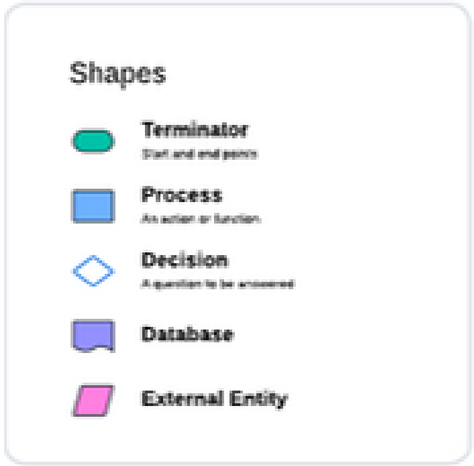
Use Case


Use case shapes
Actor
Use Case
Use Case
Use Case
Use Case

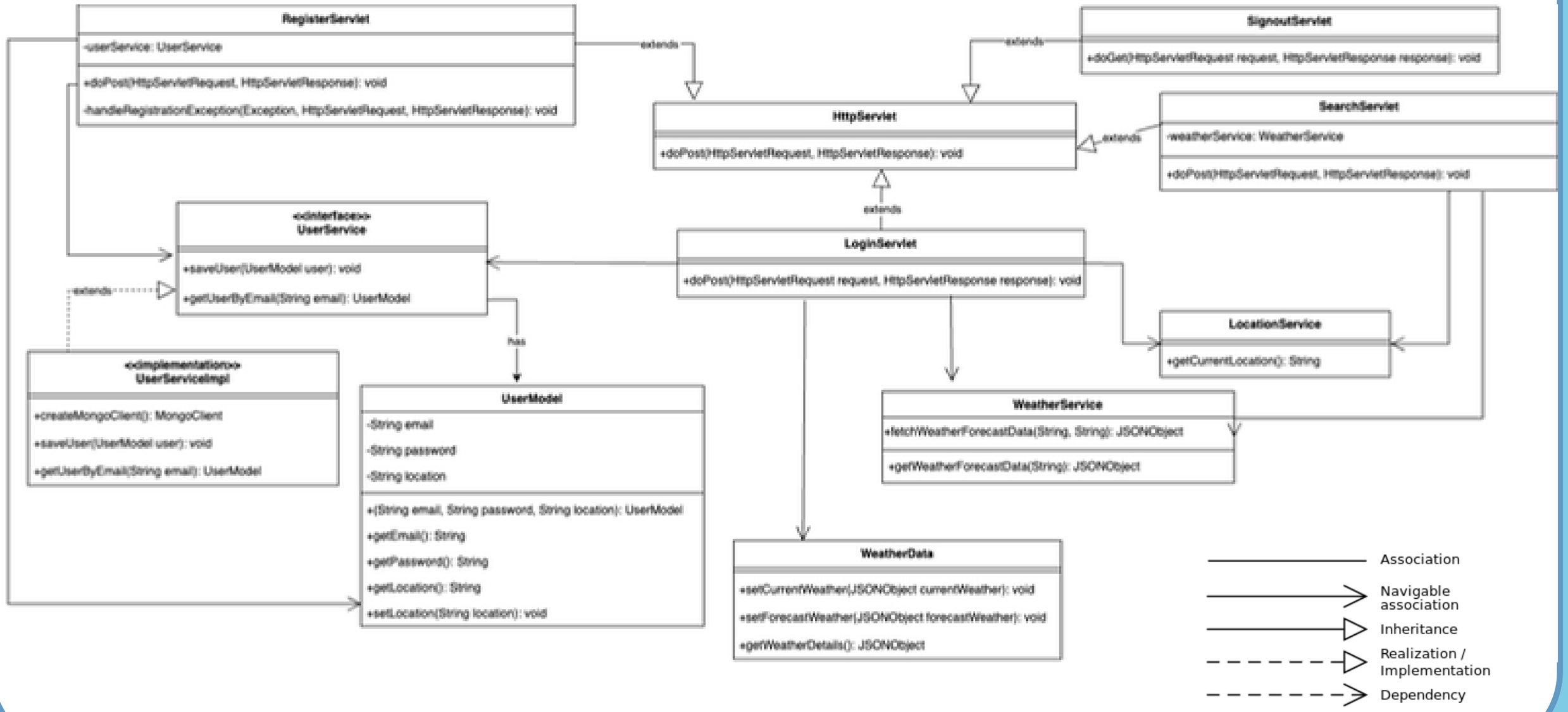

Weather Forecast System
Create an account
Account Details to Database
Login with credentials
Authenticate the User
Search for location
get data from weather api
Display data
Signout from application
<<system>> Register
<<system>> Login
<<system>> Search
<<external system>> WeatherApi
<<system>> Signout
User

FlowChart


User Opens URL
Register
Login
Yes
No
User Database
Email Present ?
Authentication Successful ?
User Database
No
Yes
Homepage
Extract Current Location
Search for Location
Get Weather data from WeatherApi
Show Forecast
Multi-Day Forecast
Severe Weather Alerts ?
Yes
No
Show Alerts
Signout


