Project Overview

Project Title: ClosetAI - A Personal AI Stylist and Shopping Assistant

Project Overview: The purpose of this project is to create an AI-powered stylist that will generate an appropriate outfit, given the user's prompts, the user's current closet, and the current weather. If the user lacks certain items under certain constraints (occasions, weather condition, or no item in the closet database), the AI will recommend new items to be purchased, to add to the current closet. The AI will have the knowledge of current trends and fashion trends in the past to help align the user's fashion vision. The end goal is to help the user simplify the daily outfit decision time and effort, always dressing with fashion trends in mind, and simplify the clothing shopping process. However, the shopping feature is a stretch goal. This semester will be mainly focused on the outfit recommendation process to ensure feasibility.

Scope:

- Closet Database (local): The user will manually input clothing items, with the color, type, and season. The AI will categorize them into structured sets (tops, bottoms…).
- Weather Data: Real-time weather will be updated and evaluated with WeatherAPI.
- User Prompts: NLP will process users' requests for the desired vision and shopping constraints.

The system will also have knowledge of different trends being implemented. The output would be the outfit or shopping recommendations. The project will be Python-based, using NLP to process users' prompts, Pandas to handle the database, Scikit-learn to order the closet, and WeatherAPI to handle real-time weather condition updates. There will be an API that handles the online shopping portion as well. Shiny Express will handle the UI portion of the system.

Stakeholder:

- Users: The ones who are seeking an AI stylist outfit recommendation. The users will update their closets, asking the system on what to wear and what items to shop for.
- Vendors/E-commerce platforms: Vendors and brands can help integrate their websites or recommendations for the primary user on the application.
- Data Scientist/ AI Engineers: Develop the logic for the system (appropriate outfits while working with the stylist/doing research, trends, available clothing items, online shopping handling, and image processing)
- Software Developers: Creating a User Interface that's appropriate.
- Fashion designers or stylists: Helping validate outfit recommendations.

Computer Infrastructure

1. **Project Needs Assessment**

- **Objective & Task:** Outfit selection and (classification, ranking, and NLP) & shopping recommendation.
- **Data Types:** User-entered closet (text), image of clothes (future implementation), weather data (API), user prompts.
- **Performance Benchmarks:**

    - Latency: Less than 1 second for outfit general.

    - **Accuracy:** Must take the weather into consideration, more than 80% aligned with the user satisfaction (through the surveys).
- **Deployment Constraints:** Runs on cloud through a mobile application interface.

    - **Decision**: Prioritize speed and intuitiveness over model complexity.

    - **Options considered:** Complex image-based outfit generation vs. simple rule/query system.

    - **Tradeoff:** Intuitiveness vs. quick deployment

    - **Risk/trigger:** If the generation time passes 3 seconds, reduce model size.

2. **Hardware Requirements Planning**

- **Training Hardware:** Hipergator's NVIDIA A100
- **Inference Hardware:** CPU-based inference for simple prompts, GPU-based for more complex tasks.
- **Minimum Specifications:** 16-40 GB of GPU memory, 32 GB of RAM, 200 GB of storage.

- **Decision:** Start with cloud GPU (A100) for training, interfere on CPU for a less complex prototype demo.

- **Options considered:** Physical vs cloud GPU.

- **Tradeoff statement:** Cloud offers room for future implementations and scalability, but comes at a higher cost.

- **Risk/trigger:** If experiencing a GPU lack of quota, switch to a lower tier GPU (V100) or optimize the models.

3. **Software Environment Planning**

- **Operating System:** Windows 11
- **Frameworks/Libraries:** Python 3.11, Pytorch (Machine Learning),, Pandas (Database Handling), scikit-learn for output, spaCy, NLTK., Streamlit for the UI. WeatherAPI for the weather data.
- **Visualization/Containers:** Docker Desktop for Windows

- **Decision:** Windows was chosen for the development because of familiarity and convenience. Docker ensures consistency.

