Programming Language 2:
Course Work Report

Coursework title:

wea-rCloth

Team Members and their responsibilities

Name1 and responsibilities: **Name2 and responsibilities:**

1. Nurali Bakytbek uulu: Project Management and Programming
2. Aijan Tilekova: Data Collection

Objective:

The objective of this coursework is to design and develop an intelligent web-based application titled **“wea-rCloth”**, which functions as a personal wardrobe assistant. The goal is to provide users with a platform that helps them digitally manage their clothing items, create and rate outfit combinations, and gain insights into their fashion preferences through visual analytics.

This system leverages **Streamlit** as a rapid web development framework, enabling an interactive and responsive user interface, while **Airtable** serves as the backend database for storing user data, wardrobe items, and outfit combinations. The application supports secure user authentication and includes role-based access control — regular users can manage their own wardrobes, while administrators have the ability to oversee and manage all users and records.

A key feature of the application is its ability to generate outfit recommendations based on user-defined filters such as **season** and **style**. It also provides options to manually create and rate combinations, and mark favorites. Users can update personal profile details including their avatar and bio, making the experience more personalized.

Through data visualization using libraries like **matplotlib** and **seaborn**, the system provides analytical insights into wardrobe trends, usage patterns, and top-rated combinations. Ultimately, this coursework demonstrates the integration of full-stack development concepts including front-end UI design, backend database management, session handling, and user-centric feature design in a cohesive, real-world application.

Methodology

The development of the **wea-rCloth** project followed a structured and iterative software engineering methodology that combines elements of the **Agile model** with **modular development** principles. The methodology emphasizes rapid prototyping, real-time user interface feedback, and continuous integration of components. The application was built using **Python**, with **Streamlit** for the frontend, and **Airtable** as a cloud-based backend database.

1. Requirement Analysis

The first step was gathering functional and non-functional requirements for the system. The primary goals were:

- Enable users to maintain a virtual wardrobe.
- Generate and rate clothing combinations based on season and style.
- Provide personalized analytics and visualizations.
- Offer profile management with avatar uploads and bio updates.
- Include an admin panel for data oversight and user management.

Key non-functional requirements included data persistence (using Airtable), quick user authentication, and an intuitive, user-friendly interface.

2. Technology Stack Selection

The technology choices were driven by rapid development goals and easy deployment:

- **Frontend/UI:** Streamlit (Python-based UI framework ideal for data-driven apps).
- **Backend:** Airtable (a lightweight, spreadsheet-style cloud database, accessed via the pyairtable API).
- **Data Processing:** Pandas for data wrangling and transformation.
- **Visualization:** Matplotlib and Seaborn for charts and analytics.
- **Authentication:** Bcrypt for password hashing and session state for login persistence.

3. Modular Architecture

To ensure clarity and maintainability, the codebase was divided into logical modules:

- **auth.py and auth_ui.py**: Handle authentication, signup, login, and session management.
- **app.py**: Acts as the central controller, handling routing, page rendering, and layout.
- **profile_ui.py**: Displays and manages user profile data, including avatar upload.
- **wardrobe_editor.py**: Interface for editing and deleting wardrobe items.
- **airtable_utils.py**: Abstracts interaction with Airtable for loading and saving data.
- **admin_panel.py**: Provides admin-exclusive tools to manage users and their data.
- **constants.py**: Central repository for static lists (e.g., categories, styles).
- **state_management.py**: Initializes and maintains session variables.
- **wardrobe_helpers.py and ui_components.py**: Utility functions and reusable UI components.

This modular approach supports future expansion, such as adding new features like outfit calendars, machine learning-based recommendations, or community sharing.

4. User Authentication and Session Handling

Authentication was implemented using secure password hashing via bcrypt. Upon successful login, the user's information is stored in st.session_state, which persists across pages during a session.

Two user roles were implemented:

- **User (status = 0)**: Limited to their own wardrobe and combinations.
- **Admin (status = 1)**: Full visibility and control over all user data, including deletion.

To prevent unauthorized access, every protected view checks the session state and redirects unauthenticated users to the login interface.

