Banker's Algorithm Visual Simulator

This project is a visual simulator for the Banker's Algorithm, a deadlock avoidance algorithm used in operating systems. 🚀 It allows users to input process allocation, maximum resource needs, and available resources, then visualizes the algorithm's execution to determine if the system is in a safe state. This tool is designed for educational purposes, helping students and professionals understand the Banker's Algorithm through interactive simulation.

Table of Contents

Features

- **Interactive Simulation:** Visual representation of the Banker's Algorithm in action. 🎮
- **Input Configuration:** Allows users to define the number of processes and resources, and their allocation and maximum needs. ⌨️
- **Safe State Validation:** Determines whether the system is in a safe state and displays the safe sequence. ✅
- **Dynamic Process Addition:** Supports adding new processes during simulation to observe the impact on system safety. ➕
- **Step-by-Step Execution:** Provides controls to step through the algorithm execution, making it easier to understand. 🚶
- **Visualization:** Uses a canvas to render the processes and their states. 🎨
- **Export and Screenshot:** Features to export simulation traces and take screenshots for documentation. 📤
- **Speed Control:** Ability to adjust the simulation speed. ⏱️
- **Process Status Indicators:** Clear visual indicators for process states (waiting, executing, finished, unsafe). 🚦

Tech Stack

- **Frontend:**
  - React - A JavaScript library for building user interfaces. ⚛️
  - TypeScript - A typed superset of JavaScript that compiles to plain JavaScript. 🟦
  - Vite - A build tool that aims to provide a faster and leaner development experience for modern web projects. ⚡
  - Tailwind CSS - A utility-first CSS framework for rapidly building custom designs. 🎨
  - Lucide React - Beautifully simple, pixel-perfect icons. 🔆
- **Utilities:**
  - ESLint - A tool for identifying and reporting on patterns found in ECMAScript/JavaScript code. ⚙️
  - PostCSS - A tool for transforming CSS with JS plugins.
- **Dependencies**
  - @supabase/supabase-js
- **Development Dependencies**
  - globals, typescript-eslint, autoprefixer,

Installation 🛠️

1. **Clone the repository:**

```
git clone https://github.com/rohit29032005/23BCE0052_OS_ROHIT.git
cd 23BCE0052_OS_ROHIT
```

2. **Install the dependencies:**

```
npm install
```

Usage 🚀

1. **Start the development server:**

```
npm run dev
```

2. Open your browser and navigate to the address provided in the console (usually `http://localhost:5173/`). 🌐
3. **Configure the Simulation:**
   - Enter the number of processes and resources.
   - Define the Allocation and Max matrices for each process.
   - Set the Available vector for system resources.

   Example:

Process	Allocation (R1, R2, R3)	Max (R1, R2, R3)
P0	0, 1, 0	7, 5, 3
P1	2, 0, 0	3, 2, 2
P2	3, 0, 2	9, 0, 2

Available: 10, 5, 7

4. **Run the Simulation:**

- Click the "Run Simulation" button to start the algorithm.

5. **Control the Simulation:**

- Use the Play, Pause, Step Forward, and Step Backward buttons to control the simulation flow.
- Adjust the speed using the speed slider.
- Add new processes using the "Add Process" button.

6. **Observe Results:**

    - The canvas visualization will display the processes and their states.
    - The statistics panel shows the Available vector, Need matrix, and process status.
    - The Safe Sequence section indicates whether the system is safe and displays the safe sequence.

How to use ⚙️

This project can be used to visually understand the Banker's Algorithm, a crucial concept in operating systems for deadlock avoidance. Use the simulator to:

- Experiment with different allocation scenarios to see how they affect system safety.
- Observe how the algorithm determines a safe sequence of process execution.
- Add new processes dynamically and observe how the system responds.
- Export simulation traces and screenshots for documentation or presentations.

Real-world use cases

- **Educational Tool:** To visualize and understand the Banker’s Algorithm in operating systems.
- **Resource Allocation Simulation:** To simulate and manage resource allocation in complex systems, ensuring system safety.
- **Deadlock Avoidance:** To analyze and prevent potential deadlocks by dynamically adding processes and monitoring resource availability.

Project Structure 📂

```
23BCE0052_OS_ROHIT/
├── .bolt/
│   └── config.json
│   └── prompt
├── src/
│   ├── components/
│   │   ├── BankerSimulator.tsx
│   │   ├── CanvasVisualization.tsx
│   │   ├── ControlPanel.tsx
│   │   ├── Header.tsx
│   │   ├── InputPanel.tsx
│   │   └── StatisticsPanel.tsx
│   ├── types/
│   │   └── index.ts
│   ├── utils/
│   │   ├── bankerAlgorithm.ts
│   │   └── canvasRenderer.ts
│   ├── App.tsx
│   ├── index.css
│   ├── main.tsx
│   └── vite-env.d.ts
├── .eslintignore
├── eslint.config.js
├── index.html
├── package.json
├── postcss.config.js
├── README.md
├── tailwind.config.js
├── tsconfig.app.json
├── tsconfig.json
├── tsconfig.node.json
└── vite.config.ts
```

- `src/`: Contains the source code.
  - `components/`: React components for the UI.
  - `types/`: TypeScript interfaces.
  - `utils/`: Utility functions for the Banker's Algorithm and canvas rendering.
  - `App.tsx`: Main application component.
  - `index.css`: CSS styles.
  - `main.tsx`: Entry point for the React application.

- `index.html`: Main HTML file.
- `package.json`: Lists project dependencies and scripts.
- `vite.config.ts`: Configuration file for Vite.
- `tsconfig.json`: Configuration file for TypeScript.
- `tailwind.config.js`: Configuration file for Tailwind CSS.
- `eslint.config.js`: Configuration file for ESLint.
- `postcss.config.js`: Configuration file for PostCSS.

Contributing 🙌

Contributions are welcome! Here are the steps to contribute:

1. Fork the repository.
2. Create a new branch with a descriptive name.
3. Make your changes and commit them with clear messages.
4. Push your changes to your forked repository.
5. Submit a pull request to the main repository.

License 📝

This project is **unlicensed**.

Important links 🔗

- Repository: https://github.com/rohit29032005/23BCE0052_OS_ROHIT

Footer 👣

23BCE0052_OS_ROHIT • rohit29032005 • Contact

⭐ Fork and like the repository! ⭐

Generated by ReadmeCodeGen