**1. HTML (index.html)**

This file structures the application into meaningful sections and links external resources (CSS and JavaScript). Here's a breakdown:

**a. Metadata and Resources**

* <meta> ensures proper text encoding and responsiveness.
* <link> connects the CSS file (style.css) for styling.
* <script> connects the JavaScript file (script.js) for interactivity.

**b. Main Sections**

The application has four main sections:

1. **Header Section**
   * Displays the app's title <h1> and the author's name <h3>.
2. **Welcome Section (#welcome-section)**
   * Contains a brief introduction and a **"Proceed to Calculator"** button (<button id="proceed-button">).
   * Clicking the button transitions the user to the **Options Section**.
3. **Options Section (#options-section)**
   * Allows the user to choose between:
     + **Growth/Decay**
     + **Heating/Cooling**
   * Includes a dropdown (<select id="calc-type">) and a **"Start"** button (<button id="start-button">).
4. **Input Section (#input-section)**
   * Dynamically generates input fields based on the selected calculation type (managed by JavaScript).
5. **Solution Section (#solution-section)**
   * Displays the **detailed solution** for the user's problem in an organized format.

**c. Interactivity**

* Sections use classes (.visible and .hidden) to toggle visibility. These are managed via JavaScript.

**2. CSS (style.css)**

The CSS file applies a **modern green-themed design** and ensures responsiveness.

**a. General Styling**

* **Body Styling**:
  + Font: Modern sans-serif (Inter).
  + Background: Light green (#F0F6E8).
  + Text Color: Dark green (#2C3E2C).
* **Container Styling**:
  + Adds a **white card** with rounded corners and a subtle shadow.

**b. Interactive Elements**

* **Buttons**:
  + Default: Bright green (#4CAF50) with uppercase text.
  + Hover: Slightly darker green (#43A047) and a lift effect (transform: translateY(-2px)).
* **Input Fields**:
  + Styled with green borders and background (#F1F8E9).
  + Focus Effect: Bright green outline (#4CAF50) with a shadow.

**c. Responsive Design**

* Adjusts sizes and padding for screens smaller than 768px.
* Ensures buttons, inputs, and containers remain user-friendly on mobile devices.

**d. Solution Styling**

* Highlights the solution area (#solution) with a **light green background** and **monospace font** for readability.

**3. JavaScript (script.js)**

This file handles **logic**, **UI interactivity**, and **calculations**. Key components include:

**a. Utility Functions (utils)**

* **Unit Conversions**:
  + Converts temperatures between Celsius, Fahrenheit, and Kelvin.
  + Normalizes time to hours for calculations.
* **Input Validation**:
  + Ensures all user inputs are numeric and non-empty.

**b. UI Management (UIManager)**

* **Event Listeners**:
  + "Proceed to Calculator" button: Hides the welcome section and shows the options section.
  + "Start" button: Generates input forms dynamically based on the selected calculation type.
* **Dynamic Form Generation**:
  + Uses templates for Growth/Decay and Heating/Cooling forms.
  + Adds dropdowns, text inputs, and buttons dynamically to the DOM.

**c. Calculator Factory (CalculatorFactory)**

* Creates instances of the appropriate calculator (GrowthDecayCalculator or HeatCoolCalculator) based on user selection.

**d. BaseCalculator Class**

* Provides common functionality:
  + **addStep(step)**: Stores a step-by-step explanation of the calculation.
  + **displaySolution()**: Renders the solution on the webpage.

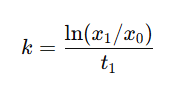
**e. Growth/Decay Calculator**

Implements calculations based on exponential growth and decay. Example methods:

1. **findAmount()**: Determines the value of x(t) at a specific time.
2. **findInitialValue()**: Solves for the initial value x.
3. **findTime()**: Calculates the time required to reach a target amount.

**Example Steps for findAmount()**:

1. **Solve for growth rate K:**



1. Calculate the amount x(t2):



**f. Heating/Cooling Calculator**

Uses Newton's Law of Cooling/Heating to perform calculations. Example methods:

1. **findTemp()**: Determines the temperature at a specific time.
2. **findInitialTemp()**: Calculates the initial temperature.
3. **findTime()**: Solves for the time required to reach a target temperature.

**Example Steps for findTemp()**:

1. **Convert all temperatures to Kelvin**.
2. **Normalize time units** to hours.
3. **Calculate the cooling rate K:**



1. Solve for T(t2):  
   

**How It All Works Together**

1. The user starts on the **Welcome Section** and proceeds to select a calculation type.
2. Based on their selection, a dynamic form is generated in the **Input Section**.
3. The user enters their data, and the relevant calculator performs the computation.
4. The detailed, step-by-step solution is displayed in the **Solution Section**.