Trendify

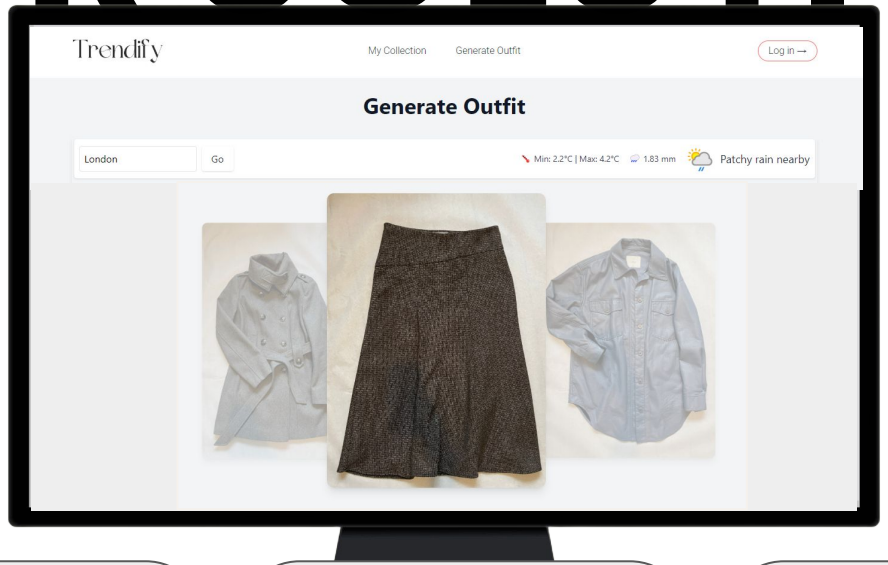
Susan Zheng, Robert Crellin, Agnes Robert, Bart Bielski

Table of Contents.

THE PROBLEM.

- Wasting time picking out appreciate outfits
- Outfits ill-suited for the weather
- Losing track of your purchases