5. Wardrobe and Outfit Combination Logic

Each wardrobe item includes attributes such as:

- Category (Upper body, Lower body, Footwear)
- Type (e.g., Shirt, Jeans)

- Style (e.g., Casual, Formal)
- Season (e.g., Summer, Winter)
- Color
- Image (optional upload)
- Auto-generated unique model ID

When adding items, a **model ID** is generated based on category, type, style, and season to maintain uniqueness.

For outfit generation:

- Users select a **style** and **season**.
- The system filters items that match these attributes.
- An outfit is formed by randomly selecting one item from each required category.
- The user can then rate the outfit and optionally mark it as a favorite.

Manual combination creation is also supported, giving users control to build and save looks.

6. Airtable Integration

Airtable was chosen for its ease of setup, real-time sync, and spreadsheet-like interface. It stores three core datasets:

- users_data: Contains email, password hash, bio, role, and avatar.
- wardrobe_data: Clothing item entries with associated metadata.
- combinations_data: Saved outfit combinations, ratings, and favorites.

The pyairtable Python SDK was used for reading and writing data. Airtable table keys and base IDs were securely embedded in utility modules, though they could be further abstracted into environment variables for deployment.

7. Profile and Avatar Management

Each user can view and edit their profile. Avatar uploads are handled using Streamlit's file uploader. Uploaded images are saved locally with timestamps to avoid filename collisions. The avatar URL is updated in Airtable and displayed dynamically in the profile view.

Users can also edit their bio, which is reflected in the UI and persisted in the database.

8. Visualization and Analytics

Users can analyze their wardrobe composition using:

- **Pie charts** for distribution across types or styles.
- **Bar charts** for item counts by attribute.
- **Line charts** for changes over time.
- **Scatter plots** comparing style vs. season.

For combinations, users can:

- Filter combinations by style, season, and rating.
- View the top-rated outfits.
- See rating distribution with histogram + KDE plots.

This adds an engaging, data-driven layer that helps users discover their personal trends.

9. Admin Controls

Admins can:

- View a list of all registered users.
- View clothing and combo counts per user.
- Permanently delete a user and all their data (wardrobe + combinations).

This is a critical feature for data cleanup, testing, and ensuring proper role separation.

10. Theming and UX

Users can toggle between **Light** and **Dark** themes, improving accessibility and aesthetic preference. This is applied dynamically using inline CSS injection in Streamlit.

The UI was designed to be clean, minimal, and mobile-responsive as much as Streamlit allows. Each form and interaction provides real-time feedback through success, warning, or error messages.

11. Testing and Iteration

The app was tested locally with various dummy accounts. Particular focus was given to:

- Data consistency between frontend actions and Airtable.

- Resilience against bad inputs (e.g., missing fields, upload errors).
- Session and role access behavior.
- Usability of forms and filters.

Bugs and logic issues were identified and resolved iteratively through logging and test inputs.

Novelty of work compared with the existing works

Wea-rCloth is the first structured digital wardrobe assistant in Kyrgyzstan, The development of **wea-rCloth** introduces a unique approach to personal wardrobe management that is not commonly found in Kyrgyzstan’s tech ecosystem. While fashion apps and e-commerce platforms are gaining popularity in the region, there is a noticeable gap when it comes to **personalized digital wardrobe systems** — especially those that offer smart outfit suggestions, visual analytics, and user-driven customization.

In contrast to global fashion-tech giants that often rely on heavy infrastructure, machine learning, or high-cost app development, wea-rCloth is designed to be lightweight, accessible, and easily deployable. By using **Streamlit** for the interface and **Airtable** for the backend, the system remains low-cost while providing a responsive and user-friendly experience. This makes it an excellent candidate for implementation in educational settings, startups, or individual use in resource-constrained environments.

Globally, wardrobe planning apps often focus solely on visual outfit mixing or shopping recommendations. wea-rCloth adds value by incorporating features like:

- Auto and manual outfit creation based on seasons and styles
- User ratings and favorite marking
- Profile-based wardrobe analytics
- Admin-level user and data management tools

This balance of personalization, data visualization, and practical functionality sets it apart from both regional tools and many mainstream fashion apps. As such, this work contributes a **novel, user-centered, and scalable approach** that can serve as a foundation for future smart fashion tools in both local and global contexts.