- **Option considered:** Windows was chosen over Linux. Pytorch was chosen for its flexibility, and it is easy to implement.

- **Tradeoff statement:** Though it will be more familiar to develop the system on Windows, it might introduce a lot of compatibility issues and limit scalability.

- **Risk/trigger:** If using Windows introduces compatibility issues, the project development might have to migrate to Ubuntu

4. **Cloud Resources Planning**

- **Provider:** AWS

- **Storage & Scaling:** Data will be stored in S3.
- **Cost Estimation:** Roughly $100 a month to use the AWS services

    - Decision: AWS was chosen as the main cloud service

    - Options considered: Azure, GCP, Oracle

    - Tradeoff statement: It has to be exclusively AWS instead of a wider multi-cloud compatibility

    - Risk/trigger: If the cost becomes too much, move to Azure or GCP.

5. **Scalability, and Performance Planning**

- **Scaling Strategy:** Automatic scaling for the EC2 instances.
- **Optimization Techniques:** Reducing model complexity for faster inference.
- **Performance Monitoring:** Latency, user satisfaction

    - Decision: Automatic scaling in AWS.

    - Options considered: Accuracy vs. optimization

    - Tradeoff statement: Reducing the model's complexity may reduce its accuracy.

    - Risk/trigger: If latency is greater than 2 seconds, scaling will be triggered.

Security, Privacy, and Ethics (Trustworthiness)

1. **Problem Definition:**

- **Goal:** The system will define how the user will present themselves in society and a professional environment without a pre-determined notion, bias, or stereotypes. The system will support users to dress appropriately in certain situations, introducing sustainable fashion practices (Utilize what the user already has, eliminating unnecessary shopping).
- **Strategies:** Meetings between stakeholders to identify fashion bias (race, gender assumptions) utilizing AI Blindspot, introducing constraints for the model by introducing a stereotypical dataset that should be avoided.

2. **Data Collection:**

- **Goal:** Gather high-quality, representative, and unbiased data to support reliable AI model development.
- **Strategies:** Implement differential privacy for the user's closet using IBM Diffprivlib, and implement statistical tests to detect imbalances in the dataset (overrepresentation in fashion). This particular issue can be fixed with data augmentation.

3. **AI Model Development:**

- **Goal:** Develop fair, interpretable, and robust AI models that perform reliably across diverse scenarios.
- **Strategies:** Use Fairlearn to assess and mitigate the model's bias in outfit generation across all types of people.

4. **Deployment:**

- **Goal:** Deploy the AI model securely and ensure its performance and reliability in real-world conditions.
- **Strategies:** Utilize BentoML to guarantee secure model serving and in-depth model monitoring and feedback.

5. **Monitoring and Maintenance:**

- **Strategies:** Use NannyML to detect data and concept drift in ML models (changing trends).

Human-Computer Interaction (HCI)

1. **Understand User Requirements:**

- The user requirements can be gathered by providing the users with a survey on how they pick what to wear and the thought process behind it. The surveys will be created with Google Forms.
- The user requirements that needed to be gathered is their definition of an intuitive user interface, so the model can have a mass appeal across all demographics. This process can be done through interviews.

2. **Create Personas and Scenarios:** Use role-playing exercises to create user scenarios:
    - Lisa is a college student and wants a quick outfit for a lecture on a rainy day.
    - It's 80s night at the college bar that I'm going to, what should I wear?
    - Peter is trying to choose his outfit to go golfing, it's pretty windy and cold today.
    - How can Jason incorporate his baggy jeans into his outfit to go out?
3. **Identifying Accessibility Requirements:**

- Ensure the system complies with the WCAG standard to guarantee usability. Implement a screen reader for users with blindness.

4. **Outline Usability Goals:**

- Generate an outfit recommendation within 5 seconds.
- Reduce decision effort and time.
- User-friendly interface.
- Easy closet database update process.

5. **Github Repository:**

https://github.com/itsmekhang/ClosetAI-Development

Risk Management Strategy

1. Problem Definition

The goals and the scope that are not well-defined will lead to the model being misaligned. This section guarantees the goals and scope are obtainable, ethical, and align with stakeholders' vision.

