1. **Utworzenie potrzebnych kontrolerów dla stron**

   - *WeatherController.java*

```
package com.pbs.edu.webapp.controller;

import com.pbs.edu.webapp.model.User;
import com.pbs.edu.webapp.model.WeatherData;
import com.pbs.edu.webapp.service.AuthService;
import com.pbs.edu.webapp.service.WeatherServiceClient;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import jakarta.servlet.http.HttpSession;
import java.util.List;

@Controller
public class WeatherController {

    private final WeatherServiceClient weatherServiceClient;
    private final AuthService authService;

    public WeatherController(WeatherServiceClient weatherServiceClient,
AuthService authService) {
        this.weatherServiceClient = weatherServiceClient;
        this.authService = authService;
    }

    @GetMapping("/weather")
    public String getWeather(@RequestParam(value = "city", required =
false) String city,
                             @RequestParam(value = "country", required =
false) String country,
                             Model model,
                             HttpSession session) {
        User authenticatedUser = (User) session.getAttribute("user");
        if (authenticatedUser == null) {
            return "redirect:/login"; // Przekierowanie na stronę
logowania, jeśli użytkownik nie jest zalogowany
        }

        if (city == null || country == null) {
            city = authenticatedUser.getCity();
            country = authenticatedUser.getCountry();
        }

        WeatherData currentWeather =
weatherServiceClient.getCurrentWeather(city, country);
        model.addAttribute("currentWeather", currentWeather);
        model.addAttribute("cityName", city);
        model.addAttribute("user", session.getAttribute("user"));
        return "weather";
    }

    @GetMapping("/history")
    public String getWeatherHistory(@RequestParam(value = "city",
required = false) String city,
                                    @RequestParam(value = "country",
required = false) String country,
                                    Model model,
                                    HttpSession session) {
        User authenticatedUser = (User) session.getAttribute("user");
        if (authenticatedUser == null) {
            return "redirect:/login"; // Przekierowanie na stronę
logowania, jeśli użytkownik nie jest zalogowany
        }

        if (city == null || country == null) {
            city = authenticatedUser.getCity();
            country = authenticatedUser.getCountry();
        }

        List<WeatherData> historicalWeather =
weatherServiceClient.getHistoricalWeather(city, country);
        model.addAttribute("historicalWeather", historicalWeather);
        model.addAttribute("cityName", city);
        model.addAttribute("user", session.getAttribute("user"));
        return "history";
    }
}
```

- *NotificationsController.java*

```
package com.pbs.edu.webapp.controller;

import com.pbs.edu.webapp.model.NotificationRequest;
import com.pbs.edu.webapp.model.User;
import com.pbs.edu.webapp.service.NotificationServiceClient;
import com.pbs.edu.webapp.service.UserServiceClient;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import jakarta.servlet.http.HttpSession;
import java.util.List;

@Controller
public class NotificationsController {

    private final NotificationServiceClient notificationServiceClient;
    private final UserServiceClient userService;

    public NotificationsController(NotificationServiceClient
notificationsServiceClient, UserServiceClient userService) {
        this.notificationServiceClient = notificationsServiceClient;
        this.userService = userService;
    }

    @GetMapping("/notifications")
    public String getNotifications(Model model, HttpSession session) {
        User user = (User) session.getAttribute("user");
        if (user == null) {
            return "redirect:/login";
        }
        String userEmail = user.getEmail();
        List<NotificationRequest> notifications =
notificationServiceClient.getNotificationsForUser(userEmail);
        model.addAttribute("notifications", notifications);
        model.addAttribute("user", session.getAttribute("user"));
        return "notifications";
    }

}
```

2. Utworzenie plików html będących templateami oraz modyfikacja pliku css

- *weather.html*

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Current Weather</title>
    <link th:href="@{/motyw.css}" rel="stylesheet">
</head>
<body>
<nav>
    <ul class="nav-menu">
        <li><a href="/">Home</a></li>
        <li><a href="/weather">Current Weather</a></li>
        <li><a href="/history">Historical Weather</a></li>
        <li><a href="/settings">Settings</a></li>
        <li><a href="/notifications">Notifications</a></li>
        <li></li>
        <li></li>
        <li></li>
        <li></li>
        <li th:if="${user != null}"><p th:text="${user.email}"></p></li>
        <li th:if="${user != null}"><a href="/logout">Logout</a></li>
    </ul>
</nav>
<br>
<h1>Weather Data</h1>

