WEAR: WEather Attire Recommender

Electrical and Computer Engineering Department

Capstone Design Project II:

COE 596

Project I Proposal Report

Supervised by Dr. Maria Abi Saad

Rabih Ayoub (201903639)

Ghassan Awwad (201904117)

Karim Hamawi (201602739)

Abstract

The difficulty of choosing the ideal attire based on one's preferences and the weather conditions is a common worry in today's fast-paced world. In response, WEAR (WEather Attire Recommender) presents a creative solution in the form of an intelligent outfit suggestion system that aims to streamline and improve the wardrobe picking process. WEAR is a novel combination of technology and personal fashion intelligence that makes use of machine learning algorithms, calendar scheduling, user preferences, and weather forecasts. This project elaborates on the WEAR software implementations, system design, hardware requirements, and restrictions. By taking these factors into account, WEAR is presented as a user-focused solution ready to simplify the outfit selection process and offer a more practical and customized approach to everyday clothing decisions.

In this project, we created a sophisticated system that combines a set of technological elements to provide outfit recommendations using up-to-date data and user inputs. We employed machine learning algorithms, such as K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naive Bayes, to examine weather conditions and user preferences in order to provide precise recommendations for outfits. The system utilizes the OpenWeatherMap API to retrieve real-time weather information and the FullCalendar API to handle user events. In addition, a DHT22 sensor offers instantaneous temperature and humidity measurements, guaranteeing the system's operation even in the absence of internet connectivity. The Python-based backend manages data preprocessing, model training, and API interactions, while the frontend, created using HTML, CSS, and JavaScript, provides a user-friendly interface for managing clothes inventories and scheduling events. The incorporation of a text-to-speech functionality improves accessibility, particularly for individuals with visual impairments. The Naive Bayes model achieved an accuracy score of 65%, the SVM model achieved 82%, and the KNN model achieved 77.5%. These results demonstrate the system's ability in giving personalized and practical outfit recommendations. By implementing these advancements, WEAR effectively diminishes the duration and exertion necessary for daily clothing choices, rendering it a helpful tool for modern life.

Contents

Table of Figures

List of Tables

1 Introduction

Many individuals face the same difficulty in the rush of modern life: the routine of selecting the perfect attire that complements their unique style and the weather. According to research by London-based store Marks & Spencer [8], the amount of time we spend considering our wardrobe choices—whether for work, school, or social engagements—can add up to months over the course of a lifetime. Between the ages of 18 and 60, men lose about 13 minutes a day, or four months of their life, while women lose about 17 minutes a day, or six months.

1.1 Motivation

In this context, WEAR (WEather Attire Recommender) emerges as a shining example of creativity, providing a revolutionary answer to the age-old problem of what to wear. WEAR is a revolution in wardrobe management, not merely a clever outfit recommendation engine. By utilizing weather forecasts, calendar schedules, user preferences, and sophisticated machine learning algorithms, this innovative combination of technology and personalized fashion intelligence streamlines the wardrobe selection process and drastically cuts down on the time and effort required to make wardrobe decisions.

1.2 Project Scope

The WEAR project's scope includes developing an advanced gadget that can be installed inside a wardrobe. This device has an intuitive interface that suggests outfits based on user preferences and a variety of parameters, including daily activities, the weather, and individual tastes. The suggested outfits are taken from a SQL database that stores every item of clothing that a person owns. The system's interface with the online FullCalendar, temperature and humidity sensors, and access to local weather forecasts are its key components. Creating an intuitive and accessible user interface (UI) is another aspect of the project that will enable smooth interaction for users of all ages and abilities, including those who are visually impaired. To further aid visually impaired people, we have incorporated a text-to-speech feature that reads out the proposed outfits.

1.3 Target Audience

WEAR is made to appeal to a wide range of consumers. This applies to kids, teens, and adults of both sexes, as choosing an appropriate wardrobe for each day is a universal need. Additionally, the approach takes into account people with visual impairments, such as colorblind or totally blind people. The device's user-friendly UI and accessibility features make it suitable for people with visual impairments. Therefore, WEAR makes sure that everyone benefits from the time and effort savings associated with streamlining the outfit selection process.

1.4 Limitations

Although the WEAR system takes a unique approach to outfit selection, it is not without its limits. Its dependency on the user's current wardrobe database is a major drawback. The efficiency and user satisfaction of the system may be negatively impacted by its inability to provide diverse and fitting outfit choices for people who own a restricted wardrobe. Furthermore, shoes and accessories are important components of an ensemble that can greatly influence how well-fitting it is for a given occasion and how well-rounded it looks. These are not taken into consideration by the present version of the system. A further constraint pertains to the possibility of imprecision in weather forecasts or calendar entries, which may result in suggestions for improper attire. Additionally, the system makes an assumption about user comfort and preference that might not match human variances in style choice and temperature tolerance. Furthermore, the suggestions could eventually become old or repetitive if they are unable to adjust to shifting fashion trends or the user's changing style. Finally, the system is not designed to account for the different patterns in clothes and how they match with each other, nor for the different shades of colors (dark and light tones). These drawbacks point up areas that should be developed further in order to increase the system's customization, comprehensiveness, and adaptability.

1.5 General System Architecture

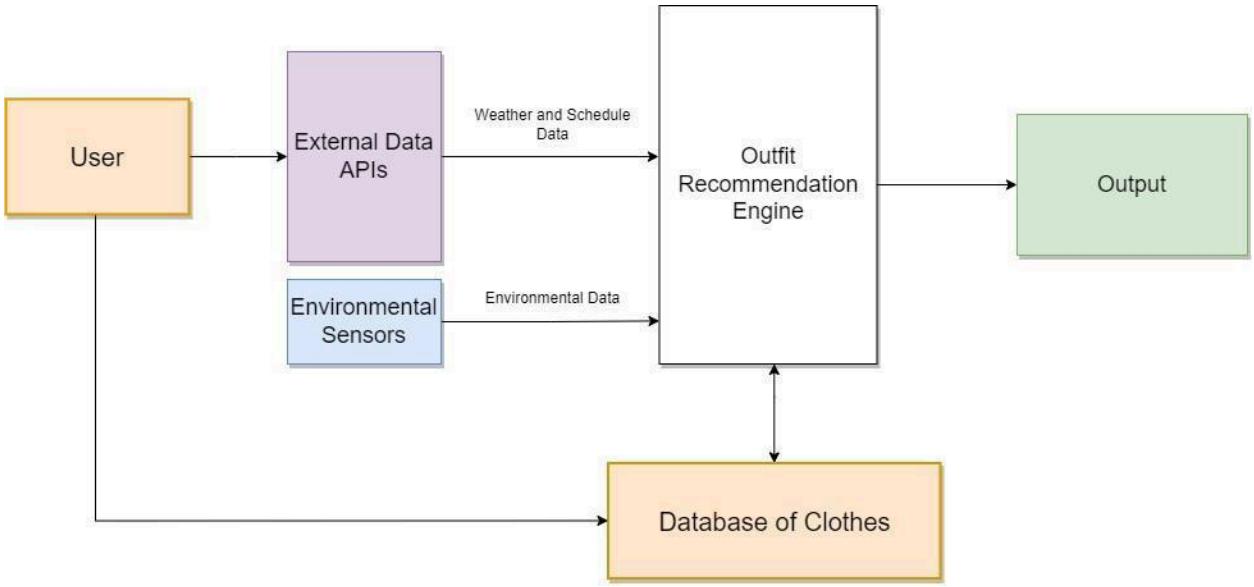


Figure 1 - General System Architecture

Note: The above diagram represents the general system architecture, showing the main components without naming them or giving details about them. In section 4: System Architecture, a more detailed diagram will be provided.

In the above diagram, we have the following:

1. **User**: The individual who uses the system and receives outfit recommendations.
2. **External Data APIs**: These are interfaces that retrieve information from outside sources, such as user schedules and weather predictions. This information aids the system in comprehending the environment (such as the kind of event and the weather) in which the outfit will be worn.
3. **Environmental Sensors**: These devices gather environmental information from the area around the user. Real-time temperature and humidity data are examples of the kind of information that can influence clothing selection.

4. **Database of Clothes**: A database containing information about the user's clothes. This contains details about each article of apparel, such as its kind, color, fabric, style, and adaptability for different environments.
5. **Outfit Recommendation Engine (ML, AI)**: The central element that applies algorithms for artificial intelligence (AI) or machine learning (ML) to process input data. To provide suitable outfit recommendations, it takes into account information from the clothes database, environmental data, and external data.
6. **Output**: The finished product, which shows the user the suggested ensembles as a list of listed clothing pieces on the webpage. Additionally, a speaker will provide an auditory output for those who are blind or visually impaired.

2 Limitations and Standards

2.1 Project Limitations

Table 1 - Limitations of the Project

Hardware Limitations	Sensors	Weather sensors have limitations in terms of range, accuracy, and durability.
	Arduino	The complexity of machine learning algorithms that we can execute directly on the device is limited by memory and processing resource constraints.
Software Limitations	Algorithm Complexity	The complexity of the machine learning algorithm would need to be simplified to run efficiently on limited hardware.
	Real-Time Processing	Processing the sensor data in real-time limits providing immediate outfit suggestions.
Data Limitations	Data Availability	The availability and quality of the wardrobe data and personal schedules limits the effectiveness of the outfit suggestion.
	Data Security	Ensuring the security of personal data is challenging, especially with the system being connected to the internet.
	User Adoption	There might be a need to convince users to input their wardrobe and daily events into the system.
	Variability in Personal Preference	Our system may not always align with the user's personal style preferences or may not account for all variables affecting outfit choice (e.g., mood, fashion trends).
Economic Limitations	Cost	The cost of sensors and additional hardware components is a limiting factor, especially for a prototype.
	Maintenance	Our system will require regular updates and maintenance, which would be a limitation for users with limited technical expertise. In addressing these standards and limitations, we'll need to balance technical feasibility with user needs and practicality. It's also important for us to consider iterative testing and user feedback to refine the system.

2.2 Standards

Table 2 - Standards Used in the Project

Standard	Code	Description
HTTP/2	RFC7540	Data exchange protocol used between the client and the host of the web application, for the user to request access to resources on the server or send information to the server.
Wi-Fi	IEEE 802.11	The system can use internet access to retrieve weather data in addition to relying on the sensors
Electric Safety Standard	IEC 60950	We will follow safety guidelines to prevent electrical hazards. This includes proper wiring, isolation, and the use of components that meet safety standards.

