Summary for FLAME Technical Challenge

Team Name:
OutfitLab

Team Members:

Ippatapu Venkata Srichandra, Student
Pathange Omkareshwar Rao, Student
Harish Vijay V, Student
Amrit Subramanian, Student
Dr Premjith B, Mentor/professor

Project Vision:
Our project aims to enhance personalized wardrobe and accessory recommendations by utilizing large language models (LLMs) and advanced computational intelligence. The core objective is to provide users with highly customized attire and accessory suggestions based on images of their wardrobe, their personal preferences for specific occasions, and context-based insights such as event types or weather conditions. By analyzing visual and textual data, the system will generate outfit combinations that are not only visually appealing but also contextually relevant, ensuring that users are well-dressed for any situation.

Problem Statement:

Modern users increasingly seek personalized fashion recommendations but often struggle to coordinate their outfits and accessories for specific events or occasions. Existing solutions frequently fall short in effectively integrating deep learning-based visual and textual embeddings, which are essential for accurately capturing user preferences. Our project aims to bridge this gap by combining image-based embeddings, classification techniques, and graph databases to provide seamless, tailored recommendations. By utilizing powerful AI models, we not only suggest outfits but also accessories like belts, watches, and jewelry that complement the overall look. This approach helps users get fashion suggestions that are both suitable for the occasion and visually well-matched, ensuring their outfits and accessories work together perfectly.

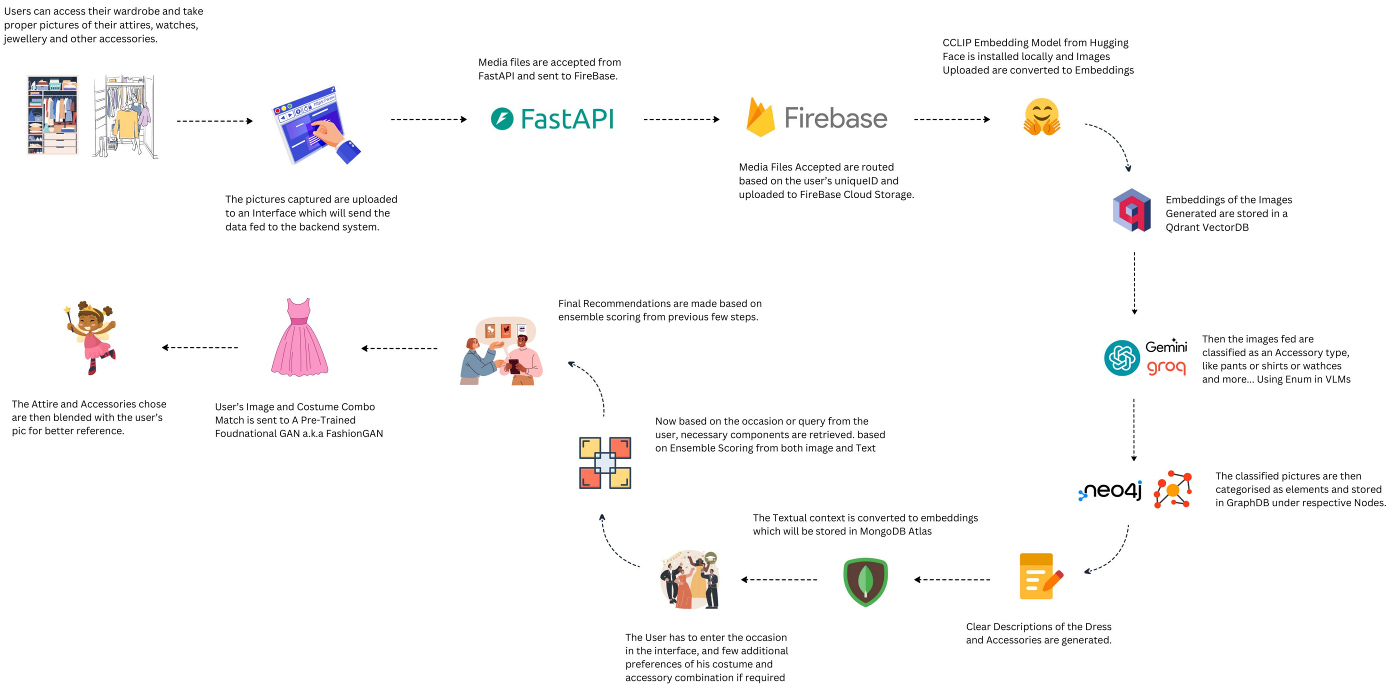
Progress Overview:

We have developed a system that allows users to capture images of their wardrobe and body to process them through a backend pipeline that generates embeddings. These embeddings represent key features of clothing items, such as color, texture, and style, enabling personalized outfit recommendations. Images are uploaded via an interface using **FastAPI**, which sends the data to **Firebase** for media storage. The system then converts these images into embeddings using the **CLIP embedding model** from Hugging Face. These embeddings are stored in **Qdrant VectorDB** for efficient retrieval during outfit recommendations. Additionally, we use **Gemini groq** for image classification, identifying whether the item is a shirt, pants, watch, belt, or other accessory, and **Neo4j** for managing categorized images in a knowledge graph. This knowledge graph helps organize items like clothing and accessories under respective nodes for better, more structured recommendations.

Our current focus is on optimizing the backend, particularly by refining efficient storage, the knowledge graph, and pipelines to ensure smooth data flow and fast retrieval. The knowledge graph, stored in **Neo4j**, plays a crucial role in linking related clothing items and accessories, making the outfit selection process more intuitive and accurate. We're also working to improve accessory recommendations, including watches, belts, and jewelry, by enhancing the embedding and classification processes for these items. These ongoing efforts will further enrich the user experience by providing more detailed and comprehensive outfit suggestions.

Appendix for FLAME Technical Challenge

Methodology:



Users can access their wardrobe and take proper pictures of their attires, watches, jewellery and other accessories.

The pictures captured are uploaded to an Interface which will send the data fed to the backend system.

Media files are accepted from FastAPI and sent to FireBase.

Media Files Accepted are routed based on the user's uniqueID and uploaded to FireBase Cloud Storage.

CCLIP Embedding Model from Hugging Face is installed locally and Images Uploaded are converted to Embeddings

Embeddings of the Images Generated are stored in a Qdrant VectorDB

Then the images fed are classified as an Accessory type, like pants or shirts or wathces and more... Using Enum in VLMs

The classified pictures are then categorised as elements and stored in GraphDB under respective Nodes.

Clear Descriptions of the Dress and Accessories are generated.

The Textual context is converted to embeddings which will be stored in MongoDB Atlas

The User has to enter the occasion in the interface, and few additional preferences of his costume and accessory combination if required

Now based on the occasion or query from the user, necessary components are retrieved. based on Ensemble Scoring from both image and Text

Final Recommendations are made based on ensemble scoring from previous few steps.

User's Image and Costume Combo Match is sent to A Pre-Trained Foudnational GAN a.k.a FashionGAN

The Attire and Accessories chose are then blended with the user's pic for better reference.

Fig-1 Proposed Methodology

The proposed methodology works by accepting wardrobe images from users, converting them to embeddings, and storing them for future recommendations.

The workflow involves the following stages:

Data Capture:
Users can upload images of their clothes, accessories, watches, and their body using a website interface. These images are then sent to the backend system via FastAPI, allowing users to easily submit both their wardrobe items and body images for processing, ensuring personalized outfit recommendations.

Media File Storage:
The images are then routed to Firebase Cloud Storage based on user IDs. This organized storage system allows for efficient media management. It ensures that each user's wardrobe images are securely stored and easily accessible.

Embedding Generation:
The CLIP model from Hugging Face converts the uploaded images into embeddings. These embeddings are stored in Qdrant VectorDB, enabling fast retrieval for future matching. This process transforms visual data into a format suitable for recommendation algorithms.

Classification:
The images are categorized into groups such as shirts, pants, watches, and accessories using an Enum model

running on Groq VLMs. This classification provides a structured organization of the wardrobe. It allows users to navigate their items more easily and facilitates better recommendations.