<!-- Formularz do sprawdzenia pogody dla wybranego miasta -->
<form method="get" action="/weather">
    <label for="city">City:</label>
    <input type="text" id="city" name="city" required>
    <label for="country">Country:</label>
    <input type="text" id="country" name="country" required>
    <button type="submit">Submit</button>
</form>

<h2>Current Weather Data</h2>
<div th:if="${currentWeather != null}">
    <h3>Weather for your city: <span style="color: #00FFFF;"
th:text="${cityName}"></span></h3>
    <p>Temperature: <span th:text="${currentWeather.temp}"></span> °C</p>
    <p>Feels Like: <span th:text="${currentWeather.feelsLike}"></span>
°C</p>
    <p>Min Temperature: <span th:text="${currentWeather.tempMin}"></span>
°C</p>
    <p>Max Temperature: <span th:text="${currentWeather.tempMax}"></span>
°C</p>
    <p>Pressure: <span th:text="${currentWeather.pressure}"></span>
hPa</p>
    <p>Humidity: <span th:text="${currentWeather.humidity}"></span>%</p>
    <p>Wind Speed: <span th:text="${currentWeather.windSpeed}"></span>
m/s</p>
```

```
    <p>Wind Direction: <span
th:text="${currentWeather.windDeg}"></span>°</p>
    <p>Cloudiness: <span th:text="${currentWeather.clouds}"></span>%</p>
    <p>Visibility: <span th:text="${currentWeather.visibility}"></span>
m</p>
</div>

<div id="openweathermap-widget-21"></div>
<script
src="//openweathermap.org/themes/openweathermap/assets/vendor/owm/js/d3.m
in.js"></script>
<script th:inline="javascript">
    const apiKey = '353dc6ca3e2e48a7296714f30d5f88c4';

    // Dane dynamiczne: nazwa miasta, kraj lub współrzędne
    const cityName = /*[[${cityName}]]*/;
    const countryCode = /*[[${currentWeather.country}]]*/;

    // Funkcja pobierająca City ID z OpenWeatherMap API
    async function fetchCityId(cityName, countryCode) {
        const url =
`https://api.openweathermap.org/data/2.5/weather?q=${cityName},${countryC
ode}&appid=${apiKey}`;
        const response = await fetch(url);
        const data = await response.json();
        return data.id;
    }

    // Dodanie widgetu do strony
    async function loadWeatherWidget() {
        const cityId = await fetchCityId(cityName, countryCode);

        // Parametry widgetu
        window.myWidgetParam = window.myWidgetParam || [];
        window.myWidgetParam.push({
            id: 21,
            cityid: cityId,
            appid: apiKey,
            units: 'metric',
            containerid: 'openweathermap-widget-21',
        });

        // Ładowanie skryptu widgetu
        const script = document.createElement('script');
        script.async = true;
        script.charset = 'utf-8';
        script.src =
'//openweathermap.org/themes/openweathermap/assets/vendor/owm/js/weather-
widget-generator.js';
        document.getElementsByTagName('head')[0].appendChild(script);
    }

    // Inicjalizacja
    loadWeatherWidget();
</script>
</body>
</html>
```

- *notifications.html*

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Notifications</title>
    <link th:href="@{/motyw.css}" rel="stylesheet">
</head>
<body>
<nav>
    <ul class="nav-menu">
        <li><a href="/">Home</a></li>
        <li><a href="/weather">Current Weather</a></li>
        <li><a href="/history">Historical Weather</a></li>
        <li><a href="/settings">Settings</a></li>
        <li><a href="/notifications">Notifications</a></li>
        <li></li>
        <li></li>
        <li></li>
        <li></li>
        <li th:if="${user != null}"><p th:text="${user.email}"></p></li>
        <li th:if="${user != null}"><a href="/logout">Logout</a></li>
    </ul>
</nav>
<br>
<h1>Your Notifications</h1>

<ul class="wthrData">
    <li th:each="notification : ${notifications}">
        <p>
            <strong>Email:</strong> <span
th:text="${notification.email}"></span><br>
            <strong>Subject:</strong> <span
th:text="${notification.subject}"></span><br>
            <strong>Message:</strong> <span
th:text="${notification.message}"></span><br>
            <strong>Time:</strong> <span
th:text="${notification.timestamp}"></span><br>
            <strong>Sent:</strong> <span
th:text="${notification.sent}"></span><br>
        </p>
    </li>
</ul>
</body>
</html>
```