- **Key Risks:**
  - System goals misaligning with stakeholder expectation.
  - Potential negative impact for society.
  - Potentially introducing bias.
  - Undefined success metric.
  - Exclusion of marginalized user.
  - Legal/Regulatory Compliance.
- **Resources:**
  - AI Ethics Guidelines
  - NIST AI RMF
- **Mitigation Strategies:**
  - Conduct meetings to discuss the project scope and goals with stakeholders.
  - Define success metrics: latency less than 2 seconds, weather compliance is greater than 95%, and the user satisfaction is greater than 80%.
  - Conduct legal and regulatory compliance reviews.
- **Technical Mitigation Strategies:**
  - Conduct the residual risk assessment.
  - Requirements checklist with Github Issues.

- Providing a flow diagram with Lucidchart.

2. Data Collection: Good data results in a more accurate and robust AI model

- **Key Risks:**
  - Poor data quality (noisy or incomplete).
  - Bias in data: over-representation of certain data categories.
  - Risk of privacy being compromised.
- **Resources:**
  - Data Ethics and Bias
  - Data Privacy Laws
- **Mitigation Strategies:**
  - Enforce consistent labels for categorical attributes, automate schema validation.
  - Implement a duplicate/missing handling system.
  - Monitor data distribution and targeted augmentation for under-represented data.
  - Apply privacy protection.
- **Technical Mitigation Strategies:**
  - Implement automated data cleaning and augmentation techniques via Pandas.

3. AI Model Development

- **Key Risks:**
  - Algorithmic Bias (biased learning patterns).
  - Explainability (risk of the model becoming a “black box”).
  - Overfitting/Underfitting Issues.
- **Resources:**
  - Fairness-Aware Algorithms (e.g., Adversarial Debiasing, Fairness Constraints).

- Model Explainability Tools (e.g., LIME, SHAP).

- **Mitigation Strategies:**
  - Implement Cross-Validation/Regularization.
  - Using LIME/SHAP to introduce explanations of the model's functionality and flow.
- **Technical Mitigation Strategies:**
  - Scikit-learn GridSearchCV function with a timing aware CV.
  - LIME/SHAP notebook preparation.

4. AI Deployment

- **Key Risks:**
  - Integration difficulties.
  - Security Liability.
- **Resources:**
  - Containerization with Docker.
  - A/B Testing.
- **Mitigation Strategies:**
  - Tracking the performance of the system and applying security measures.
  - Implement A/B testing to smoothen the integration process without altering the existing workflows.
  - Redact traceable personal information.
- **Technical Mitigation Strategies:**
  - Implement the use of TLS
  - Deploy a BentoML API with JSON Web Tokens access control and logging within Docker; automate CI with Github Actions.

- Use CI/CD pipelines with gates.

5. Monitoring and Maintenance

- **Key Risks:**
  - Concept/data drift
  - WeatherAPI outage
  - Newly formulated security threats
- **Resources:**
  - Automated Monitoring (e.g., Prometheus, Grafana)
- **Mitigation Strategies:**
  - Introducing a drift detection system to retrain pipelines.
  - Set up Service-level objectives (latency requirement, weather refresh.
  - Set up an alert system if there's a sign of data drift with NannyML

6. Residual Risk Assessment

			Impact			
			0 Acceptable	1 Tolerable	2 Unacceptable	3 Intolerable
			Little or no effect	Not critical	Serious Impact	Disasterous
Likelihood	Improbable	Unlikely	typo in User Interface	Overfitting/ underfitting	1. Personal data leakage 2. Legal issues	
	Possible	Likely	API outage	1. Representativeness bias 2. Intergration issues 3. Stakeholder exclusion 4. Schema drift	data drift	S3 misconfiguration
	Probable	Will occur	API timeout	user input typo	Seasonal model drift	Security breach

- Mitigation tactics for these risks are mentioned above

Data Collection Management and Report

1. Data Type

- Structured tabular data (Item ID, Type, Color, Season, Occasion, Material)
- External WeatherAPI data.
- Data granularity: user inputs vs. formatted tables.

