Tailr: An AI-Powered Outfit Generation App

Georgia Barrand

***Abstract*—Choosing an outfit each day can feel overwhelming, leaving many consumers frustrated and underutilising the clothes they already own. This potentially leads to excess consumerism and outfit dissatisfaction. Tailr is an AI-driven outfit generation app designed for women aged 18-34, offering a solution that empowers users to create stylish outfits from their existing wardrobe. This could lead to better use of existing clothes, reducing consumerism and waste, and increasing people's satisfaction with their own outfits. This project presents three core deliverables: a software architecture and UX design, a proof-of-concept mobile app, and a user evaluation of this app. The app was developed entirely using Flutter/Dart for iOS and Android. A user evaluation of the app conducted on participants within the target demographic confirmed that there is a keen market for this product. Participants found the app to be extremely usable and intuitive, with 100% of participants stating that they'd use a fully developed version. An evaluation of the AI tools used found that the integrated AI returned responses within an average of 2.2s. Overall, Tailr not only simplifies the daily challenge of outfit selection but also promotes a more sustainable and satisfying approach to fashion, positioning itself as a valuable tool for modern, conscious consumers.**

I. Introduction

It's the perpetual question: 'What do I wear?'. The process of determining what to wear each day is an integral part of our lives. While some may view this as a frivolous activity, clothing is our first interface with society and has a significant impact on our mood, productivity and self-perception [1]–[4].

However, selecting an outfit for the day is often draining and time-consuming, leaving many feeling frustrated. In fact, 20% of participants in a poll reported feeling mad or frustrated when they couldn't decide what to wear, with 28% admitting to throwing clothes out of frustration [5]. Furthermore, according to a Stitch Fix survey, over half of participants felt overwhelmed when choosing clothing [6]. These feelings intensify for important events like weddings, first meetings, or a night out. On average, women spend 17 minutes per day selecting an outfit, while men spend 13 minutes [5]. This accounts for almost a year's worth of waking hours over a lifetime. This ongoing struggle can contribute to decision fatigue – the phenomenon where the quality of decisions declines as the number of decisions we make increases [7]. Making excess decisions depletes mental resources and has been linked to a reduced ability to exert self-control, leading to poorer performance on tasks that require discipline, persistence and focus [8].

Moreover, how we dress has been found to impact our mood, productivity and internal perception. Dressing in a self-affirming outfit encourages a sense of pride and achievement [2]. Outfits have been found to both reflect and change individuals' moods [2]. In particular, outfits with positive sentiments were found to foster pride, accomplishment and resilience. Connotations associated with clothing can also impact productivity [1], [3]. The impact of our relationship with fashion highlights its importance within our personal lives and broader society [4]. The significant impact of personal style highlights the need to optimise and accelerate the outfit selection process to maximise these benefits.

Furthermore, our clothing items are significantly underutilised, with only 44% of items worn regularly and 28% left untouched for over a year [5], [9]. The paradox of choice when selecting an outfit can lead to people feeling they have 'nothing to wear'. For example, 61% of Americans struggle to find something to wear despite having sufficient items [9]. Consumers are driven to continuously purchase new clothing to simplify decision-making. However, this exacerbates environmental harm. The fashion industry is estimated to be responsible for 10% of global carbon emissions [10]. To meet the Paris Agreement's goal of limiting global temperature rise, reducing new purchases to five garments per year is recommended [11]. As such, maximising the use of existing clothing is crucial to achieving this target.

Evidently, the current outfit-selection process is time-consuming, mentally taxing and driving overconsumption. Furthermore, this ineffective process hinders individuals from enjoying the full benefits of a confidence-boosting outfit. This project presents the solution: Tailr. Tailr is an AI-powered outfit selection app that takes items that a user already owns and generates outfits based on provided prompts.

Tailr's functionality relies on 3 key components: an image metadata extractor (IME), an outfit-matching AI model, and an outfit selection model. The IME identifies key metadata characteristics from the user's uploaded photos. The outfit-matching AI model uses this metadata to generate a comprehensive set of outfits. The outfit selection AI model selects an outfit based on information from the provided prompt.

Our user study reinforces the need and desire for Tailr. Furthermore, the app clearly satisfies usability requirements. Participants also evaluated two different interfaces for selecting an outfit: a freeform AI prompt and a set of property dropdown selectors. While the AI prompt was preferred, the results indicated that participants would prefer to have both interfaces available. However, participants indicated the need for further refinement of the outfits generated along with an increased ability to customise the app to their tastes. Furthermore, a performance evaluation of the AI models used demonstrates that the app is able to provide satisfactory and correct responses within an average of 2.2 seconds.

A. Environmental and Sustainability Considerations

Several of the UN's Sustainable Development Goals (SDGs) [12] are relevant to Tailr.

Primarily, we must ensure sustainable consumption and production patterns. Tailr may contribute positively to this

goal since it encourages users to reduce their consumption of new clothing. However, the app also needs to use resources responsibly. This is addressed by storing images as textual metadata representations in the database rather than storing the images directly. The images themselves are saved on-device. This requires significantly less storage space and, therefore, reduces the amount of database resources used.

Moreover, the number of times that the outfit generation algorithm is run is minimised. Given that the users' amount of clothes is predominantly stable, the outfit generation functionality can be run once, and then the outfit combinations generated are stored. This outfit store is then queried as the user provides prompts. When the user adds a new clothing item, the algorithm is rerun to update the outfit store. This minimises the number of times the algorithm is run and thereby the resources used. This also improves the time it takes for the user to receive a response. The alternative would be to run the outfit generation algorithm every time the user provides a prompt, which requires significantly more resources.

While the app encourages better use of existing clothes, it risks increasing outfit self-consciousness. This may negatively impact users' mental health (SDG 3) as they are more conscious of their outfits and compare them to others. Furthermore, the app may promote increased consumerism as users may wish to generate more and more varied outfits (SDG 12).

Tailr aims to improve users' mental health and well-being by reducing mental load and providing better outfit selection (SDG 3). However, a key concern is that users' mental health may be negatively impacted if the app gives the perception that their wardrobe is unfashionable. This is mitigated by a flexible outfit generation algorithm that ensures that outfits are always generated.

B. Requirements

While this application could be expanded to encompass all ages and genders, to refine the scope of development and inform the requirements an initial target audience of women aged 18-34 was identified. This was due to a variety of reasons. Firstly, the decision was made to exclude children from the project due to their unique clothing items, and the increased ethical complexity of conducting user evaluations with them. Women were considered over men as typically they have a broader range and variety of clothing, with most 'men's' garment categories being a subset of 'women's' ones. As such, if the app were to be expanded to explicitly include 'men's' clothes, it is expected that this would not require a significant amount of additional work. The age range was limited to 18-34 as this is the age group most likely to be using shopping and beauty mobile apps [13].

To inform the requirements, two key personas were identified based on the target demographic: 'Emily' and 'Serafina'. Emily is the primary persona and Serafina is the secondary persona. The full persona analysis can be found in Appendix C. Based on these personas and the overall solution's goals, the following requirements guided the project's development.

1) Functional Requirements: **FR1.** The application must generate a compendium of outfits that adhere to outfit structure rules (e.g. must only wear one bottom garment) and general outfit-matching best practices.
FR2. Based on the provided prompt, the application must select an appropriate outfit from the generated outfit set.
FR3. Users must be able to view all the clothing items that they have uploaded.
FR4. Users must be able to view all the outfits that they have favourited.
FR5. The application must extract key metadata about uploaded images that accurately describe the clothing item. An exact list of supported metadata is set out in Appendix B.
FR6. Users can permanently delete clothing items from their uploaded items.
FR7. Users can filter the clothing items included in the outfits selected for them based on weather and temperature.

2) Non-Functional Requirements: **NFR1.** The application must provide an outfit selection efficiently when prompted by the user.
NFR2. The application is deployable on mobile phones running at or above iOS 17.2 or Android 14.0.
NFR3. The application is functional on mobile phones with an internet connection, camera hardware, and a screen size of at least 750px × 1334px.
NFR4. The app's interface is intuitive and appealing to women aged 18-34.
NFR5. The app can be interacted with via touch-based interactions.
NFR6. The application's interface must align with the SMASH mobile app heuristics [14].
NFR7. The application can support at least 200 clothing items and 1000 generated outfits per user.