Shapes
Terminator
Process
Decision
Database
External Entity

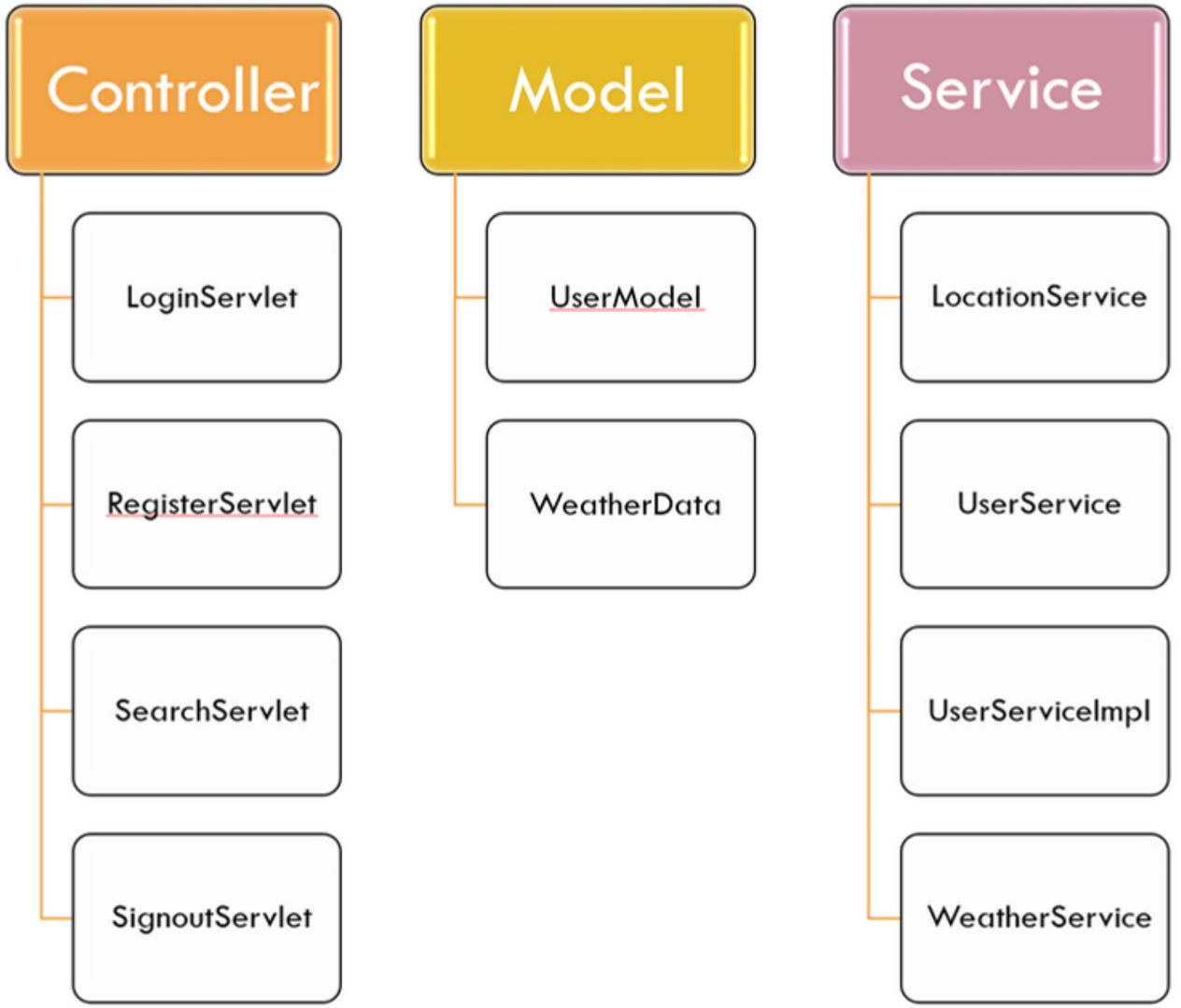
Class Diagram



...

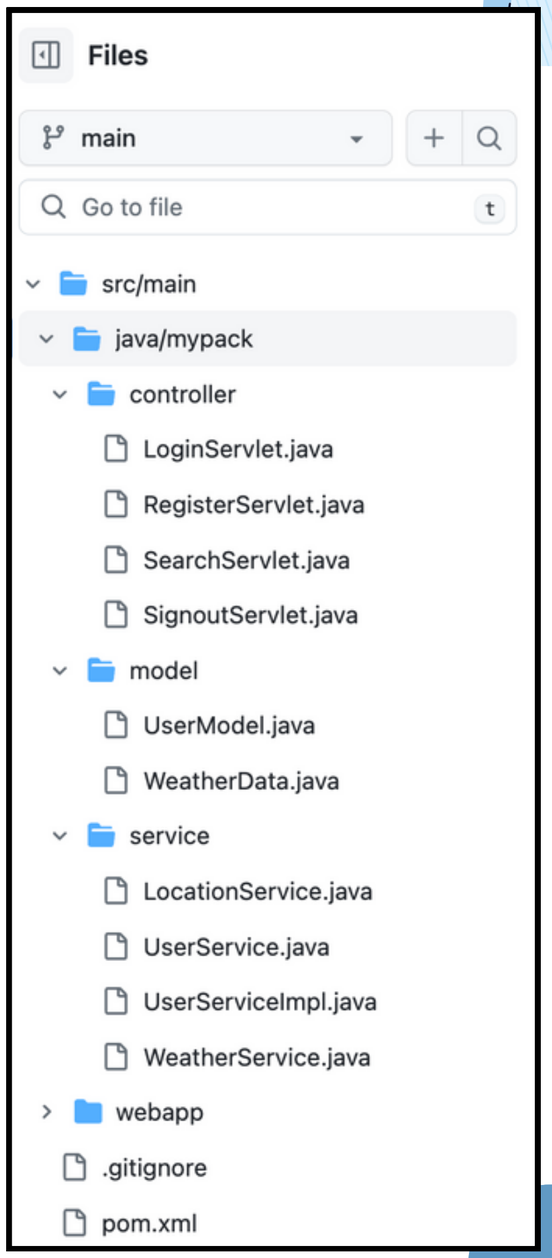
Implementation

Classes


Controller
LoginServlet
RegisterServlet
SearchServlet
SignoutServlet
Model
UserModel
WeatherData
Service
LocationService
UserService
UserServiceImpl
WeatherService

Design Patterns

- **MVC Pattern:**
  - **Model -** UserModel and WeatherData contains essential data
  - **View -** Consist of JSP files to display information from UserModel and WeatherData classes
  - **Controller -** LoginServlet, RegisterServlet, SearchServlet, SignoutServlet
    - Handles incoming HTTP requests
    - Interact with models to retrieve data
    - Perform actions based on received requests
    - Direct user to appropriate view



Design Patterns

- **Repository Pattern:**
  - To Create an abstraction layer
  - Interfaces with classes like UserServiceImpl.
  - These classes contain the actual logic to interact with the database

- **Service Pattern:**
  - Encapsulate Business Logic separating from presentation and data access logic
  - Represent business actions in interfaces like UserService and WeatherService.
  - Declare methods that the rest of the application can use to perform business logic
  - UserServiceImpl and other service implementations would contain the actual business logic

OO Concepts Incorporated

1. ABSTRACTION
2. INHERITANCE
3. POLYMORPHISM
4. ENCAPSULATION

1 ABSTRACTION

```java
package mypack.service;

import mypack.model.UserModel;

5 usages  1 implementation  srikar
public interface UserService {
    1 usage  1 implementation  srikar
    void saveUser(UserModel user);
    1 usage  1 implementation  srikar
    UserModel getUserByEmail(String email);
}
```

2 INHERITENCE

```
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
```

```
@WebServlet("/SearchServlet")
public class SearchServlet extends HttpServlet {
```

```
@WebServlet("/SignoutServlet")
public class SignoutServlet extends HttpServlet {
```

```
@WebServlet("/RegisterServlet")
public class RegisterServlet extends HttpServlet {
```