Results and Discussions

The development and implementation of the **wea-rCloth** system yielded promising results in terms of functionality, user experience, and performance. This section presents the outcomes of the project and discusses its strengths, limitations, and areas for future enhancement.

Functional Outcomes

The system successfully met its intended objectives, with all major features implemented and functioning as expected. The following key modules were developed, tested, and integrated into the final build:

1. User Authentication

Users are able to sign up, log in, and manage their session securely. Passwords are hashed using bcrypt, ensuring data confidentiality. Session persistence is maintained through streamlit.session_state, which prevents unauthorized access to protected views. The role-based access model effectively differentiates between regular users and administrators.

2. Wardrobe Management

Users can add clothing items by specifying details such as category, type, color, style, and season. Each item is assigned a unique **model ID** that encodes key attributes, making it easier to track and filter items. Image uploads allow users to visually represent their wardrobe, improving engagement. Editing and deletion are also supported, enabling users to keep their wardrobe up to date.

3. Outfit Combination

The system supports both **automatic** and **manual** generation of outfit combinations:

- **Automatic generation** filters clothing based on selected **season** and **style**, and randomly combines items from different categories (upper, lower, footwear).
- **Manual creation** allows users to hand-pick each item, ensuring full control over the combination.

Users can then **rate** the combination and **mark it as a favorite**, making it easier to revisit preferred outfits.

4. Visualization and Analytics

One of the standout features of wea-rCloth is its interactive visualization tools. Users can generate:

- **Pie charts** to understand distribution of wardrobe items by type, category, or season.
- **Bar charts** to visualize counts of styles or colors.
- **Line charts** and **scatter plots** to explore attribute relationships.
- **Histograms** with KDE plots to analyze rating distributions.

These tools offer meaningful insights into personal style preferences and wardrobe diversity, providing both practical and aesthetic value.

5. Profile Customization

Users can update their bio and upload a profile picture. Uploaded images are locally stored and referenced in Airtable for persistent access. Profile stats such as total clothes and saved combinations are displayed to encourage usage and exploration.

6. Admin Tools

Admins are granted access to:

- A dashboard listing all users.
- Metrics for total wardrobe items and combinations.
- Ability to view, edit, and delete users and their associated data.

This ensures that system administrators can maintain data hygiene and support troubleshooting.

Usability Testing

The system was tested with several dummy user accounts, covering various roles and actions. The feedback collected during manual testing included the following observations:

- **Intuitive Interface**: Most users found the Streamlit interface easy to navigate, with clearly labeled sections and minimal learning curve.
- **Responsive Feedback**: Success messages, error handling, and real-time updates (e.g., `st.rerun()`) provided confidence during interactions.
- **Useful Visuals**: Charts and analysis tools were highlighted as one of the most enjoyable and informative features.

Performance Observations

The system performed reliably across multiple test sessions with varying input sizes. Airtable's API integration proved sufficient for small to medium datasets. However, the following performance considerations were noted:

- **Data Fetching**: Repeated calls to Airtable (especially `.all()`) introduced some latency, particularly in admin views that process large record sets.

- **Local File Handling**: Image and avatar uploads rely on local file paths, which may not be viable in a cloud-hosted deployment.

These observations point to the potential need for backend caching or transitioning to cloud storage (e.g., AWS S3 or Firebase) for media assets in a production setting.

Discussion of Innovation

Compared to traditional fashion or wardrobe apps, wea-rCloth offers several innovative elements:

- **Dual outfit creation paths**: Users are not limited to automated suggestions; they can create precise combinations manually.
- **Embedded analytics**: Most wardrobe tools focus only on cataloging; wea-rCloth introduces visual self-analysis of style patterns.
- **Airtable integration**: Rather than building a full backend from scratch, the app leverages Airtable to streamline development without sacrificing functionality.
- **Streamlit-based design**: Streamlit is rarely used for consumer-facing apps, but in this case, it enabled rapid prototyping and real-time UI feedback, making it an ideal choice for this academic project.

These innovations make the system accessible, flexible, and extensible, particularly for individual users, educators, or small fashion startups.