- *history.html*

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Historical Weather Data</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <link th:href="@{/motyw.css}" rel="stylesheet">
</head>
<body>
<nav>
    <ul class="nav-menu">
        <li><a href="/">Home</a></li>
        <li><a href="/weather">Current Weather</a></li>
        <li><a href="/history">Historical Weather</a></li>
        <li><a href="/settings">Settings</a></li>
        <li><a href="/notifications">Notifications</a></li>
        <li></li>
        <li></li>
        <li></li>
        <li></li>
        <li th:if="${user != null}"><p th:text="${user.email}"></p></li>
        <li th:if="${user != null}"><a href="/logout">Logout</a></li>
    </ul>
</nav>
<br>
<h1>Historical Weather Data</h1>

<form method="get" action="/history">
    <label for="city">City:</label>
    <input type="text" id="city" name="city" required>
    <label for="country">Country:</label>
    <input type="text" id="country" name="country" required>
    <button type="submit">Submit</button>
</form>

<h2>Historical Data for city: <span style="color: #00FFFF;"
th:text="${cityName}"></span></h2>
<ul class="wthrData">
    <li th:each="data : ${historicalWeather}">
        <span th:text="${#temporals.format(data.date, 'yyyy-MM-dd
HH:mm:ss')}"></span> -
        Temperature: <span th:text="${data.temp}"></span> °C,
        Feels Like: <span th:text="${data.feelsLike}"></span> °C,
        Pressure: <span th:text="${data.pressure}"></span> hPa,
        Wind Speed: <span th:text="${data.windSpeed}"></span> m/s,
        Humidity: <span th:text="${data.humidity}"></span>%
    </li>
</ul>
<br>
<br>
<!-- Sekcja wykresów -->
<div>
    <h2 for="chartSelect">Select a chart:</h2>
    <select id="chartSelect">
        <option value="temperature">Temperature Chart</option>
        <option value="feelsLike">Feels Like Temperature Chart</option>
        <option value="pressure">Pressure Chart</option>
        <option value="wind">Wind Speed Chart</option>
        <option value="humidity">Humidity Chart</option>
    </select>
</div>

<br>
<h2 id="temperatureLabel" style="text-align: center; display:
block;">Temperature Chart</h2>
<canvas id="temperatureChart" width="400" height="200" style="display:
block;"></canvas>

<h2 id="feelsLikeLabel" style="text-align: center; display: none;">Feels
Like Temperature Chart</h2>
<canvas id="feelsLikeChart" width="400" height="200" style="display:
none;"></canvas>
```

```
<h2 id="pressureLabel" style="text-align: center; display:
none;">Pressure Chart</h2>
<canvas id="pressureChart" width="400" height="200" style="display:
none;"></canvas>

<h2 id="windLabel" style="text-align: center; display: none;">Wind Speed
Chart</h2>
<canvas id="windChart" width="400" height="200" style="display:
none;"></canvas>

<h2 id="humidityLabel" style="text-align: center; display:
none;">Humidity Chart</h2>
<canvas id="humidityChart" width="400" height="200" style="display:
none;"></canvas>


<script th:inline="javascript">
    const historicalWeather = /*[[${historicalWeather}]]*/ [];

