

Final Project Documentation

ForecastNewsFeed

Objective

The primary objective of this project is to create a web application that utilizes WebSockets for real-time communication and integrates weather updates and news headlines.

Functionalities and Features

- ◆ Real-time Communication: Enables live data exchange between clients and the server using WebSockets.
- ◆ Weather Updates: Fetches and displays real-time weather information.
- ◆ News Headlines: Fetches and displays the latest news headlines
- ◆ User Interface: Provides a web-based interface for users to interact with the application.
- ◆ Styling: Includes CSS for a responsive and visually appealing interface.

INDEX.HTML:

- ◆ Sets up the basic structure of the web page, including sections for real-time communication, weather, and news
- ◆ **Dropdown for Country Selection.** The select element allows the user to choose a country. The id attribute locations are used to identify and manipulate this element in JavaScript.
- ◆ **Weather and News Sections.** These sections (#weather-data and #news-data) will be populated dynamically with weather and news data fetched using APIs.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dynamic Web App</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="app">
    <h1>Weather and News Updates</h1>

    <div id="location-selector">
```

```

<label for="locations">Select Country:</label>
<select id="locations">
  <option value="Australia">Australia</option>
  <option value="China">China</option>
  <option value="India">India</option>
  <option value="Japan">Japan</option>
  <option value="South Korea">South Korea</option>
  <option value="Indonesia">Indonesia</option>
  <option value="Malaysia">Malaysia</option>
  <option value="New Zealand">New Zealand</option>
  <option value="Philippines">Philippines</option>
  <option value="Singapore">Singapore</option>
  <option value="Thailand">Thailand</option>
  <option value="Vietnam">Vietnam</option>
</select>
</div>

<div id="weather">
  <h2>Current Weather</h2>
  <div id="weather-data"></div>
</div>
<div id="news">
  <h2>Latest News</h2>
  <div id="news-data"></div>
</div>
</div>
<script src="app.js" defer></script>
</body>
</html>

```

STYLE.CSS

- ◆ Provides styling for the main content, weather, and news sections
- ◆ Body Styling: The body styles ensure the entire page is centered and has a light background.

```

body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}

```

```
}

#app {
  background: white;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  width: 80%;
  max-width: 800px;
  padding: 20px;
  box-sizing: border-box;
}

h1 {
  text-align: center;
  color: #333;
}

#location-selector {
  margin-bottom: 20px;
  text-align: center;
}

#location-selector label {
  font-weight: bold;
  margin-right: 10px;
}

#locations {
  padding: 5px;
  font-size: 16px;
}

#weather, #news {
  margin-top: 20px;
}

#weather-data {
  background: #e0f7fa;
  padding: 15px;
  border-radius: 5px;
  text-align: center;
}

#news-data {
  margin-top: 20px;
}
```

```

}

.news-article {
  background: #ffffff;
  margin-bottom: 20px;
  padding: 15px;
  border-radius: 5px;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}

.news-article img {
  max-width: 100%;
  height: auto;
  border-radius: 5px;
  margin-top: 10px;
}

.news-article h3 {
  margin: 0;
  font-size: 18px;
  color: #00796b;
}

.news-article p {
  margin: 10px 0;
  color: #555;
}

```

APP.JS:

- ◆ Establishes a WebSocket connection to the server.
- ◆ Listens for the open event to send a greeting message to the server.
- ◆ **Country-Capital Mapping:** A dictionary maps each country to its capital city, used for fetching weather data.
- ◆ **Utility Function:** capitalizeFirstLetterOfEachWord capitalizes the first letter of each word in a string.
- ◆ **Display Functions:** displayWeather and displayNews functions update the DOM to display the fetched weather and news data.
- ◆ **Fetch Functions:**
 - fetchWithRetry handles API request retries.
 - fetchWeatherData fetches weather data from the OpenWeatherMap API and formats it.

- fetchNewsData fetches news data from the NewsAPI and displays it.
- ◆ **Initial Data Fetch:** When the page loads, it fetches weather and news data for the default selected country.

```
document.addEventListener('DOMContentLoaded', () => {
  const weatherDataDiv = document.getElementById('weather-data');
  const newsDataDiv = document.getElementById('news-data');
  const locationsDropdown = document.getElementById('locations');

  // Define a mapping of countries to their capital cities
  const countryCapitalMap = {
    "Australia": "Canberra",
    "China": "Beijing",
    "India": "New Delhi",
    "Japan": "Tokyo",
    "South Korea": "Seoul",
    "Indonesia": "Jakarta",
    "Malaysia": "Kuala Lumpur",
    "New Zealand": "Wellington",
    "Philippines": "Manila",
    "Singapore": "Singapore",
    "Thailand": "Bangkok",
    "Vietnam": "Hanoi"
  };

  function capitalizeFirstLetterOfEachWord(str) {
    return str.replace(/\b\w/g, char => char.toUpperCase());
  }

  function displayWeather(data) {
    const { temp, weather, forecast } = data;

    let weatherHtml = `
      <p>Temperature: ${temp}°C</p>
      <p>Weather: ${capitalizeFirstLetterOfEachWord(weather)}</p>
    `;

    if (forecast && forecast.length > 0) {
      weatherHtml += '<h3>Weekly Forecast</h3>';
      weatherHtml += '<div class="weekly-forecast">';
      forecast.forEach(day => {
        weatherHtml += `
          <div class="forecast-item">
            <p><strong>${day.day}</strong></p>
            <p>Temperature: ${day.temp}°C</p>
          `;
      });
    }
  }
});
```