II. Literature Review

This literature review primarily focuses on existing work related to outfit recommendation systems and fashion applications, as this is at the core of Tailr. Several key terms are used to understand the examined work.

A. Background

1) Decision Fatigue: As previously mentioned, a key goal of Tailr is to reduce *decision fatigue* by reducing the number of decisions required to select an outfit. This is the phenomenon that as you make more decisions within a day, the quality of those decisions declines [7]. There are four key symptoms associated with decision fatigue: procrastination, impulsivity, avoidance and indecision. Streamlining choices, delegating decisions, and developing daily routines are key decision fatigue mitigators. Tailr addresses decision fatigue by delegating outfit decisions to an external app.

2) Artificial Intelligence (AI): AI is a field of study that focuses on simulating human-like intelligence using machines. There are several different methods of accomplishing this. Many recommendation systems use *neural networks*. These are computational models inspired by the human brain, consisting of interconnected nodes (neurons) that process and transmit information to perform tasks such as classification, regression, and pattern recognition.

3) Foundation Models: Foundation models such as ChatGPT and Gemini have recently risen to prominence due to their ability to provide accurate and fast responses to a huge range of text and image-based prompts. These models are trained on a vast amount of data which allows them to learn patterns and relationships between items [15]. This allows them to then respond by predicting the next item in a sequence based on the context of previous items and the provided prompt. While these models could suggest outfits based on a user's wardrobe, their general-purpose design makes the UI cumbersome, and managing uploaded items (e.g., deleting or updating) would be difficult. However, many foundation models also have APIs that allow developers to use their capabilities in a custom context. As such, Tailr can utilise their capabilities for image metadata extraction and prompt analysis, as these are complex tasks that would be beyond this project's scope to implement from scratch.

4) Search: Search algorithms are step-by-step procedures used to locate specific data or solutions within a dataset or problem space. There are two fundamental search strategies: uninformed search and informed search. Uninformed search strategies operate solely upon the information provided in the problem definition and have no indication of whether a particular exploration direction is 'more promising' than another [16]. In contrast, informed search strategies utilise problem-specific knowledge beyond the problem definition to find solutions more efficiently [16]. One such method is a *greedy best-first search* - an algorithm that uses a heuristic function to determine what decisions to make. A well-defined search algorithm can simulate system intelligence with less complexity than an explicit artificial intelligence approach.

B. Related Work

1) Fashion Recommendation Systems: Fashion recommendation systems are a substantial area of research, with multiple techniques being employed to explore item compatibility and item recommendation. Work within this area typically focuses on achieving one or more of the following tasks: item pairing recommendations, fill-in-the-blank (FITB) outfit completion and outfit recommendation.

The item pairing task focuses on suggesting individual clothing items of a specified category that pair well with a specified item. For example, if provided a pair of pants, recommend a top. FITB outfit completion builds upon this task. For this task, the goal is to suggest a suitable missing item that best completes a partially complete outfit. This task is typically used to evaluate fashion outfit composition. Finally, outfit recommendation constructs a complete outfit from scratch. This may be prompted by providing an item to build the outfit off or may be informed by metrics such as user attention to particular items.

Existing works have used a large variety of models to attempt to accomplish these tasks. However, most state-of-the-art works are based on a deep learning (DL) architecture.

Prato et al. [17] present a two-stage DL algorithm to accomplish FITB tasks. The first stage extracts tensor representation from multimodal inputs. The second stage leverages the Transformer architecture to learn compatibility between clothing items. Similarly, Sarkar et al. [18] propose OutfitTransformer, a scalable framework that uses task-specific tokens and the self-attention mechanism to learn relationships between items in an outfit. This is used to accomplish item pairing and FITB tasks. Both models outperformed standard state-of-the-art models on the Polyvore dataset. Furthermore, Prato et al.'s model was evaluated by industry fashion experts who found the outfits produced to be compatible 88% of the time.

Lin et al. also incorporate a transformer-inspired attention mechanism in their two-stage neural network OutfitNet [19] for personalised outfit recommendations. In the first stage, the Fashion Item Relevancy network is created which learns the compatibility between items. In the second stage, the Outfit Preference network incorporates user attention to generate personalised outfit recommendations. OutfitNet was also evaluated using FITB testing and was able to outperform standard state-of-the-art models. A key differentiator of this model is its integration of user attention to inform predictions. Essentially, the model takes note of how much attention a user pays to an item and is more likely to recommend outfits with items that have higher attention. This allows OutfitNet to generate more personalised predictions that may be more likely to satisfy users.

Diffusion models are also a significant area of research for fashion recommendation systems. For example, Xu et al. propose DiFashion [20], a generative diffusion model for personalised outfit recommendations. It gradually corrupts images with Gaussian noise and then uses a conditional denoising process to generate multiple fashion items simultaneously. The process is guided by three conditions: category prompt, mutual compatibility, and user attention based on interaction history. This model is evaluated on FITB and outfit recommendation tasks. Again, it was able to achieve superior performance to other baseline state-of-the-art models.

While all of these models are able to recommend compatible outfits, they have several key limitations. Primarily, most of the models proposed are only able to accomplish item pairing or FITB tasks, rather than outfit recommendations from scratch. They require the model to be supplied with the majority of an outfit in order to provide an outfit recommendation. This wouldn't address this project's goal of reducing decision fatigue as the user would still have to decide the majority of items within their outfit.

Furthermore, while these models may be able to make compatible predictions, there is a significant lack of ability to specify context or user preference. While OutfitNet and DiFashion use user attention to inform recommendations, they are unable to use context-specific factors to generate an outfit. For example, factors such as weather, occasion and temperature significantly impact the appropriateness of outfits on a given day. Furthermore, factors such as the user's mood or general idea of what they want to wear for an outfit aren't taken into consideration. So while these outfits may be to generate compatible outfits, they are not context-specific and may not align with the user's needs.

The models surveyed also appear to be fairly limited in the matches recommended. In particular, the models don't appear to be creative when pairing items based on colour. Many of

the outfits generated appear to be monochrome or limited to two colours. Bolder choices like complementary, analogous or triadic colour combinations don't appear to be recommended.

Finally, the most evident shortcoming of these models is the lack of a usable interface for users to interact with. In some studies, users are able to supply an item for the algorithm to generate an outfit based on. However, none of the works examined allow users to provide more complex inputs such as freeform prompts. Furthermore, they are all tested using existing datasets rather than allowing users to trial using their own clothes. Adding this component is essential for these models to actually provide benefits to users.

Overall, the works demonstrated that DL architectures are the current state-of-the-art for fashion recommendation tasks. Several different models have been found to be effective including Transformer, diffusion and general neural networks. The majority of these works, however, focus on accomplishing outfit pairing or FITB tasks rather than complete outfit generation. Furthermore, there is significant scope for further expanding the models to integrate user preferences, more experimental pairings and adding a user-friendly interface.

2) Fashion Applications: There is a significant selection of fashion-related apps available in the current market. These can include e-commerce, visual discovery, or explicit outfit-generation apps.

There are several fashion-related e-commerce platforms that are relevant to Tailr. We focus on evaluating ASOS, Depop and Stitch Fix. ASOS is an online fashion retail platform targeting users aged 18–31 [21]. Users can browse and purchase fashion items via its website and mobile app. Users can filter items by various attributes or category-specific qualities. Item pages provide detailed descriptions and styled outfit suggestions. The app also offers personalised recommendations based on shopping habits, saved items and recently viewed categories.

Notably, ASOS offers a Style Match tool that allows users to upload photos for metadata analysis; identifying colour, pattern, and clothing type to suggest similar items. This demonstrates that user-taken photos can be effective inputs, even in non-ideal conditions. For example, a test photo of black jeans taken on a mixed backdrop still yielded relevant results (see Appendix D).

Depop is another e-commerce app where users can buy and sell second-hand or vintage fashion items. It targets a younger demographic interested in sustainable and unique fashion choices. While Depop facilitates clothing discovery and purchases, it does not offer outfit recommendations or styling assistance. Its focus is on individual items rather than complete outfits, leaving the styling decision entirely up to the user.

Stitch Fix provides personalised styling services by combining AI with human stylists to curate outfits for users. The service ships a box of curated clothing items to the user, who can keep or return them. While Stitch Fix offers personalised recommendations based on style profiles, it cannot factor in users' existing wardrobes. It also emphasizes new purchases rather than maximising the use of owned clothing. Tailr's approach to suggesting outfits from an existing wardrobe addresses this gap, offering a more sustainable and personalised solution by helping users style clothes they already own.