    // Dane do wykresów
    const labels = historicalWeather.map(data => new
Date(data.date).toLocaleString());
    const temperatures = historicalWeather.map(data => data.temp);
    const feelsLikeTemps = historicalWeather.map(data => data.feelsLike);
    const pressures = historicalWeather.map(data => data.pressure);
    const windSpeeds = historicalWeather.map(data => data.windSpeed);
    const humidities = historicalWeather.map(data => data.humidity);


   const ctxTemperature =
document.getElementById('temperatureChart').getContext('2d');
    new Chart(ctxTemperature, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Temperature (°C)',
            data: temperatures,
            borderColor: 'rgba(255, 99, 132, 1)',
            backgroundColor: 'rgba(255, 99, 132, 0.2)',
            borderWidth: 1,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: {
                display: true,
                position: 'top'
            },
            tooltip: {
                mode: 'nearest',
                intersect: false
            }
        },
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Date & Time'
                },
                ticks: {
                    display: false
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Temperature (°C)'
                }
            }
        }
    }
});

const ctxFeelsLike =
document.getElementById('feelsLikeChart').getContext('2d');
new Chart(ctxFeelsLike, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Feels Like Temperature (°C)',
            data: feelsLikeTemps,
            borderColor: 'rgba(255, 159, 64, 1)',
            backgroundColor: 'rgba(255, 159, 64, 0.2)',
            borderWidth: 1,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: {
                display: true,
                position: 'top'
            },
            tooltip: {
                mode: 'nearest',
                intersect: false
            }
        },
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Date & Time'
                },
                ticks: {
                    display: false
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Feels Like Temperature (°C)'
                }
            }
        }
    }
});
```

```
const ctxPressure =
document.getElementById('pressureChart').getContext('2d');
new Chart(ctxPressure, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Pressure (hPa)',
            data: pressures,
            borderColor: 'rgba(153, 102, 255, 1)',
            backgroundColor: 'rgba(153, 102, 255, 0.2)',
            borderWidth: 1,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: {
                display: true,
                position: 'top'
            },
            tooltip: {
                mode: 'nearest',
                intersect: false
            }
        },
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Date & Time'
                },
                ticks: {
                    display: false
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Pressure (hPa)'
                }
            }
        }
    }
});

const ctxWind = document.getElementById('windChart').getContext('2d');
new Chart(ctxWind, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Wind Speed (m/s)',
            data: windSpeeds,
            borderColor: 'rgba(54, 162, 235, 1)',
            backgroundColor: 'rgba(54, 162, 235, 0.2)',
            borderWidth: 1,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: {
                display: true,
                position: 'top'
            },
            tooltip: {
                mode: 'nearest',
                intersect: false
            }
        },
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Date & Time'
                },
                ticks: {
                    display: false
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Wind Speed (m/s)'
                }
            }
        }
    }
});

const ctxHumidity =
document.getElementById('humidityChart').getContext('2d');
new Chart(ctxHumidity, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Humidity (%)',
            data: humidities,
            borderColor: 'rgba(75, 192, 192, 1)',
            backgroundColor: 'rgba(75, 192, 192, 0.2)',
            borderWidth: 1,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: {
                display: true,
                position: 'top'
            },
            tooltip: {
                mode: 'nearest',
                intersect: false
            }
        },
        scales: {
            x: {
                title: {
                    display: true,
                    text: 'Date & Time'
                },
                ticks: {
                    display: false
                }
            },
            y: {
                title: {
                    display: true,
                    text: 'Humidity (%)'
                }
            }
        }
    }
});

     document.getElementById('chartSelect').addEventListener('change',
function() {
        const selectedChart = this.value;

        // Hide all charts
        document.getElementById('temperatureChart').style.display =
'none';
        document.getElementById('feelsLikeChart').style.display = 'none';
        document.getElementById('pressureChart').style.display = 'none';
        document.getElementById('windChart').style.display = 'none';
        document.getElementById('humidityChart').style.display = 'none';

        // Hide all labels
        document.getElementById('temperatureLabel').style.display =
'none';
        document.getElementById('feelsLikeLabel').style.display = 'none';
        document.getElementById('pressureLabel').style.display = 'none';
        document.getElementById('windLabel').style.display = 'none';
        document.getElementById('humidityLabel').style.display = 'none';