OUR SOLUTION



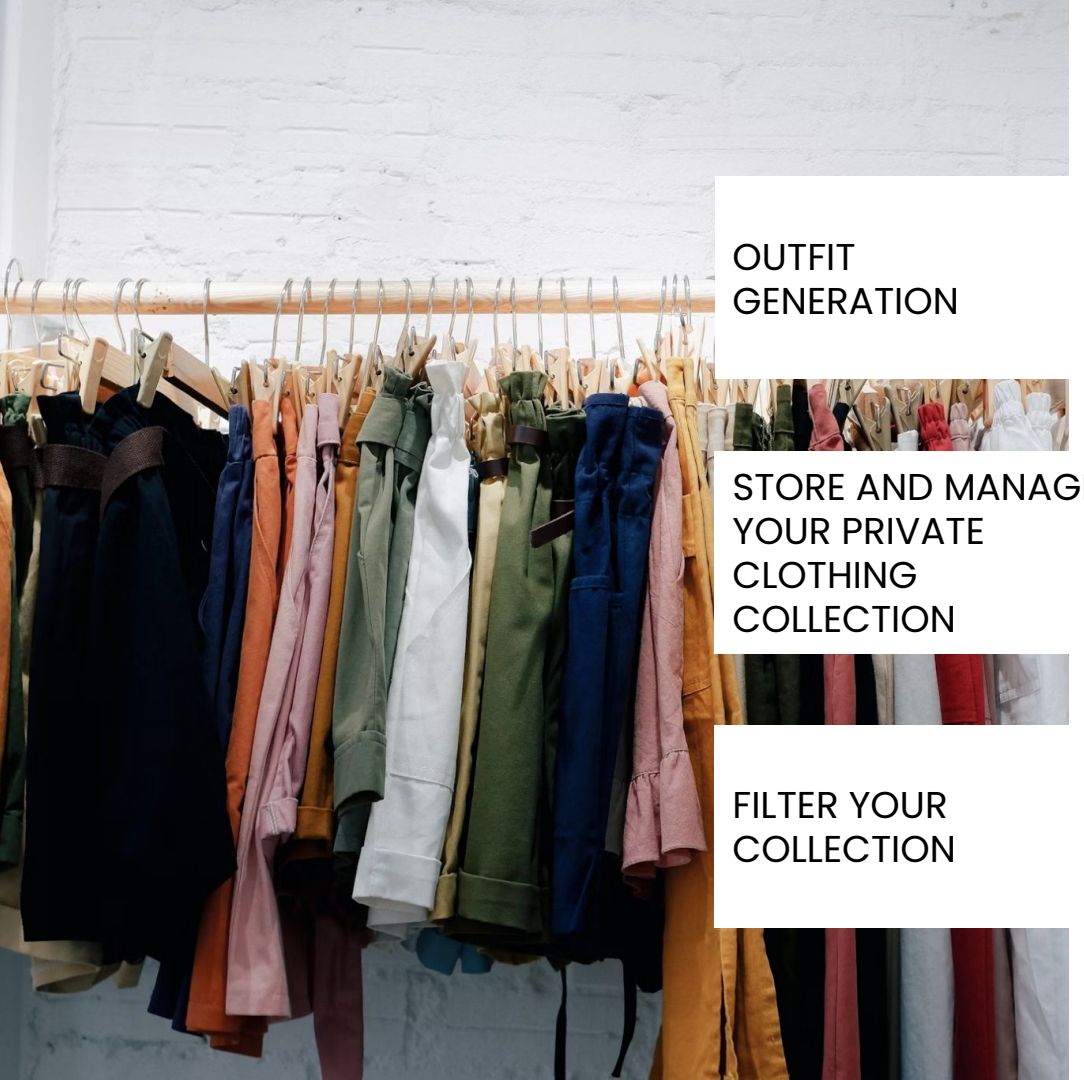
KEEPING TRACK OF
ALL OF YOUR CLOTHES

LOOKING YOUR BEST

ALWAYS DRESSED
FOR THE WEATHER


BEFORE
AFTER

FEATURES.



OUTFIT GENERATION

Receive outfit recommendations based on the current weather in your desired location

STORE AND MANAGE YOUR PRIVATE CLOTHING COLLECTION

Update and manage your private clothing collection. Add or remove items as needed.

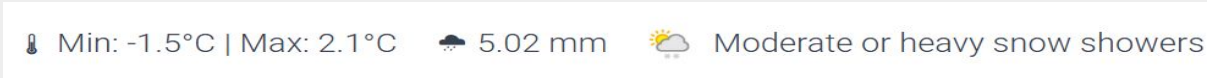
FILTER YOUR COLLECTION

Looking for a specific item? Use the filter menu to find each item with ease!


Live demo.

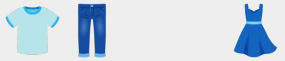
CODE SHOWCASE.

Integrating Weather API to Fetch Real-Time Weather Data




```
8 references
public async Task<(ExecutionStatus, WeatherInfo?)> GetWeatherForecast(string location)
{
    try
    {
        string url = $"{ApiUrl}?key={ApiKey}&q={location}&days=1";
        HttpResponseMessage response = await _httpClient.GetAsync(url);
        response.EnsureSuccessStatusCode();

        string jsonResponse = await response.Content.ReadAsStringAsync();

        var doc = JsonDocument.Parse(jsonResponse);

        var root = doc.RootElement;
        string loc = root.GetProperty("location").GetProperty("name").GetString() ?? "Unknown";
        var forecastDay = root.GetProperty("forecast").GetProperty("forecastday")[0].GetProperty("day");

        var weather = new WeatherInfo()
        {
            Location = loc,
            MinTemp = forecastDay.GetProperty("mintemp_c").GetSingle(),
            MaxTemp = forecastDay.GetProperty("maxtemp_c").GetSingle(),
            AvgTemp = forecastDay.GetProperty("avgtemp_c").GetSingle(),
            Precipitation = forecastDay.GetProperty("totalprecip_mm").GetSingle(),
            Condition = forecastDay.GetProperty("condition").GetProperty("text").GetString() ?? "Unknown",
            ConditionIconUrl = "https:" + forecastDay.GetProperty("condition").GetProperty("icon")
        };
        return (ExecutionStatus.SUCCESS, weather);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error fetching weather: {ex.Message}");
        return (ExecutionStatus.NOT_FOUND, null);
    }
```