The primary difference between e-commerce apps and Tailr is the use case. Fashion retail platforms encourage users to purchase new clothes, without considering whether these items will match their existing wardrobe or suit their personal style. While apps like Depop improve on this by reselling existing clothes, they still promote consumerism. Clothing recommendations are subtly integrated into these apps, with items ranked higher in search results rather than explicitly suggested. While apps like Stitch Fix take into account the user's preferences, these recommendations don't take the user's existing wardrobe into account. Furthermore, while ASOS's product images show items within outfits, this does not help users make better styling choices, as these outfits are not user-specific.

Visual discovery platforms such as Pinterest are also relevant to Tailr. Pinterest allows users to explore and save ideas, including fashion inspiration. While Pinterest enables users to discover styling ideas and curate mood boards, it is not a fashion-specific app and does not provide personalised outfit suggestions. It may inadvertently recommend clothing items based on the user's saved posts, however, this is not the app's primary purpose. Tailr's structured and goal-oriented approach to outfit creation contrasts with Pinterest's broader, inspiration-focused model.

Finally, Acloset is a key example of an existing outfit generation app. Acloset allows users to create a digital version of their closet and create their own outfits [22]. With more than a million worldwide users, this app demonstrates the value of Tailr due to its similarity.

However, there are some key distinctions between Acloset and Tailr. Acloset primarily focuses on users creating their own outfits. This somewhat simplifies outfit selection as the user doesn't have to physically look through their closet, however, it doesn't fully alleviate decision fatigue. Furthermore, while Acloset does have some AI-powered recommendations, these are limited to outfits based on weather, colour and location. By allowing users to craft their own prompts, we are able to suggest outfits using multiple metadata factors, crafting more specific and tailored outfits.

Acloset emphasises the importance of FR2 and FR7 for removing the mental load and decision fatigue of selecting an outfit. These are key requirements that distinguish our app.

Overall, the apps surveyed highlight key UI attributes that our target demographic values. In particular, simplistic and minimal design was consistent throughout all the apps. Additional features such as filtering also help to aid in item discovery, even without precise searches. Furthermore, efficient and intuitive navigation was key for navigating quickly through the apps.

Moreover, while our specific use case is unique, these apps validate the interest in and value of fashion-based apps.

C. Summary of Existing Work

A summary of current existing work in regard to how they fulfil the established functional and non-functional requirements can be found in Tables I and II.

Existing Work	FR1	FR2	FR3	FR4	FR5	FR6	FR7
Prato et al.	Partial	No	No	No	Yes	No	No
OutfitTransformer	Partial	No	No	No	Yes	No	No
OutfitNet	Partial	No	No	No	Yes	No	No
DiFashion	Partial	No	No	No	Yes	No	No
ASOS	No	No	No	Partial	Yes	No	Partial
Acloset	Partial	Partial	Yes	Yes	Yes	Yes	Partial
Depop	No	No	Partial	Partial	No	Yes	Partial
Pinterest	No	No	Partial	Partial	Yes	Partial	Partial

TABLE I: Comparison of Existing Work Against Functional Requirements

Existing Work	NFR1	NFR2	NFR3	NFR4	NFR5	NFR6	NFR7
Prato et al.	Partial	No	No	No	No	No	Yes
OutfitTransformer	Partial	No	No	No	No	No	Yes
OutfitNet	Partial	No	No	No	No	No	Yes
DiFashion	Partial	No	No	No	No	No	Yes
ASOS	No	Yes	Yes	Yes	Yes	Yes	Partial
Acloset	Yes	Yes	Yes	Partial	Yes	Partial	Partial
Depop	No	Yes	Yes	Yes	Yes	Yes	Partial
Pinterest	No	Yes	Yes	Partial	Yes	Yes	Partial

TABLE II: Comparison of Existing Work Against Non-Functional Requirements

D. Tools and Methodology

Several key tools enabled the development and assisted in the engineering process of Tailr.

1) ChatGPT: ChatGPT was leveraged as a development tool to streamline the Flutter app development process. It was primarily used to help identify key Flutter components and libraries for use in the app, which would otherwise have required time-consuming manual searches to find. This was especially crucial as I have limited Flutter experience, and lacked familiarity with the full range of in-built components. Using these components was key to ensuring design consistency across the app and avoiding unnecessarily rewriting code. As such, ChatGPT was used to find relevant choices to select from.

Furthermore, ChatGPT's ability to generate baseline code for classes was particularly helpful in kickstarting the coding process. It was able to provide a basic class template that I could then build upon. This allowed me to focus on the key functionality of the app rather than being slowed down by setup.

However, ChatGPT wasn't without limitations. While it excelled at surfacing relevant components and generating foundational code, it often struggled with the more nuanced task of integrating these components seamlessly. Occasionally, it also misinterpreted prompts, leading to suggestions that didn't quite fit the intended use case. It also occasionally recommended out-of-date or deprecated functions or libraries. As a result, the developer needed to carefully assess each response, extracting useful insights while filtering out less relevant or incorrect information. Despite this, the iterative process of refining responses still offered value, as it helped me to clarify my understanding of the tools at hand and progress more quickly than if they had worked unaided.

2) XCode and TestFlight: The development process was conducted on an M3 MacBook Pro using Apple's development tools: XCode and TestFlight. XCode was a key tool for deploying the app to an iPhone test device, though the initial setup posed several challenges. One significant issue involved difficulties in signing the app for approval, as XCode did not initially recognise the test phone's certificate. However, after resolving these setup challenges, XCode proved to be a reliable and efficient tool for development. It was particularly useful for archiving builds, which could then be distributed via TestFlight – Apple's platform for internal testing.

TestFlight played an important role in facilitating user testing. Its simplicity allowed the app to be easily deleted and reinstalled after each test, ensuring that any uploaded data was removed and the app could be tested in a fresh environment. This capability also allowed the app to be distributed to additional testers, expanding the pool of feedback. As a result, several bugs and usability issues were uncovered that would likely have gone unnoticed without this broader testing.

3) Methodology: The project followed a rapid prototyping methodology. Prototypes were created and updated every week. These were then shown to the project's supervisors and relevant ENGR489 students within the target audience for feedback. In particular, discussions with supervisors focused on the environmental effects of potential decisions, allowing us to create an informed and conscious design. Based on this feedback, changes were made. This methodology was selected as it is efficient, suitable for an individual project, and still provides development accountability. Furthermore, given the novelty and uniqueness of the project, it was key to trial different implementation methods to determine the most effective way of doing things. A key example of this methodology working was when implementing the database. Initially, Firebase – a NoSQL database – was trialled due to its scalability and ease of integrating with a Flutter app. However, upon implementation, it was discovered that Firebase's querying abilities weren't comprehensive enough for the app's requirements. For example, there was no ability to select distinct values. However, due to the rapid prototyping methodology, I could rapidly review and evaluate the tool and pivot to another.

Development was driven by working through a feature list, tracked using GitLab. Features were broken down into atomic

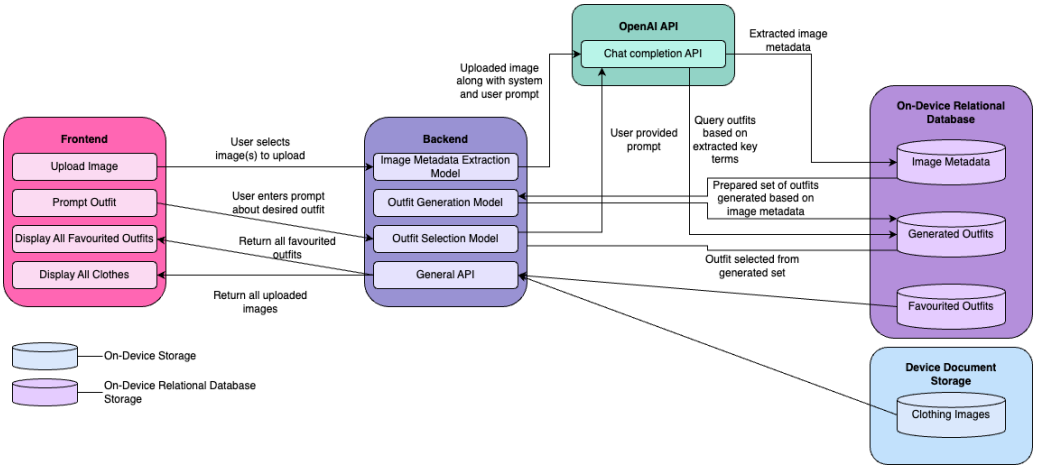

Fig. 1: Application Structure

parts, which were then divided into Issues. From each Issue, an associated branch and Merge Request were created. Once the feature was implemented, a self-review of the issue was conducted before merging the branch.

