I. Special Topic Content and Technical Requirements

Front-end Technology

The project is a practical website developed using React. The application is a Single Page Application (SPA) built with Vite and styled using Tailwind CSS. It is ready for deployment to any cloud platform that supports modern JavaScript applications (e.g., Vercel, Netlify).

Back-end / API Integration

The application uses two different external APIs to provide its core functionality:

1. OpenWeather API:

   - Role: This API is used to fetch real-time weather data for any city the user inputs. The data includes temperature, weather conditions, humidity, and wind speed.

   - How it helps: Provides the essential data for the application to generate accurate outfit recommendations.

2. Google Gemini API:

   - Role: Acts as the “brain” of the app. It receives structured weather data and generates outfit recommendations.

   - How it helps: Translates raw weather data into actionable advice to help users decide what to wear.

Awareness of Thematic Issues

- What problem needs to be solved?

  Many people still struggle to choose the right outfit even after checking the weather. The app solves this by giving concrete outfit suggestions.

- Who are the target users?

Anyone who gets dressed daily, especially those in unpredictable climates, travelers, or users wanting to save time.

- Why the functions help:

The app automates decision-making, providing instantly usable outfit recommendations rather than just data.

II. Demo Video Content

Special Motivation and Problem Description

The project was inspired by the common experience of checking the weather but still choosing the wrong outfit. WeatherWear aims to eliminate this daily confusion.

Solution and Function Introduction

1. User enters a city.

2. App fetches real-time weather data.

3. Gemini generates outfit recommendations based on that data.

Technical Architecture and Code Implementation Process

- Overall Architecture:

  1. User inputs city.

  2. React fetches data from OpenWeather API.

  3. Weather data is sent to Gemini API.

  4. Gemini returns recommendations.

  5. React displays weather + outfit suggestion.

- Key Code:

WeatherWearApp.jsx contains main logic.

fetchWeather handles:

- API calls
- Error handling
- Prompt generation
- Gemini call
- State updates

Website Practical Operation Demo

1. User opens homepage.
2. Navigates to main app.
3. Types a city (e.g., Paris).
4. Loading indicator appears.
5. Weather card + outfit recommendation card appear.
6. User can search again.

Conclusion and Future Development

Potential improvements:

- User style preferences
- Activity-based suggestions
- API key security using backend/serverless functions

Future functions:

- Multi-day forecast
- User accounts to save locations
- Wardrobe integration for personalized outfits