Spec from o3  
  
Below is a detailed specification document that you can use to instruct a large language model (LLM) to create (or improve) a single‑file, web‑based Early Retirement Monte Carlo Simulation program. This document covers the functional requirements, simulation logic, UI elements, external dependencies, persistence of parameters, and potential improvements.

**1. Overall Objective**

Create a self‑contained HTML file that implements a Monte Carlo simulation for early retirement planning. The application should allow users to adjust simulation parameters, run multiple simulation scenarios, and visualize the results in a dual‑axis chart. The program must include a way to save and load simulation parameters persistently (using the browser’s localStorage).

**2. Technical Requirements**

**2.1 File Structure and External Dependencies**

* **Single HTML File:** All HTML, CSS, and JavaScript must reside in one file.
* **CSS Framework:** Use Bootstrap (via CDN) for styling the UI.
* **Charting Library:** Use Chart.js (via CDN) for data visualization.
* **Local Storage:** Use the browser’s localStorage API for persisting simulation parameters between sessions.

**2.2 Programming Style**

* **Modular Code:** Organize JavaScript into functions (e.g., for reading parameters, running the simulation, calculating percentiles, updating charts, saving/loading parameters).
* **Commenting and Readability:** Include inline comments to explain the logic, especially within the simulation functions.
* **Real‑Time Updates:** All changes to inputs should immediately re‑run the simulation and update the visualizations.

**3. Input Parameters**

The program must include the following parameter groups, each editable via UI elements:

**3.1 Fixed Parameters**

* **Current Age:**
  + Type: Number
  + Default: 54
* **Legal Retirement Age:**
  + Type: Number
  + Default: 67
* **Fixed Monthly Pension:**
  + Type: Number (in €)
  + Default: 5000
* **Current Assets:**
  + Type: Number (in €)
  + Default: 600000
* **Capital Gains Tax Rate:**
  + Type: Number (percentage)
  + Default: 26.25%
* **Annual Savings (while working):**
  + Type: Number (in €)
  + Default: 18000

**3.2 Variable Parameters**

* **Intended Retirement Age:**
  + Input Type: Range slider
  + Range: 55–66
  + Default: 60
  + (Also display the numeric value beside the slider.)
* **Average ROI:**
  + Input Type: Range slider (as a percentage)
  + Range: 5%–11%
  + Default: 7%
* **Average Inflation:**
  + Input Type: Range slider (as a percentage)
  + Range: 2%–5%
  + Default: 3%

**3.3 Volatility Parameters**

* **ROI Volatility:**
  + Type: Number (decimal, absolute standard deviation; e.g., 0.02 means 2 percentage points)
  + Default: 0.02
* **Inflation Volatility:**
  + Type: Number (decimal, absolute standard deviation; e.g., 0.005 means 0.5 percentage points)
  + Default: 0.005

**3.4 Expense Categories**

**Monthly Expenses (in €)**

* **Health:** Default 1500
* **Food:** Default 1500
* **Entertainment:** Default 500
* **Shopping:** Default 500
* **Utilities:** Default 500

**Annual Expenses (in €)**

* **Vacations:** Default 12000
* **Repairs:** Default 2000
* **Car Maintenance:** Default 1500

**3.5 Simulation Control**

* **Number of Simulation Runs:**
  + Type: Number
  + Default: 1000
* **Simulation End Age:**
  + Fixed at 90 (hard‑coded)

**4. Simulation Logic**

**4.1 Simulation Phases**

* **Accumulation Phase:**
  + Simulate from the current age until the intended retirement age.
  + Each year add the annual savings and apply a random ROI (drawn from a normal distribution with the specified average ROI and ROI volatility).
  + For positive ROI, apply capital gains tax.
* **Distribution Phase:**
  + Simulate from the intended retirement age until the simulation end age (90).
  + For each year, subtract the annual expenses (calculated from the sum of monthly expenses \* 12 plus the annual expenses) from the assets.
  + If the age is at or beyond the legal retirement age, add the fixed monthly pension (times 12).
  + Apply a random ROI (with tax considerations as in accumulation) to the assets.
  + Update the annual expenses each year using a random inflation rate drawn from a normal distribution with the specified average inflation and inflation volatility.
  + Record the monthly spending for each year (annual expense divided by 12) for visualization.

**4.2 Failure Detection**

* A simulation run is considered a "failure" if assets drop below zero at any time.
* The overall "success probability" is the percentage of simulation runs that do not fail.

**4.3 Random Number Generation**

* Use the Box–Muller transform to generate normally distributed random numbers for ROI and inflation rate.

**4.4 Percentile Calculation**

* For each simulated year (across all simulation runs), compute:
  + 10th percentile (worst-case scenario)
  + Median (50th percentile)
  + 90th percentile (best-case scenario)
* Compute these percentiles separately for asset values and monthly spending.

**5. Visualization**

**5.1 Combined Chart**

* Use Chart.js to create a combined chart with dual y‑axes.
  + **X‑axis:** Age (from current age to simulation end age)
  + **Left Y‑axis (Assets):**
    - Display asset trajectories as line charts representing the 10th, median, and 90th percentiles.
  + **Right Y‑axis (Monthly Spending):**
    - Display monthly spending values as bar charts for the 10th, median, and 90th percentiles.
* Ensure that the tooltips and legends clearly differentiate between asset lines and spending bars.