III. Design

This section details Tailr's design and the rationale behind the design decisions made.

Overall, the app uses a layered architecture, where the presentation layer (frontend), the data access layer (backend) and the database layer are separated. This allows us to separate concerns so that each layer addresses a distinct purpose. There is an additional database layer to store image metadata, outfits and favourites. Clothing images are stored on-device to reduce the number of database resources used (aligning with SDG 12) and to provide data ownership and security.

A. User Interface

The user interface (UI) is primarily responsible for addressing FR3, FR4, FR6, FR7, NFR4, NFR5 and NFR6. It is responsible for providing a touch-based, intuitive interface for the user to navigate the app. As shown in Figure 1, the UI provides functionality to upload images, prompt outfits, display favourited outfits, and display all clothes.

The initial design for the UI was guided by the created personas (see Appendix C). The personas identified several key tasks that the app needed to support: uploading and automatically extracting metadata from clothing images, viewing and filtering all clothing items, generating an outfit from a text-based prompt and saving a generated outfit. From this, initial paper wireframes were created (see Appendix F). Design decisions were informed by several factors. Iterative feedback was solicited from relevant ENGR489 peers within the target audience of women aged 18-34. Furthermore, relevant mobile app heuristics such as SMASH [14] and Apple's Human Interface Guidelines [23] were used. The design is deliberately simple for ease of use, clarity and overall aesthetic appeal.

B. Database

The database is responsible for storing and persisting information related to clothing items, outfits and favourites. It is primarily responsible for NFR7, however, it supports all functional requirements as it provides the data necessary to perform the required tasks.

For data storage, there were 4 key options considered: a NoSQL database, a relational database, a simple file and Flutter's SharedPreferences. NoSQL databases were considered predominantly due to their ability to scale and handle huge volumes of data. NoSQL databases are also schemaless, meaning that they have increased flexibility and ability to handle unstructured data. However, this means that they sacrifice strong data consistency and complex querying capabilities. Given that our data is structured, there was no benefit to using a schemaless architecture. Using a simple file (e.g. JSON, CSV, etc.) was also considered due to the small scale of our data. This would be simpler to implement than a NoSQL or relational database. However, it would require all querying capabilities to be custom-implemented, manual data concurrency handling, and manual entry index handling. This would result in a high likelihood of error or inconsistency in our data. Flutter's SharedPreferences was considered again due to the simplicity of implementation. However, SharedPreferences can only store small-scale, key-value pairs, making it unsuitable for the scale and complexity of our data. A relational database was ultimately selected as the data storage method due to its ability to scale, complex querying capabilities and ACID compliance.

Furthermore, a key design decision was whether to use on-device database storage or to create a database server. Using a database server offers better scalability, centralised data management, and data backup and recovery. However, database servers require more infrastructure and would have significant development complexity. Similarly, on-device storage also offers several advantages. It provides faster data access, as there is no need to rely on network connectivity or experience latency from remote database servers. It reduces the need for constant network traffic, which can lower both bandwidth usage and power consumption in alignment with SDG 12. Additionally, on-device storage enhances user privacy by keeping data locally without the need to transmit it over the internet. While scalability is limited by each user's device

storage, each user's data scale is relatively small and should be able to be handled by most modern mobile devices. Due to these factors, on-device storage was ultimately selected.

C. Backend and AI

The backend is primarily responsible for addressing FR1, FR2, FR5 and NFR1. As shown in Figure 1, the backend provides functionality for extracting metadata features using computer vision, generating the outfit compendium, and selecting an outfit based on a prompt. The backend interfaces with databases to store and retrieve relevant data.

A key component is the metadata extraction model. Instead of creating a custom image analysis tool, the application interfaces with an external AI API. This decision was made due to the limited scope and timeframe of the project. Creating an effective image analysis tool is a complex task and likely a custom tool would produce inferior results to the AI APIs currently available. To extract the metadata, the app's backend sends a request to the API with the respective image attached. The API then responds with the metadata specified in Appendix B, in a parseable format.

Furthermore, the app interfaces with an external AI API in order to extract and infer key information from the user's provided prompt. This extracted information is then used to find suitable outfits that match the extracted criteria.

Finally, the backend is responsible for generating new outfits each time a new item is uploaded. These outfits are generated based on compatibility rules regarding colour, weather, temperature, occasion, pattern, and category.

IV. Implementation

Building Tailr involved several key technical decisions and challenges, from the selection of development tools to the implementation of its AI-driven backend. This section provides a detailed account of how the app was implemented and the process behind this.

A. User Interface

Selecting the right tools for implementing Tailr's UI was crucial for ensuring cross-platform compatibility and ease of development. There were three key candidates for the UI programming language: Flutter, Swift, and Objective-C. These are all languages used for frontend mobile development. Objective-C and Swift were considered as I have significant experience with Objective-C, which is also the basis of Swift. However, this experience was mostly with building on existing applications. Furthermore, Objective-C has an unusual and clunky syntax, which may have slowed development. By comparison, Swift's syntax is much more simplified.

However, Objective-C and Swift are only compatible with iOS - meaning that NFR2 would not be met. Flutter, however, automatically provides cross-platform compatibility, including iOS and Android. Additionally, Objective-C and Swift don't offer 'hot-reloading', meaning that the app needs to be recompiled every time a change is made. This would make development cumbersome. Furthermore, Flutter offers a built-in 'refactor' feature, which allows the developer to instantly extract components, apply styling, or adjust the layout. This accelerates the coding process. While I was unfamiliar with Flutter, it is based on the Dart language, which has a syntax similar to Java. Furthermore, there is robust documentation available along with official libraries.

Flutter was ultimately selected as the front-end programming language based on its development features, cross-platform compatibility, and significant documentation.

The implemented UI successfully provides functionality to upload images, prompt outfits, display favourited outfits, and display all clothes. The UI utilises the Material UI base components to ensure a consistent, familiar design that adheres to Google's Material Design guidelines, providing a cohesive user experience across platforms. These components are pre-built, customisable, and highly performant, offering smooth animations and responsive layouts. Additionally, they simplify creating accessible apps while maintaining native-level performance.

A key focus during app implementation was to ensure it would be aesthetically pleasing to the target audience of women aged 18-34. This was done by implementing simple, refined components so that visual clutter was reduced as much as possible. Decorative features were kept to a minimum so that the clothes and outfits presented were the focal point of the app. Furthermore, a purple colour scheme was utilised throughout. Purple is commonly linked to creativity and imagination, which aligns with the creative purpose of the app. Furthermore, it is frequently used across brands that are marketed towards Gen-Z, who take up a significant portion of the target demographic [24].

The app also implements light and dark mode colour schemes. Users may utilise the app at night to generate an outfit for the following day. As such, offering a dark mode can help reduce eye strain. Dark mode also allows us to cater to different user preferences as many users now expect dark mode as a standard feature, thereby helping to improve user experience and satisfaction.

The final UI is shown in Figures 2–10.

B. Database

The overall structure of the database is shown in Figure 11. There are 4 key tables implemented: clothing items, outfits, outfit items and favourites. The clothing items table stores all relevant information about each clothing item. Additional fields such as image, description, title and date have been added alongside the metadata specified in Appendix B. The outfits table keeps track of outfits added, and its primary purpose is to specify outfit IDs, which can then be used to group items in the outfit items table. Finally, the favourites table stores the favourited outfits' IDs, so that the corresponding outfits and items can be retrieved from the outfit items and clothing items tables.

The database serves as the key source of data throughout the app. This helps to ensure that data is consistent throughout and that any changes are immediately propagated.

