Technical Specification for Bitcoin (BTC) Retirement Calculator with Google Chrome Extension

Overview

The Bitcoin Retirement Calculator is a web-based application designed to help users plan their retirement using Bitcoin. The tool will allow users to input their financial details and receive real-time updates on their retirement portfolio value, helping them determine if their BTC holdings are sufficient for their retirement goals. The application will be mobile responsive, ensuring a seamless user experience on both desktop and mobile devices. Additionally, a Google Chrome extension will provide easy access to the calculator directly from the browser.

Figma design

https://www.figma.com/design/8aI16e0hGsgAOsVPQvVXlV/%F0%9F%92%B0-BTC-Retirement-Calculator?node-id=1-5&t=AJZYxiveKMMM1m9h-0

User Input Fields

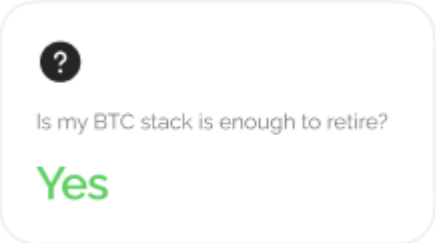
1. **Your Current Age**
   - Type: Integer
   - Validation: Positive integer, greater than 0
2. **Your Current BTC Stack**
   - Type: Float
   - Validation: Positive number, greater than 0
3. **Your Ideal Monthly Expenses During Retirement**
   - Type: Float
   - Validation: Positive number, greater than 0
4. **Your Monthly Contribution**
   - Type: Float
   - Validation: Positive number, greater than 0
5. **Retirement Age**
   - Type: Integer
   - Default Value: 65
   - Validation: Positive integer, greater than current age
6. **Life Expectancy Age**
   - Type: Integer
   - Default Value: 85
   - Validation: Positive integer, greater than retirement age
7. **Annual Inflation Rate**
   - Type: Float
   - Default Value: 5%

        - Validation: Percentage value (0-100)
    8. **Forecast BTC Price**
        - Type: String (with predefined options)
        - Default Value: "Power Law"
        - Options:
            i. Power Law
            ii. Linear Growth (Hidden)
            iii. Exponential Growth (Hidden)

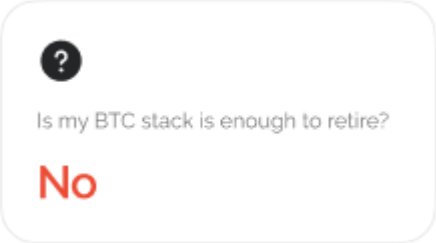
Dashboard View

The dashboard will update immediately when the user updates any of the input fields. It will display the following:

1. **Current BTC Portfolio Value**
    - Formula:
      `Current BTC Stack * Current BTC Price in USD`
    - Example: If the current BTC stack is 1 BTC and the current BTC price is $30,000, the portfolio value will display $30,000.
2. **Portfolio Balance at Age 65**
    - Formula:
      `(Current BTC Stack + SUM(Monthly BTC Contributions until Retirement)) * Forecasted BTC Price at Retirement`
    - Calculation Steps:
        - Determine the number of months to retirement:
          `(Retirement Age - Current Age) * 12`
        - Calculate monthly BTC contribution based on the forecasted BTC price for each month.
        - Sum the monthly contributions in BTC until the retirement age.
        - Add the current BTC stack to the total contributions.
        - Multiply the result by the forecasted BTC price at the retirement age.
3. **Is My BTC Stack Enough to Retire?**
    - Formula:
      `RetirementAmount = ((Ideal Monthly Expenses during Retirement * (1 + Annual Inflation Rate)^(Life Expectancy Age - Retirement Age))* 12) * (Life Expectancy Age - Retirement Age)`
    - Logic:
        - If `RetirementAmount <= Portfolio Balance at Age 65`, display "Yes".