3 POLYMORPHISM

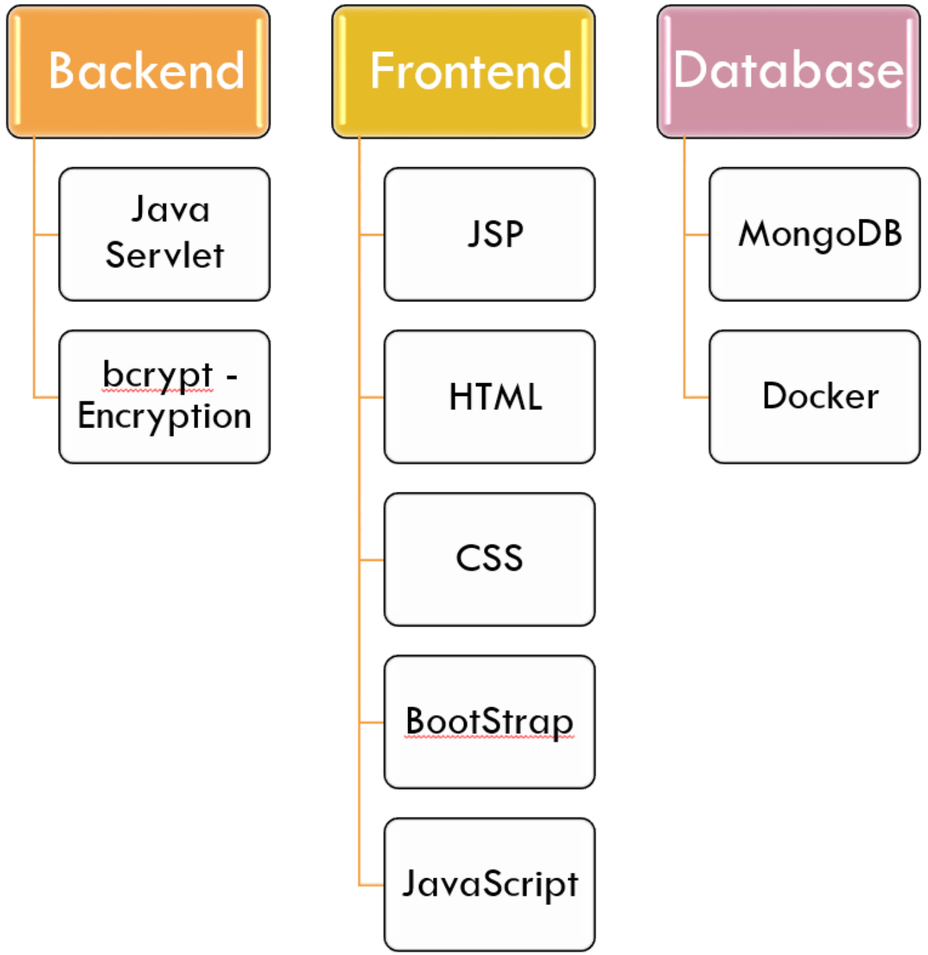
```
package mypack.service;

import ...

4 usages  srikar
public class UserServiceImpl implements UserService {

    1 usage  srikar
    private MongoClient createMongoClient() { return MongoClients.create("mongodb://localhost:27017"); }

    1 usage  srikar
    @Override
    public void saveUser(UserModel user) {...}

    1 usage  srikar
    public UserModel getUserByEmail(String email) {...}
}
```

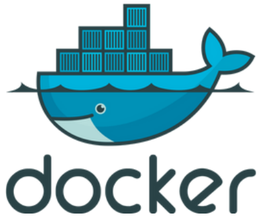
4 ENCAPSULATION

```java
package mypack.model;

12 usages  srikar +1
public class UserModel {
    2 usages
    private  String email;
    2 usages
    private  String password;
    5 usages
    private String location;

    // Constructor
    2 usages  srikar
    public UserModel(String email, String password, String location) {...}

    // Getters and Setters
    srikar
    public String getEmail() { return email; }

    srikar
    public String getPassword() { return password; }

    srikar
    public String getLocation() {...}

    srikar
    public void setLocation(String location) { this.location = location; }
}
```

```java
package mypack.model;

import org.json.JSONObject;

3 usages  srikar
public class WeatherData {
    4 usages
    private JSONObject weatherDetails;

    1 usage  srikar
    public WeatherData() { this.weatherDetails = new JSONObject(); }

    1 usage  srikar
    public void setCurrentWeather(JSONObject currentWeather) { this.weatherDetails.put("weather", currentWeather); }

    1 usage  srikar
    public void setForecastWeather(JSONObject forecastWeather) {...}

    1 usage  srikar
    public JSONObject getWeatherDetails() { return weatherDetails; }
}
```