3 Background and Existing Solutions

3.1 Commercial Solutions

3.1.1 Weather Fit: iOS Application

The application [18] functions as a virtual wardrobe consultant, offering clothing recommendations based on prevailing weather conditions. A whimsical cartoon character on the screen undergoes outfit changes corresponding to the weather, donning a jacket for colder temperatures or wielding an umbrella during rainfall. However, a notable limitation is its failure to consider the user's existing wardrobe, potentially suggesting attire the user does not possess. Furthermore, the application overlooks the user's specific activities, such as a formal business meeting, a beach excursion, or a casual outing with friends, resulting in generic suggestions solely based on weather conditions. Compounded by its digital nature, the system lacks the capacity to account for the user's current clothing or the hardware to measure real-time temperature and humidity at their precise location. Instead, it relies on online weather sources to provide information for its recommendations.

3.1.2 Acloset – AI Fashion Assistant

This software *[19]*, which was initially created as a digital closet organizer, transforms wardrobe management by using the user's wardrobe to suggest outfits based on the weather. By uploading images of their clothing items, users allow the algorithm to provide customized recommendations based on what they already own. In addition to curating outfits, the platform enables users to purchase and trade clothing among themselves. Interestingly, the software takes into account several events and offers unique costume recommendations for each one depending on the user's wardrobe. Future plans call for the system to be integrated with the user's calendar, allowing for automatic recommendation of what to wear each day without the need for explicit user input. The application only uses software and doesn't use any hardware, even with its sophisticated features.

3.1.3 Mixdress

This simple program *[20]* acts as an intuitive wardrobe helper by providing a variety of outfit options based on the user's input. The user only logs the items of clothing they currently own,

adding details like how often they have been worn recently. Users of the system can designate whether it is chilly or warm outside and choose from a variety of style categories, such as business, casual, or special occasions. With the help of this simple input technique, the program creates customized suggestions for outfit pairings, making it easier for users to put together an appropriate outfit depending on their wardrobe and personal tastes.

3.1.4 Twelve70: Android and iOS Application

The Twelve70 is a unique computerized wardrobe management and men's outfit calculator. The Twelve70 website claims to be able to expedite the process of choosing what to wear, while also providing information on various styles. This addresses the widespread problem of taking 13 minutes to decide what to wear. Notably, the program ensures diversity in its design by accommodating users who are color blind. Twelve70 takes a comprehensive approach to clothing recommendations, taking into account social gatherings and weather circumstances in addition to simple outfit choices. With a focus on accessibility, style exploration, and efficiency, Twelve70 seeks to completely transform the way consumers make clothing decisions.

3.1.5 Pureple: iOS Application

By automatically classifying users' belongings to create a virtual closet, this creative program *[21]* streamlines wardrobe management. In contrast to conventional wardrobe apps, it takes into account each individual's unique style in addition to just organizing. Using its knowledge of each user's own fashion tastes, the application creates personalized outfit recommendations based on their choices. Notably, this places more focus on style than on the weather, deviating from traditional systems of wardrobe suggestion. This software offers a fresh method for selecting clothes that meet the user's personal style preferences because it prioritizes personal preferences over weather-based recommendations.

3.2 Solutions based on academic research

3.2.1 WEATHER-TO-GARMENT

The paper [17] discusses the development of a practical system for automatic clothing recommendations based on weather conditions. The creation of a workable system for weather-based automatic clothes recommendations is covered in this study. The algorithm can offer appropriate outfits from the user's own wardrobe or suggest combinations with a wardrobe piece of reference. Taking into account a variety of parameters under varying weather conditions presents a problem. The study employs clothing traits as a link between high-level weather categories and low-level features in order to remedy this. They have gathered a Weather-to-Garment (WoG) dataset for model building and used a scoring function with three terms to model the relationship. The efficiency of their model for weather-related outfit recommendations as well as garment pairing is validated by experiments conducted on this dataset. But unlike WEAR, this system doesn't account for an individual's schedule or activities. Thus, WEAR can provide a more customized and suitable suggestion compared to WoG.

3.2.2 Clothing Recognition and Segmentation

Based on a single, metadata-free image, the authors [13] offer a scalable approach for making product recommendations related to clothes. They tackle this issue as cross-scenario retrieval, where the products originate from online shopping catalogs, usually in a pristine environment, and the query is an actual image. There are two primary phases to the method:

1. To segment the individual in the picture and cluster potential image regions, they first do articulated pose estimation. This aids in identifying the clothing classes included in the image under query.
2. To locate visually comparable products within each of the identified clothing classifications, they apply image retrieval techniques. More than 50 times faster than the state-of-the-art, the approach obtains garment detection performance on a recent annotated dataset. They also use a large-scale clothing suggestion example, where the product database has over a million products, to show the scalability of their approach.

3.2.3 Apparel Classification

The authors [14] offer a thorough pipeline that can be used in online advertising, e-commerce, and event and activity detection to identify and categorize people's attire in natural settings. The pipeline consists of multiple phases that utilize cutting-edge components such as visual characteristics, multiple feature channels, and upper body detectors. Their main component is a multi-class learner that uses strong discriminative learners as decision nodes and is built on a random forest. Even though the automatically gathered training data from the web may be noisy and contain irrelevant photos, they incorporate it into the learning process to improve automation. To facilitate transfer learning across domains, they expand Random Forests. To assess their method, they define 15 clothing classes and introduce a benchmark dataset for clothing classification, featuring over 80,000 images, which is made publicly available.

3.2.4 Magic Closet

The demo [15] introduces a practical system called "Magic Closet," which focuses on automatic occasion-oriented clothing pairing. Users input an occasion (e.g., wedding or shopping), and the Magic Closet system intelligently pairs their specified reference clothing (upper-body or lower-body) with the most suitable options from online shops. Two key criteria guide this system:

1. Wearing properly: It ensures that the clothing choice matches the occasion appropriately. For example, it suggests a cocktail dress for a banquet rather than suit pants.
2. Wearing aesthetically: This criterion emphasizes the visual appeal of the outfit, such as suggesting that a red T-shirt pairs better with white pants than with green pants.

To bridge the gap between low-level visual features and high-level occasion categories, the system uses middle-level clothing attributes, such as clothing category, color, and pattern, treated as latent variables. These attributes play a crucial role in their proposed latent Support Vector Machine (SVM) based recommendation model. The "wearing properly" criterion is

described using a feature-occasion potential and an attribute-occasion potential, while the "wearing aesthetically" criterion is expressed by an attribute-attribute potential.

4 System Architecture

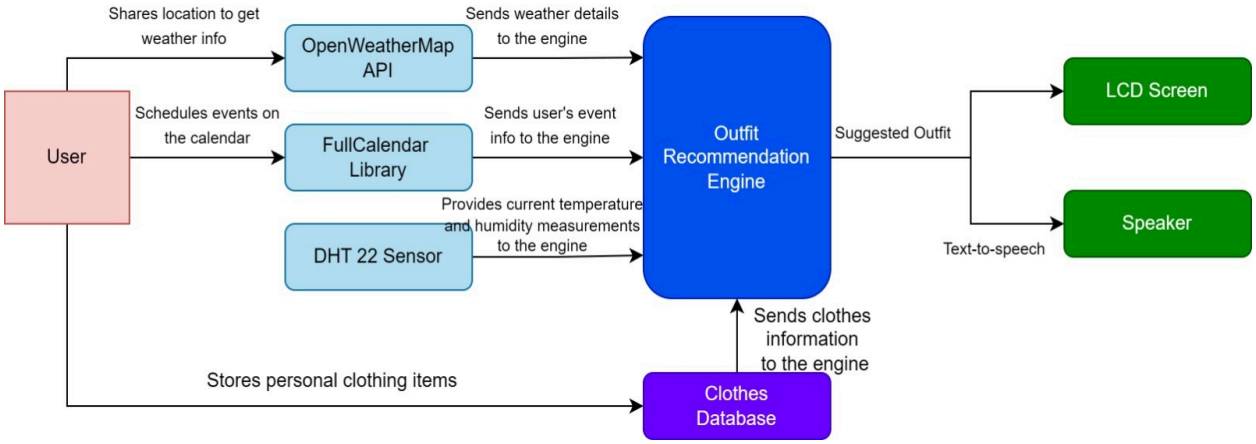


Figure 2 - Detailed System Architecture

As seen in figure 2, the user must first enter data into the WEAR system. Through the web application's user interface, these data types can be separated into categories such as location data, calendar data, and style preferences. These bits of information are meant to give the system the knowledge it needs to produce an acceptable outfit depending on the previously listed parameters.

The following is a detailed description of the system architecture:

4.1 User

The user is the starting point of the system architecture where they interact with the system. They provide their location for weather data, input events into their calendar, and store personal clothing items in the database. Their role also extends to choosing a specific style of clothing in the web application.

4.2 OpenWeatherMap API

Our choice of the OpenWeatherMap API is as a result of the following factors:

1. Its easy-to-use architecture enables rapid application integration and implementation.

2. It provides a wide range of weather data, which is essential for weather-analysis-dependent systems like WEAR. This includes historical data, forecasts, and current conditions.
3. The free usage tier's availability meets the small-scale projects' financial and developmental needs.
4. Its worldwide reach guarantees accurate weather information in a variety of geographical areas, including complete coverage for Lebanese regions and other nations alike.

In terms of how this API works, it takes user location data and provides the recommendation engine with weather information. API calls to OpenWeatherMap's endpoints are made during this interaction, usually over HTTP/HTTPS. After that, the system gets a JSON payload including weather-related information via a RESTful response. Information like temperature, humidity, and weather conditions are essential for selecting the right clothing.

4.3 FullCalendar API

Widely praised for its versatility and integration capabilities, FullCalendar is a powerful and flexible calendar API. For developers who want to integrate event handling and personalized calendar views into their apps, this is an excellent option. The extensive feature set of FullCalendar facilitates a wide range of calendar functions, including the ability to view, add, edit, and delete events. Because of this, it's perfect for applications that need an interactive, detailed calendar without requiring syncing with other calendar services. FullCalendar is well-known for its performance; it is essential for dynamic applications since it can manage a high volume of interactions and intricate event manipulations. Furthermore, because of its complete customization capabilities, developers can adjust the calendar's features and design to meet the demands of particular user groups.

4.4 DHT22 Sensor

A hardware sensor that gives the recommendation engine immediate access to real-time environmental data. It takes measurements of the local humidity and temperature, providing

the engine with more context when choosing an outfit. This is especially helpful when internet connection isn't available.

4.5 File Database

Before deciding on which database we would use, we explored a few different possibilities for the database:

File Database:

A file database is a type of database system where data is stored in flat files, such as text files or CSV files. These databases are characterized by their simplicity and straightforward data structure, making them an excellent choice for small-scale applications or situations where the complexity of a full-fledged database management system (DBMS) is unnecessary. File databases are typically easy to set up and manage, as they do not require extensive configuration or specialized software beyond basic file-handling capabilities. They are best suited for handling relatively static data with minimal relationships between records. Due to their simplicity, file databases can be less efficient for large datasets or applications that require complex querying, data integrity, and relational data management.

