Chrono Minerals app PSEUDOCODE

PROGRAM: CHRONO MINERALS - African Critical Minerals Platform

MODULE: Application Setup

IMPORT Flask, pandas, plotly, folium, os

CREATE Flask app

SET secret key for sessions

DEFINE login\_required decorator

MODULE: Data Loading

FUNCTION load\_csv\_data(filename):

TRY:

READ CSV file from data folder

CONVERT to list of dictionaries

PRINT success message

RETURN data

EXCEPT FileNotFoundError:

PRINT error message

RETURN empty list

LOAD:

minerals\_data = load\_csv\_data('minerals.csv')

countries\_data = load\_csv\_data('countries.csv')

production\_data = load\_csv\_data('production\_stats.csv')

sites\_data = load\_csv\_data('sites.csv')

users\_data = load\_csv\_data('users.csv')

IF no CSV data loaded, USE sample hardcoded data

MODULE: Authentication System

DEFINE users\_data = [

{username: "admin", password: "admin123", role: "administrator"},

{username: "investor", password: "invest123", role: "investor"},

{username: "researcher", password: "research123", role: "researcher"}

]

MODULE: Role-Based Access Control (RBAC)

DEFINE role\_required(required\_roles) decorator:

CHECK if user logged in

CHECK if user role in required\_roles

IF not authorized, REDIRECT to dashboard

DEFINE role permissions:

ADMIN\_ROLES = ['administrator']

RESEARCHER\_ROLES = ['administrator', 'researcher']

INVESTOR\_ROLES = ['administrator', 'investor']

ALL\_ROLES = all roles

MODULE: Routes

ROUTE: '/'

REDIRECT to login

ROUTE: '/login' [GET, POST]

IF POST request:

GET username and password from form

CHECK against users\_data

IF valid:

SET session user data

REDIRECT to dashboard

ELSE:

SHOW error message

ELSE:

RENDER login template

ROUTE: '/dashboard' [PROTECTED - ALL ROLES]

CALCULATE statistics:

total\_minerals = count minerals

total\_countries = count countries

total\_sites = hardcoded value

avg\_price = average mineral price

RENDER dashboard template with stats

ROUTE: '/minerals' [PROTECTED - RESEARCHERS+ADMIN]

RENDER minerals template with minerals\_data

ROUTE: '/countries' [PROTECTED - ALL ROLES]

RENDER countries template with countries\_data

ROUTE: '/analytics' [PROTECTED - RESEARCHERS+ADMIN]

CREATE charts using plotly:

Chart 1: Mineral Prices Bar Chart

Chart 2: Country Revenue Bar Chart

Chart 3: GDP vs Revenue Scatter Plot

Chart 4: Production Trends Line Chart

Chart 5: Price Distribution Box Plot

Chart 6: Export Value vs Price Scatter

Chart 7: Production vs Price Scatter

Chart 8: Average Export Value Bar Chart

CONVERT charts to HTML

RENDER analytics template with all charts

ROUTE: '/interactive\_map' [PROTECTED - ALL ROLES]

CREATE folium map centered on Africa

ADD country markers with:

location coordinates

popup with GDP and revenue data

red flag icons

ADD mineral deposit circles with:

location coordinates

color-coded by mineral type

popup with production data

CONVERT map to HTML

RENDER map template

ROUTE: '/logout'

CLEAR user session

REDIRECT to login

MODULE: Admin Data Management [PROTECTED - ADMIN ONLY]

ROUTE: '/admin/minerals' [GET, POST]

IF POST request:

CREATE new mineral from form data

ADD to minerals\_data

SAVE updated data to minerals.csv

REDIRECT back to admin minerals

ELSE:

RENDER admin minerals template

ROUTE: '/admin/countries' [GET, POST]

IF POST request:

CREATE new country from form data

ADD to countries\_data

SAVE updated data to countries.csv

REDIRECT back to admin countries

ELSE:

RENDER admin countries template

ROUTE: '/admin/delete\_mineral/<mineral\_id>'

REMOVE mineral from minerals\_data

SAVE updated data to minerals.csv

REDIRECT to admin minerals

ROUTE: '/admin/delete\_country/<country\_id>'

REMOVE country from countries\_data

SAVE updated data to countries.csv

REDIRECT to admin countries

MODULE: Template Features

ALL templates include:

Navigation bar with role-based menu items

Conditional display based on user role

Professional styling with gradients

Responsive design

SPECIFIC templates:

login.html: Authentication form with demo credentials

dashboard\_final.html: Welcome message and feature cards

minerals\_final.html: Mineral cards with images and prices

countries\_final.html: Country profiles with economic data

analytics\_final.html: Grid of interactive charts

map.html: Interactive folium map with legend

admin\_\*.html: Data management forms

MODULE: Image Handling

FOR each mineral in minerals\_data:

ASSIGN image filename based on mineral name

IMAGES stored in static/images/ folder

TEMPLATE displays images with fallback handling

MODULE: Error Handling

TRY-EXCEPT blocks for:

File loading errors

Map creation errors

Authentication failures

Database operations

MODULE: Main Execution

IF \_\_name\_\_ == '\_\_main\_\_':

PRINT startup messages

PRINT demo credentials

RUN Flask app in debug mode

END PROGRAM