Technology Stack Used





Implementation

https://github.com/jvsaisrikar/weather

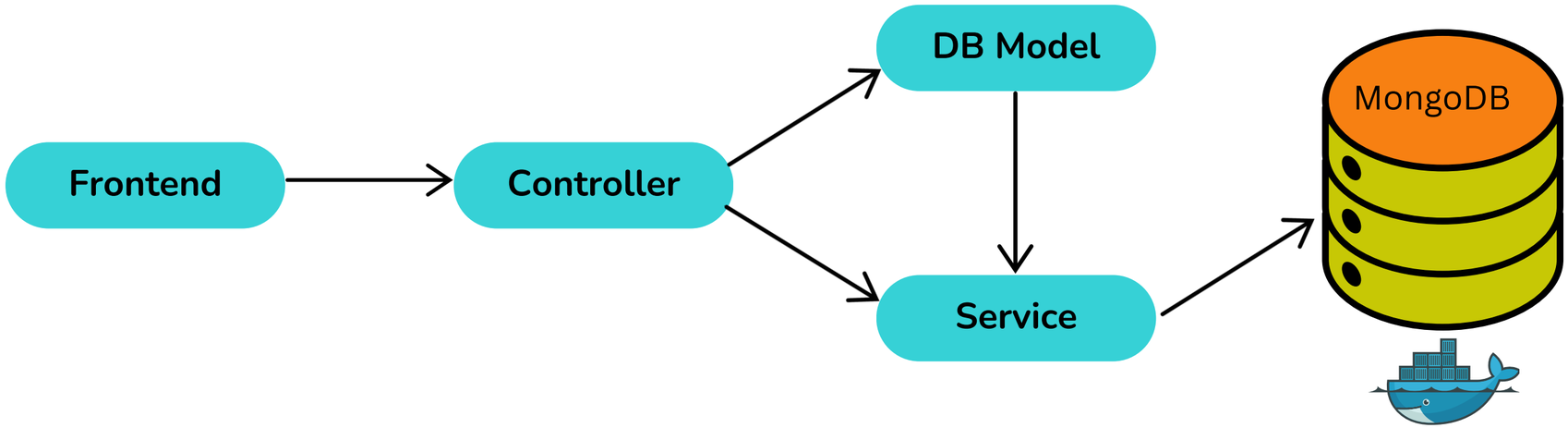
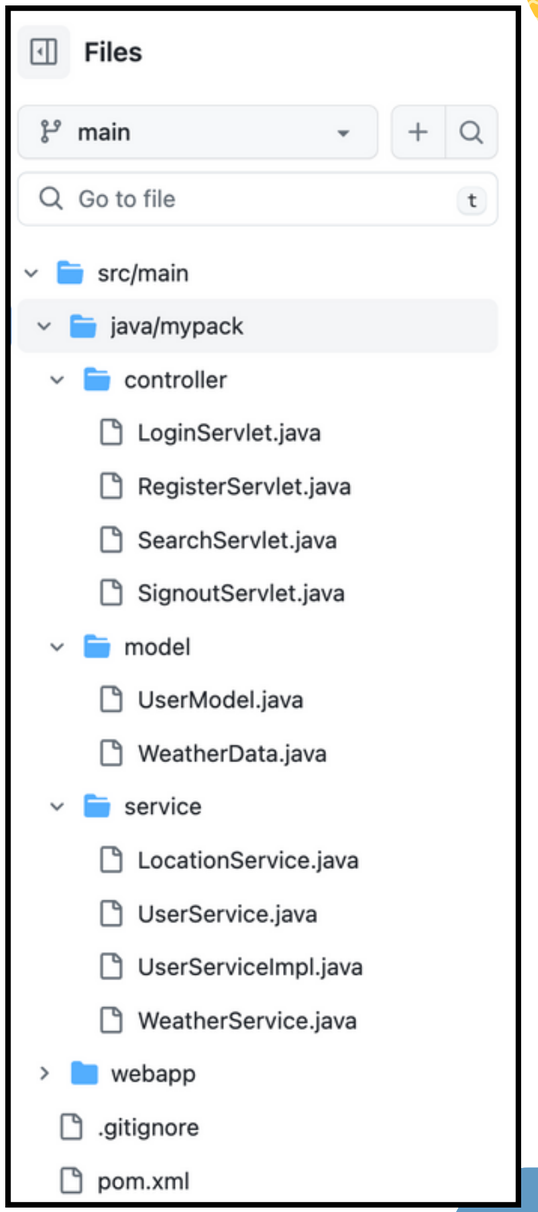
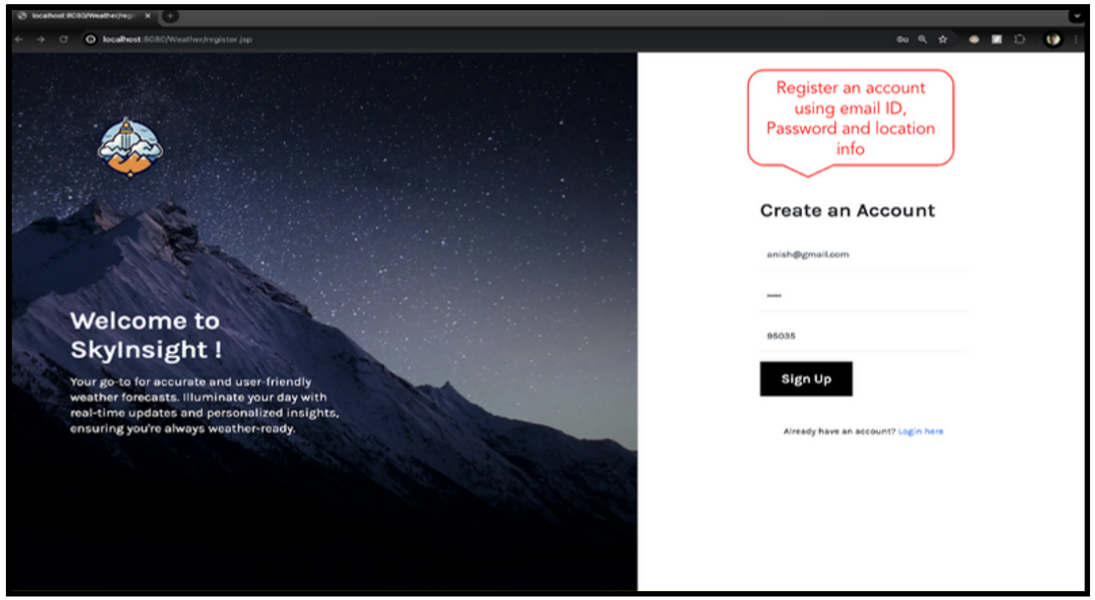