SQLite was ultimately chosen as the relational database management system to implement the database. It was selected as it is lightweight, efficient, and reliable for managing local data storage. Furthermore, it has widespread Flutter support

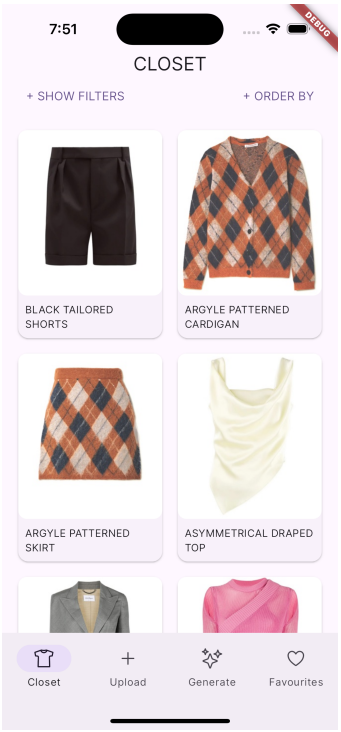

Fig. 2: Closet Page

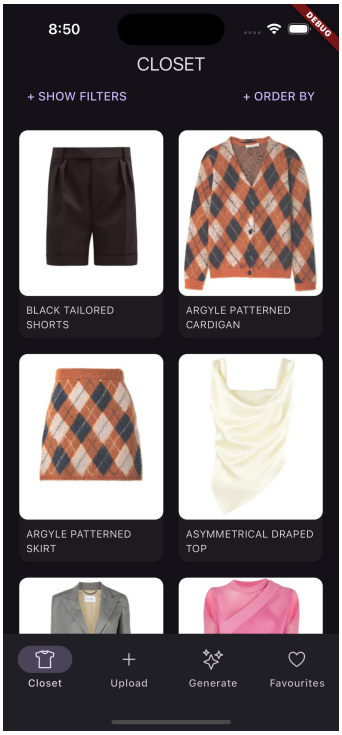

Fig. 3: Dark Mode Closet Page

Fig. 4: Item Page

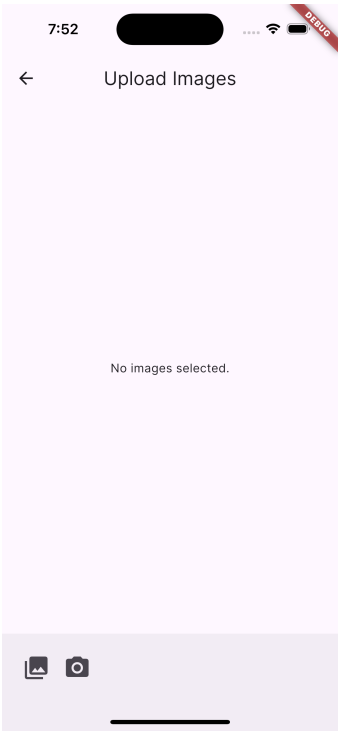

Fig. 5: Upload Page

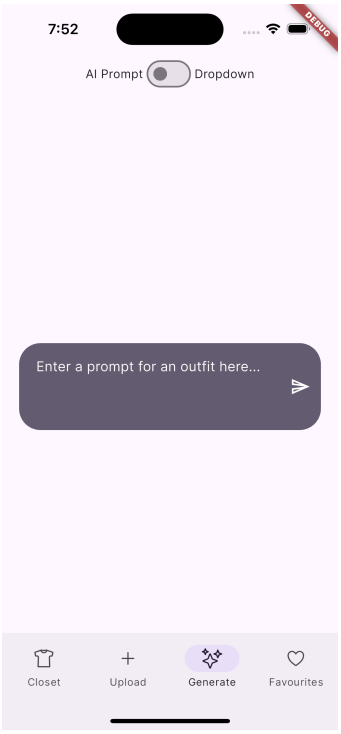

Fig. 6: Generate Page with AI Prompt Interface

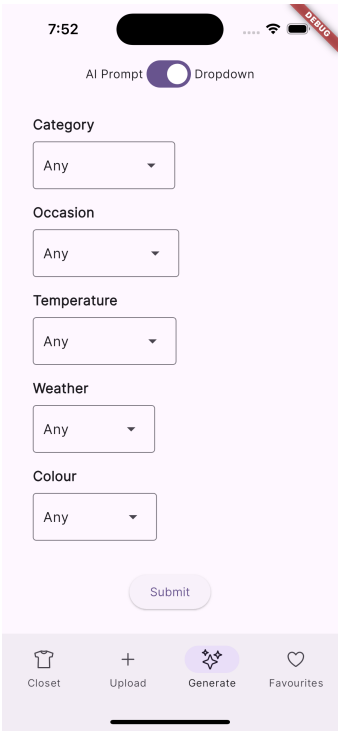

Fig. 7: Generate Page with Dropdown Interface

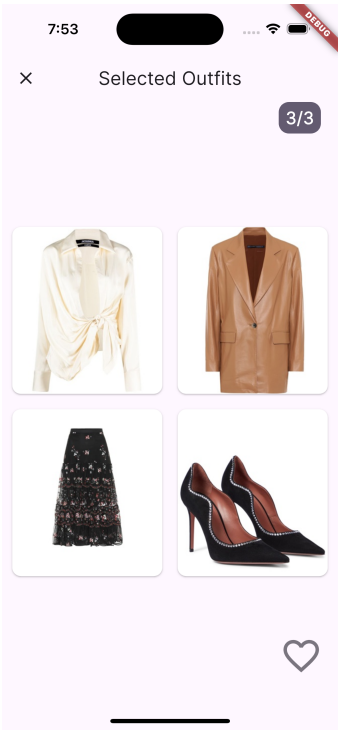

Fig. 8: Generated Outfit Page

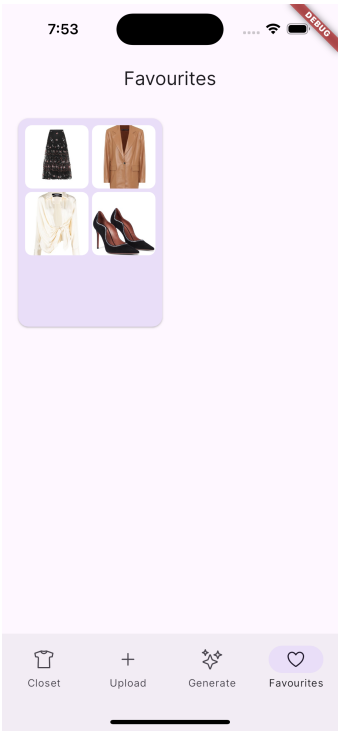

Fig. 9: Favourites Page

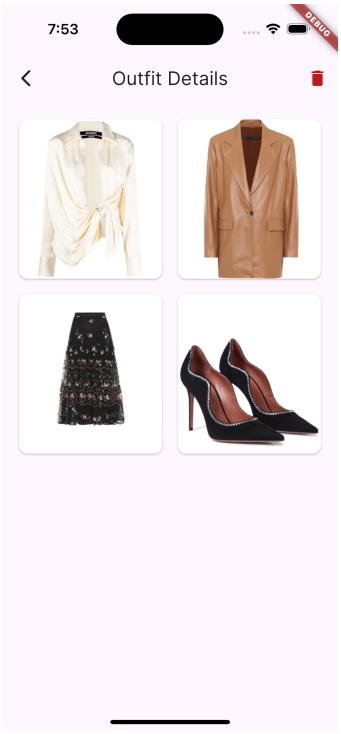

Fig. 10: Favourited Outfit Page

and documentation, with official Google documentation also recommending it for on-device storage.

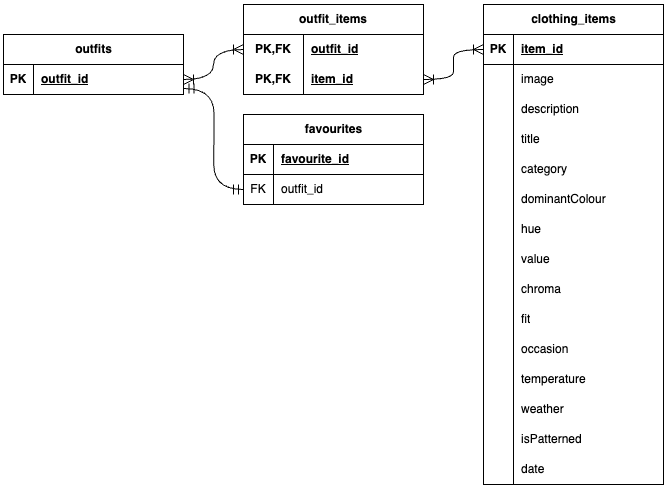

Fig. 11: Entity Relationship Diagram of Database Structure

C. Backend and AI

Two AI APIs were considered to assist in the development of the IME and prompt analysis. These were Gemini's API and OpenAI's API. They were considered due to their prominence, affordability, extensive documentation, and reliability.