2. Data Collection Methods

- **Source of Data:**
    - User-generated entries that will populate a CSV/User Interface.
    - External API: WeatherAPI (HTTPS).
- **Methodologies Applied:**
    - Batch CSV ingestion with the Pandas library.
    - REST APIs for automatic weather data retrieval.
- **Ingestion for Training:**
    - Load data with Pandas
    - Stratified splits the data by Season and Occasion for a more accurate evaluation
    - The private and encrypted data will be stored in the S3 bucket.
- **Ingestion for Deployment:**
    - Real-time REST communication between client and server, regular weather fetching with caching.
    - Kafka/RabbitMQ) if throughput grows.

3. Compliance with Legal Frameworks

- **Applicable Laws and Standards:**
    - GDPR/CCPA

- NIST standards for cybersecurity
- ISO/IEC 27001 for information security

- **Compliance Strategy and Results:**
  - No personal data for core features; ensuring user privacy.
  - TLS while transferring data, encrypting data in the S3 bucket; ensuring user privacy.
  - Ask for consent of the user for non-crucial analytical data collection.
  - Providing an option for users to download/delete their data.

4. Data Ownership and Access Rights

- **Ownership and Access Control:**
  - Users have ownership to their item catalog data; the model will only act as a processor.
  - JSON Web Token roles for APIs
  - S3 bucket will have the most limited privilege
  - Regular access rights and logging assessment.
- **Lessons Learned:**
  - Reducing personal data will significantly reduce compliance complexity and logging risk.

5. Metadata Management

- **Metadata Content and Management System:**
  - Item metadata: ID, color, season, occasion, material, timestamps
  - Dataset metadata: count, features, generation timestamps,
  - System metadata: JSON

6. Data Versioning

- **Version Control System and Strategy:**
    - Git for CSV and JSON files; include commit messages
    - Commit after ingestion cycles, record milestones

7. Data Preprocessing, Augmentation, and Synthesis

- **Preprocessing Techniques:**
    - Normalize strings variables, standardize attribute names, nulls and duplicate handling.
- **Data Augmentation and Synthesis:**
    - Targeted for under-represented categories.
    - Introducing safe synthetic data for unusual scenarios.
    - Feature scaling for numerical categories
    - LIME/SHAP notebooks to clarify preprocessing purpose to ensure transparency.

8. Data Management Risks and Mitigation

- Poor data quality: Use Pandas and schema checks with warning messages.
- Bias: Distribution reports and targeted data augmentation.
- Privacy concerns: Diffprivlib, avoid personal data in datasets.
- Data drift concerns: item name validations

9. Data Management Trustworthiness and Mitigation

- Ensure model fairness: stratified split Season/Occasion; ensure user satisfaction
- Privacy: Collect minimal features; redact logs; implement differential-privacy noise to protect user-level data statistics
- Transparency: Users can view and edit closet data.

Github Repository:

https://github.com/itsmekhang/ClosetAI-Development

Model Development and Evaluation

1. Model Development

ClosetAI integrates a hybrid AI structure that combines semantic similarity modeling, data analysis with differential privacy.

- **Algorithm Selection:**
    1. SentenceTransformer: handles and generate clothing categories and occasions based on the provided context embeddings. It was selected for its strong semantic understanding and requires minimal performance.
    2. SpaCY NLP Pipeline: to parse user prompts and extract tokens and keywords, to guarantee efficient human-language understanding without using a pre-tuned model.
    3. Rule-based weather adaption logic: Uses data from WeatherAPI to fine tune the recommendation based on the weather condition (hot or cold, clear or rainy).
    4. Laplace Mechanism (diffprivlib): Ensure private data analysis of data distribution for bias and privacy audits.
- **Feature Engineering and Selection:**
    a. **Features:**
        - Category: derived various items to the appropriate categories using sentence transformers to detect keywords. Hard coded libraries were implemented to help with the sorting process.
        - Occasion: Similar process where the occasions of the items are determined by a weighted score and predetermined logic.