Fig. Software Architecture Diagram

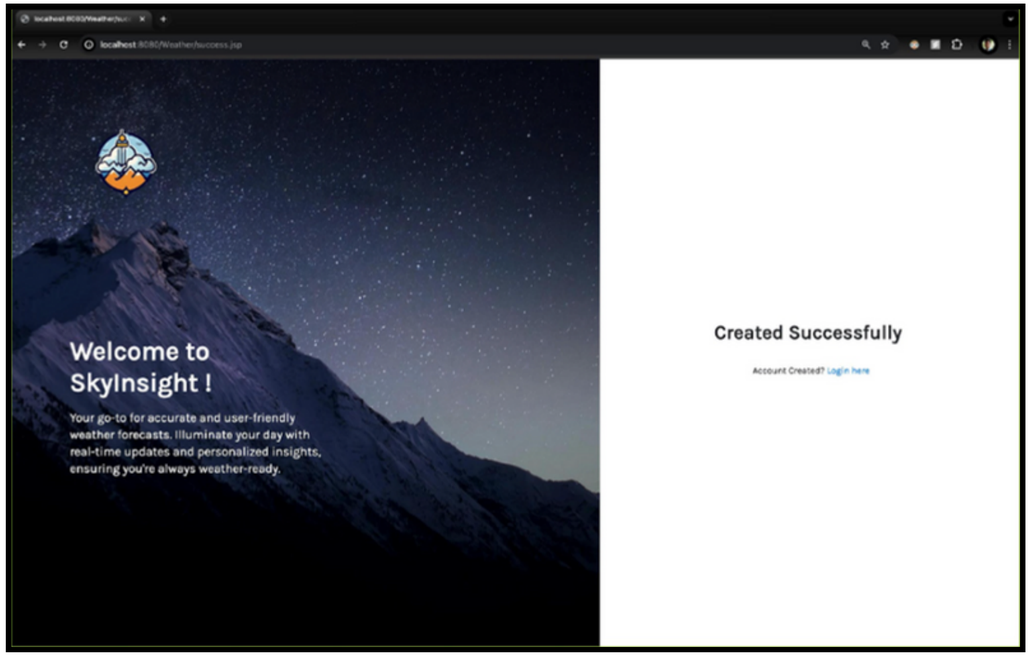


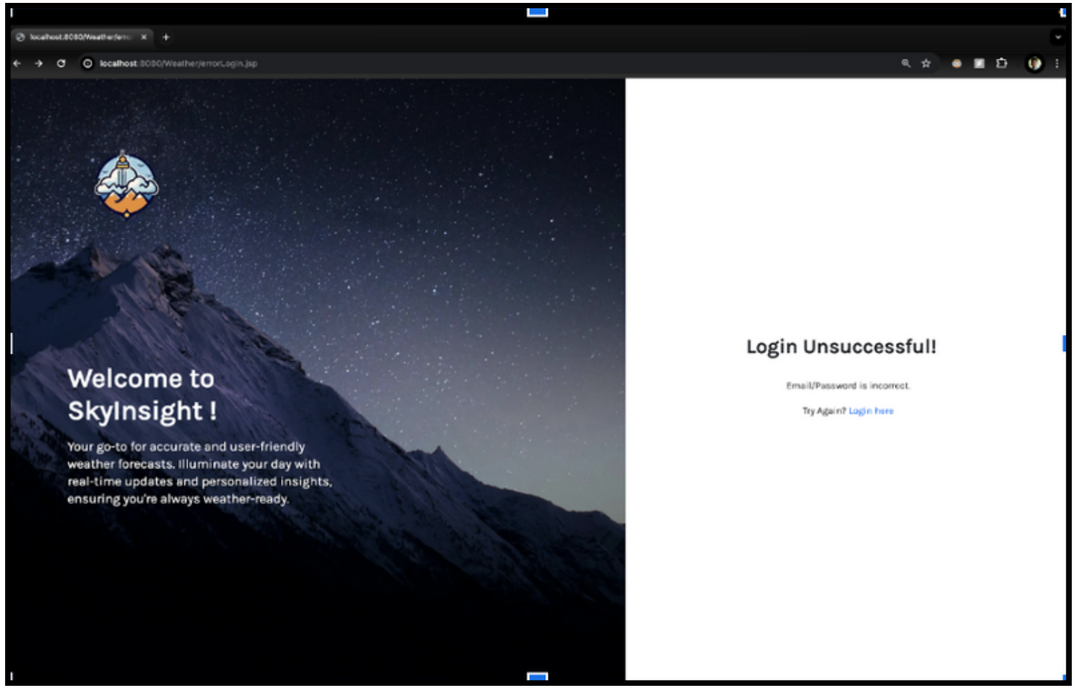
DEMO



Register an account using email ID,
Password and location info

Testcase 1:
Account Created Successfully



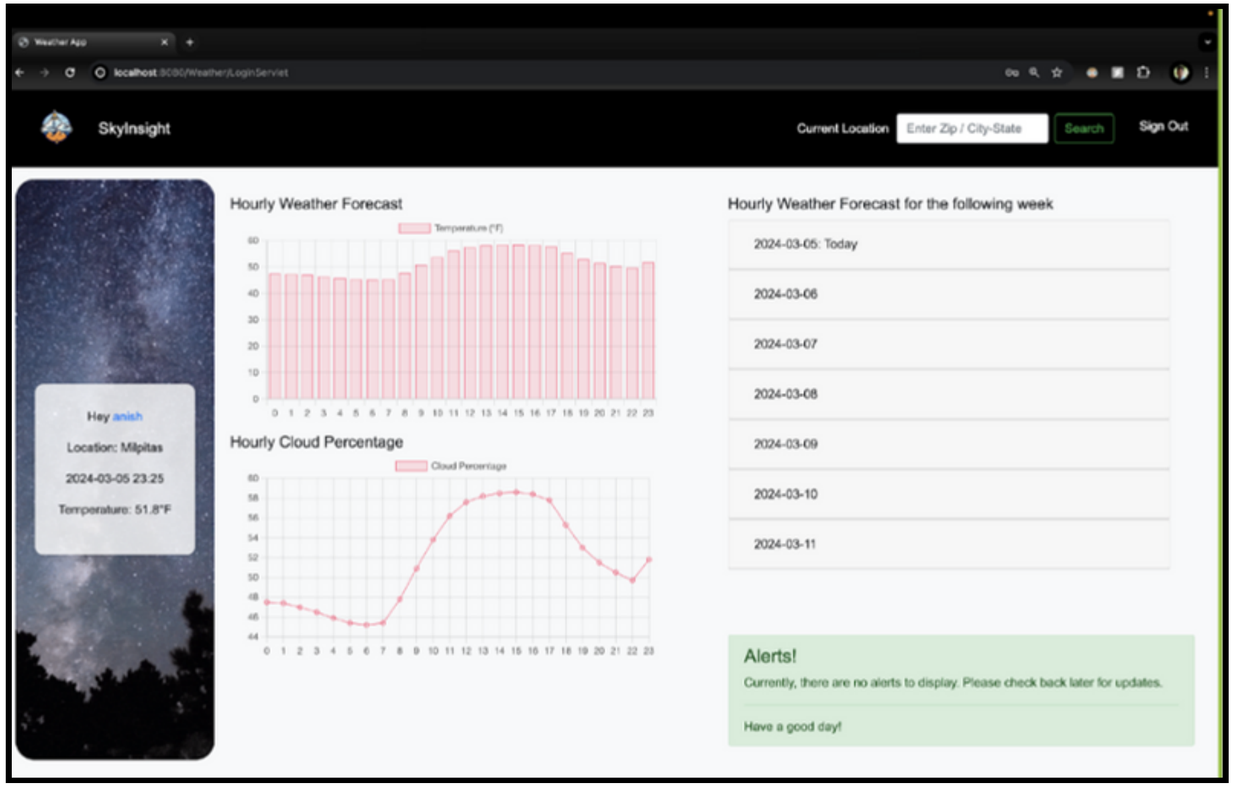


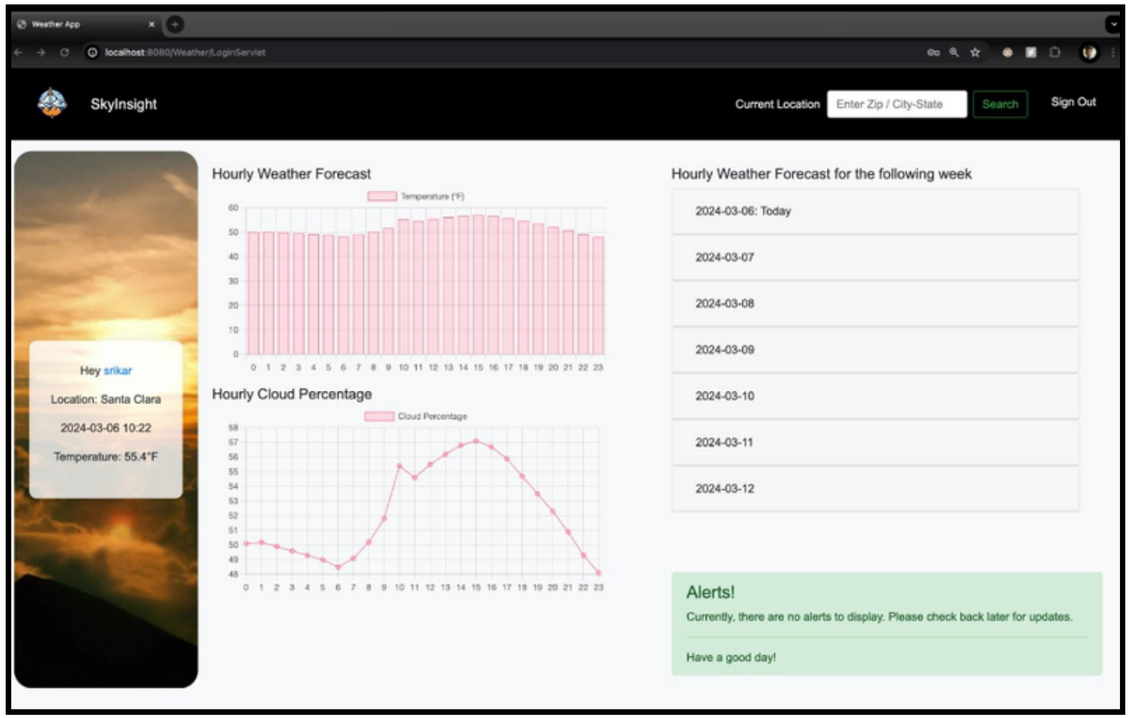
