Jet Francis Q. Podaca
Jayvi M. Mandalihan
MIT 221 Advanced Web Systems and Technologies

Output #4: Web Application with State Management

Documentation Report

A. STRUCTURE AND COMPONENTS

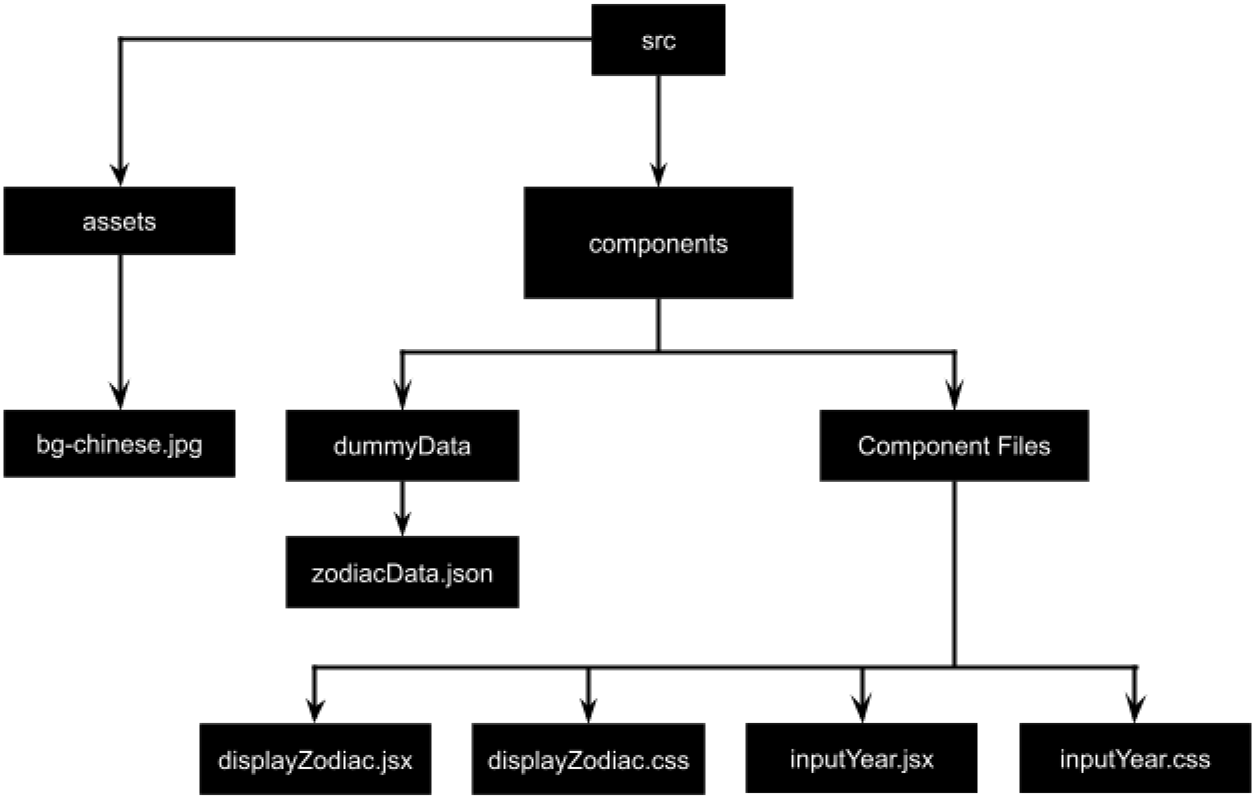
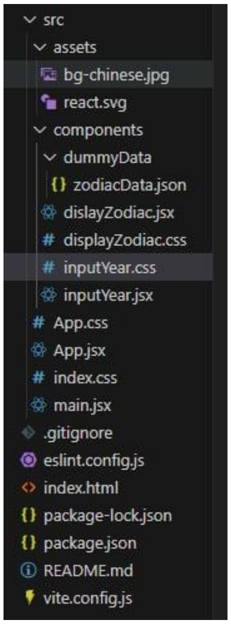