Limitations

Despite its many strengths, wea-rCloth has some limitations:

1. **Scalability Constraints**
   - Airtable has API call limits, and performance may degrade as the number of users and wardrobe items grows.
   - No pagination or lazy loading is implemented, which could cause slowdowns for large datasets.
2. **File Storage**
   - Avatar and clothing images are stored locally, making them inaccessible across multiple devices unless deployed on a centralized server.
3. **Lack of Mobile Optimization**
   - Streamlit offers limited mobile responsiveness. Some forms and visuals may not display properly on smaller screens.
4. **No AI/ML-Based Recommendations**
   - Outfit suggestions are rule-based (style/season matching) and do not include intelligent recommendations or user behavior analysis.

Future Enhancements

Based on the insights gained during development, several enhancements are envisioned:

- **Cloud Storage Integration**: Move image handling to services like Firebase or AWS S3 to ensure scalability and cross-device access.
- **AI Outfit Suggestions**: Introduce machine learning models to recommend outfits based on usage patterns, weather data, or user preferences.
- **Calendar Planning**: Allow users to plan outfits for specific days, creating a visual timeline of wardrobe usage.
- **Sharing & Community**: Enable users to share combinations or view public profiles for style inspiration.
- **Improved UI Framework**: Consider migrating to frameworks like React or Flutter for a more responsive and polished front-end experience.

Real Time Applicability of Wea-rCloth

The **wea-rCloth** system offers strong real-time applicability for both personal and administrative use. For individual users, the platform acts as a **digital wardrobe assistant**, enabling them to instantly add, update, and manage clothing items. Users can generate outfit combinations on-the-fly based on changing factors like weather, season, or occasion, making it a practical tool for daily fashion planning.

In real-time scenarios, users benefit from immediate feedback during outfit generation, clothing uploads, and profile changes. The outfit rating and visualization features support reflective decision-making by helping users track what they wear most and what styles dominate their wardrobe.

From an admin perspective, real-time access to all user data enables timely moderation, data correction, or user support. Since the system is built on **Streamlit with Airtable integration**, all updates are synced live with the cloud backend, ensuring data consistency across sessions and devices.

Overall, wea-rCloth bridges fashion organization and decision-making with interactive, real-time functionality, making it highly relevant for students, professionals, or style enthusiasts looking to manage their wardrobe more efficiently.

Conclusion

The development of the **wea-rCloth** application demonstrates a successful integration of modern web technologies with practical lifestyle utility. Designed as a digital wardrobe assistant, the system provides users with a personalized platform to organize clothing items, generate outfit combinations, and analyze fashion preferences through visual insights.

By leveraging **Streamlit** for an interactive user interface and **Airtable** as a cloud-based backend, the project achieved its goals of creating a user-friendly, accessible, and real-time system. The incorporation of essential features such as secure login, role-based access control, image uploads, and smart combination generation added significant value to the overall functionality.

Through its modular architecture, the application ensures code reusability and scalability. Regular users are empowered to manage and explore their personal wardrobe efficiently, while administrators are equipped with tools to manage users and data globally. The visual analytics component provides users with meaningful wardrobe insights, encouraging self-reflection on clothing usage and preferences.

The novelty of wea-rCloth lies in its balanced blend of practicality, personalization, and simplicity. Unlike typical fashion or e-commerce apps, it focuses on helping users make better use of what they already own rather than encouraging new purchases. This makes it not only innovative but also sustainable.

In conclusion, wea-rCloth represents a practical and impactful project with real-world relevance. It lays a strong foundation for future enhancements such as AI-based outfit recommendations, mobile deployment, and community sharing. Overall, the project highlights how thoughtful software design can transform everyday activities—like getting dressed—into smarter, data-driven experiences.