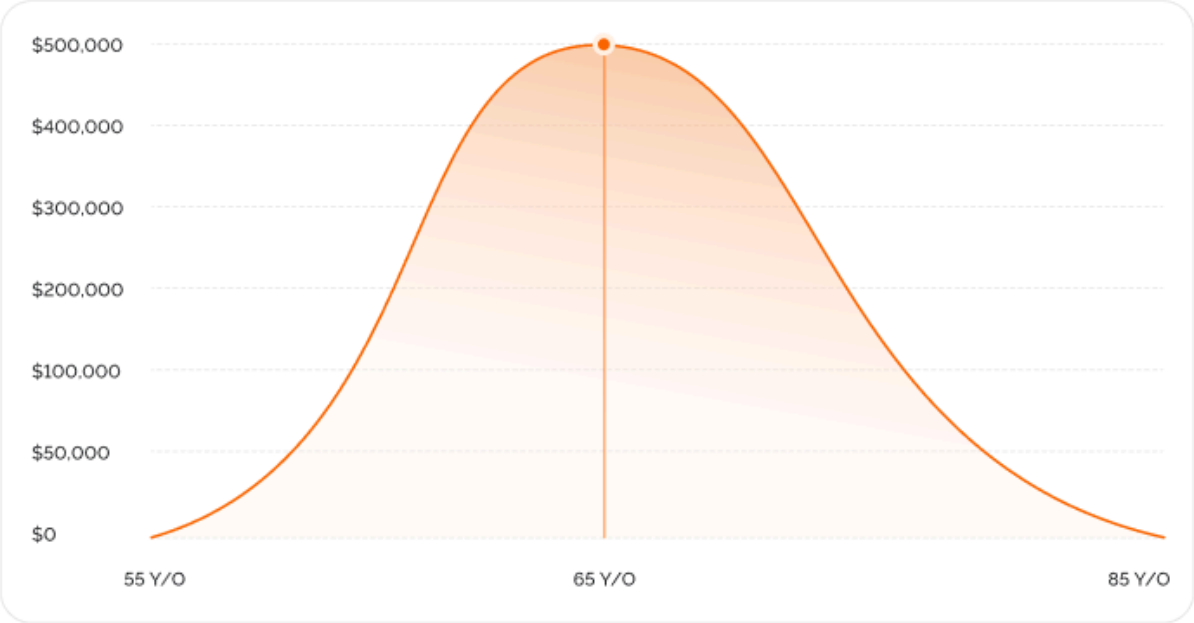


- If `RetirementAmount > Portfolio Balance at Age 65`, display "No".



- Retirement

4. **BTC Stack Growth and Usage Graph**



- **Graph Details**:
  - X-axis: Age (from current age to life expectancy age)
  - Y-axis: BTC Value in USD
  - Two lines:
    1. **BTC Stack Growth**: Displays the increment of the BTC stack up to the retirement age.
    2. **BTC Stack Usage**: Displays the increase of the BTC stack up to retirement age and the decrease of the BTC stack post-retirement based on the ideal monthly expenses during retirement.

Footer View

Content:

Made with 💛 by SaasCraft Ventures Suggest our next feature 📢

- Made with 💛 by SaasCraft Ventures **(Left aligned)**
- Suggest our next feature 📢 **(Right aligned)**
    - Link to canny.io

Forecast BTC Price Calculation Methods

1. **Power Law**: Uses a power law distribution to project future BTC prices. Reference: Bitcoin Power Law Corridor

Technology Stack

- **Frontend**: HTML, CSS, JavaScript (React.js or Vue.js)
- **Backend**: Node.js with Express
- **APIs**:
    - Real-time BTC price feed API (e.g., CoinGecko)
- **Hosting**: Vercel
- **Version Control and CI/CD**: GitLab

Mobile Responsiveness

- **Responsive Design**: Utilise CSS frameworks like Bootstrap or Tailwind CSS to ensure the application is mobile responsive.
- **Media Queries**: Implement media queries to adjust the layout and elements based on the screen size.
- **Touch-Friendly UI**: Ensure buttons, inputs, and other interactive elements are easily tappable on touchscreens.
- **Performance Optimization**: Optimise images and other assets for faster loading times on mobile networks.