Graph Database Integration:
The classified images are stored in Neo4j GraphDB, which enables the mapping of complex relationships between items. This system helps identify complementary clothing and accessory matches. It enhances the recommendation process by understanding how different items interact.

Textual Data Processing:
Users can input queries or describe occasions using text, which are then converted into embeddings using NLP models. These embeddings are stored in MongoDB Atlas for efficient data retrieval. This allows the system to understand user intent and context for personalized recommendations.

Ensemble Recommendation System:
Based on the user's event or query, an ensemble of image and text embeddings is used to score and recommend outfits. The recommendations are further refined using FashionGAN, which checks compatibility between selected items and user preferences. This approach ensures that the suggested outfits align with the user's style and needs.

Components:

1. **Frontend:** A user interface that allows users to upload wardrobe images, the image of their body and select their preferences.
2. **FastAPI:** A backend API service designed to handle media uploads efficiently.
3. **Firebase**: A storage solution for media files based on user IDs, ensuring secure and organized storage.
4. **Qdrant VectorDB:** A database for storing image embeddings, enabling quick similarity searches.
5. **Groq VLMs:** A model used for the classification of wardrobe items.
6. **Neo4j GraphDB:** A database that organizes items and their relationships, enhancing the recommendation process.
7. **MongoDB Atlas:** A storage solution for textual descriptions and embeddings, facilitating user queries.
8. **FashionGAN:** A pre-trained Generative Adversarial Network used to validate the compatibility of outfits and accessories.

Initial Results:

Image Upload & Storage:
Successfully implemented an interface where users can upload images of their wardrobe and body.
Images are uploaded via FastAPI and stored in Firebase for easy media management.

Embedding Generation:
Images are processed using the CLIP embedding model from Hugging Face to generate embeddings that capture key features like color, texture, and style.

Efficient Embedding Storage:
Embeddings are stored in Qdrant VectorDB, enabling fast and efficient retrieval during the recommendation process.

Image Classification:
Integrated Gemini groq to classify items such as shirts, pants, watches, belts, and other accessories.

Knowledge Graph Management:
Created a knowledge graph in Neo4j, categorizing wardrobe items and accessories into respective nodes for structured, intuitive recommendations.

To-Do:

·Backend Efficiency Optimization:

Refined storage systems to improve performance and scalability. Enhanced the data pipelines for smoother flow and faster retrieval.

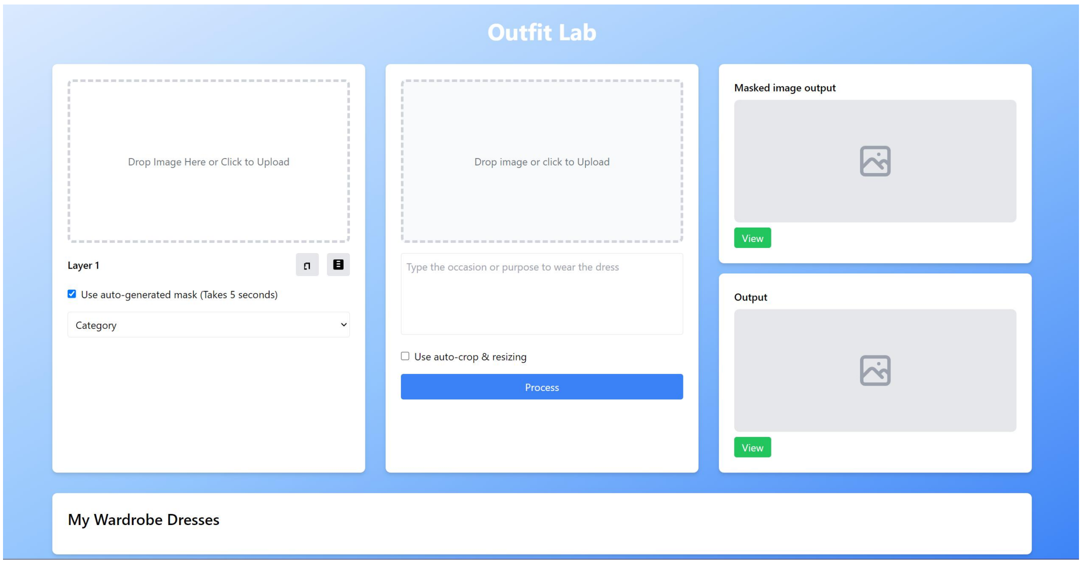
Knowledge Graph Improvement:

Continued building and refining the Neo4j knowledge graph to link related clothing items and accessories, ensuring more intuitive and accurate outfit selection.

Accessory Recommendations Enhancement:

Focused on improving recommendations for watches, belts, jewelry, and other accessories. Fine-tuned the embedding and classification processes for better integration of accessories into outfit suggestions.

Results:

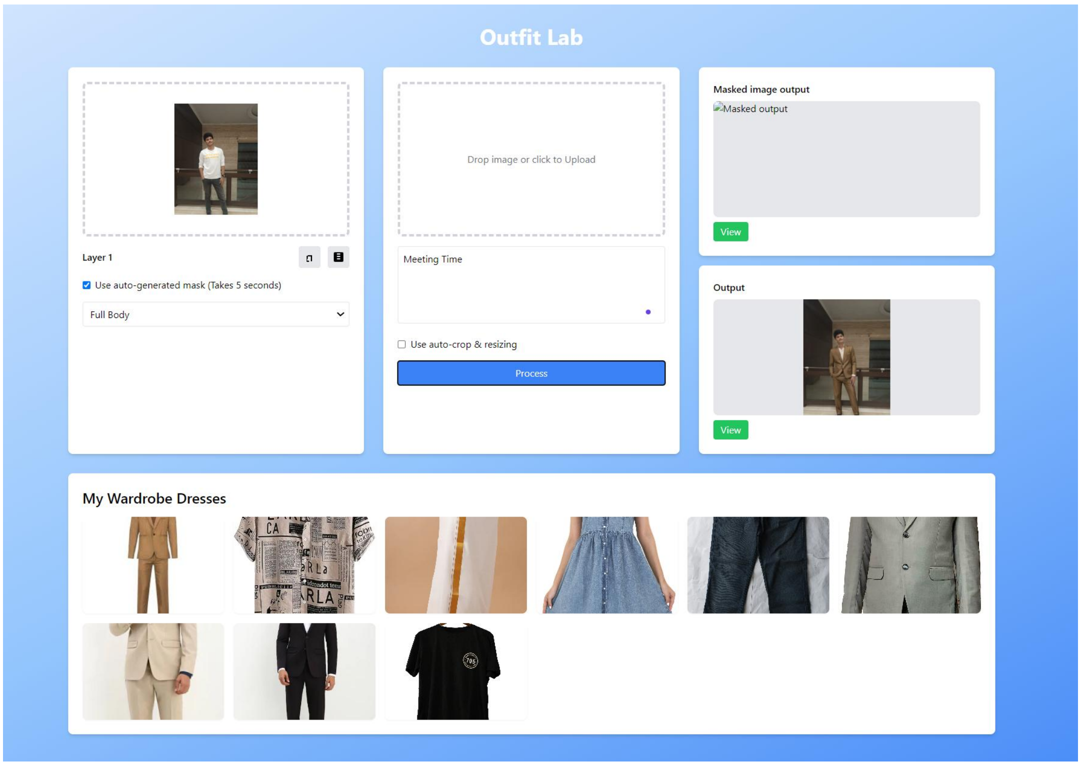


Outfit Lab
Drop Image Here or Click to Upload
Layer 1
Drop image or click to Upload
Type the occasion or purpose to wear the dress
Process
Masked image output
View
Output
View
My Wardrobe Dresses

☑ Use auto-generated mask (Takes 5 seconds)

Category

□ Use auto-crop & resizing



☑ Use auto-generated mask (Takes 5 seconds)

Full Body

Meeting Time

□ Use auto-crop & resizing