Frontend Development Steps (React + MUI + ThreeJS) for NASA Mars Weather App

1. Set Up React App:

- Use Create React App: npx create-react-app mars-weather-app

- Alternatively, use Vite for faster builds.

- Navigate into the project directory: cd mars-weather-app

2. Install Required Libraries:

- Material UI: npm install @mui/material @emotion/react @emotion/styled

- Axios for HTTP requests: npm install axios

- Three.js and React Three Fiber: npm install three @react-three/fiber @react-three/drei

3. Set Up Environment Variables:

- Create a .env file in the root of your project.

- Example: REACT_APP_API_BASE_URL=http://localhost:8080

- Access it in code via process.env.REACT_APP_API_BASE_URL

4. Build Components:

- Create components like WeatherCard, MarsScene, Header, etc.

- Use MUI components for consistent design (Cards, Typography, Grid).

5. Fetch Data from Backend:

- Use axios to fetch weather data from Spring Boot backend.

- Example:

  axios.get(`${process.env.REACT_APP_API_BASE_URL}/weather`)

6. Display Weather Information:

- Use MUI Cards or Tables to present the data.

- Optionally add charts using a library like Chart.js or Recharts.

7. Create 3D Visualization (Optional):

- Use react-three-fiber and drei to render Mars or atmospheric visuals.

- Integrate inside a component like <Canvas> in MarsScene.js

8. Add Delete Operation (If applicable):

- If backend supports DELETE, add a button to trigger axios.delete().

9. Handle Loading and Error States:

- Show loading indicators using MUI CircularProgress.

- Handle API errors gracefully.

10. Style and Polish UI:

- Use MUI theming to match app branding.

- Use responsive layout techniques with MUI Grid/Flexbox.

11. Test and Build:

- Test functionality across browsers.

- Run build for production: npm run build