**1. Define Scope & Requirements**

* **Project Architecture:**
  + **Frontend:** Use Next.js for a fast, React-based UI.
  + **Backend:** Build API endpoints in Python (using Flask, FastAPI, or Django) that serve data and analysis results.
* **Design Aesthetic:**
  + Emulate Apple’s website style with a minimalist, clean, and modern design (lots of white space, elegant typography, and subtle animations).

**2. Data Exploration & Preparation**

* **Understand Your Data:**
  + Review the CAMELS-AUS v2 data description to understand the variables (e.g., streamflow, precipitation, catchment attributes) and their formats.
  + Identify key files: time series (e.g., streamflow, precipitation, temperature), catchment boundaries, and attribute tables.
* **Data Cleaning:**
  + Handle missing data and outliers.
  + Convert dates into proper datetime objects for time series analysis.
* **Preprocessing:**
  + Aggregate or smooth data if needed for clarity.
  + Normalize or scale variables for comparison.

**3. Analysis & Visualization Ideas (4 Tabs/Pages)**

* **Tab 1: Overview/Dashboard**
  + **Summary Stats:** Display key metrics (e.g., average streamflow, total precipitation, number of catchments).
  + **Interactive Charts:** Use graphs to show overall trends or distributions.
* **Tab 2: Time Series Analysis & Prediction**
  + **Visual Time Series:** Plot historical streamflow or precipitation data over time.
  + **Decomposition:** Show seasonal, trend, and residual components.
  + **Forecasting:** Implement a simple forecasting model (e.g., ARIMA, Prophet, or even an LSTM for a more advanced approach) to predict future streamflow.
  + **Interactivity:** Allow users to adjust parameters (e.g., forecast horizon) and view prediction confidence intervals.
* **Tab 3: Correlation & Comparative Analysis**
  + **Scatter Plots & Heatmaps:** Display correlations between variables such as precipitation vs. streamflow, or catchment attributes vs. hydrological responses.
  + **Interactive Filtering:** Let users select different catchments or time periods to see how relationships vary.
* **Tab 4: Spatial & Catchment Analysis**
  + **Mapping:** If you have catchment boundary data, integrate a map to visualize spatial distribution.
  + **Attribute Visualizations:** Use choropleth maps or interactive overlays to show how different catchment attributes (e.g., land cover, geology) influence hydrological behavior.

**4. Backend & API Development**

* **Data Analysis Scripts:**
  + Write Python scripts (using libraries like Pandas, NumPy, SciPy, statsmodels, or Prophet) to perform the data analysis.
  + Precompute heavy analysis tasks (such as forecasting) if real-time processing is not necessary.
* **API Endpoints:**
  + Create RESTful endpoints to serve:
    - Raw and processed data.
    - Visualizations (as JSON data for interactive charts or pre-rendered images).
    - Prediction results.
* **Integration:**
  + Ensure that the API endpoints are secure and efficient.
  + Use CORS settings to allow Next.js to fetch data from the Python backend.

**5. Frontend Development in Next.js**

* **Page Routing:**
  + Set up 4 pages (or tabs) corresponding to the analyses outlined above.
* **Design & Styling:**
  + Use custom CSS or a CSS-in-JS library (like styled-components) to mimic the clean, minimalist aesthetic of the Apple website.
  + Incorporate responsive design principles for a smooth user experience on both desktop and mobile.
* **Interactivity:**
  + Utilize libraries such as Chart.js, D3.js, or Plotly for interactive charts.
  + Implement dynamic data fetching from your Python API.

**6. Testing, Integration & Deployment**

* **Testing:**
  + Test API endpoints and data integrity.
  + Verify that the Next.js frontend correctly displays data and handles user interactions.
* **Integration:**
  + Connect the frontend with the backend, and test the full data flow.
* **Deployment:**
  + Consider deploying the Next.js app on platforms like Vercel.
  + Deploy the Python backend on a service like Heroku, AWS, or DigitalOcean.

**7. Creative Enhancements & Future Ideas**

* **User Interactivity:**
  + Allow users to select different catchments or time periods to explore “what-if” scenarios.
* **Model Exploration:**
  + Provide options to toggle between different forecasting models or analysis methods.
* **UI/UX Tweaks:**
  + Use smooth transitions and subtle animations to make the user experience engaging and elegant, staying true to the Apple aesthetic.

**TAB 1:**

**Key Metrics (Summary Cards)**

1. **Total Catchments:**  
   Count the number of catchments included in the master attribute table. This gives an immediate sense of the dataset's scale.
2. **Average Catchment Area:**  
   Use the catchment\_area field (from Table 2) to calculate and display the average area. You could also show the minimum and maximum values for context.
3. **Streamflow Metrics:**
   * **Average Streamflow:** Compute the average of streamflow values (using data from streamflow\_MLd.csv or the infilled version).
   * **Key Streamflow Signature:** For example, display the average or median of a signature like sig\_mag\_BaseMag (from Table 4).
4. **Precipitation Metrics:**  
   Show the average daily precipitation (from the hydrometeorology data in Table 7) or total annual precipitation, giving insights into the water inputs to the catchments.

**Interactive Charts**

* **Time Series Trends:**  
  Create a line chart displaying overall streamflow or precipitation trends over time. This could be an average across catchments or allow users to filter by region.
* **Distributions:**
  + **Histogram of Catchment Areas:** Show how catchment sizes are distributed.
  + **Distribution of Streamflow Values:** A histogram or box plot to highlight variability in streamflow across catchments.
* **Scatter Plots:**  
  Plot relationships such as catchment area versus average streamflow or precipitation, helping users visually identify correlations.
* **Optional Map View:**  
  If you have spatial data (e.g., catchment outlet coordinates), include a map to show the geographical distribution of the catchments.

**Dashboard Layout & UI**

* **Summary Cards:**  
  Use card components (styled minimally like the Apple aesthetic) to display each key metric prominently.
* **Interactive Chart Section:**  
  Below or alongside the cards, include interactive charts (using libraries like Chart.js, Plotly, or D3) that update based on user interaction (e.g., filtering by region or time period).
* **Additional Information:**  
  Use tooltips or modal popups to provide more details on each metric, linking back to the dataset documentation (e.g., detailed descriptions from the PDF ).