Generate Outfit based on location and weather



```
3 references
public async Task<(ExecutionStatus, Outfit?)> MakeOutfitBasedOnWeather(string location)
{
    var (status, weather) = await _weatherService.GetWeatherForecast(location);
    if(status != ExecutionStatus.SUCCESS || weather == null)
        return (ExecutionStatus.NOT_FOUND, null);

    var (sts, outfit) = MakeOutfit();
    if (sts != ExecutionStatus.SUCCESS || outfit == null)
        return (sts, null);

    var modifiedSelection = new List<int>(outfit.ClothingItemsIds);

    try
    {
        if (weather.Precipitation > 2.5 && weather.MaxTemp < 20 && weather.AvgTemp > 10)
        {
            modifiedSelection.Add(GetRandomClothingItem(ClothingKind.Overall));
        }
        else if (weather.AvgTemp <= 10)
        {
            //add jumper AND coat
            modifiedSelection.Add(GetRandomClothingItem(ClothingKind.Outer));
            modifiedSelection.Add(GetRandomClothingItem(ClothingKind.Overall));
        }
        else if (weather.MaxTemp < 15 && weather.MaxTemp > 10)
        {
            //add jumper or coat
            var rand = new Random().Next(0, 2);
            if (rand == 0) //jumper
            {
                modifiedSelection.Add(GetRandomClothingItem(ClothingKind.Outer));
            }
            else //coat
            {
                modifiedSelection.Add(GetRandomClothingItem(ClothingKind.Overall));
            }
        }

    }
    catch(Exception ex)
    {
        Console.WriteLine($"Error in Outfit Generation: {ex.Message}");
        return (ExecutionStatus.INTERNAL_SERVER_ERROR, null);
    }
```

```
8 references
public int GetRandomClothingItem(ClothingKind kind)
{
    var clothingList = _repository.FindAllClothingItems();
    var filteredListOfIds = clothingList
        .Where(c => ClothingKindMapper.GetClothingKind(c.Category) == kind)
        .Select(c => c.Id)
        .ToList();
    var newId = GetUniqueId(filteredListOfIds, new List<int>() { });
    return newId;
}

3 references
public (ExecutionStatus, Outfit) MakeOutfit()
{
    Outfit newOutfit = new Outfit();
    //decide top+bottom OR single
    //filter list based on EITHER single
    //OR random top+ random bottom
    //temperature + jumper and coat
    var clothingList = _repository.FindAllClothingItems();
    var rand = new Random().Next(0, 2);
    List<int> filteredList = new();

    if(rand == 0) //single
    {
        filteredList = new List<int>() { GetRandomClothingItem(ClothingKind.Single) };

    }
    else if(rand == 1) //top + bottom
    {
        filteredList = new List<int>() {
            GetRandomClothingItem(ClothingKind.Top),
            GetRandomClothingItem(ClothingKind.Bottom)
        };
    }

    newOutfit.ClothingItemsIds = filteredList;

    return(ExecutionStatus.SUCCESS, newOutfit);

}
```

Managing User Image Upload: Preview, Processing, and Backend Storage

Frontend - Blazor component

Upload

```
1 reference
public static async Task<Response<int>> UploadImageAsync(IBrowserFile file)
{
    var result = new Response<int>();
    try
    {
        if (file == null)
        {
            result.HasError = true;
            result.ErrorMessage = "No file selected.";
            return result;
        }
        var content = new MultipartFormDataContent();
        var fileContent = new StreamContent(file.OpenReadStream(maxAllowedSize: 10_000_000))
        {
            Headers =
            {
                ContentType = new MediaTypeHeaderValue(file.ContentType)
            }
        };
        content.Add(fileContent, "file", file.Name);

        HttpResponseMessage response = await HttpClient.PostAsync("ClothingItems/Image", content);
        result.StatusCode = response.StatusCode;
```

Preview Image

```
private IBrowserFile? SelectedFile { get; set; } = null;

private async Task HandleFileUpload(InputFileChangeEventArgs e)
{
    SelectedFile = e.File;
    if (SelectedFile != null)
    {
        FileName = SelectedFile.Name;
        var format = SelectedFile.ContentType;
        using var stream = SelectedFile.OpenReadStream(1048576);
        using var memoryStream = new MemoryStream();
        await stream.CopyToAsync(memoryStream);
        var fileBytes = memoryStream.ToArray();

        if (format.StartsWith("image/"))
        {
            IsImage = true;
            FileDataUrl = $"data:{format};base64,{Convert.ToBase64String(fileBytes)}";
            ErrorMessage = String.Empty;
        }
    }
}
```

