**Summary of the Files**

There are 2 main subfolders in the chemical-web-app folder:

1. **Backend**– Has 3 subfolders

* **\_\_pycache\_\_:** Created automatically when running .py files.
* **Database**: This folder has all the chemical csv files. Each file contains different columns and use cases based on the type of data preprocessing that has been performed on them.
* **figs**: AEGL graphs for liquid absorption, flux and vapor absorption, flux are stored here.
* Multiple files
* **aegl\_results.csv**: This file compiles AEGL toxicity thresholds and derived dose kinetics for multiple chemicals for each AEGL level and time. It was generated by running analyzeAllCompounds, which processes each compound’s AEGL values and exports the results into a structured summary table.
* **kr\_predictions\_135.csv**:Output file for kinetics prediction
* **Python\_CWA\_Tool.py**: Original monolithic script containing all the core analysis, data loading and Flask app logic including AEGL and kinetic predictions. This file became too big so I decided to break it down into smaller parts which are mentioned below.
* **app.py**: The Flask API server that handles endpoints for compound analysis, AEGL graphs, summary tables and links together the other modules.
* **analysis\_core.py**: Core scientific engine that computes AEGL dose/flux models, reactivity scoring, kr predictions, absorption graphs and summary table generation
* **aqueous\_code.py**: Code for aqueous absorption/flux AEGL modeling (used when mode="aqueous" in the API)
* **data\_loader.py**: Loads and preprocesses the master chemicals database (Database/combined\_chemicals.csv) into Pandas structures for downstream use
* **pubchem\_utils.py**: Utility functions for PubChem integration; fetches SMILES, CAS, molecular properties, and assigns chemical classes, with fallbacks to local data
* **mo\_chi\_d\_chemical\_csv\_maker\_script.py**: Python script that computes dermal absorption parameters (Mo, D, chi, kp, kg, Ksc, Csat) for each compound based on formulas from the excel, merges them with kr predictions and outputs a new CSV (chemicals\_with\_mo\_chi\_d.csv) without AEGL columns.
* **deconarticle1\_updated.ipynb**: Dermal penetration and decontamination model for chemicals, originally for VX but was modified to take parameters (Mo, D, χ, Ksk, ttrans) from a precomputed CSV (chemicals\_with\_mo\_chi\_d.csv made from mo\_chi\_d\_chemical\_csv\_maker\_script.py) or from user input and runs different numerical models to simulate skin absorption and flux under exposure/decontamination scenarios.
* **requirements.txt**: Libraries that need to be downloaded for the files to run, using “pip install -r requirements.txt”

1. **Frontend** – Has 1 main subfolder

* **components**: stores all the different UI components displayed on frontend
  + **DropdownMenu.tsx**: Provides a dropdown menu UI for selecting chemicals or options.
  + **AEGLButtons.tsx**: Displays buttons to view AEGL values, exposure times, and interactive AEGL graphs in modals.
  + **AnalysisView.tsx**: Main analysis page that toggles between table view, kr graph, and search mode.
  + **CombinedSummaryTable.tsx**: Fetches and renders a searchable summary table of compounds with AEGL values and dose times, with modal details.
  + **GraphModal.tsx**: A reusable modal component for displaying AEGL or other interactive graphs.
  + **AEGLGraph.tsx**: Renders dermal absorption and flux vs time AEGL plots using Plotly.
  + **KrPrediction.tsx**: Card component showing the predicted k value and model confidence for a selected compound.
  + **AEGLModal.tsx**: A generic modal window used for displaying AEGL values and times in a styled overlay.
  + **ReactivityInfo.tsx**: Displays reactivity details of a compound (chemical class, reactive groups, scores, etc.).
  + **CompoundInfo.tsx**: Shows basic compound information such as formula, MW, CAS, class, and SMILES string.
  + **TopBar.tsx**: Displays the dashboard header with the title and a button to toggle between table mode and search mode.
  + **KrGraph.tsx**: Fetches a kr prediction scatter plot from the backend API and renders it using Plotly.
* Multiple files
* **App.css**: Defines the global styles for the frontend, including compound info cards, AEGL graph layouts, switch buttons and top bar styling.
* **App.tsx**: The main React component that renders the AnalysisView, effectively the entry point for your app’s UI logic.
* **main.tsx**: React entry file that mounts the app to the DOM with createRoot, wraps it in StrictMode and loads global CSS (Bootstrap + App.css).

1. **How to run chemical-web-app.docx** – Instructions to run the web app