Relational Databases:

This covers databases like Oracle, PostgreSQL, MySQL, and SQL Server, among others.

These are frequently utilized in corporate solutions, web development, and general-purpose applications where it's necessary to preserve links between various data types and format the data.

NoSQL Databases:

1. MongoDB: Frequently used in content management systems, real-time applications, and web development, this database is perfect for projects involving substantial volumes of unstructured or semi-structured data.

2. Cassandra: Often used in time-series data, sensor data, and other applications requiring high write throughput, this platform is well-suited to managing enormous volumes of dispersed data over several commodity servers.

Graph Databases:

Among them are Neo4j and Amazon Neptune.

These are intended for use with complicated connection data, like that found in social networks, fraud detection systems, and recommendation engines.

Taking everything into consideration, we will be using a file database because our dataset can be summed up into one table: the clothes owned by the user. This choice eliminates the necessity for the complex queries typical in other database systems, ensuring that the inherent structure and simplicity of our dataset is maintained and effectively utilized.

For the database, we will store the clothes with the following attributes:

1. Type: The clothing item's categorization. Possible values: Shirt, Trousers, Dress, Jacket, and Skirt.

2. Color: The hue of the article of apparel. Possible values: Yellow, Red, Green, Blue, Black, White, Purple, and Orange.

3. Fabric: The substance used to make the article of apparel. Possible values: Silk, Polyester, Wool, and Cotton.

4. Season Suitability: The best season for which the article of apparel is suitable. Possible values: Winter, Fall, Summer, and Spring.

5. Style: The overall look or fashion class that the article of apparel belongs to. Possible values: formal, athletic, vintage, and casual.

6. Condition: The degree of wear on the article of apparel. Possible values: New, Used, and Worn.

4.6 Outfit Recommendation Engine

In addition to real-time weather data from the DHT22 sensor, the recommendation engine will also collect information about events, activities, and schedules from the calendar API, as well as the current weather conditions from the weather API. Using machine learning (ML) and artificial intelligence (AI) technologies on this data, this outfit suggestion engine will create outfits from the user's closet depending on the previously described variables. The result will then be spoken through a speaker and shown on the web app of the user.

4.6.1 Training the Engine

With the help of AI, we produced a 1000-row dataset that was used to train the engine. We were unable to locate a dataset that precisely met our needs, particularly one that included information on attire, weather, and event type, so we made the decision to create our own. The columns were designed to work together, meaning that the temperature and the weather shouldn't conflict with one another. For instance, at 20 °C, the weather cannot be snowy.
The dataset includes the six columns covered in the preceding part as well as the next four, which are related to the weather and the kind of activity:

1. **Temp. (°C)**: The temperature at the time of usage.
2. **Weather Cond.**: The type of weather conditions at the time of usage. Possible Values: Sunny, Rainy, Snowy, Windy, Standard Weather.
3. **Humidity (%)**: The humidity levels at the time of usage. Possible Values: This would be a percentage range from 0% to 100%, with certain ranges like 0-30% (low), 31-50% (moderate), 51-100% (high).
4. **Type of Event**: The kind of event/activity at the time of usage. Possible Values: Casual Outing, Formal Event, Sports Activity.

The following is a screenshot of the first 10 rows of the dataset:

Type	Color	Fabric	Season Suitability	Style	Condition	Temp. (°C)	Weather Cond.	Humidity (%)	Type of Event	Recommend
Jacket	White	Cotton	Spring	Casual	Used	16	Standard Weather	42	Casual Outing	1
Jacket	Blue	Wool	Spring	Vintage	Used	14	Standard Weather	59	Casual Outing	1
Dress	Black	Wool	Fall	Athletic	New	9	Windy	34	Sports Activity	0
Dress	White	Wool	Winter	Athletic	Used	-4	Snowy	40	Formal Event	0
Shirt	Red	Cotton	Fall	Athletic	New	24	Rainy	56	Formal Event	0
Skirt	Red	Polyester	Spring	Vintage	Worn	12	Rainy	27	Formal Event	0
Shirt	Blue	Polyester	Spring	Casual	Used	14	Rainy	44	Casual Outing	1
Skirt	Blue	Silk	Spring	Casual	Used	8	Windy	57	Casual Outing	1
Skirt	Black	Silk	Fall	Vintage	New	13	Rainy	88	Casual Outing	1

Figure 3 - First 10 Rows of the Dataset

The “Recommend” column is filled manually row by row. For example, the first row checks whether a white casual cotton jacket suitable for a general spring climate which is also moderately used is suitable for a temperature of 16 °C standard weather with a humidity of 42% and for a casual outing. The rest of the rows are filled in a similar fashion.

Effect of Fabric

Different fabrics have varying levels of warmth, breathability, and comfort, which are crucial in different weather conditions. For example:

1. Silk is not as good in cold, windy weather since it is lightweight and not very insulating, despite its elegance and luxury.
2. Heavier fabrics like wool or thick cotton are more suitable for colder weather as a result of their insulating properties.
3. Occasion Appropriateness: Fabrics generally reflect a certain level of formality or casualness of a garment. Silk, for instance, is often associated with formal or elegant outfits, as compared to cotton which is more casual.
4. Durability for Activity: For casual outings, more durable and less delicate fabrics are generally preferable. Silk can be delicate leading to damage easily, which may not be ideal for an active (sports) or a casual setting.

Moreover, different fabrics have different suitability in varying levels of humidity.

1. **Low Humidity (Below 30%):**
    i. Cotton: Beneficial because it prevents skin dryness.
    ii. Wool: Excellent, particularly for holding onto heat.
    iii. Polyester: Acceptable, especially if it's a variety that wicks away moisture.

iv. Silk: Adequate, but in arid, cold areas, it might not offer adequate warmth.

2. **Moderate Humidity (30% to 50%):**
   i. Cotton: Superior, cozy in most weather conditions.
   ii. Wool: High-quality wool, especially merino, which is lightweight.
   iii. Polyester: Excellent types for active clothes, particularly those that wick away sweat.
   iv. Silk: Good; it offers just the right amount of breathability and comfort.
3. **High Humidity (Above 50%):**
   i. Cotton: Superior due to its high absorption capacity and breathability.
   ii. Wool: Lightweight, moderate wool can nevertheless be cozy.
   iii. Polyester: Variables; kinds that drain away moisture are appropriate for sportswear, while normal polyester may feel stuffy because of its reduced breathability.
   iv. Silk: Limited; if fitted too tightly, it may feel damp and sticky.

4.6.2 Color Matching

The color matching feature is effectively managed through a separate color compatibility matrix or table, which works alongside the main dataset. This approach keeps the dataset streamlined while still allowing for complex color matching logic within the recommendation system.

Table 3 - Matrix of Color Matching

	Black	*White*	*Red*	*Blue*	*Green*	*Yellow*	*Purple*	*Orange*	*Brown*	*Gray*
Black	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
White	*1*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
Red	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*0*	*0*	*1*
Blue	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*1*	*1*
Green	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*0*	*1*	*1*
Yellow	*1*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*

Purple	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*1*
Orange	*1*	*1*	*0*	*1*	*1*	*0*	*0*	*0*	*1*	*1*
Brown	*1*	*1*	*1*	*1*	*1*	*1*	*0*	*1*	*0*	*1*
Gray	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*0*

However, as stated before, this matrix of color matching matrix does not take into account the different shades of the color, where the color blue always matches with the color green, no matter how these 2 colors may vary. [16]

4.6.3 Scoring System

Based on the above parameters, we are going to implement a scoring system for the recommended outfits, so that out of all the recommended outfits that the system generates, we can recommend only a select few of them that are the best in the current conditions. For that, we will simply give each of the parameters a weight based on their importance of determining whether an outfit is well fit or not. These parameters, in order, are as follows:

1. Fabric
2. Condition
3. Style
4. Season
5. Color

Based on this, we will calculate a recommendation score for each of the generated outfits, then the user is free to pick one of the top scoring outfits, in order to avoid redundancy in the generated outfits, as well as give the user more options to go for in case multiple outfits were needed for similar conditions

4.6.4 Machine Learning Algorithms *[7]*

An algorithm that turns variables for clothes and weather data taken from an Arduino into recommendations for an appropriate outfit is needed. We analyze the suitability of the listed algorithms, along with their advantages and disadvantages:

1. **K-Nearest Neighbors (KNN)**
    - **Advantages**:
        - Simple and easy to implement.
        - No probability distributions are assumed based on the input data. This is useful for input data that might be non-linear.
        - Naturally handles multi-class cases.
    - **Disadvantages**:
        - Computationally expensive as the algorithm stores all the training data.
        - High memory requirement.
        - Sensitive to irrelevant features and the scale of the data.
2. **Logistic Regression**
    - **Advantages**:
        - Efficient and straightforward.
        - Does not require too many computational resources.
        - Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.
    - **Disadvantages**:
        - Tends to underperform when there are multiple or non-linear decision boundaries.
        - Not flexible enough to capture more complex relationships.
3. **Support Vector Machine (SVM)**
    - **Advantages**:

- Effective in high dimensional spaces and with a clear margin of separation.
- Memory efficient as it uses a subset of training points in the decision phase.
- Works well with unstructured and semi-structured data like text and images.

- **Disadvantages**:
  - Not suitable for large datasets due to its high training time.
  - Less effective on noisier datasets with overlapping classes.
  - Requires careful tuning of the parameters and a good understanding of SVM kernels.

4. **K-means Clustering**

- **Advantages**:
  - Simple, easy to implement and interpret the results.
  - Scales well to large datasets.
  - Works well in practice for many applications.
- **Disadvantages**:
  - Assumes the clusters as spherical, so not effective if the clusters have complex shapes.
  - The number of clusters (k) must be set beforehand.
  - Sensitive to the initial choice of centroids and outliers.

5. **Hierarchical Clustering**

- **Advantages**:
  - Does not require the specification of the number of clusters beforehand.
  - Easy to interpret the dendrogram (tree diagram).

        - Can capture the relationship and hierarchy between clusters.

    - **Disadvantages**:

        - Algorithmically complex and computationally expensive, thus not suitable for large datasets.
        - Sensitive to noise and outliers.
        - Different results can be obtained if a different order of data points is used.

6. **Principal Component Analysis (PCA)**

    - **Advantages**:

        - Reduces the dimensionality of the data, which can simplify the dataset.
        - Can improve the performance of machine learning models.
        - Helps to identify and interpret the underlying structure of the data.

    - **Disadvantages**:

        - Assumes linear relationships between variables.
        - Can be substantially affected by outliers in the data.
        - PCA is a method to bring out strong patterns in a dataset, but it does not support categorical variables.

7. **Non-negative Matrix Factorization (NMF)**

    - **Advantages**:

        - Good for parts-based representation of data.
        - Useful in sparse data.
        - Can be used for feature extraction.