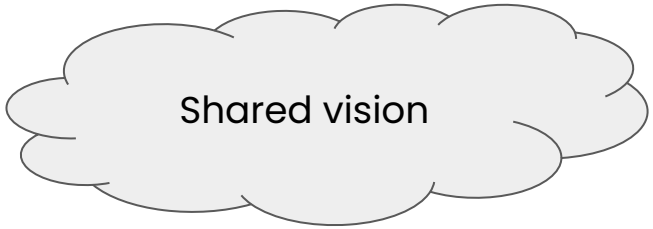
Back-End to store the uploaded image and update the database with its location

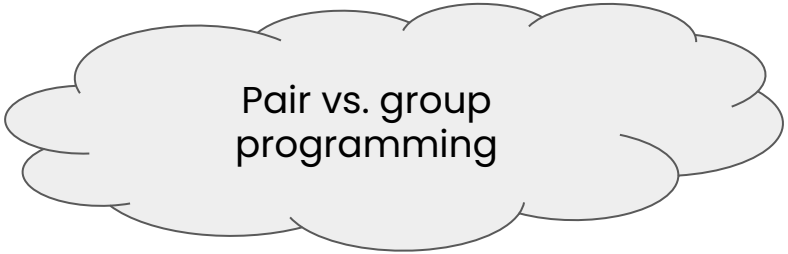
```
2 references
public string GetImageLocation(string filename) => Path.Combine(ImageFolder, Guid.NewGuid() + "_" + filename);

2 references
public (ExecutionStatus status, int? id) SaveImageToDisk(IFormFile file, int id, string locationPath)
{
    try
    {
        using (var fs = new FileStream(locationPath, FileMode.CreateNew))
        {
            file.CopyTo(fs);
        }
    }
    catch (IOException ioEx)
    {
        return (ExecutionStatus.ALREADY_EXISTS, null);
    }
    catch (Exception ex)
    {
        return (ExecutionStatus.INTERNAL_SERVER_ERROR, null);
    }

    return (ExecutionStatus.SUCCESS, id);
}

2 references
public (ExecutionStatus status, int? clothingItemId) SaveImage(IFormFile file, int? clothingItemId) =>
    AddImageLocationToDb(file.FileName, clothingItemId) switch
    {
        (ExecutionStatus.SUCCESS, int imageId, string filePath) => SaveImageToDisk(file, imageId, filePath),
        (ExecutionStatus status, _, _) => (status, null)
    };
```

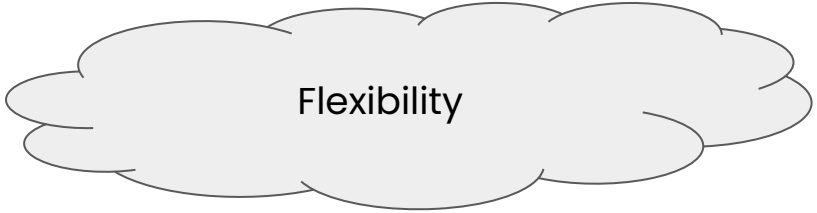
PART 2

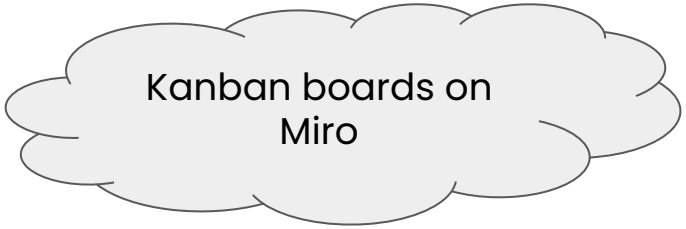
REFLECTIONS

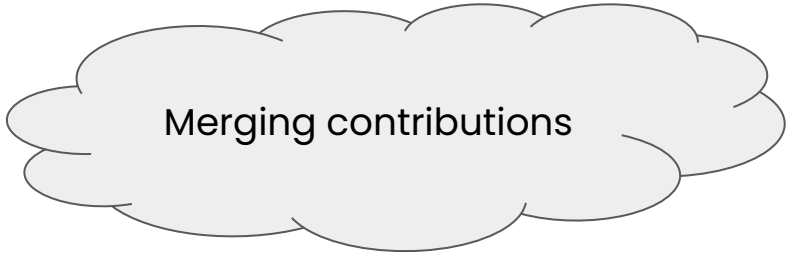
KEY CHALLENGES & TAKEAWAYS – GROUP WORKING