- Season: Standardized and cleaned for all items to fit into various seasons (Fall, Winter, Spring, Summer, All)
- Weather Features: is_hot, is_cold, is_rain, and need_pants are derived from information retrieve by WeatherAPI.

b. Validation:

- Schema and duplicates are validated with the clean_season() function; and missing categorical data are automatically classified as “All” to ensure robustness.
- Distribution is being analyzed to ensure a balanced dataset via print_share().

- **Model Complexity and Architecture:**

The model is structured as a modular multi-stage pipeline:

- Schema check, duplicate handling, and differential privacy analysis.
- A pretrained sentence transformer converts item descriptions into numbers to represent its meanings.
- Implement a scoring system that scores similarity, predefined logic, and weather conditions to generate appropriate outfits.
- The items are scored, randomized for variety, and the most appropriate items in each category are selected.
- Gradio UI was used to build the User Interface.

2. Model Training

- **Training Process:**

- ClosetAi relies on pretrained transformer from intfloat/e5-base-v2 to eliminate retraining the model from scratch and reduce computational cost.
- Each item was compared to the categories and occasions reference dictionary to retrieve the most appropriate match.
- The results of each item were checked to make sure the items are labeled correctly.

- **Hyperparameter Tuning:**

1. Normalize sentence transformer embeddings to handle similarity scores (True)
2. Similarity threshold to tuned the similarity tuning (0.814)
3. Privacy control (epsilon = 1.0)
4. Weather logic (hot weather > 80°F, cold weather < 70°F…)
5. Random noise injection (± 0.5)

3. Model Evaluation

- **Performance Metrics:**
  - Categorizing accuracy results are outputted and inspected to ensure performance (>90% match)
  - Occasion Match rate: most prompts resulted appropriate outfit generations across at least three matched categories.
  - Random noise reduces repetition for outfit generation.
  - Differential privacy noise retained confidentiality while maintaining data usability.
- **Cross-Validation:**

- Multiple user prompts were tested where the location weather and the occasions are the dependent factors for outfit generation.
- Evaluation statements are provided to justify outfit generation when switching between weather conditions and occasion types, ensuring transparency.

4. **Implementing Trustworthiness and Risk Management in Model Development**

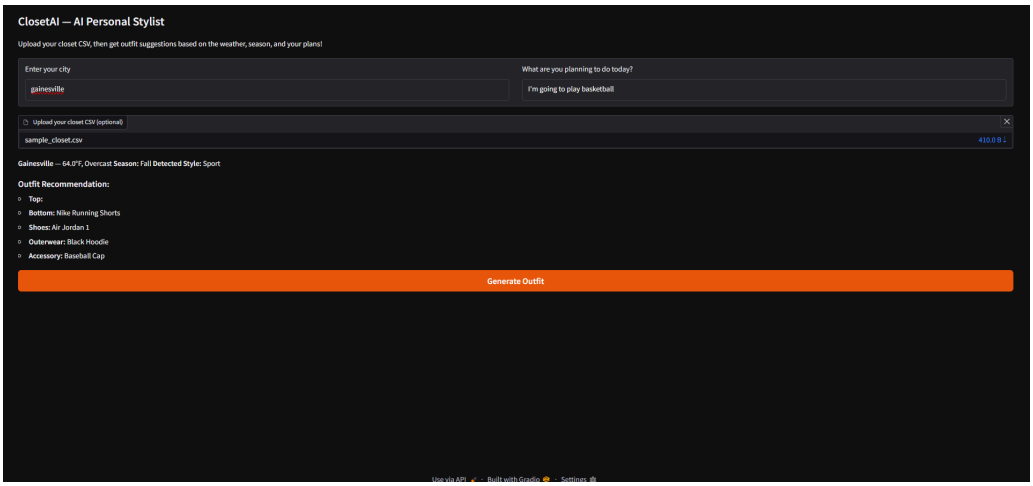
- **Risk Management Report:**
    - The system includes a Risk Management Report generation function that will analyze the identified risks, impact assessment, mitigation strategies
- **Trustworthiness Report:**
    - The Risk Management report generation function promotes transparency, accountability, and reproductivity, by providing timestamps, data information, sources, and comments.
    - Explainability is ensured by making the scoring logic clearly traceable in the code.
    - Fairness is ensured by including multiple weather conditions, items, and occasion handling with equal inclusivity.