**6. Save/Load Functionality for Parameters**

**6.1 Requirements**

* **Save Function:**
  + Provide a "Save Parameters" button that gathers all current parameters into a JSON object.
  + Display the JSON string in a dedicated text area.
  + Save this JSON string into the browser’s localStorage (using a predefined key such as "savedParameters").
* **Load Function:**
  + Provide a "Load Parameters" button that retrieves the JSON string from localStorage.
  + Parse the JSON, update all the UI input elements with the saved values, and re‑run the simulation.
  + Optionally, allow manual editing of the JSON in the text area and load those parameters as well.
* **Persistence:**
  + On page load, check localStorage for saved parameters. If found, automatically load them into the UI and update the simulation accordingly.

**7. Additional Improvements and Considerations**

* **Error Handling:**
  + Validate JSON when loading parameters. If invalid, display an alert or error message.
  + Provide user feedback if no parameters are found in localStorage.
* **Responsive Design:**
  + Ensure the UI and charts are responsive and display correctly on various screen sizes.
* **Extensibility:**
  + Consider adding additional simulation scenarios (e.g., sensitivity analysis, alternative retirement incomes, etc.).
  + Allow for customization of the simulation end age or include other retirement-specific factors.
* **Performance:**
  + Optimize the simulation logic to handle a high number of runs without lag.
  + Consider throttling input events if real‑time updates are computationally intensive.

**8. Summary of User Interface Layout**

* **Left Panel (Parameters):**
  + Sections for Fixed Parameters, Variable Parameters, Volatility Parameters, Expense Categories (Monthly and Annual), Simulation Control (number of runs), and Save/Load functionality.
* **Right Panel (Results):**
  + A combined Chart.js chart displaying asset value trajectories (lines) and monthly spending (bars) over age.
  + A text area or label displaying the success probability.
  + A note explaining that inflation is taken into account.

This detailed specification covers all aspects of the current program and suggests ways to persist settings (via localStorage) and improve functionality. You can provide this document to an LLM as instructions to either generate the current program from scratch or enhance its functionality further.

I want a monte carlo simluation for (potential early retirement). Modifiable input values with default range (inbrackets if applicable) are the following:

Current age: 54

Retirement age be law: 67 (this is the age from when a fixed rent is paid)

Fixed rent: 5000

Current (today’s) assets: 600000

Intended retirement age: (55-66)

Annual additional savings during accumulation phase (meaning Intended retirement age - Current age): 18000

Taxation of capital gains: 26.25%

Average ROI: (5%-11%)

Average YOY inflation: (2%-5%)

A Spending section; inputs for a) monthly expenses

Health, food, entertainment, shopping, utilities

b) yearly

vacations, repairs, car

The aim is to visually simulate the development of the total asset value taking into account covering expenses – whereby unspent expenses and ROI shall be added back to the total assets in order to find the sweet spot for early retirement considering various scenarios (i.e. best and worst cases and in between).  
The calculations should automatically rerun on parameter changes.

Also consider a meaning ful compact UI. The “fixed” parameters should be editable as well.

The simulation results should also be presented in a meaningful graphical way.

Put the code in a single HTML/CSS/JS file. You may however reference external CSS and JS code libraries if applicable for the tasks.

Claude:  
  
# Early Retirement Monte Carlo Simulation Specification

Create a web-based Monte Carlo simulation for early retirement planning that helps users analyze different retirement scenarios. The application should be contained in a single HTML file with embedded CSS and JavaScript, with the ability to reference external libraries as needed.

## Core Requirements

### Input Parameters

The simulation should accept the following input parameters, with specified defaults and ranges where applicable:

#### Fixed Parameters

- Current Age: 54 years

- Legal Retirement Age: 67 years (when fixed pension begins)

- Fixed Monthly Pension: €5,000

- Current Assets: €600,000

- Capital Gains Tax Rate: 26.25%

- Annual Savings During Work: €18,000 (during accumulation phase)

#### Variable Parameters with Ranges

- Intended Retirement Age: Range [55-66] years

- Average Return on Investment (ROI): Range [5%-11%]

- Average Year-over-Year Inflation: Range [2%-5%]

#### Expense Categories

Monthly Expenses:

- Health

- Food

- Entertainment

- Shopping

- Utilities

Annual Expenses:

- Vacations

- Repairs

- Car Maintenance

### Simulation Requirements

1. Asset Value Calculation

- Track total asset development over time

- Account for all expenses

- Reinvest unspent funds and investment returns

- Apply capital gains tax to investment returns

- Factor in inflation on expenses

2. Dynamic Behavior

- Automatically recalculate on any parameter change

- Consider accumulation phase (current age to retirement)

- Consider distribution phase (retirement to end of simulation)

3. Scenario Analysis

- Generate multiple scenarios (best case, worst case, median case)

- Help identify optimal early retirement timing

- Consider various market and inflation conditions

### Visualization Requirements

1. Interactive Dashboard

- Compact, intuitive user interface

- Clear input sections for all parameters

- Real-time updates

2. Results Display

- Clear graphical representation of asset development

- Show probability of successful retirement

- Display key metrics and statistics

- Compare different scenarios visually

### Technical Implementation

1. File Structure

- Single HTML file containing all code

- Embedded CSS for styling

- Embedded JavaScript for logic

2. Allowed External Dependencies

- May use CSS frameworks for styling

- May use JavaScript libraries for:

- Data visualization

- Mathematical calculations

- UI components

- Statistical analysis

3. Performance Considerations

- Efficient calculation methods

- Smooth user interface updates

- Responsive design for different screen sizes

## Expected Outputs

The simulation should provide:

1. Projected asset values over time

2. Success probability for different retirement ages

3. Impact of different market conditions

4. Effect of expense changes on retirement viability

5. Clear visualization of outcomes

## Success Criteria

The implementation will be considered successful if it:

1. Accurately models retirement scenarios

2. Provides clear, actionable insights

3. Responds smoothly to parameter changes

4. Presents results in an easily understandable format

5. Maintains performance with multiple simultaneous calculations