    - **Disadvantages**:

- Restricts to non-negative data.
- Sometimes hard to interpret the results.
- Can be computationally intensive for larger datasets.

8. **Naive-Bayes**

- **Advantages**:
  - Simple and Easy to Implement.
  - Efficient for Large Datasets.
  - Performs Well with Categorical Inputs
- **Disadvantages**:
  - Assumption of Feature Independence.
  - Poor Estimation for Low Frequency Values.
  - Biased Estimates.

The best approach is to test each of K-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Naive-Bayes in order to find which of those three models would have the better accuracy to select clothes depending on weather and other characteristics.

4.7 LCD Screen

The LCD screen displays the suggested outfit to the user having the highest score. This visual interface displays the output of the recommendation engine, listing the selected clothing items which form the outfit required.

4.8 Text-to Speech

The text-to-speech functionality can be done in one of two ways.

- **Hardware-Based Text-to-Speech**: To do this, we use the Emic 2 Text-to-Speech Module that directly synthesizes text into speech.
    - **Advantages:**
        - Fast Response Time: Offers speech synthesis instantly, without the delay that comes with cloud-based applications.
        - Customization: Provides dynamic control over aspects of speech including emphasis, pitch, and pace.
        - Ease of Interface: It is simple to interface with microcontrollers since it communicates via a serial interface.
    - **Disadvantages:**
        - Restricted Language and Voice Style Support: In contrast to cloud-based services, it provides a preset range of voice styles and only supports a small number of languages.
        - Less Flexibility: May not provide the deep customization and multilingual support that cloud-based systems do.
        - Cost Increase: The system will incur increased expenses due to the requirement for more hardware and circuits.
- **Software-Based Text-to-Speech:** To do this, we created a text-to-speech module using the pyttsx3 Python library, thus generating the speech corresponding to the input text.
    - **Advantages:**
        - Extensive Language Support: Provides a wide range of languages and different voice styles which suit global applications.
        - High-Quality Speech Output: Uses advanced neural network models to generate natural-sounding speech.

- Customization Options: Voice parameters, such as pitch, rate, volume can be adjusted, thus serving tailored speech synthesis.
- Cloud-Based Accessibility: Ensures seamless integration into web and mobile applications, with the help of RESTful APIs.
- Scalability and Updates: Can handle varying workloads which helps with the futureproofing of the product. It also receives updates for improved functionality.

- **Disadvantages:**
  - Internet Dependency: Requires a progressive internet connection to access the cloud-based service.
  - Potential Latency: Involves network latency that may cause delays in speech synthesis.
  - Additional Cost: Requires the cost of an additional speaker to produce the speech audio.

In light of these advantages and disadvantages, we have decided to choose the software-based approach as the main audio source due to its higher versatility and better-quality speech output, not to forget its lower overall costs. However, to continue the offline aspect of the system and build upon it, we decided to also use the Emic 2 Text-to-Speech Module for the system to function without an active connection.

4.9 The WebApp

In this project, users interact with a straightforward web app designed to manage and explore their clothing collection. Through the web interface, users can effortlessly add new items or delete existing ones. Additionally, the app provides a feature to view recommended outfits based on the user's collection and preferences. This intuitive system not only simplifies wardrobe management but also enhances the user experience by offering personalized styling suggestions. The user-friendly design of the app ensures that users of all technological

proficiencies can navigate and utilize the features effectively, making fashion choices both fun and accessible.

5 Hardware Components

5.1 Arduino Uno Rev3

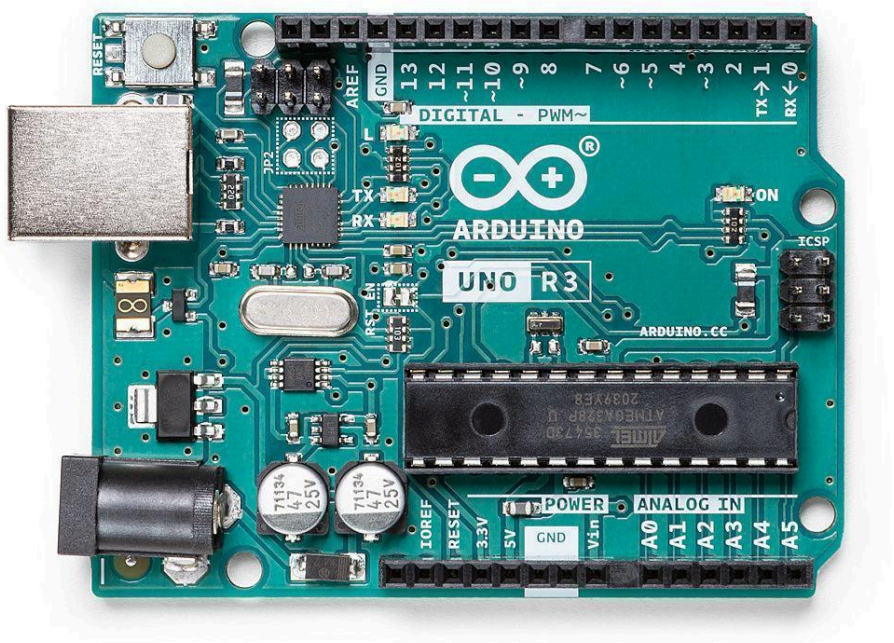


Figure 4 - Arduino Uno Rev3

The Arduino Uno Rev 3 *[9]* is a popular microcontroller board known for its versatility and ease of use in prototyping various electronic projects. The following table displays some specifications.

Table 4 - Arduino Characteristics

Parameter	Specification
Price	$27.60 (official Arduino Website)
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (6 PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA

DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

5.2 DHT22 Temperature and Humidity Sensor

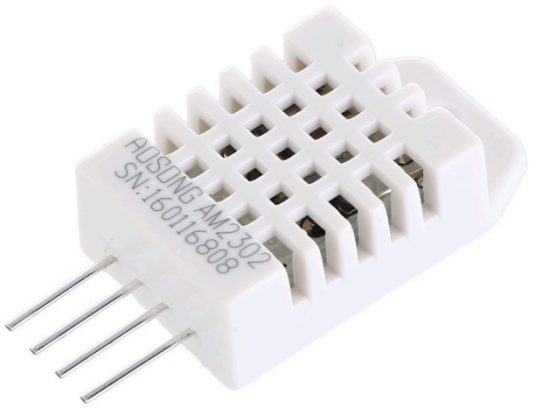


Figure 5 - DHT22 Sensor

The DHT22 *[12]* offers greater accuracy and wider operating range compared to the DHT11, making it a more suitable choice for precise temperature and humidity measurements in our application. The following table displays some specifications.

Table 5 - DHT22 Sensor Characteristics

Parameter	Specification
Price	$6.15

Supply Voltage	3.3 to 6V
Output Signal	Digital signal via single bus
Operating Range and Accuracy (Humidity)	0-100% RH; +/-2% RH
Operating Range and Accuracy (Temperature)	-40 to 80°C; +/-0.5°C
Average Sending Period	2 seconds
Dimensions (excluding pins)	15.3mm (0.6") length x 7.8mm (0.3") width x 25.3mm (1.0") height
Weight	2.4g (0.08oz)

5.3 Low-Cost Yellow Green 4004 40x4 Character LCD Display Module

Figure 6 - LCD Module

The following table displays some specifications of the LCD module [11].

Table 6 - LCD Module Characteristics

Parameter	Specification
Price	$12
Display Size	20 characters wide, 4 rows
Controller	HD44780 compatible
Interface	I2C (Inter-Integrated Circuit)
Backlight	Single LED, blue color, dimmable with resistor/PWM
LCD Type	STN (Super Twisted Nematic) LCD, positive
Text Appearance	White text on blue background
Operating Temperature Range	-20°C to +70°C
Compliance	RoHS compliant
Built-in Character Set	Supports English/Japanese text
Character Set Information	Refer to HD44780 datasheet for full character set

5.4 Middle Size Breadboard

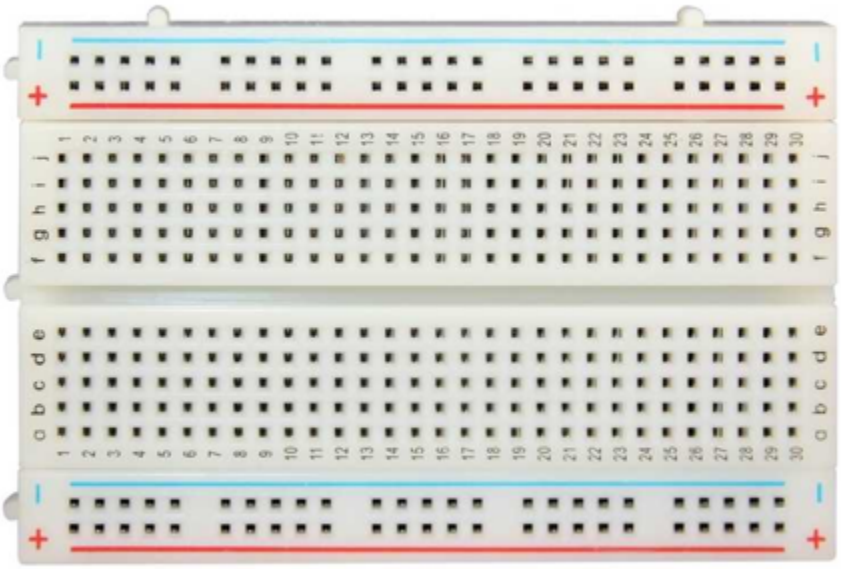


Figure 8 - Middle Size Breadboard

The following table displays some specifications.

Table 8 - Middle Size Breadboard Characteristics

Specification	Details
Price	$2.95
Bottom Side	Sticky Tape
Total Points	400
Ideal for	Electronics prototype experiments
Wire Diameter Accepted	20 AWG to 29 AWG
Size (L x W x H)	84 x 54 x 9.5 mm

5.5 Cost Analysis

In light of all the mentioned components, the total price of the system is detailed in the following table:

Table 9 - Total Cost Analysis

Component	Price
Arduino Uno Rev3	$27.6
DHT22	$6.15
LCD Display Module	$12
Jumper wires	$5
Breadboard	$2.95
	Total Price = $53.7

6 Implementation

6.1 Development Environment Setup

The setup of the software and development tools that have been essential to the creation and development of the WEAR project is covered in detail in this section. Running on a Windows operating system, Visual Studio Code (VS Code) is the centerpiece of our development environment. This configuration has given us a productive and adaptable workspace that meets our demands for programming.