SKILLS LEARNED – TECHNICAL

Image upload / download

Filtering on the frontend

Blazor / JavaScript

PROUDEST MOMENTS & SUCCESSES

Images displaying

Showing weather API icons

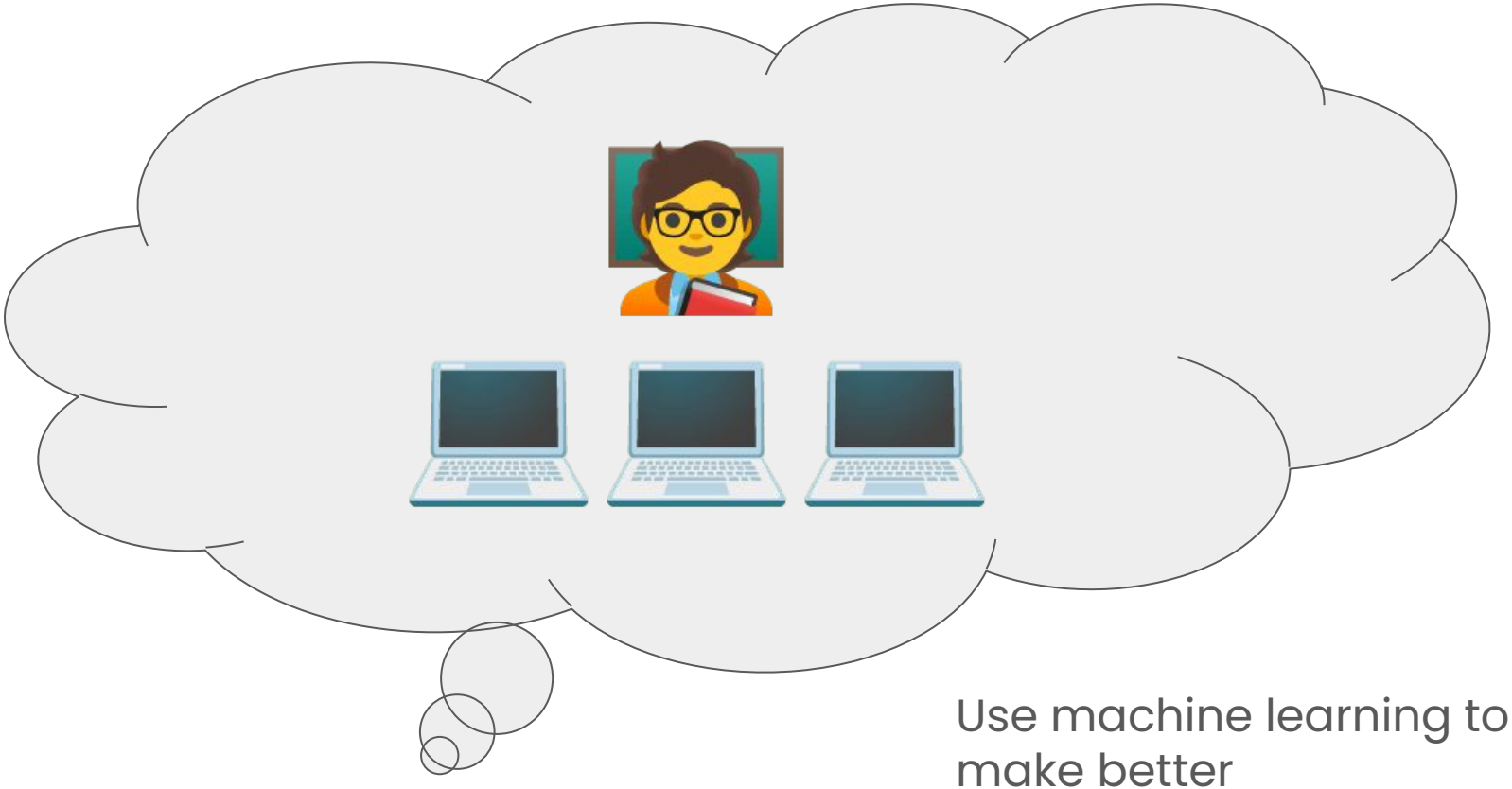
Outfit recommendation working

Clothing carousel

Filtering clothing items working

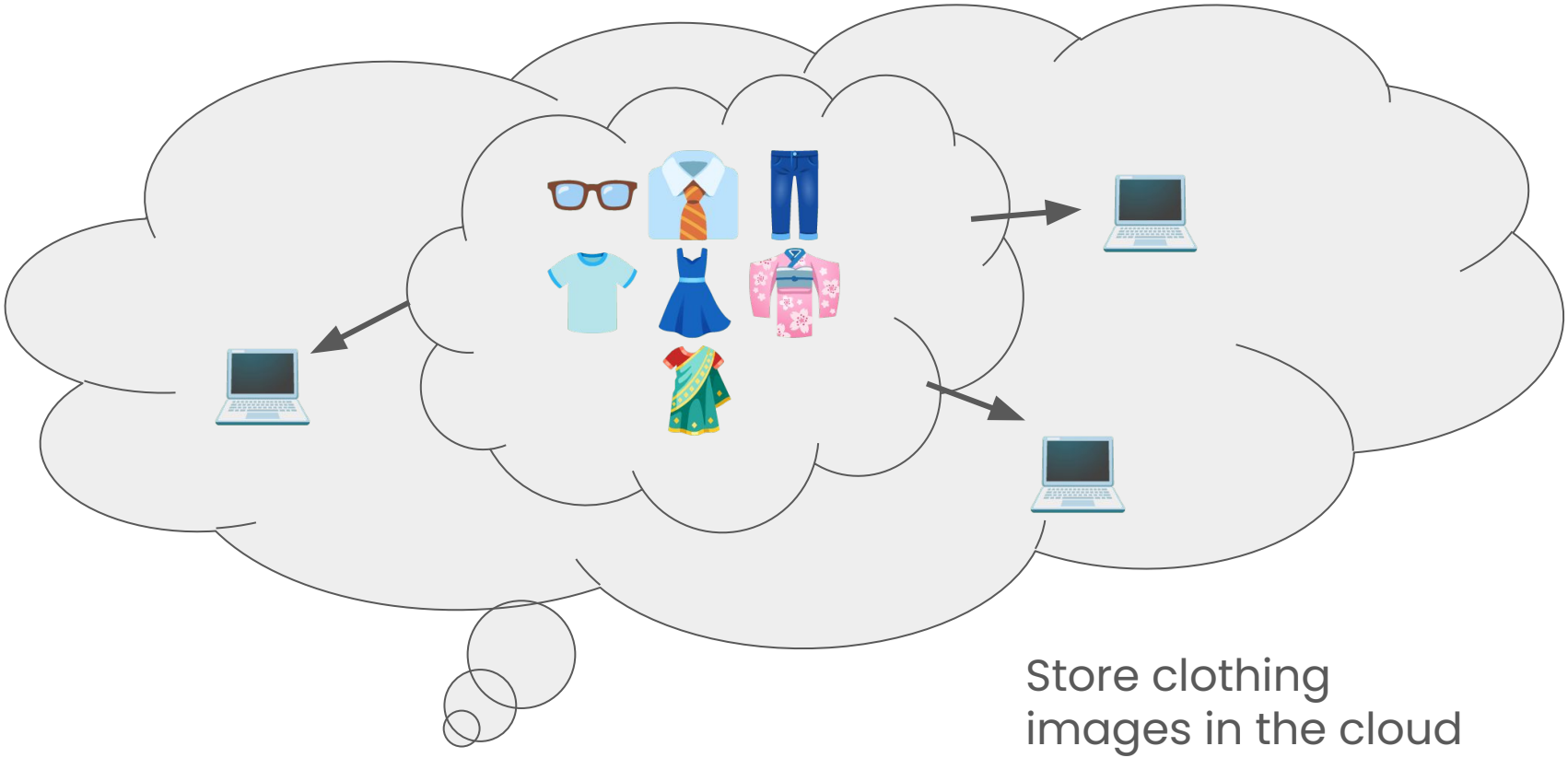
Team communication

WITH MORE TIME WE WOULD...