5. **Apply HCI Principles in AI Model Development**

- **Wireframes:**
    - The user interface layout prioritizes simplicity and clarity. The user enters two prompts into two textboxes that's asking for the occasion of the outfit and the location to retrieve the weather condition for outfit processing. The application provides an option for the user to upload their closet CSV files. The UI will include a display the location, weather condition, the detected outfit occasion, and

for the outfit recommendation. It adheres the standard wireframe principles: clear and minimal information, minimal user-computer interaction, and left-to-right and top-to-bottom input/output layout.

- **Develop Interactive Prototypes**
    - ClosetAI comes with an interface that serves as an interactive prototype. Everytime the user clicks on the “Recommend Outfit" button, it triggers the full model pipeline, which includes weather retrieval, data preprocessing, and scoring. The results change depending on the different occasion and location inputs.
- **Design Transparent Interfaces**
    - The interface displays the weather location, season, and detected style, the factors that influences the outfit creation.
    - The outfit will generate by grouping different items from different categories to generate a complete outfit. It will display the categories of the items then the item name for each categories.
- **Create Feedback Mechanisms**
    - Provide a flag button if the user is unsatisfied with the recommendation.



Deployment and Testing Management Plan

1. Deployment Environment Selection

ClosetAI will be deployed locally using a Gradio powered UI application executing through main.py within the src/ folder. This deployment plan is appropriate because the model performs lightweight computation using a SentenceTranformer and WeatherAPI API calls. The model does not require GPU or large-scale computation. Locally deploying the model can help align with the goals:

- Performance: Latency can be under 2 seconds
- Scalability: No load balancing or cloud autoscaling required.
- Security: user input does not contain sensitive data, and no information is stored.
- Efficiency: The environment is replicable using requirements.txt

2. Deployment Strategy

ClosetAI requires multiple separate modules to function: the preprocessing, model inference, trustworthiness validation, and the UI handling module.

- model_pipeline.py - item classification, appropriate scoring, and outfit generation logic.
- data_validation.py - schema checking, differential privacy implementation, distribution check.
- ui.py - Handles the Interface using Gradio, and reports inference time metric.
- main.py - Handles data upload, runs everything.

3. Security and Compliance in Deployment

a. Data Security

- No personal user information is collected.

- Uploaded closet CSV are processed in the local memory and will be discarded instantly.

b. Model Integrity

- Schema validation prevents malicious CSV inputs
- Differential privacy is applied, preventing leakage.

Evaluation, Monitoring and Maintenance Plan

1. System Evaluation and Monitoring

ClosetAI tracks a variety of performance metrics:

- Inference time (< 2 second latency goal)
- Schema drift check (ensure uploaded CSV is valid)
- Distribution drift for season and occasion (<60%)
- Weather-condition fall-back logic.

2. Feedback Collection and Continuous Improvement

- The Gradio Interface enables users to re-upload their closet CSV.
- The AI classification system will output the result for debugging purposes
- Logs to record schema and distribution drift, risk alerts, and inference time report.

+ **Continuous Improvement:**
    - Update categorization rules
    - Expand clothing-type rules
    - The item color will dictate the outfit generation logic.

3. Maintenance and Compliance Audits

- Monthly dependency updates using “pip-review” and requirements.txt
- Schema validation runs every time the user opens the application.
- API key refresh for WeatherAPI.