Visual Studio Code on Windows

- **Integrated Development Environment (IDE):** We chose Visual Studio Code because of its versatility and robust ecosystem. As a lightweight but powerful source code editor, VS Code supports Python development through an array of extensions and built-in features which enhance productivity and streamline coding tasks.
- **Benefits of Using VS Code:**
    - *Extensive Plugin Support:* VS Code's extensive library of extensions, such as Python, Pylance, and GitLens, significantly enhances the coding experience by providing intelligent code completion, advanced syntactic highlighting, and real-time feedback and linting.

- *Customizable and Configurable:* The customizable user interface and configurable settings allow developers to tailor the environment to their specific needs, optimizing workflow and productivity.
- *Debugging Tools:* Built-in debugging support for Python aids in quick identification and resolution of bugs, enhancing code reliability and reducing development time.

Programming in Python

- Because of its readability and the wide range of libraries it supports—which are essential for jobs involving both data processing and machine learning—Python was chosen as the main programming language.
- **Key Python Libraries Utilized:**
    - *Scikit-learn (sklearn):* This package is essential for effectively implementing machine learning algorithms. Classification, regression, clustering, and dimensionality reduction are just a few of the many machine learning and statistical modeling capabilities it offers. These are all essential for our outfit recommendation engine.
    - *Pandas:* Pandas provides data structures and procedures for working with time series and numerical tables, which are essential for data manipulation and analysis. This is very helpful for managing and processing the massive amounts of information that are kept in our database about user preferences, clothes products, and weather.
    - *NumPy:* NumPy, which is frequently used in tandem with Pandas, provides support for sizable multi-dimensional arrays and matrices in addition to a vast array of sophisticated mathematical operations that can be performed on these arrays. For managing the data-intensive calculations in our project, this library is essential.

6.2 Coding Standards and Practices

In the development of the WEAR project, we have adopted several coding standards and practices aimed at ensuring code quality, maintainability, and effective collaboration among team members. These practices are integral to our development strategy and include pair programming, code refactoring, modularity, and comprehensive commenting.

Pair Programming

Our development methodology has always relied heavily on pair programming, in which two programmers collaborate at a single workstation. The "navigator" reads each line as it is written by the "driver". This method not only helps the team share knowledge more easily but also improves the quality of the code by providing instant feedback. By ensuring that every team member interacts with various project components on a regular basis, role swapping helps to improve code familiarity overall and lowers the possibility of understanding single points of failure.

Code Refactoring

We constantly refactor our code to make it more readable and organized without changing how it behaves externally. This procedure has shown to be quite helpful in lowering complexity, which makes the system simpler to maintain and grow with new features. It is also necessary for keeping clean, effective code. We plan regular refactoring sessions to identify areas for improvement and simplification, making sure our codebase is resilient to changing project needs.

Modularity

Our project's architecture emphasizes modularity, with the codebase organized into distinct, interchangeable modules. Each module is designed to execute a specific part of the project's functionality, which streamlines debugging, testing, and maintenance. This approach not only simplifies updates and feature additions but also enhances scalability by isolating changes to individual modules without impacting the entire system.

Comprehensive Commenting

We use a tight commenting process across our software to make sure that every major code block has a clear, short remark accompanying it. These comments make the code easier to read and maintain by outlining the intent of the code, the parameters involved, and the anticipated results. This approach keeps the code accessible and understandable for all team members and outside reviewers, which is very helpful when onboarding new engineers and supporting the review process.

Through the integration of these practices into our everyday development efforts, we have managed to create a codebase that is clean, manageable, and functional. These principles allow successful collaboration and contribute greatly to the project's long-term success.

6.3 Brief Description of the Back-End

In the development of the WEAR project, several key code modules play crucial roles in the functionality and operation of our system. Each module addresses specific tasks within the project scope, contributing to the seamless integration of weather-based outfit recommendations. Below, we describe each module in detail:

data_preprocessing.py

- **Purpose:** Prepares and processes the raw data necessary for the machine learning models used in the outfit recommendation system.
- **Functionality:**uses scikit-learn tools for encoding, scaling, and partitioning the data into training and testing sets in addition to pandas for data processing. It guarantees that the structured and clean data supplied into the machine learning models.
- **Integration:** Feeds processed data into the model training module, ensuring high-quality inputs for accurate model training.

model_training_and_evaluation.py

- **Purpose:** Manages the training and evaluation of the machine learning models that power the recommendation engine.
- **Functionality:** trains the model using the scikit-learn KNeighborsClassifier, then assesses it using metrics such as confusion matrix, ROC AUC, and accuracy. It also has the ability to use joblib to save the learned model.
- **Integration:** obtains preprocessed data from the module data_preprocessing.py and produces a trained model that is prepared to generate predictions based on user inputs.

openAI_api.py

- **Purpose:** uses sophisticated natural language processing to classify and interpret different event kinds from textual descriptions by integrating with the OpenAI API.
- **Functionality:** contains the classify_event_with_gpt3 function, which uses the GPT-3 API to accept event descriptions and classify them in order to provide suggestions for attire based on the type of event (formal, informal, etc.).
- **Integration:** Enhances the calendar event handling by adding a layer of understanding about the event's nature, which influences outfit suggestions.

openWeather_api.py

- **Purpose:** Fetches real-time weather data that is crucial for making weather-appropriate outfit recommendations.
- **Functionality:** implements methods for obtaining weather information using city coordinates. It obtains the most recent weather conditions using the OpenWeatherMap API, which has an immediate impact on outfit suggestions.

- **Integration:** Provides the necessary weather data to the recommendation engine, allowing for dynamic outfit suggestions based on current and forecasted weather conditions.

calendarEndpoint.py

- **Purpose:** Handles the interaction with calendar data, specifically focusing on reading and organizing events from a specified CSV file.
- **Functionality:** The script begins by setting the path to a CSV file that contains event data. It includes a utility function, **read_and_sort_events**, which loads events from the CSV, ensuring they are sorted by date. This functionality supports the system in associating specific outfits with upcoming events, based on their types and dates.
- **Integration:** Works in conjunction with the outfit recommendation system to provide contextually appropriate attire suggestions for scheduled events.

color_matching.py

- **Purpose:** Implements the logic for color matching, which is pivotal in suggesting outfits that are not only weather-appropriate but also aesthetically pleasing.
- **Functionality:** Defines a function **color_matches** that uses a predefined matrix to determine compatible colors between different pieces of clothing, such as tops and bottoms. This module is essential for ensuring that the recommended outfits meet basic fashion standards.
- **Integration:** This module is integral to the outfit recommendation engine, where it helps in the selection of outfits that are color-coordinated.

arduino_sensor_reader.py

- **Purpose:** Facilitates the retrieval of environmental data directly from Arduino sensors, which includes temperature and humidity readings.
- **Functionality:** The module utilizes the **serial** library to establish a connection with the Arduino through the specified COM port. It sends a command to the Arduino to trigger data sending and reads the incoming data from the serial port, parsing it to extract temperature and humidity values.
- **Integration:** This module is crucial for environments where internet access is not available or reliable, allowing the system to continue providing weather-appropriate clothing recommendations based on locally collected data.

clothing_recommendation_engine.py

- **Purpose:** Serves as the core engine that combines weather data, event types, and user preferences to suggest appropriate outfits.
- **Functionality:** This module integrates functionalities from multiple other modules: it fetches weather data, matches clothing colors, manages calendar events, and uses machine learning models to suggest outfits. It also handles user interactions like adding or removing data from the database.
- **Integration:** Acts as the central hub for decision-making in the outfit recommendation process, utilizing input from both online APIs and local data sources.

clothing_recommendation_engine_arduino.py

- **Purpose:** An alternative version of the recommendation engine designed to operate with data from Arduino sensors, useful in offline scenarios.**Functionality:** Similar to the standard recommendation engine, but tailored to utilize local sensor data instead of

relying on online weather APIs. This adaptation allows the system to function independently of internet connectivity.

- **Integration:** Provides resilience by ensuring that the recommendation service remains operational, even in offline conditions, by adapting its inputs to available data sources.

main_clothing_recommendation.py

- **Purpose:** Determines the operational mode of the recommendation engine based on the availability of an internet connection.
- **Functionality:** Checks for internet connectivity; if available, it runs the standard recommendation engine using online data. If not, it switches to the Arduino-based engine to utilize local sensor data.
- **Integration:** Ensures that the system remains functional and adaptive to the operational environment, providing flexibility in how and where the system can be deployed.

textToSpeech_api.py

- **Purpose:** The code converts a given text input into spoken words using text-to-speech technology.
- **Functionality:** It initializes the pyttsx3 engine, sets the speaking rate and volume properties, queues the text to be spoken, and processes the speech output.
- **Integration:** This function will be integrated into our project so that it would read out the output using the arduino speaker

In conclusion, the WEAR project leverages a comprehensive suite of code modules, each specifically designed to handle distinct aspects of the outfit recommendation process. From processing raw data for machine learning to integrating with APIs for real-time weather and event classification, these modules work in harmony to deliver personalized outfit suggestions. Core functionalities include handling environmental data from Arduino sensors, managing extensive clothing databases, and operating seamlessly across different connectivity scenarios. This integrated approach ensures that outfit recommendations are not only contextually appropriate but also aesthetically pleasing, making WEAR a pioneering solution in automated wardrobe management.

6.4 Testing the Back-End

6.3.1 Testing KNN:

```python
def train_model(X_train, y_train):
    # Initializing the KNN model with specific parameters found via GridSearch
    knn = KNeighborsClassifier(n_neighbors=15, weights='distance', metric='manhattan')
    knn.fit(X_train, y_train)  # Training the model
    joblib.dump(knn, 'trained_knn_model.pkl')  # Saving the trained model to disk
    return knn
```

The KNN model is configured with 15 neighbors, uses the Manhattan distance metric, and weights points by distance, using the training set we got earlier.

We got the following results:

```
Classification Report:
0: {'precision': 0.7985074626865671, 'recall': 0.856, 'f1-score': 0.8262548262548263, 'support': 125.0}
1: {'precision': 0.7272727272727273, 'recall': 0.64, 'f1-score': 0.6808510638297872, 'support': 75.0}
accuracy: 0.775
macro avg: {'precision': 0.7628900949796472, 'recall': 0.748, 'f1-score': 0.7535529450423067, 'support': 200.0}
weighted avg: {'precision': 0.7717944369063773, 'recall': 0.775, 'f1-score': 0.7717284153454367, 'support': 200.0}
Confusion Matrix:
 [[107, 18], [27, 48]]
ROC-AUC Score: 0.8459733333333332
Cross-Validation Scores: [0.57  0.655 0.69  0.72  0.715]
Average CV Score: 0.6699999999999999
```

6.3.2 Testing SVM:

```python
def train_optimized_svm(X_train, y_train, model_path='optimized_svm_model.pkl'):
    """Train an SVM classifier with specified hyperparameters."""
    # Initialize the SVM classifier with predefined optimal parameters
    svm_classifier = SVC(C=1, class_weight=None, gamma=0.1, kernel='poly')
    svm_classifier.fit(X_train, y_train)  # Train the model
    joblib.dump(svm_classifier, model_path)  # Save the model to disk for later use
    return svm_classifier
```

This method trains a Support Vector Machine (SVM) classifier with predefined hyperparameters. It accepts three parameters: X_train (features), y_train (labels), and model_path (default path for the saved model file). The SVM is initialized with the following settings: C=1 for regularization strength, class_weight=None indicating no weights adjustments for class imbalances, gamma=0.1 controlling the influence of data points, and kernel='poly' specifying a polynomial kernel function.