However, based on preliminary testing, Gemini's API gave unexpected and erroneous results when asked to extract metadata from a provided clothing image. In some cases, it would extract the metadata correctly and store it in the desired format, whereas in others, it would result in an error stating that it "Couldn't look at images directly" (as shown in Appendix E). OpenAI's API, however, performed consistently and returned responses in the requested format. As such, OpenAI's API was selected.

To further test the effectiveness of the API, preliminary testing was conducted for the metadata extraction to ensure that the API returned prompt and accurate responses. Initially, the GPT-4o model was tasked with generating JSON objects for a series of images, and while the majority of responses were accurate, issues such as incomplete responses and incorrectly formatted objects were observed. Specific trends emerged, such as consistent invalid values for temperature and dominant colour. Prompt crafting became crucial to addressing these issues. However, refining the prompts to constrain the model's creativity within the necessary format was difficult. Lowering the temperature and top-p values helped reduce the variability in responses, making the outputs more predictable without losing the richness of detail. These changes minimized error rates and improved the consistency of metadata extraction. Nevertheless, this process required careful balancing, ensuring the model retained its creative edge while producing responses that fit a structured format.

Following creating the initial prototype, the GPT-4o and GPT-4o-mini models were compared. GPT-4o-mini is a smaller, less complex version of GPT-4o. A key benefit is that it requires fewer computational resources, which directly translates to lower energy consumption. Furthermore, GT-4o-mini costs US $0.150 per 1 million input tokens, whereas GPT-4o costs $2.50. Testing revealed that while both models produced similar types of errors, the GPT-4o-mini model demonstrated higher latency and a lower overall accuracy. The GPT-4o-mini was found to take longer to process requests, likely due to its lower priority for processing and its less sophisticated architecture. Additionally, GPT-4o-mini exhibited a higher incidence of errors related to schema adherence, such as incorrect values for colour value or invalid categories like "swimwear."

Ultimately, GPT-4o was selected because of its low latency, which reduces user wait times and higher accuracy than GPT-4o-mini. The resource-intensive nature of AI models necessitates considering both accuracy and computational efficiency. While GPT-4o-mini is less resource-intensive per query, its higher error rate would mean that responses would need to be regenerated more frequently. This would increase energy use, making its sustainability benefits negligible. As such, GPT-4o's superior performance justifies it as the more viable model for this application. The risks of significant monetary costs were mitigated by using token limits on both the system and user prompts, as well as the returned response. The total amount used across the entire project was US $7.29.

GPT-4o was also selected for extracting key terms and metadata from the user-provided prompt. This was primarily due to its faster response times compared to GPT-4o-mini.

The API is interacted with using HTTP requests to the OpenAI chat completions endpoint. Each request consists of a system prompt and a user prompt. The system prompt primes the system for how it should respond to the user prompt. For example, if it should reply with a JSON object. The user

prompt specifies the exact context or task that the AI should respond to. In the case of image metadata extraction, the image itself can be attached to the user prompt in an encoded form.

The custom outfit generation algorithm was implemented using a greedy best-first search algorithm. A key consideration when implementing this algorithm was limiting calls to it as much as possible to reduce power consumption. The outfit generation model is run each time an item is uploaded to the app. Starting with the new item, the algorithm iteratively finds compatible item matches until an outfit is formed. To limit complexity, at most 15 compatible items can be explored from each partial outfit. The algorithm terminates when there are no more partial outfits to explore. Every 100th generated outfit is then saved to an outfit store. Only storing every 100th outfit is required, as this is a combinatorial problem. Typical testing of the algorithm generated millions of outfits which would require excess storage and would be difficult to efficiently query. As such, only a subset of the outfits are stored. These outfits are then queried based on the user's prompt. Limiting the number of times the algorithm is run reduces device power consumption and reduces latency when responding to a prompt, as the outfits are simply queried rather than generated. Overall, the app can extract metadata about a provided item and generate its corresponding outfits within 10–20s on average.

Determining the rules that the outfit generation algorithm utilised was a key challenge when developing the app. Primarily, fashion is incredibly nebulous and particular rules may not apply to all individuals. Initially, the rules specified that all items in an outfit had to match exactly in terms of weather, temperature, and occasion. However, this strict approach revealed a critical flaw: certain attributes were underrepresented, leaving some items unused entirely due to a lack of matching options. The rules were then iterated upon to allow for similar matches. For example, items with a 'hot' temperature could also be paired with 'warm' items. Additionally, the initial colour-matching rules were too lenient, resulting in visually chaotic outfits with an excessive number of colours. To resolve this, the colour rules were refined to limit outfits to a maximum of four colours, ensuring greater coherence. These challenges underscored the difficulty of balancing algorithmic logic with the nuanced, often subjective nature of fashion.

D. Testing

The app was primarily tested using an iOS simulator, which worked well for most functionality but introduced key challenges. The simulator environment doesn't perfectly mirror real device conditions, particularly with persistent on-device image storage. Initially, images were stored by saving their directory path as a string in the database, but when retrieving them, they couldn't be found because the simulator's documents directory name changes on mount. Although the images still existed, the saved paths were no longer valid. This issue was resolved by saving only the image file name and dynamically loading the correct documents directory during retrieval. However, identifying the root cause of this issue took significant time to debug, highlighting the limitations of simulator-based testing.

Furthermore, the simulator is unable to simulate the device's camera hardware and as such, all of the image upload functionality could only be tested on-device. Given that the app is compatible with iOS and Android this introduced additional complexity. Along with operating system differences, there is significant variability in camera behaviour across devices. As such, validating that the image upload capabilities worked correctly required testing across a range of physical devices.

V. Evaluation

The success of Tailr is measured through both AI performance metrics and user satisfaction. This section presents the AI and user evaluation undertaken and the key findings from these. The evaluations were conducted using 476 clothing items from the A100 dataset [25] and 16945 pre-generated outfits.

A. AI Evaluation

To ensure effectiveness, the AI tools and algorithms used must provide a response within a reasonable time frame. To evaluate this, the outfit selection algorithm was run for 100 iterations for two separate tests. The first ran the algorithm with the same prompt. This allows us to evaluate the standard performance and consistency of the outfit selection algorithm. This testing gave a mean response time of 2.981s with a standard deviation of 0.803s. The second round of testing used 100 different prompts to asses how the algorithm handled a variety of prompts. Users may enter a huge range of different prompts, and the algorithm should still provide efficient responses. In this case, the mean response time was 2.237s with a standard deviation of 1.461s. Both rounds of testing provided responses efficiently, with limited variance between the iterations. This helps validate that users will not become frustrated by excessive response times, which could lead to app abandonment. The response time is significantly attributable to the fact that the app queries a pre-existing outfit store rather than generating new outfits each time. This validates this implementation decision.

B. User Evaluation

A user evaluation study was conducted to assess the app's usability, intuitiveness, and general satisfaction. The key goals of the study were to validate the need for the app and evaluate its usability and effectiveness. Furthermore, A/B testing was conducted on different methods of generating an outfit: a freeform AI prompt and a set of attribute dropdowns. This testing aimed to assess participants' preferences and determine if either generated better outfits. The study was approved by the Victoria University of Wellington Human Ethics Committee with approval HE000028. The study consisted of 3 parts: a pre-study survey, 30 minutes of app testing, and a post-study survey.

1) Participants: To align with our target demographic of women aged 18-34, the study participants were 10 women aged 18-31 (μ=23, σ=4.03). Participants were recruited using posters across the Kelburn, Pipitea, and Te Aro campuses. As koha for participating, participants were provided with a $20 supermarket voucher.

2) Testing Conditions: Testing was in CO255 using an iPhone 14 Pro running iOS 17.6. Audio and screen recordings were taken during the testing to keep a record of the actions performed and any comments made. Manual notes were also taken. Participants were encouraged to narrate their thought process and what they expected the outcome of an action to be before performing it as part of a 'speak aloud' protocol. For testing, the app was preloaded 476 clothing items and participants were provided with clothes to upload to the app. This mitigates the ethical issue of the app making any judgment on the participants themselves.