        // Show selected chart & label
        if (selectedChart === 'temperature') {
            document.getElementById('temperatureChart').style.display =
'block';
            document.getElementById('temperatureLabel').style.display =
'block';
        } else if (selectedChart === 'feelsLike') {
            document.getElementById('feelsLikeChart').style.display =
'block';
            document.getElementById('feelsLikeLabel').style.display =
'block';
        } else if (selectedChart === 'pressure') {
            document.getElementById('pressureChart').style.display =
'block';
            document.getElementById('pressureLabel').style.display =
'block';
        } else if (selectedChart === 'wind') {
            document.getElementById('windChart').style.display = 'block';
            document.getElementById('windLabel').style.display = 'block';
        } else if (selectedChart === 'humidity') {
            document.getElementById('humidityChart').style.display =
'block';
```

```
            document.getElementById('humidityLabel').style.display =
'block';
        }
    });
</script>
</body>
</html>
```

- *motyw.css*

```
body, h1, h2, h3, p, ul, li {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    background-color: #121212;
    color: #FFFFFF;
    font-family: 'Arial', sans-serif;
    padding: 20px;
}

h1 {
    color: #FFFFFF;
    margin-bottom: 10px;
    text-align: center;
}
h2, h3 {
    color: #FFFFFF;
    margin-bottom: 10px;
}

nav {
    background-color: #1e1e1e;
    padding: 10px;
}

.nav-menu {
    list-style-type: none;
    padding: 0;
    display: flex;
    justify-content: center;
    background-color: #333;
}

.nav-menu li {
    margin: 0 15px;
}

.nav-menu p {
    padding: 10px;
    display: block;
    font-weight: bold;
    color: #00FFFF;
}

.nav-menu a {
    color: #f1f1f1;
```

```css
    text-decoration: none;
    padding: 10px;
    display: block;
}

.nav-menu a:hover {
    background-color: #575757;
    color: white;
}

form {
    background-color: #1E1E1E;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);
    max-width: 400px;
    margin: 20px auto;
}

label {
    display: block;
    margin-bottom: 5px;
    color: #FFFFFF;
}

input[type="text"], input[type="password"], input[type="email"], select,
button {
    width: 100%;
    padding: 6px;
    border: 1px solid #555555;
    border-radius: 4px;
    background-color: #2D2D2D;
    color: #FFFFFF;
    font-size: 16px;
    margin-bottom: 15px;
}

button {
    background-color: #87CEFA;
    color: white;
    border: none;
    padding: 12px;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    text-align: center;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #ADD8E6;
}

.wthrData ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

.wthrData {
    background-color: #1e1e1e;
    padding: 10px 15px;
    border-radius: 4px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    max-height: 500px;
    overflow-y: auto;
    transition: background-color 0.3s, transform 0.2s;
    margin-bottom: 8px;
}

.wthrData li {
    background-color: #1e1e1e;
    padding: 10px 15px;
    border-radius: 4px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    transition: background-color 0.3s, transform 0.2s;
}

.wthrData li:hover {
    background-color: #575757;
    color: white;
    transform: translateY(-2px);
}

select {
    background-color: #1E1E1E;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);
    max-width: 400px;
    margin: 20px auto;
}


/* Wykresy */
canvas {
    border: 1px solid;
    border-radius: 5px;
    border-color: #FFFFFF;
}

h2.chart-title {
    color: #FFFFFF;
    margin-bottom: 10px;
    text-align: center;
}

.chartjs-tooltip {
    background-color: #333333;
    color: #FFFFFF;
    border-radius: 5px;
    padding: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);
}

.chartjs-legend {
    background-color: #2D2D2D;
    color: #FFFFFF;
    padding: 10px;
    border-radius: 5px;
}
```

```
#openweathermap-widget-21 {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    width: 100%;
    margin: 0 auto;
    padding: 20px;
    box-sizing: border-box;
}

#errorDiv {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    width: 100%;
    margin: 0 auto;
    padding: 20px;
    box-sizing: border-box;
}
```

3. **Uruchomienie aplikacji.**

```
docker-compose up --build
```

4. **Uruchom przeglądarkę**

Przejdź pod adres: http://localhost:8083/ i sprawdź czy aplikacja działa poprawnie.