For the SVM, we got the following results:

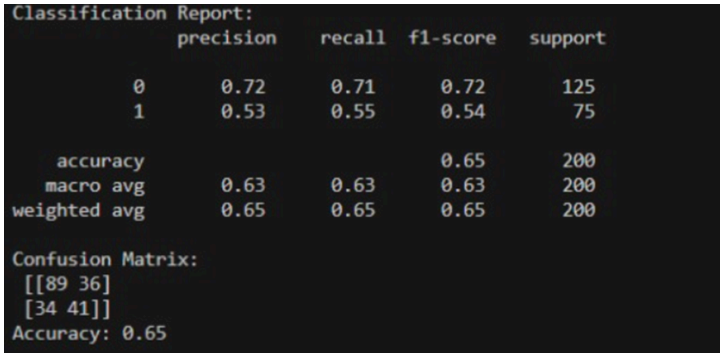
```
Classification Report:
 {'0': {'precision': 0.8503937007874016, 'recall': 0.864, 'f1-score': 0.8571428571428571, 'support': 125.0}, '1': {'precision': 0.7671232876712328, 'recall': 0.7466666666666667, 'f1-score': 0.7567567
567567568, 'support': 75.0}, 'accuracy': 0.82, 'macro avg': {'precision': 0.8087584942293172, 'recall': 0.8053333333333333, 'f1-score': 0.8069498069498069, 'support': 200.0}, 'weighted avg': {'precis
ion': 0.8191672958688383, 'recall': 0.82, 'f1-score': 0.8194980694980694, 'support': 200.0}}
Confusion Matrix:
 [[108  17]
 [ 19  56]]
Accuracy: 0.82
```

6.3.3 Testing Naive-Bayse:

```python
def train_naive_bayes(X_train, y_train, model_path='optimized_nb_model.pkl', var_smoothing=1e-9):
    """Train Gaussian Naive Bayes model with specific var_smoothing."""
    nb = GaussianNB(var_smoothing=var_smoothing)  # Initialize GaussianNB with specific smoothing
    nb.fit(X_train, y_train)  # Fit model on training data
    joblib.dump(nb, model_path)  # Save the model to disk
    return nb
```

The function is designed to train a Gaussian Naive Bayes model with specific settings for variable smoothing. The function accepts three parameters: X_train (features), y_train (labels), and the file path. Additionally, it has an optional parameter var_smoothing set to a default value of 1e-9, which is used to specify the smoothing factor to handle low variance features.

And this model gave us the following results:



```
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.71      0.72       125
           1       0.53      0.55      0.54        75

    accuracy                           0.65       200
   macro avg       0.63      0.63      0.63       200
weighted avg       0.65      0.65      0.65       200

Confusion Matrix:
 [[89 36]
 [34 41]]
Accuracy: 0.65
```

We noticed that SVM has the best accuracy out of the three models. As for the performance of each of the models:

- KNN: Relatively balanced Precision, Recall, and F1-Score for Class '0' and '1', indicating a fairly even performance across classes.
- SVM: Higher Precision, Recall, and F1-Score for Class '0' compared to Class '1', showing better performance on this class
- Naïve-Bayes: Lower scores compared to the other models, especially for Class '1', suggesting it struggles more with this classification.

In general, SVM exhibits the highest accuracy, making it potentially the most reliable for outfit recommendations in this specific application. KNN follows, with slightly reduced accuracy but still reasonable performance. Naive Bayes lags behind, particularly in handling Class '1', which could indicate issues with handling data features or model assumptions not being met.

6.5 Brief Description of the Front-End

The front end of the WEAR project is meticulously designed using HTML, CSS, and JavaScript to offer a user-friendly and interactive interface. The webpage is structured to include elements for both clothing inventory management and event scheduling. Users can input details about clothing items, such as type, color, fabric, and condition, through a dedicated form. These details are then displayed in a dynamic table on the same page, allowing for easy viewing and management.

The CSS styling ensures that the interface is visually appealing and easy to navigate. It styles everything from form elements and tables to buttons and the calendar, creating a cohesive and engaging user experience.

JavaScript enhances the page's interactivity, particularly through the integration with the FullCalendar library, which is crucial for handling calendar functionalities. Users can interact directly with the calendar to add, edit, and delete events, with these changes reflected immediately without page reloads. The JavaScript also handles the form submissions for adding clothing items to the inventory, ensuring data validation and dynamic updates to the HTML table.

The following is a screenshot of the Clothing Inventory section in the front-end, with one row being added to the person’s wardrobe:

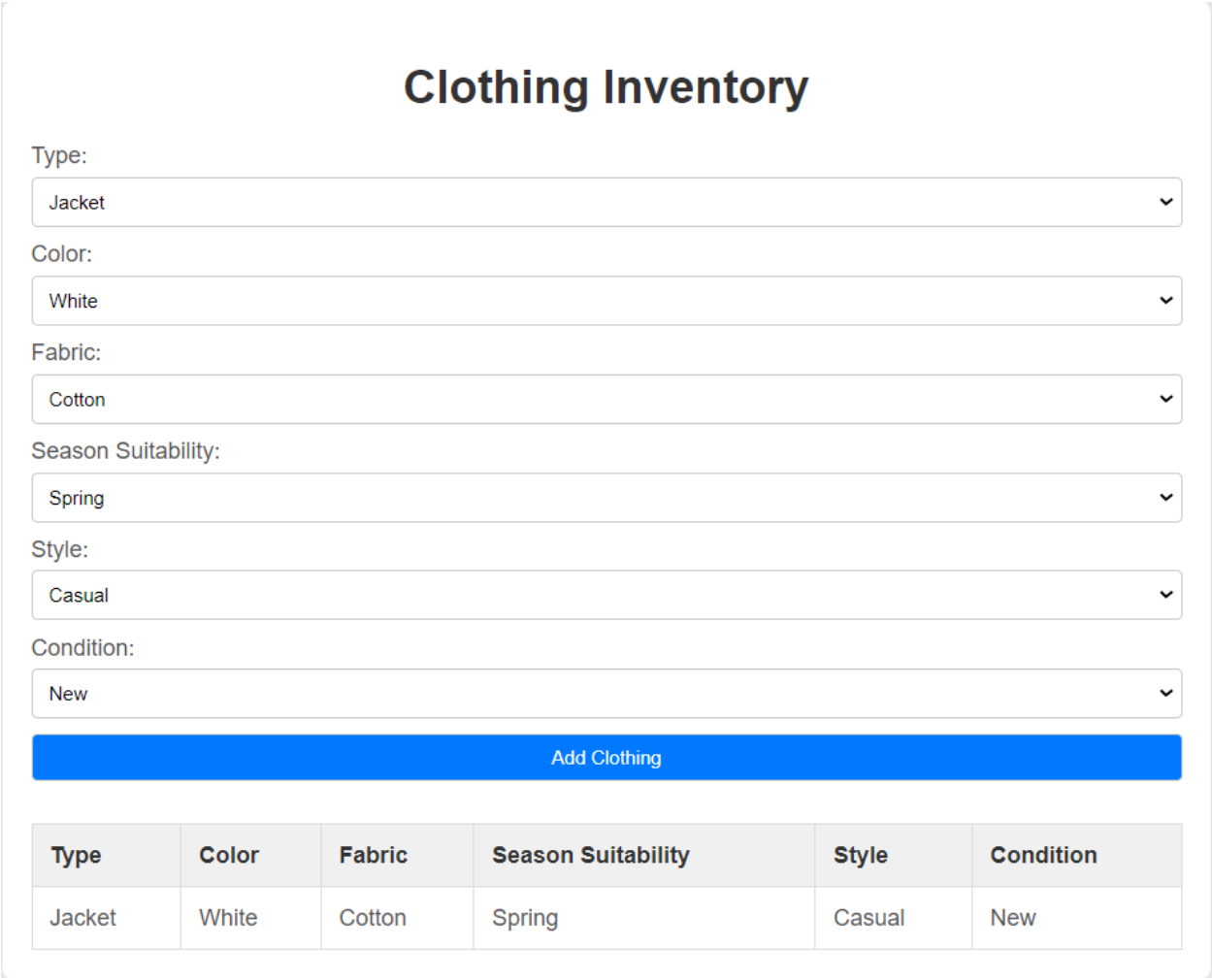
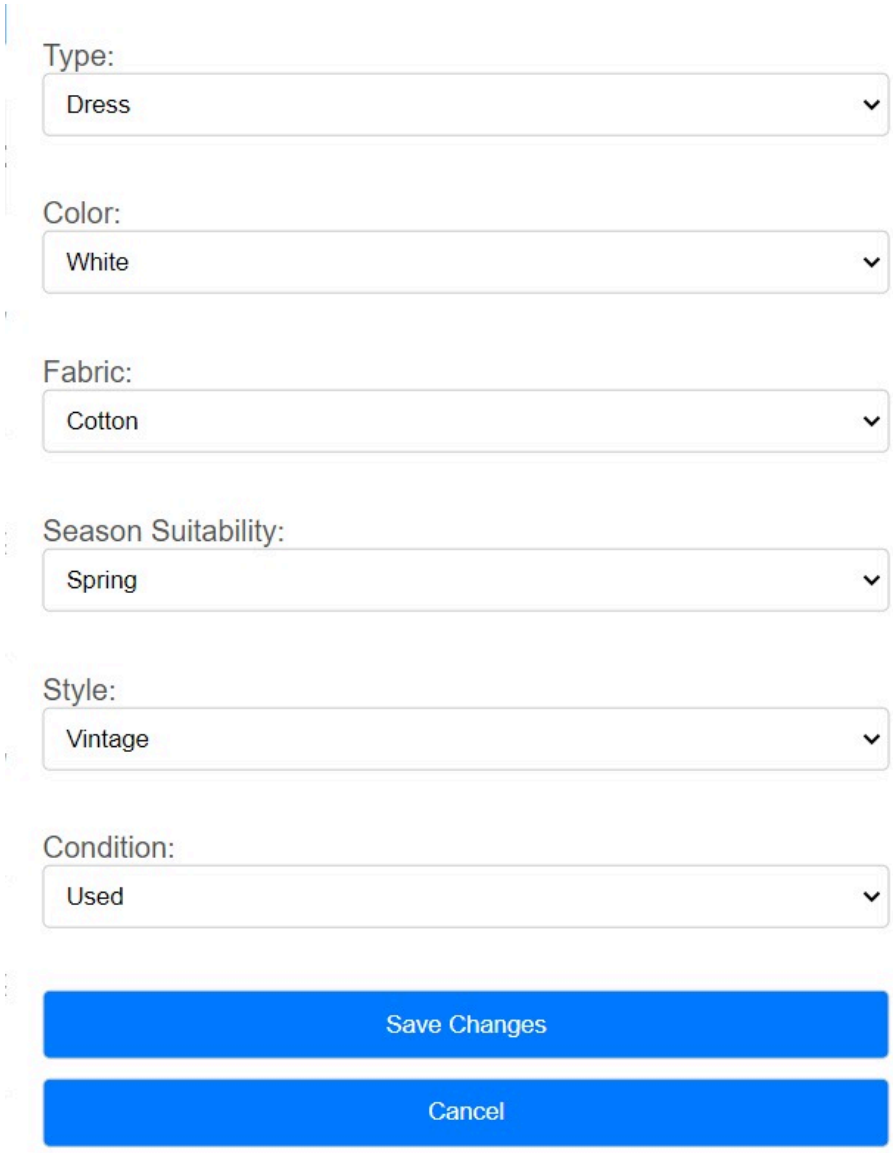