3) Process: During each testing session, participants were first presented with a pre-study survey to complete. This focused on assessing each participant's level of experience with mobile apps – in particular fashion apps – and AI tools. Furthermore, participants were also asked questions regarding how long it takes them to get ready and their experience with selecting outfits. This was to validate our hypothesis that the current process of getting ready is time-consuming and frustrating. A full list of the questions asked can be found in Appendix G-A.

Following the pre-study survey, participants were asked to conduct user testing of the app. For the first 15 minutes, participants were asked to complete a list of tasks. One of the task sets involved generating an outfit. The presentation of the two conditions was fully counterbalanced across the ten participants to avoid carryover bias in the A/B testing. A full task list can be found in Appendix G-B. Following this, participants were asked to freely experiment with the app for a further 15 minutes

After the testing, participants were asked to complete a post-study survey. This survey aimed to understand how usable participants found the app, their experience with the different interfaces presented for generating an outfit, and their general sentiments about the app. The first section of the survey asked participants to rate how much they agreed with a set of statements using a 5-point Likert scale.

The statements rated by participants are as follows:

1) The user interface was intuitive
2) The application was easy to use
3) I felt confident using the application
4) I needed to learn a lot of things before I was able to use the application properly
5) I found it difficult to navigate the application
6) The app provided me with reasonable outfit recommendations
7) I preferred the AI prompt over the dropdown menu for specifying an outfit selection
8) The AI prompt gave me more flexibility than the dropdown menu for specifying an outfit selection
9) I was more satisfied with the outfits provided from the dropdown than those provided by the AI prompt
10) If I don't have any outfits that satisfy my provided prompt, I would prefer to be shown a random selection of outfits rather than nothing
11) I would feel upset if the app could not suggest any outfits with my own clothing items

Participants were also prompted to respond to the following freeform questions.

12) What did you like most about the app?
13) What would you like to see added to the app?
14) If the dropdown menu was the primary interface for generating an outfit, what other options would you like to see added to choose from?
15) Would you see yourself using a fully developed version of the app?
16) To what degree should your personal style be integrated into the app's recommendations?
17) How comfortable would you feel with the AI making recommendations based on your personal wardrobe?
18) Would you have any concerns about using this app?
19) Please add any further comments or feedback below

4) Study Results: The pre-study survey elucidated several key findings regarding the need for the app. Notably, 60% of the participants reported owning over 80 items of clothing, with 40% of the participants owning over 100. This indicates that our target audience owns a significant number of clothing items, potentially making them difficult to manage. This is reinforced by the result that 20% of the participants reported that their collection was 'Very Large', which corresponded to the description that their collection was 'Extensive, with many items and hard to manage'. Furthermore, 70% of the participants reported spending more than 5 minutes selecting an outfit each day. One participant reported spending between 20-30 minutes each day. Additionally, 20% of the participants responded that they frequently have trouble picking an outfit, and it takes them a while to decide. Evidently, these results reinforce our hypothesis that the current process of selecting an outfit is time-consuming. Furthermore, 60% of the participants reported putting moderate to considerable thought into selecting an outfit. This exemplifies how this selection process could be contributing to decision fatigue.

Anecdotal feedback from participants emphasised the stress and mental effort put into selecting an outfit. For example, P2 stated, "I'm such an overthinker, I think like the day ahead [about what I'm going to wear]. Like I was sitting in bed last night thinking about it." Furthermore, P2 and P3 noted the stress that they feel when comparing their outfits to others. P3 said, "You go through your teen angst phase of 'oh my god, I have nothing to wear.' Meanwhile, your closet is jam-packed. But sometimes I'm just like, especially coming to campus, I feel really underdressed compared to other people. Yeah, I do find that [picking an outfit] can get frustrating at times."

Furthermore, the post-study survey gave key insights into how participants perceived the usability and utility of the app. Overall, all participants were able to complete all tasks on the task list (see Appendix G-B) within the 15-minute time limit. The overall responses to the Likert scale questions are shown in Figure 12.

Participants strongly found the interface to be intuitive, easy to use and easy to navigate. P7, P9 and P10 specifically remarked on the app's ease of use and intuitiveness. P7 stated, "This is just how I would shop. I think that's wonderful cause I have a lot of stuff that I just don't wear cause it's just out of sight. I have used other apps that have a similar capacity in

the past, but they've never been so easy to use 'cause they get caught in the minutia. But this is very, very easy to use." In particular, P3, P5, P6, P7, P8 and P10 all remarked that they liked the loading messages when uploading an item to the app. This indicates a successful user experience that meets the commercial need of engaging the user early on.

Furthermore, several participants noted the convenience of the app. Both P3 and P4 noted that they would use the app in bed before getting up in the morning. Furthermore, P1, P3, P4, P5 and P7 remarked that the app would help them to easily visualise and remember what clothes they owned. P3 stated, "For me personally, I'm like a visual learner, so instead of getting my clothes out of the wardrobe, I could just do it on my phone while I'm still in bed." Furthermore, P1 and P4 remarked that being able to browse and select outfits from their phone would help them to keep their room tidier. P4 stated, "For me if things are put away in drawers or closets, I just forget them. So, this would be a really easy visual reminder. Rather than pulling everything out of drawers and closets, laying them on the bed, and seeing if they suit. So I'd love being able to do it all from my phone to keep my room tidier."

Furthermore, the majority of participants remarked that the app provided aesthetically pleasing and prompt-appropriate outfit recommendations. P2, P5, P6, P8 and P10, in particular, commented on the wearability and the appropriateness of the outfits. P2 stated, "[The app] is good at producing outfits that people would wear." Upon being shown an outfit selection, P5 remarked, "Ooooh, these [outfits] are so cute!" Additionally, P2, P3 and P9 remarked that they liked the variety in the outfits shown, with P3 saying, "For me personally, I don't have a lot of clothes, so like finding a variety of outfits is like hard, so this is really handy." P2 and P3 also appreciated that they were shown 3 outfits. However, in contrast, P8 stated, "It makes me trust the results less when I know that I always get 3 [outfits]." This highlights the challenges with designing the system, as while some people appreciate the selection, the selection may imply to some users that the AI isn't necessarily making 'the best' selection for them.

However, there were some instances in which the app recommended items that did not suit the given prompt. For example, outerwear was paired with an outfit for the beach, or shoes were provided for an outfit that was intended to be indoors. This indicates that the outfit generation algorithm could use further refinement, especially when matching items based on temperature and weather. In some cases, the outfit returned may have been selected as there was no outfit available that met the specified criteria. In these cases, these outfits should be prefaced with a message that no exact matches were found. Responses were mixed as to whether participants would prefer to be shown a random outfit selection rather than nothing.

While most participants indicated that they preferred the AI prompt interface and found it more flexible, the results also indicate that participants preferred the outfits provided by the dropdown interface. However, P1 and P3 also specifically remarked that they would use both interfaces but for different purposes. P3 stated, "I like both [the AI prompt and the dropdown]. I think the dropdown is really handy cause it breaks down your day into like categories. But the prompt is like, [more creative and freeform]. [...] I really like both of them." The AI prompt interface can capture more abstract prompts, such as specific events, moods, or aesthetics. However, the dropdown appeared to be more efficient to use for simple prompts such as colours, weather or temperature. P2 noted a clear preference for the dropdown, saying, "If I was trying to use this to find an outfit, I would immediately go to the dropdown." As such, this indicates that both interfaces should be available within the app rather than selecting one or the other. However, the AI prompt should remain the primary interface.

Participants also identified several key features that they'd like to see in a fully developed app. A key theme throughout the feedback was introducing the ability to tailor the recommendations to the users' preferences. For example, P2 and P8 mentioned that they'd like for the AI to learn from the outfits that they favourite. Furthermore, P2 stated, "I wish there was a way to be like 'no' to get the AI to be better" in reference to the outfits displayed on the AI prompt. P1, P2 and P5 mentioned that they'd like to be able to set their preferences regarding aesthetic or outfit rules to help inform the outfits generated. For example, P2 mentioned that they would wear dresses or skirts with pants – a combination that is not supported. Another consistent theme that came up was being able to specify the number of colours that participants would like in an outfit. P4 stated that while they would normally have 3-4 colours in an outfit, they know friends who would ordinarily wear at least 10.