References

Technical References

1. Streamlit Documentation – https://docs.streamlit.io/
2. Airtable API – https://airtable.com/api
3. pyAirtable (Python SDK) – https://github.com/gtalarico/pyairtable
4. bcrypt Python Library – https://pypi.org/project/bcrypt/
5. Matplotlib Documentation – https://matplotlib.org/stable/contents.html
6. Seaborn Documentation – https://seaborn.pydata.org/
7. Pandas Documentation – https://pandas.pydata.org/docs/

Project Assets & Links

- **GitHub Repository**:
  https://github.com/i-nura-l/Projects/tree/main/wea-rCloth
- **Live App (if deployed)**:
  https://wea-rcloth-app.streamlit.app/
- **Demo Video or Screenshots**:
  on README.md in github

Course Name:	Programming 2 Course Work	Project Name:	wea-rCloth
Advisor:	Dr. Tauheed Khan	GitHub Link:	wea-rCloth
Team Name:	DataS	WebApp:	Project_App
Team Members:	Nurali Bakytbek uulu	AirTable Data:	Database
	Aijan Tilekova	Report:	Dock

Project Progress Tracker

Members:	Responsibility:	Week_1	Week_2	Week_3	Week_4	Week_5	Week_6	Week_7	Week_8	Week_9	Week_10
Nurali Bakytbe uulu	Programming			Finalize wardrobe & combination dataset formats	Implement outfit generator logic + rating system	Design and test wardrobe filters		Refactor code structure for readability	Implement access control (user-specific data)	Add filtering and search refinements (if it is user or admin)	
	Data Collecting + Cleaning			Get to know Airtable and Streamlit library	Handle file loading/saving, duplicates, sessions	Build charts (bar, pie, line, scatter)	Bug testing and edge case handling	User data: count clothes and combinations.	Collecting user data and saving it in database	Collecting data from user, and ability to delete users from database	Bug testing and edge case handling
	App building	**BrainStorming** (searching Internet for ideas, wrting down several ideas, and collecting the needed information)	1. Choosing the Project 2. Collecting information on the project 3. Creating a Road-Map and future plans and deviding responsibilites.	Implement data loading/saving (CSV)	Create basic navigation (sidebar pages)	Build wardrobe input form UI		Build Profile page	Build Login page	Build Admin page	
				~~Using PyQt6 as app MVP~~							
Aijan Tilekova	Data Collecting + Cleaning			Create base CSV datasets or template	Provide wardrobe test data (diverse examples)	Refining the data		New columns in Airtable. Updated wardrobe and combination tables	New data table: user_data (to collect user's data)	Test multi-user data separation	
	Searching			Design data structure: wardrobe + combinations	~~Prepare visualizations for use in reports~~		Complete report (results, discussion, conclusion)		Verify missing image/file handling		Complete final report (results, discussion, conclusion)
	Reporting			Draft project plan, goals, and methodologya	Start drafting final report: intro, goals, methodology	~~Create presentation slides (short, visual, clear)~~				Create presentation slides (short, visual, clear)	

Future Goals (What I have envisioned:)	Color - if Done						
1. Real-Time Weather Integration	Use weather APIs (like OpenWeatherMap) to automatically suggest outfits based on live weather in the user's location.	→ "It's 3°C and windy? Wea-rCloth has your back with a coat + boots combo."					

2. Cloud-Based Storage & Sync	Replace or complement CSV with Airtable, Firebase, or PostgreSQL for:	Real-time syncing across devices	Multi-user capability	Easier mobile expansion							
3. Image Upload + Color Detection	Upload user's cloth and avatar image	Automatically extract dominant colors using OpenCV or other image processing tools.	→ Helps with color-coordinated outfit generation.								
4. Machine Learning Recommendations	Train an ML model using:	Combination ratings	User preferences	Weather + event context	To predict and suggest the best outfits.	→ Personalized fashion AI, locally tuned.					
5. Mobile App Version	Streamlit App	Convert the Streamlit prototype into a PWA (Progressive Web App)	or build a Flutter app so users can manage their wardrobe on the go.								
6. Social + Community Features	Share outfit combos	Like/comment others’ styles	Public “lookbook” for seasonal inspo								
7. Outfit Calendar & Planning	Add a timeline where users can plan outfits ahead of time (e.g., for trips, events, workdays).	→ “Tomorrow is rainy and formal? Here's your plan.”									
8. Notifications & Daily Suggestions	Push notifications with “Outfit of the Day” based on weather, mood, and occasion.	→ Hook users into using the app daily.									