Testcase 2:

Account Creation failed as the email is already registered

Home Screen After Login:

It will show the forecast for the location you used while creating the account



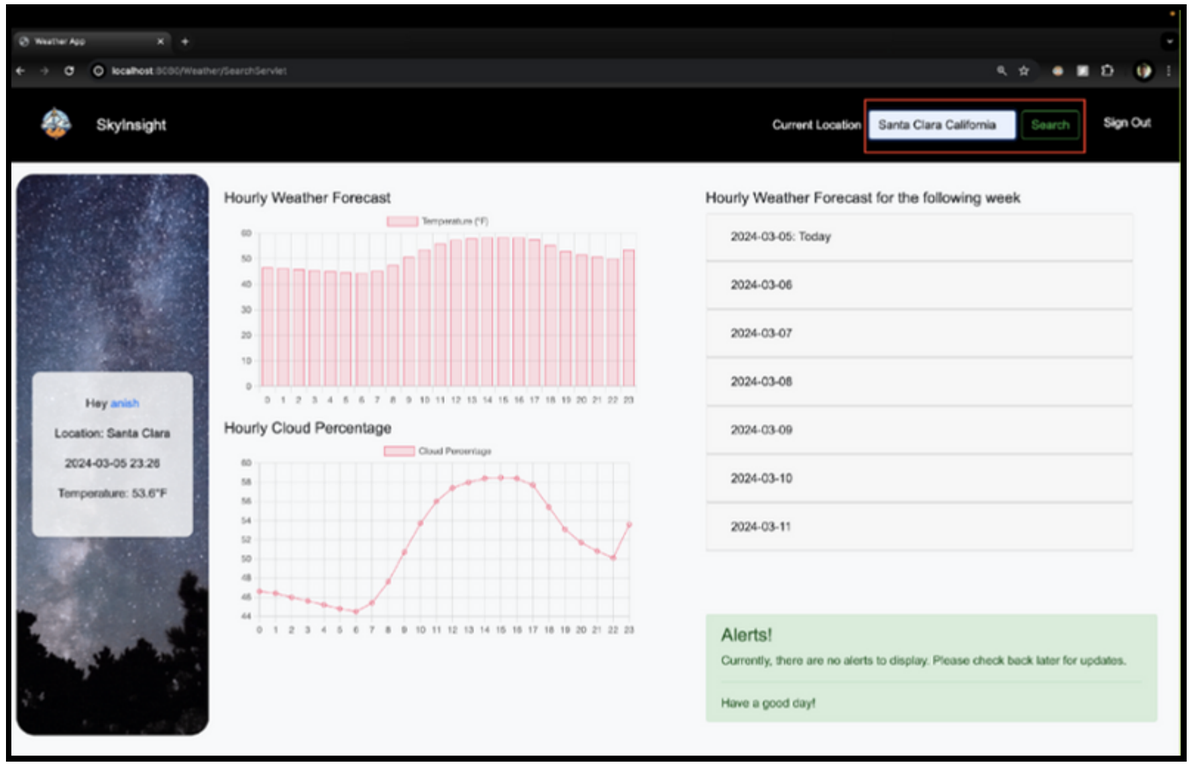


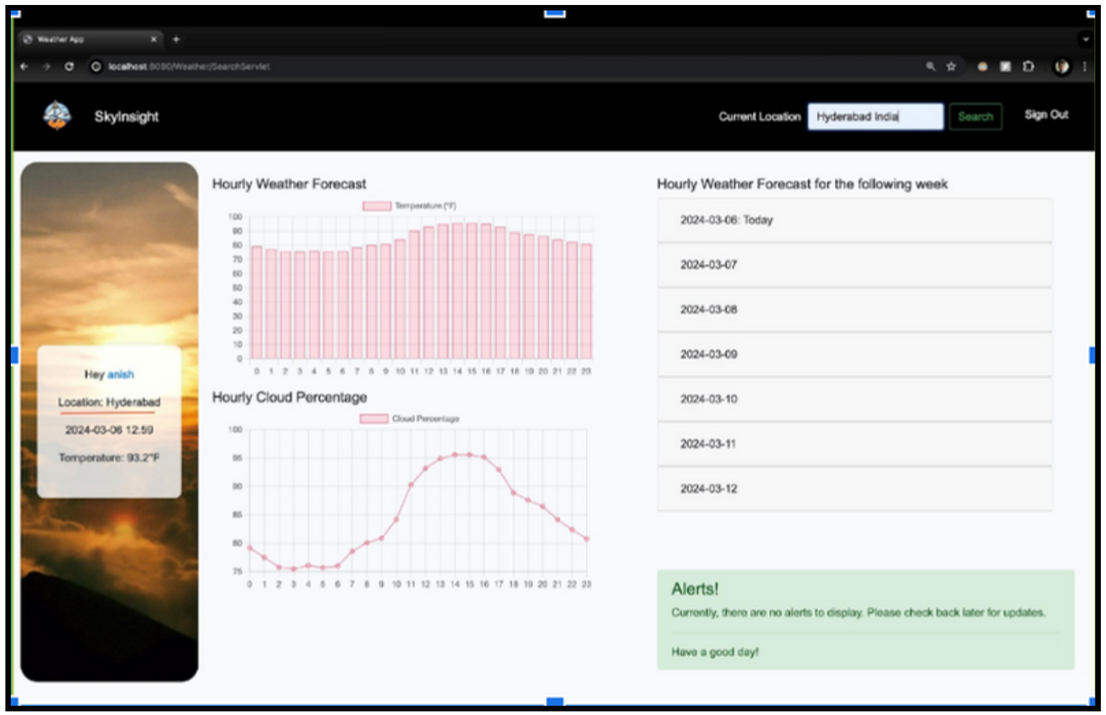
Home Screen After Login:

Location information is not provided while creating account, it will show current location's weather information

Searching for New Location:

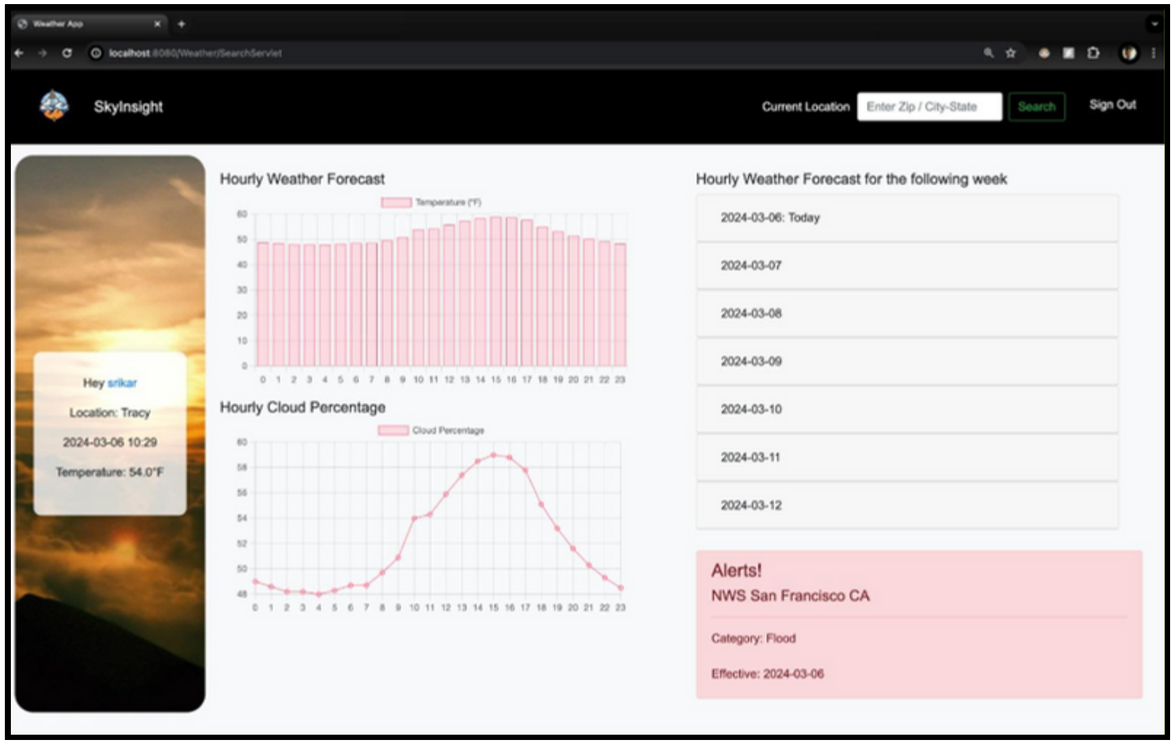
Santa Clara, CA

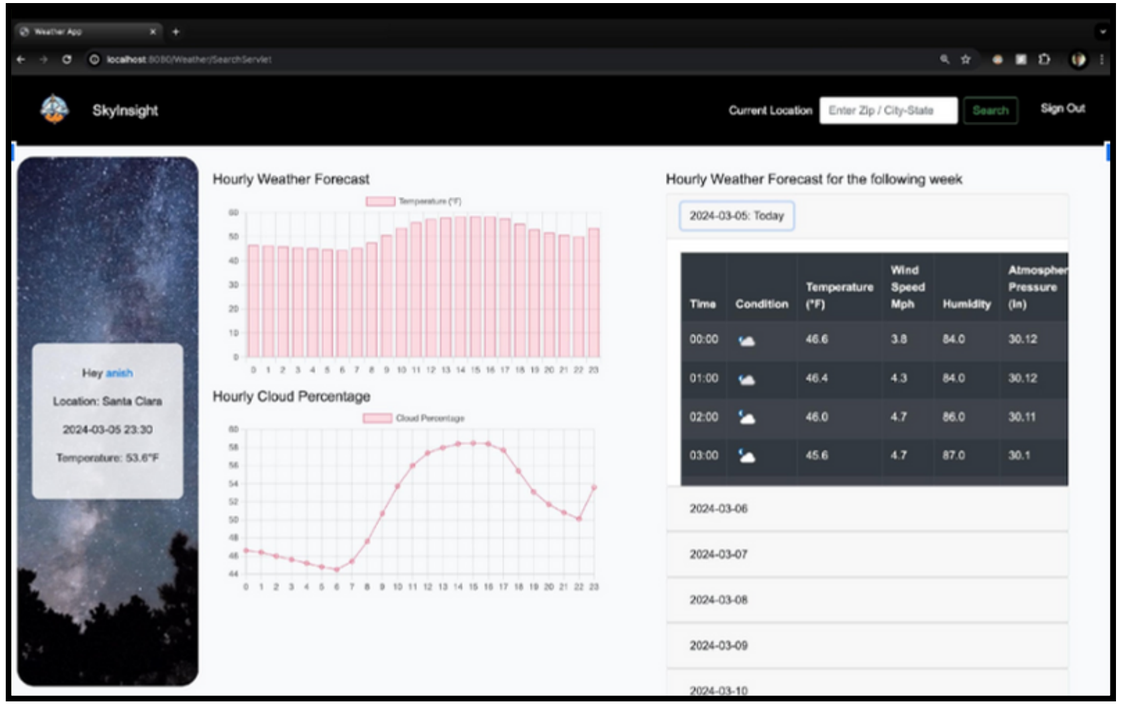




Searching for New Location:
Hyderabad, India

Severe Weather Alerts:
If weather warnings are present for the selected location



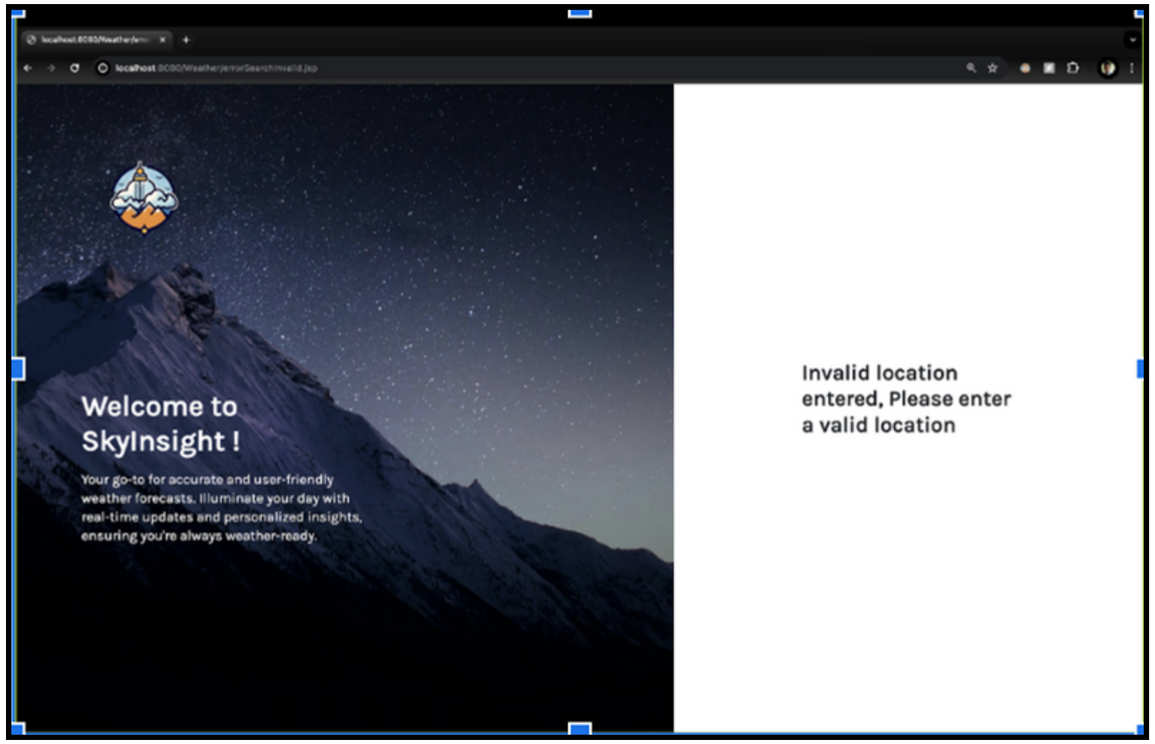


Multiday Forecast:

Shows hourly weather information such as Condition, Temperature, Wind Speed, Humidity, Atmospheric Pressure, Precipitation