Furthermore, participants stated that they'd like the ability to edit the items' metadata and add more than one metadata tag to an item. While participants appreciated that the AI could automatically extract the metadata, some mentioned that they'd like to be able to edit the information to fit their own perceptions. For example, P2 stated regarding the generated metadata, "[The AI] has said that this is yellow, but I'd call it bright green." Colour perception is particularly user-specific, so adding an ability to edit items would help ensure that their closet is tailored to their preferences.

Additionally, P1, P2 and P6 mentioned that they'd like to be able to organise and categorise their favourites into folders so that they could easily find outfits, especially as they added more favourites.

VI. Conclusions and Future Work

This project successfully achieves its aim of creating a proof-of-concept prototype app that integrates AI to accelerate and optimise the outfit selection process. The app successfully implements the key functionality requirements that allow the user to upload items, view their items, generate outfits, and create favourites. The user study validated and reinforced the need for and desire for this product. Furthermore, this evaluation demonstrated that the app met the usability performance requirements and met participants' expectations for the utility and performance of the app.

However, A key area of future work is improved outfit customisation, particularly tailoring recommendations to suit users' specific preferences. Wearing outfits that align with an

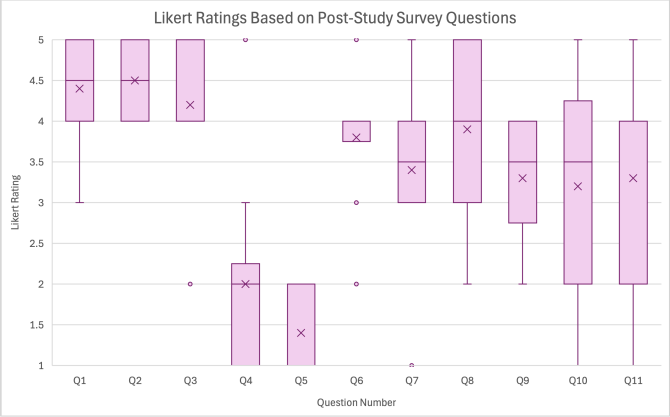

Fig. 12: Box and whisker distributions of Ratings given in response to post-study survey

individual's personal style is key for harnessing their mood-enhancing benefits. Furthermore, participants indicated in Q16 of the post-study survey that personal style should factor heavily into the app's recommendations and would be a major factor in why they would use the app. Improving the app's outfit-generation algorithm by integrating a more sophisticated AI would be key for a subsequent version of the product. The AI could learn users' preferences based on the outfits they favourite to inform their recommendation. Furthermore, additional feedback interfaces could be introduced so that users can also provide negative feedback if they dislike a generated outfit. Users could set personal style preferences, aesthetic goals, or even daily moods to serve as indicators that could inform the outfits generated.

Furthermore, automatic integrations could further help reduce the user's mental load. For example, calendar and weather integrations could help to inform the prompts provided based on the current weather, temperature and planned events for the day.

ACKNOWLEDGMENTS

My sincere thanks to Dr Stuart Marshall and Dr Andrew Lensen for supervising this project. Their support, guidance and quick wit were crucial throughout. Thank you to the user study participants for their valuable time and opinions. Additionally, thank you to the ENGR489 cohort for their support, feedback and caffeine supply.

REFERENCES

[1] M. L. Slepian, S. N. Ferber, J. M. Gold, and A. M. Rutchick, "The cognitive consequences of formal clothing," *Social Psychological and Personality Science*, vol. 6, no. 6, pp. 661–668, 2015. [Online]. Available: https://doi.org/10.1177/1948550615579462

[2] R. Smith and J. Yates, "Flourishing fashion: An interpretive phenomenological analysis of the experience of wearing a happy outfit," *Fashion Studies*, vol. 1, no. 1, pp. 1–39, 2018.

[3] H. Adam and A. D. Galinsky, "Enclothed cognition," *Journal of Experimental Social Psychology*, vol. 48, no. 4, pp. 918–925, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022103112000200

[4] D. Kodzoman, "The psychology of clothing: Meaning of colors, body image and gender expression in fashion," *Textile & Leather Review*, vol. 2, no. 2, p. 90–103, Jun 2019.

[5] E. Euse, "Men will spend four months of their lives deciding what to wear," Complex. [Online]. Available: https://www.complex.com/style/a/erica-euse/men-spend-four-months-of-lives-deciding-what-to-wear

[6] Stitch Fix, "2024 style forecast," Dec 2023. [Online]. Available: https://newsroom.stitchfix.com/2024-style-forecast.pdf

[7] S. Berg, "What doctors wish patients knew about decision fatigue," American Medical Association, Nov 2021. [Online]. Available: https://www.ama-assn.org/delivering-care/public-health/what-doctors-wish-patients-knew-about-decision-fatigue

[8] K. Vohs, R. Baumeister, B. Schmeichel, J. Twenge, N. Nelson, and D. Tice, "Making choices impairs subsequent self-control: A limited-resource account of decision making, self-regulation, and active initiative," *Journal of personality and social psychology*, vol. 94, pp. 883–98, 05 2008.

[9] Z. Gervis, "You're not the only one who constantly feels "wardrobe panic"," Mar 2018. [Online]. Available: https://nypost.com/2018/03/07/youre-not-the-only-one-who-constantly-feels-wardrobe-panic/

[10] European Parliament, "The impact of textile production and waste on the environment (infographics)," 2020, accessed: 2024-10-03. [Online]. Available: https://www.europarl.europa.eu/topics/en/article/20201208STO93327/the-impact-of-textile-production-and-waste-on-the-environment-infographics

[11] L. Coscieme, L. Akenji, E. Latva-Hakuni, K. Vladimirova, K. Niinimäki, C. E. Henninger, C. Joyner-Martinez, K. Nielsen, S. Iran, and E. D´Itria, *Unfit, Unfair, Unfashionable: Resizing Fashion for a Fair Consumption Space.*, 11 2022.

[12] United Nations, "The 17 goals," United Nations, 2015. [Online]. Available: https://sdgs.un.org/goals

[13] L. Ceci, "Age group distribution of mobile app users worldwide in the Google Play Store in 2nd quarter 2022, by category," Aug 2023. [Online]. Available: https://www.statista.com/statistics/1333429/google-play-store-apps-age-distribution-by-category/

[14] R. Inostroza, C. Rusu, S. Roncagliolo, V. Rusu, and C. A. Collazos, "Developing smash: A set of smartphone's usability heuristics," *Computer Standards & Interfaces*, vol. 43, pp. 40–52, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548915000926

[15] Amazon Web Services, "What are foundation models?" 2024. [Online]. Available: https://aws.amazon.com/what-is/foundation-models/

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. [Online]. Available: http://aima.cs.berkeley.edu/

[17] G. Prato, F. Sallemi, P. Cremonesi, M. Scriminaci, S. Gudmundsson, and S. Palumbo, "Outfit completion and clothes recommendation," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–7. [Online]. Available: https://doi.org/10.1145/3334480.3383076

[18] R. Sarkar, N. Bodla, M. I. Vasileva, Y.-L. Lin, A. Beniwal, A. Lu, and G. Medioni, "OutfitTransformer: Learning outfit representations for fashion recommendation," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 3590–3598.

[19] Y. Lin, M. Moosaei, and H. Yang, "OutfitNet: Fashion outfit recommendation with attention-based multiple instance learning," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 77–87. [Online]. Available: https://doi.org/10.1145/3366423.3380096

[20] Y. Xu, W. Wang, F. Feng, Y. Ma, J. Zhang, and X. He, "Diffusion models for generative outfit recommendation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1350–1359. [Online]. Available: https://doi.org/10.1145/3626772.3657719

[21] ASOS, "ASOS," 2024. [Online]. Available: https://www.asos.com/women/

[22] H. Ko, "Acloset," 2024. [Online]. Available: https://acloset.app/

[23] Apple Inc., "Designing for ios - human interface guidelines," 2024. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/designing-for-ios

[24] A. Pockros, "How gen-z purple became the new millennial pink," https://www.wix.com/studio/blog/lilac-purple, 2022, accessed: 2024-10-03.

[25] A. Chow, "AiDLab fAshIon Data: A100 Dataset," 2024. [Online]. Available: https://github.com/AemikaChow/AiDLab-fAshIon-Data/blob/main/Datasets/A100.md