```

        <p>Weather:
    ${capitalizeFirstLetterOfEachWord(day.weather)}</p>
    </div>
    `;
    });
    weatherHtml += '</div>';
}

weatherDataDiv.innerHTML = weatherHtml;
}

function displayNews(localArticles) {
    const filterArticlesWithImages = (articles) => {
        return articles.filter(article => article.urlToImage);
    };

    const createArticleHtml = (article) => {
        return `
            <div class="news-article">
                <h3><a href="${article.url}"
target="_blank">${article.title}</a></h3>
                ${article.description ? `<p>${article.description}</p>` : ''}
                
                <p>Published on: ${new
Date(article.publishedAt).toLocaleString()}</p>
                <p>Source: ${article.source.name}</p>
            </div>
        `;
    };

    const localNewsHtml =
filterArticlesWithImages(localArticles).map(createArticleHtml).join('');

    newsDataDiv.innerHTML = localNewsHtml;
}

async function fetchWithRetry(url, retries = 3, delay = 3000) {
    for (let i = 0; i < retries; i++) {
        const response = await fetch(url);
        if (response.ok) {
            return await response.json();
        } else if (response.status === 429 && i < retries - 1) {
            await new Promise(resolve => setTimeout(resolve, delay));
        } else {

```

```

        throw new Error(`Request failed with status ${response.status}`);
    }
}

}

async function fetchWeatherData(location) {
    try {
        const weatherData = await
fetchWithRetry(`https://api.openweathermap.org/data/2.5/forecast?q=${location}&ap
pid=987aee4e091d8eca75e5293553f6490c&units=metric`);
        const formattedData = formatWeatherData(weatherData);
        displayWeather(formattedData);
    } catch (error) {
        weatherDataDiv.innerHTML = `

Error fetching weather data:
${error.message}</p>`;
    }
}

function formatWeatherData(data) {
    const currentWeather = data.list[0];
    const currentTemp = currentWeather.main.temp;
    const currentWeatherDesc = currentWeather.weather[0].description;

    // Extract weekly forecast
    const weeklyForecast = [];
    let currentDate = null;
    data.list.forEach(weatherItem => {
        const itemDate = new Date(weatherItem.dt *
1000).toLocaleDateString('en-US', { weekday: 'long' });
        if (currentDate !== itemDate && new Date(weatherItem.dt *
1000).getHours() === 12) {
            weeklyForecast.push({
                day: itemDate,
                temp: weatherItem.main.temp,
                weather: weatherItem.weather[0].description
            });
            currentDate = itemDate;
        }
    });

    return {
        temp: currentTemp,
        weather: currentWeatherDesc,
        forecast: weeklyForecast
    };
}


```

```

    }

    async function fetchNewsData(country) {
        try {
            // Fetch local news specific to the selected country
            const localNewsData = await
fetchWithRetry(`https://newsapi.org/v2/everything?q=${country}&apiKey=ee87bf38206
f4de8bd380ab3c2fd46ca`);

            // Display local news
            displayNews(localNewsData.articles);
        } catch (error) {
            newsDataDiv.innerHTML = `

Error fetching news data:
${error.message}</p>`;
        }
    }

    locationsDropdown.addEventListener('change', () => {
        const selectedCountry = locationsDropdown.value;
        const capitalCity = countryCapitalMap[selectedCountry];
        fetchWeatherData(capitalCity);
        fetchNewsData(selectedCountry);
    });

    // Initial fetch for default location and news
    const defaultCountry = locationsDropdown.value;
    const defaultCapitalCity = countryCapitalMap[defaultCountry];
    fetchWeatherData(defaultCapitalCity);
    fetchNewsData(defaultCountry);
});


```


WEBSOCKET-SERVER.JS:

- ◆ Uses the ws library to create a WebSocket server listening on port 8080.
- ◆ Listens for incoming connections and messages.
- ◆ **Mock Updates:** Sends mock weather updates every 5 seconds and mock news updates every 10 seconds to connected clients.
- ◆ **Connection and Disconnection Handlers:** Logs messages when clients connect or disconnect.

```
const WebSocket = require('ws');

const server = new WebSocket.Server({ port: 8080 });

server.on('connection', socket => {
  console.log('Client connected');

  // Send mock weather updates for Manila
  setInterval(() => {
    const weatherUpdate = {
      type: 'weather-update',
      data: {
        temp: (Math.random() * 10 + 25).toFixed(1), // Simulate
temperature for Manila
        weather: ['Sunny', 'Cloudy',
'Rainy'][Math.floor(Math.random() * 3)] // Simulate weather
      }
    };
    socket.send(JSON.stringify(weatherUpdate));
  }, 5000);

  // Send mock news updates for the Philippines
  setInterval(() => {
    const newsUpdate = {
      type: 'news-update',
      data: [
        { title: 'Mock Philippine News 1', url: '#', publishedAt: new
Date().toISOString(), urlToImage: 'https://via.placeholder.com/150', description:
'This is a mock Philippine news update 1.' },
        { title: 'Mock Philippine News 2', url: '#', publishedAt: new
Date().toISOString(), urlToImage: 'https://via.placeholder.com/150', description:
'This is a mock Philippine news update 2.' }
      ]
    };
    socket.send(JSON.stringify(newsUpdate));
  }, 10000);
});
```

```
socket.on('close', () => {  
  console.log('Client disconnected');  
});  
});  
  
console.log('WebSocket server running on ws://localhost:8080');
```