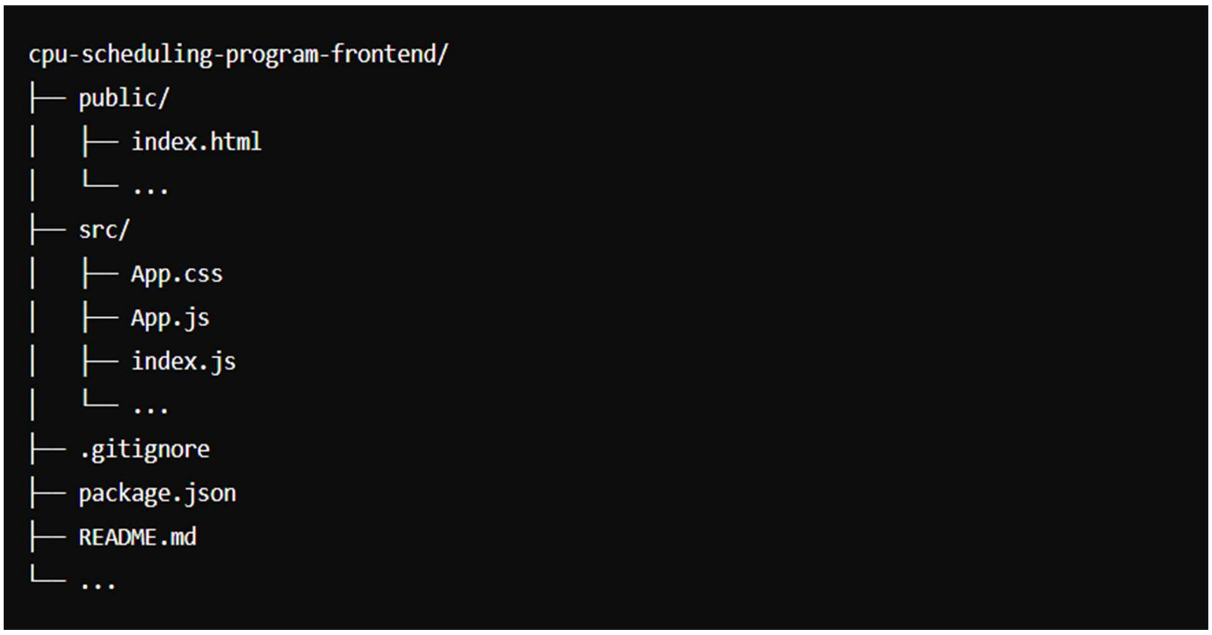
CPU Scheduling Program Frontend

This project is a frontend for a CPU Scheduling program built using React. The application allows users to input processes with their arrival time, burst time, and priority, and specify a time quantum period. The user can then run the scheduling algorithm to see the output. The frontend is styled using CSS and communicates with a backend server to execute the scheduling algorithm.

Features

- Add process details (arrival time, burst time, priority).
- Set the time quantum period.
- View the list of added processes in a table format.
- Run the scheduling algorithm and display the output.
- Reset the input fields and process list.
- Error handling for invalid inputs and server communication issues.

File Structure



```
cpu-scheduling-program-frontend/
├── public/
│   ├── index.html
│   └── ...
├── src/
│   ├── App.css
│   ├── App.js
│   ├── index.js
│   └── ...
├── .gitignore
├── package.json
├── README.md
└── ...
```

`App.js`

The main component that contains the application logic:

- Uses React hooks (`useState`) to manage state.
- Handles form submissions to add processes.
- Validates input data.
- Communicates with the backend server to execute the scheduling algorithm.

- Displays error messages and output results.