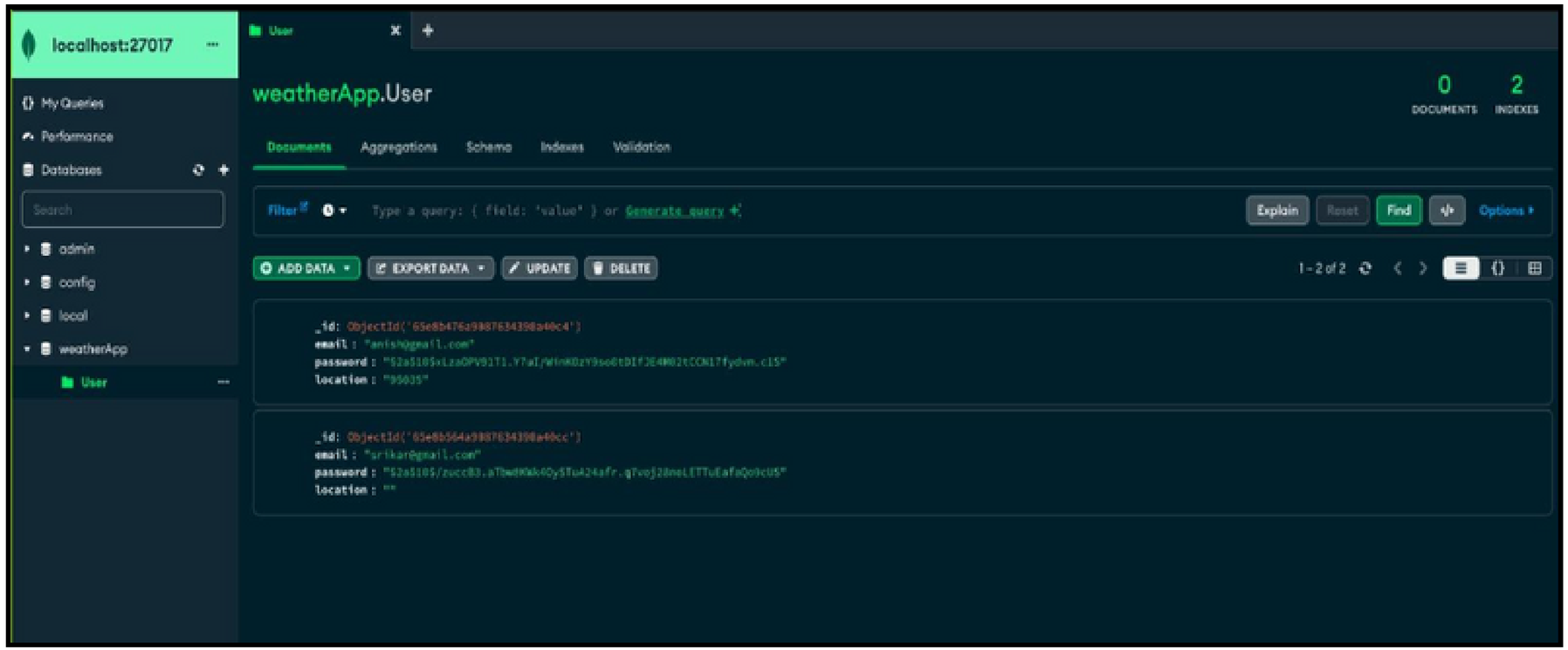
Invalid Location:

If you enter an invalid location while creating an account



Database:

MongoDB Database Snapshot



QR Code

You are about to visit:

3164-129-210-115-226.ngrok-free.app

Website IP: 129.210.115.226

- This website is served for free through ngrok.com.
- You should only visit this website if you trust whoever sent the link to you.
- Be careful about disclosing personal or financial information like passwords, phone numbers, or credit cards.

Visit Site

Thank You !