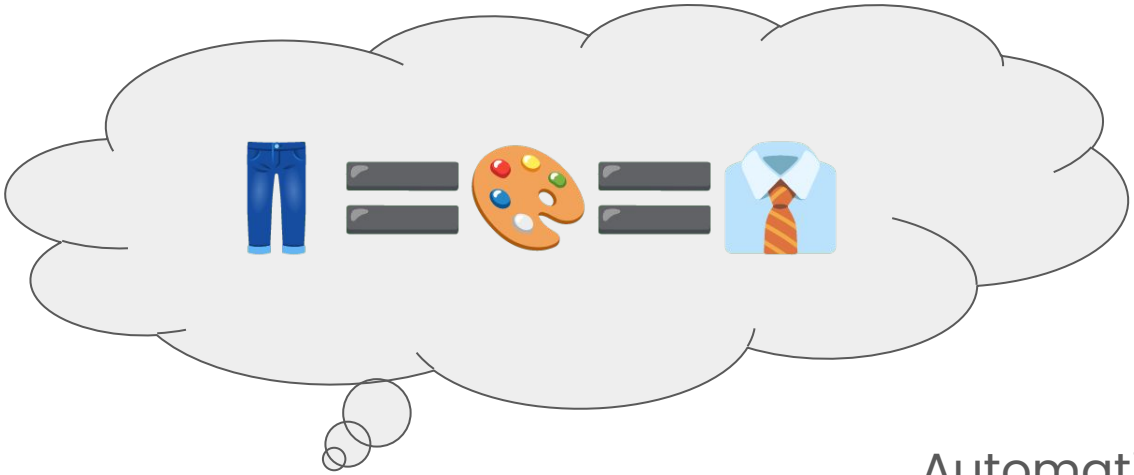
Use machine learning to make better recommendations

WITH MORE TIME WE WOULD...



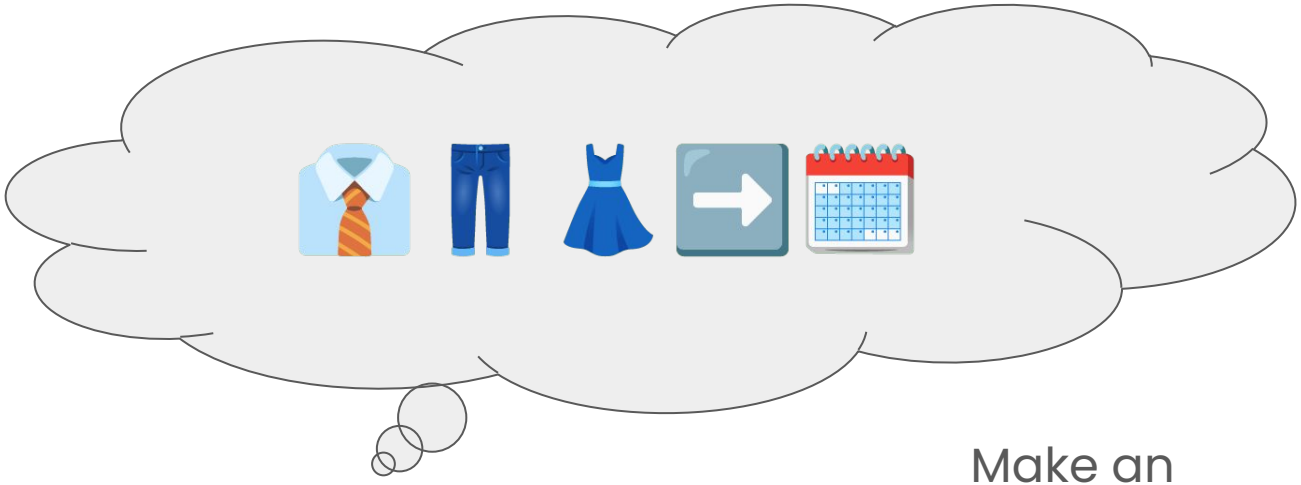
Store clothing images in the cloud

WITH MORE TIME WE WOULD...