`App.css`

The CSS file for styling the application:

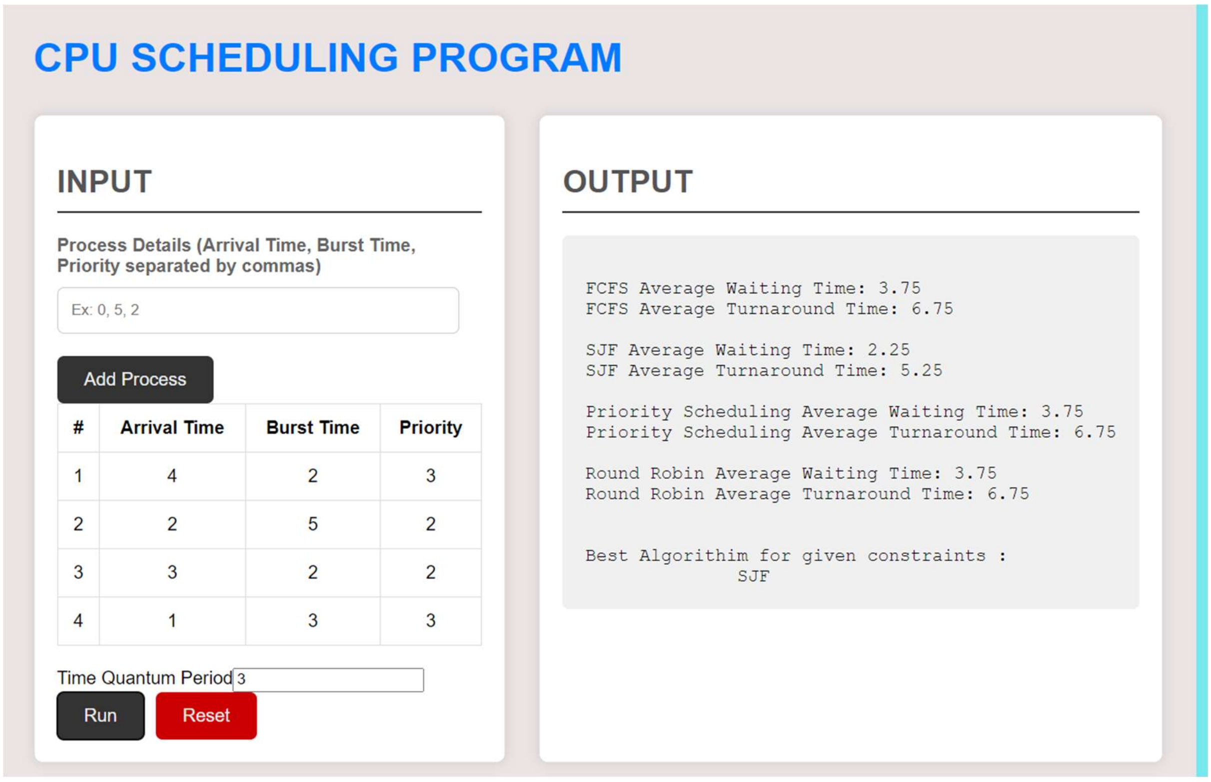
- Provides a modern and clean look.
- Uses Flexbox for layout.
- Styles form elements, buttons, tables, and error messages

Usage

1. Enter the process details (arrival time, burst time, priority) separated by commas in the input field and click "Add Process".
2. Add more processes as needed.
3. Enter the time quantum period in the specified input field.
4. Click "Run" to execute the scheduling algorithm.
5. View the output in the output section.
6. Click "Reset" to clear all inputs and output.

Example

1. Add process details: `0, 5, 2` and click "Add Process".
2. Add another process: `2, 3, 1` and click "Add Process".
3. Set the time quantum period: `4`.
4. Click "Run" to see the output.



#	Arrival Time	Burst Time	Priority
1	4	2	3
2	2	5	2
3	3	2	2
4	1	3	3

```
FCFS Average Waiting Time: 3.75
FCFS Average Turnaround Time: 6.75

SJF Average Waiting Time: 2.25
SJF Average Turnaround Time: 5.25

Priority Scheduling Average Waiting Time: 3.75
Priority Scheduling Average Turnaround Time: 6.75

Round Robin Average Waiting Time: 3.75
Round Robin Average Turnaround Time: 6.75


Best Algorithim for given constraints :
              SJF
```

Error Handling

- Displays an error message if the number of processes exceeds 10.
- Validates the format of process input.
- Ensures the time quantum is a positive number.
- Handles server communication errors.

1. .

Contributing

If you would like to contribute to this project, please follow these steps:

1. Fork the repository.
2. Create a new branch: `git checkout -b feature/your-feature-name`.
3. Make your changes.
4. Commit your changes: `git commit -m 'Add some feature'`.
5. Push to the branch: `git push origin feature/your-feature-name`.
6. Open a pull request.