User Interface

- **Input Form**: A simple form where users can input and update their financial details. Designed to be user-friendly on both desktop and mobile.
- **Real-time Dashboard**: Displays the calculated values based on user input. Includes:
    - Current BTC Portfolio Value
    - Portfolio Balance at Age 65
    - Retirement sufficiency status
    - BTC Stack Growth and Usage Graph
- **Responsive Layout**: Ensures the dashboard adapts to different screen sizes, maintaining usability on both desktop and mobile devices.

Google Chrome Extension

- **Extension Popup**: A mini version of the retirement calculator will be accessible directly from the Chrome toolbar.
- **Synchronisation**: User inputs and calculated values will sync between the web application and the extension.
- **Quick Access**: Users can quickly input data and see their portfolio status without navigating away from their current tab.
- **Notification Alerts**: The extension can send notifications for significant portfolio changes or milestones.

GitLab Integration

- **Version Control**: Use GitLab for version control, enabling collaboration among developers.
- **CI/CD Pipelines**: Set up GitLab CI/CD pipelines to automate testing, building, and deployment processes.
- **Issue Tracking**: Use GitLab's issue tracking features to manage bugs, enhancements, and new feature requests.
- **Merge Requests**: Implement merge requests for code reviews and to ensure high code quality before merging changes into the main branch.

Deliverables

1. Bug-Free Product

The final product will be thoroughly tested to ensure it is free of bugs and operates as intended. This includes:

- **Functionality Testing**: Verify that all features, including input fields, real-time calculations, and the graph, work correctly.
- **Performance Testing**: Ensure the application loads quickly and performs efficiently on both desktop and mobile devices.
- **Usability Testing**: Confirm that the user interface is intuitive and user-friendly.

2. Documentation

Comprehensive documentation will be provided, covering the following aspects:

1. **User Guide**
    - **Introduction**: Overview of the Bitcoin Retirement Calculator and its purpose.
    - **Getting Started**: Instructions on how to access and use the calculator, both via the web application and the Chrome extension.
    - **Input Fields**: Detailed explanation of each input field and its required data.
    - **Dashboard Features**: Description of the dashboard elements, including the current BTC portfolio value, portfolio balance at age 65, retirement sufficiency status, and the BTC stack growth and usage graph.
    - **Troubleshooting**: Common issues and their solutions.

2. **Developer Guide**
    - **Project Structure**: Overview of the project's architecture, including frontend and backend components.
    - **Setup Instructions**: Steps to set up the development environment, including dependencies, and configuration.
    - **API Documentation**: Detailed documentation of the APIs used for real-time BTC prices
    - **Codebase Overview**: Explanation of the key modules and their functions.
    - **Testing Procedures**: Guidelines for running unit tests, integration tests, and end-to-end tests.
    - **CI/CD Pipelines**: Instructions on how to use GitLab CI/CD pipelines for testing and deployment.
3. **API Documentation**
    - **Endpoint Descriptions**: List of all API endpoints with their methods, parameters, and responses.
    - **Error Handling**: Common errors and their meanings.

Warranty

A warranty will be provided for the product, ensuring its reliability and usability. The warranty terms are as follows:

1. **Duration:** The warranty will be valid for **6 months** from the date of delivery.
2. **Coverage:**
    - **Bug Fixes:** Any bugs or issues reported within the warranty period will be promptly addressed and resolved.
    - **Performance Issues:** Any performance-related issues will be investigated and optimised to ensure the application runs smoothly.
3. **Exclusions:**
    - **Third-party Integrations:** Issues caused by third-party APIs or services are not covered under this warranty.
    - **User-caused Issues:** Problems arising from incorrect use or unauthorised modifications of the code are not covered.
4. **Support:** During the warranty period, support will be provided for any queries or issues related to the usage of the application.