Figure 1 - Front-end Clothing Inventory Section

The user could also edit any of their clothing items to change any of it's attributes by selecting the edit button next to the clothing. The following is an example of the editing window that is

opened once the edit button is selected:



As for the calendar section, the user is able to schedule events or remove events from any day throughout the year by simply clicking on a cell in the calendar and writing its description. The following is a screenshot of the calendar:

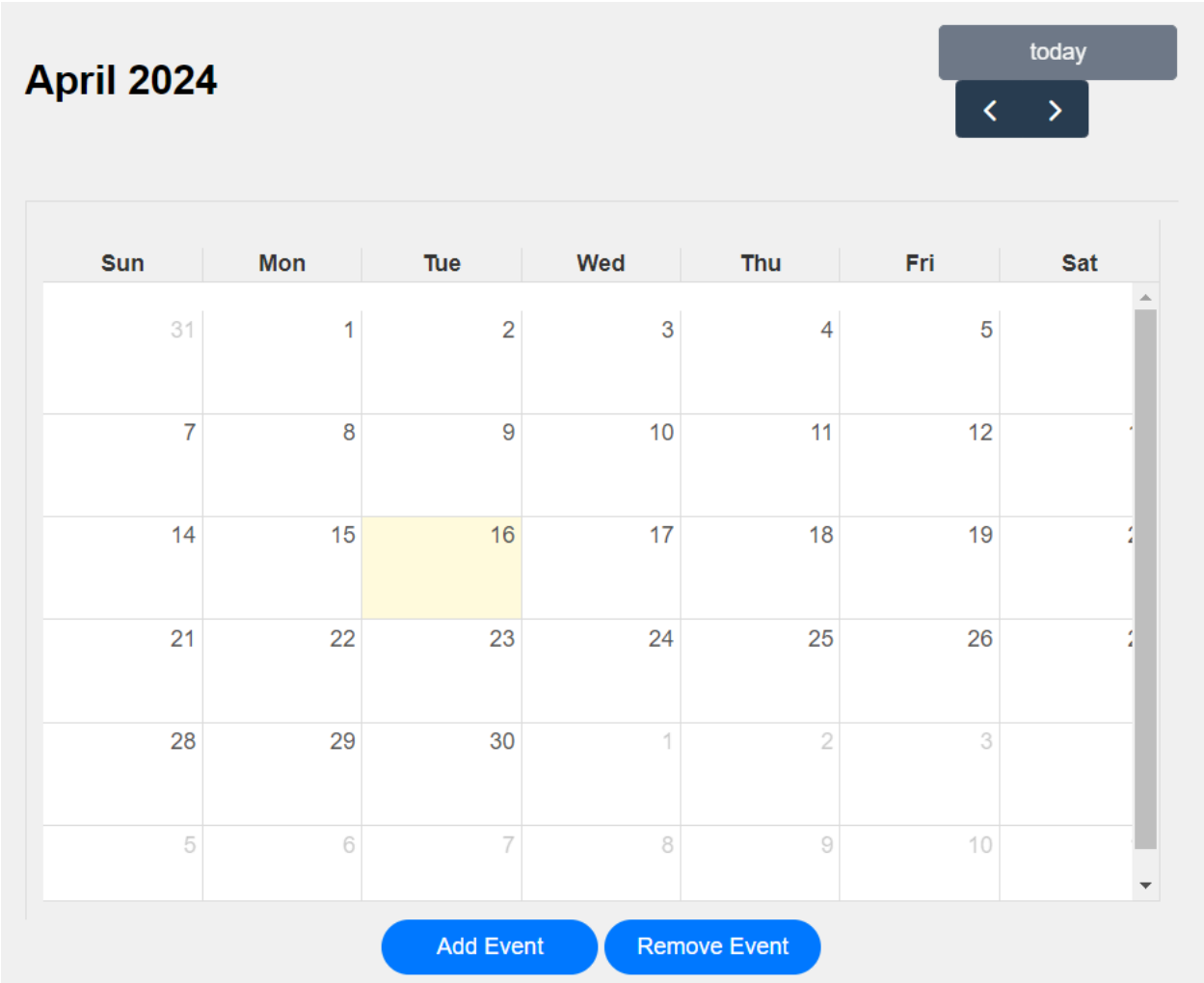
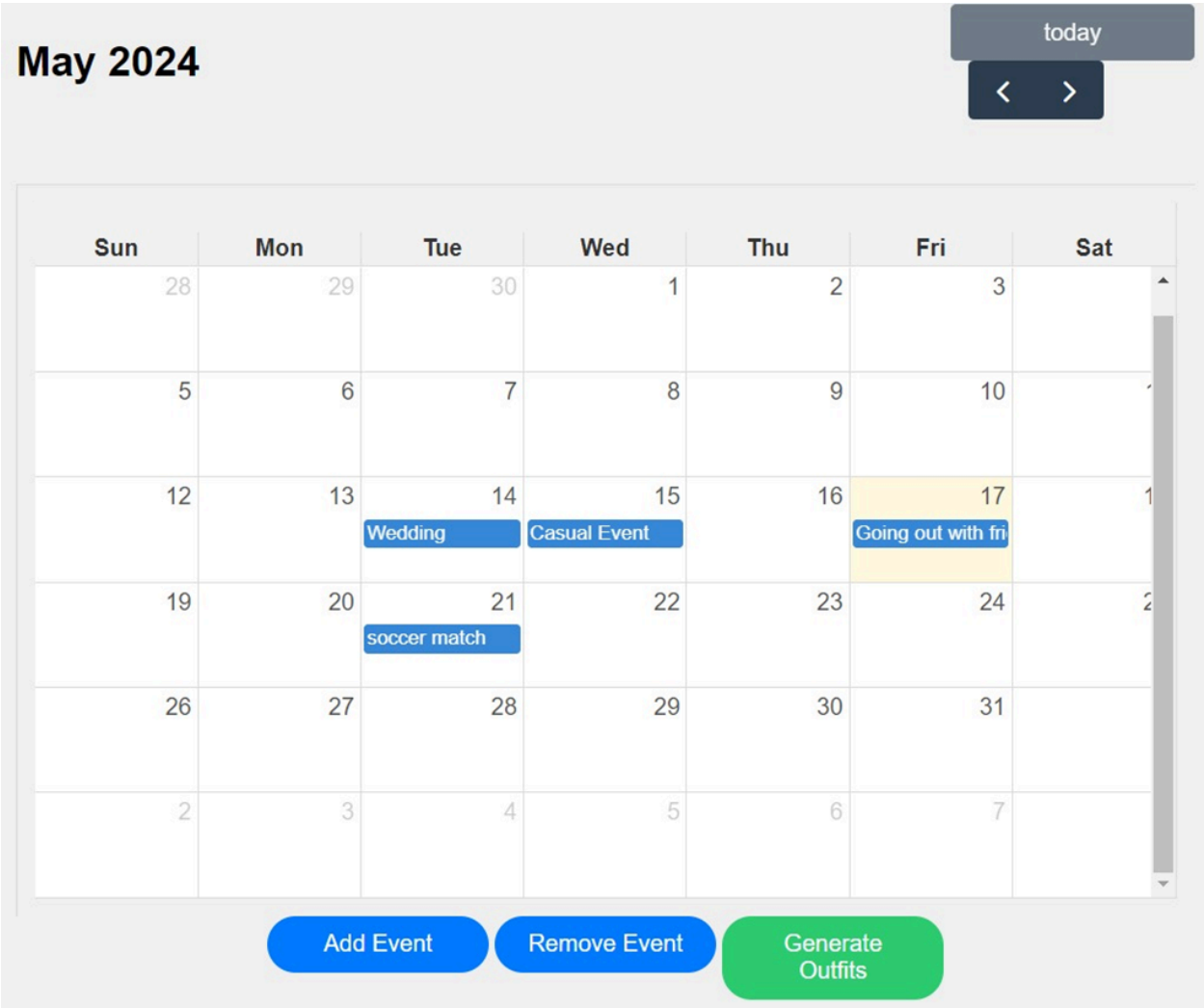


Figure 2 - Front-end Calendar

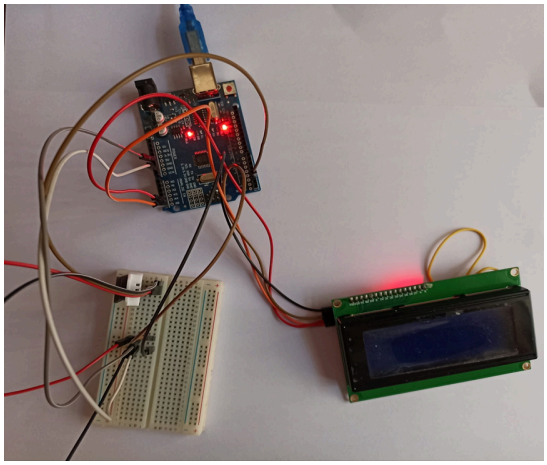
Below is an example of the calendar, but with the user's events being displayed after being read from the csv:



And this is the corresponding csv:

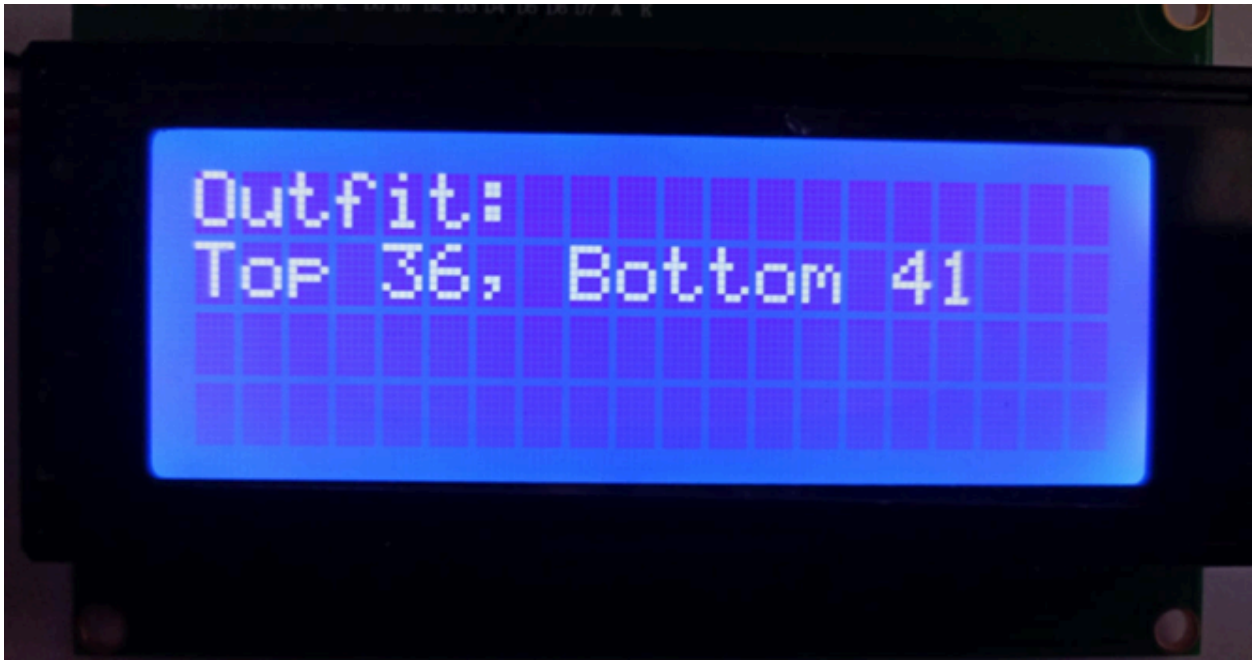
year	month	day	description
2024	5	14	Wedding
2024	5	15	Casual Event
2024	5	21	soccer match
2024	5	17	Going out with friends

6.6 Brief Description of the Hardware



The hardware setup for this project consists of an Arduino board, a DHT22 sensor, a breadboard, and an LCD screen. The DHT22 sensor, placed on the breadboard, is used to read temperature data. This sensor is connected to the Arduino, which processes the temperature readings. The Arduino is also connected to a laptop, providing access to the user's clothing database. Additionally, an LCD screen is connected to the Arduino, allowing for the display of the recommended outfit. In addition to this, the arduino speaker will be reading out the results after being processed with the text-to-speech.

This is an example output displayed on the LCD screen:



7 Testing the System

Testing the WEAR system was pivotal to ensure its efficiency and reliability. Our comprehensive approach focused on evaluating the system's functionality, integration, and user interaction capabilities. These tests aimed to pinpoint and rectify any discrepancies, enhancing the system's overall performance and user experience.

Functional Testing:

- **Adding Clothing and Events:** We conducted extensive testing by inputting various types of clothing and scheduling different events to observe how the system handled diverse data inputs. This involved adding clothes with various attributes (like color, fabric, and type) and creating events on different dates and times. The goal was to ensure that the system could handle and accurately reflect these entries without errors, such as crashes or incorrect data handling.

- **Interaction Tests:** We also tested how well different components of the system interacted with each other. For instance, ensuring that the clothing recommended by the system matched the requirements of the added events, and verifying that the calendar correctly updated and displayed new events.

Debugging and Validation:

- **Console Output:** Throughout the development process, we utilized console outputs to trace and validate the flow of data through different parts of the system. By strategically placing print statements within the code, we could observe the operations being performed, verify the integrity of data at various stages, and identify any discrepancies or unexpected behaviors.
- **Error Handling Tests:** We introduced known errors into the data inputs to ensure that the system could gracefully handle such situations. This included inputting invalid dates, unsupported clothing attributes, and contradictory event details to test the robustness of error handling mechanisms.

User Interface Testing:

- **UI Responsiveness:** Testing was performed to ensure that the user interface remained responsive and functional when interacting with the system. This included verifying that forms accepted inputs as expected, buttons triggered the correct actions, and information displayed on the UI was accurate and updated in real-time.

The testing phase was instrumental in refining the WEAR system. It enabled us to validate the functional integrity of our project, ensuring that the system performed optimally under various scenarios and user interactions. This robust testing framework significantly bolstered the reliability and user satisfaction of the WEAR project.

8 Future Development and Enhancements

8.1 Short-Term Goals

In the near term, we plan to focus on enhancing the accessibility of our interface to better support visually impaired users. This will include integrating simpler voice command capabilities and improving the existing text-to-speech system to make navigation and interaction more intuitive. Additionally, we aim to refine the outfit recommendation algorithm by fine-tuning it based on user feedback, enabling more personalized and relevant outfit suggestions based on user preferences and historical selections.

8.2 Long-Term Goals

Looking ahead, our goal is to enhance the system's infrastructure by gradually moving to a cloud-based environment. This shift will allow for more efficient data management and easier scalability as user numbers grow. We also plan to integrate additional weather forecasting services and calendar functionality to provide more accurate and contextually relevant clothing recommendations under varying conditions and schedules.

8.3 Potential Features

Future enhancements could include the development of a basic mobile app to allow users easy access to the system from their smartphones, improving user convenience. We also consider adding simple social features, enabling users to share their outfits and receive input from friends within the app. Further down the line, we could explore basic data analysis features to detect patterns in user preferences and seasonal clothing trends, which could inform the system's recommendation logic.

9 Conclusion

To sum up, the creation and use of the WEAR system mark a substantial breakthrough in the field of customized clothing technology. With the integration of many technologies, including the FullCalendar API, DHT22 sensors, OpenWeatherMap API, and a file database, the system shows a holistic approach to outfit suggestion. Its capacity to handle calendar

obligations, user preferences, and real-time environmental data enables highly customized and contextually appropriate outfit recommendations. Nevertheless, there are several limitations to the system, such as its dependence on the user's current wardrobe and the lack of suggestions for shoes and accessories. Subsequent improvements can concentrate on adding these components to the database and integrating changing fashion trends and styles. This system sets a precedent for the future of personalized fashion, highlighting the ability of AI and ML to enhance our daily lives and personal style choices.

References

[1] Ai Clothing Detection: Use Cases for fashion and E-commerce. PostIndustria. (n.d.). https://postindustria.com/ai-clothing-detection-use-cases-for-fashion-and-e-commerce/

[2] Nalwan, A. (2020, January 27). *Teaching ai to identify clothes*. Medium. https://agustinus-nalwan.medium.com/teaching-ai-to-identify-clothes-1feb0a8b3446

[3] Daolio, F. (2018, September 6). *Deep learning for fashion attributes*. Medium. https://medium.com/asos-techblog/deep-learning-for-fashion-attributes-763c8c95034c

[4] Nirmal, N. A. (n.d.). *The "closet" database: Industry turns to AI algorithms for categorizing and predicting fashion*. ScienceWriters (www.NASW.org). https://www.nasw.org/article/closet-database-industry-turns-ai-algorithms-categorizing-and-predicting-fashion

[5] Team, I. E. (2023, September 19). *10 key benefits of machine learning (with uses and faqs)*. 10 key benefits of machine learning (with uses and FAQs). https://uk.indeed.com/career-advice/career-development/benefits-of-machine-learning

[6] Uses and benefits of Machine Learning for Your Enterprise. Matillion. (n.d.). https://www.matillion.com/blog/the-uses-of-machine-learning-and-the-benefits-for-your-enterprise

[7] Adservio team. (2022, September 16). *Machine learning: Types: Benefits*. Adservio. https://www.adservio.fr/post/machine-learning-types-benefits

[8] Cliff, M. (2016, June 5). *Women spend six months of their working lives deciding what to wear - and suffer "Wardrobe rage" over trying to choose the right outfit* . Daily Mail Online. https://www.dailymail.co.uk/femail/article-3626069/Women-spend-SIX-MONTHS-working-lives-deciding-wear.html

[9] Arduino Uno REV3. Arduino Official Store. (n.d.). https://store.arduino.cc/products/arduino-uno-rev3

[10] Industries, A. (n.d.). *Emic 2 text-to-speech module*. adafruit industries blog RSS. https://www.adafruit.com/product/924

[11] Low-cost yellow green 4004 40X4 charcter LCD display module. BuyDisplay.com. (n.d.). https://www.buydisplay.com/low-cost-yellow-green-4004-40x4-charcter-lcd-display-module

[12] Addicore DHT22 (AM2302) temperature and humidity sensor. Addicore. (n.d.). https://www.addicore.com/products/dht22-temperature-and-humidity-sensor

[13] Kalantidis, Y. (2013, April). *Getting the look: Clothing recognition and segmentation for automatic ...* Getting the Look: Clothing Recognition and Segmentation for Automatic Product Suggestions in Everyday Photos. https://www.researchgate.net/publication/243973142_Getting_the_Look_Clothing_Recognition_and_Segmentation_for_Automatic_Product_Suggestions_in_Everyday_Photos

[14] Bossard, L., Dantone, M., Leistner, C., Wengert, C., Quack, T., & Gool, L. V. (1970, January 1). *Apparel classification with style*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-642-37447-0_25

[15] Liu, S. (2012, October). *(PDF) hi, Magic Closet, tell me what to wear! - researchgate*. Hi, magic closet, tell me what to wear! https://www.researchgate.net/publication/233988887_Hi_magic_closet_tell_me_what_to_wear

[16] Deep Learning In Fashion (Part 3): Clothing matching tutorial. Indico Data. (2021, October 22). https://indicodata.ai/blog/fashion-matching-tutorial/

[17] Liu, Y. (2017, July). *Weather-to-garment: Weather-oriented clothing recommendation - researchgate*. Weather-to-garment: Weather-oriented clothing recommendation. https://www.researchgate.net/publication/319569282_Weather-to-garment_Weather-oriented_clothing_recommendation

[18] Weather fit - IOS weather app that tells you what to wear. Weather Fit - iOS weather app that tells you what to wear. (n.d.). https://weatherfit.com/

[19] Acloset. (n.d.). https://www.acloset.app/

[20] Mixdress-combine clothes app. MixDress - clothes matching app. (n.d.). https://combineclothes.com/

[21] Outfit organizer App. Pureple. (n.d.). https://pureple.com/