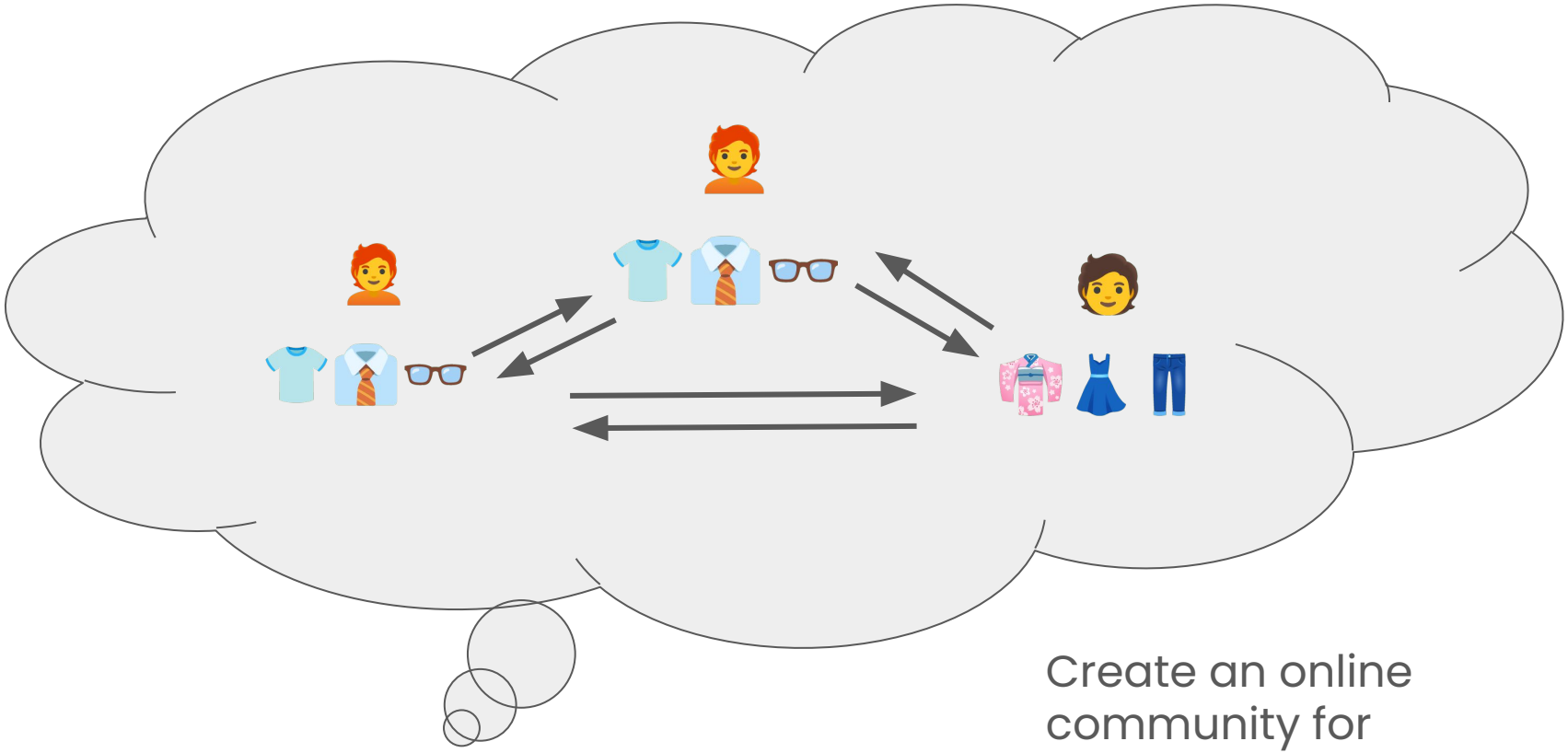
Automatically colour-match clothes

WITH MORE TIME WE WOULD...



Make an outfit-planner

WITH MORE TIME WE WOULD...



Create an online community for exchanging clothes

THANK YOU!

Now you know where to turn when you don’t know what to wear