Figure 1. Chinese Zodiac Classifier Web App Structure



```
src
  assets
    bg-chinese.jpg
    react.svg
  components
    dummyData
      zodiacData.json
    dislayZodiac.jsx
    displayZodiac.css
    inputYear.css
    inputYear.jsx
  App.css
  App.jsx
  index.css
  main.jsx
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js
```

Figure 2. Screenshot of the directories of the
Chinese Zodiac Classifier Web App

Brief description of each directory:

1. Assets Directory
   bg-chinese.jpg: A background image for the application featuring Chinese zodiac theme

2. Components Directory
   dummyData/
   zodiacData.json: Contains the mapping of years to zodiac animals and their characteristics

   Main Components
   displayZodiac.jsx: The component responsible for showing the zodiac animal and its characteristics
   displayZodiac.css: Styling for the displayZodiac component
   inputYear.jsx: The component handling user input for birth year
   inputYear.css: Styling for the inputYear component

   Additional Components
   App.jsx: Main application component that stages other components
   App.css: Global application styles
   index.css: Root-level styles
   main.jsx: Application entry point

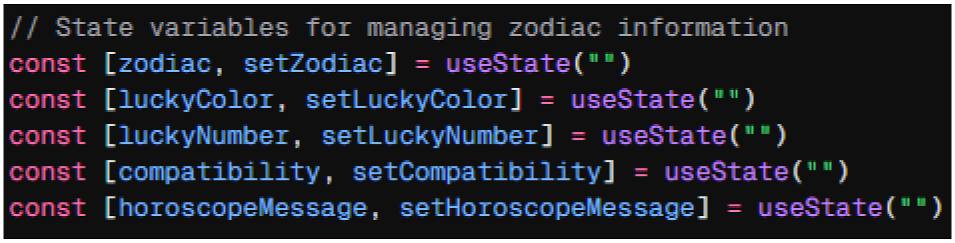
3. Configuration Files
   vite.config.js: Vite build tool configuration
   package.json: Project dependencies and scripts
   eslint.config.js: ESLint configuration for code quality
   index.html: Main HTML file
   .gitignore: Git ignore rules

In conclusion, the application follows a modular structure where:

- User input is handled by the **inputYear** component
- Results are displayed through the **displayZodiac** component
- Data processing uses the **zodiacData.json** for classifications

B. THE STATE AND CONDITIONAL RENDERING

1. State Management in **App.jsx**

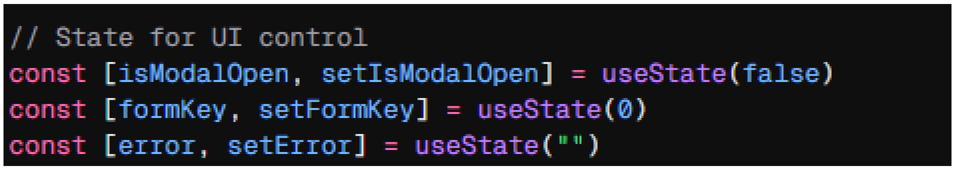


```
// State variables for managing zodiac information
const [zodiac, setZodiac] = useState("")
const [luckyColor, setLuckyColor] = useState("")
const [luckyNumber, setLuckyNumber] = useState("")
const [compatibility, setCompatibility] = useState("")
const [horoscopeMessage, setHoroscopeMessage] = useState("")
```

Figure 3. useState Rendering for managing zodiac characteristics

These lines create *state variables*, e.g., *zodiac*, and their *setter functions*, e.g., *setZodiac*, for storing zodiac-related information:

- **zodiac:** Stores the determined zodiac sign
- **luckyColor:** Holds the lucky color for the zodiac
- **luckyNumber:** Contains the lucky number
- **compatibility:** Stores compatible zodiac signs
- **horoscopeMessage:** Holds a personalized horoscope message



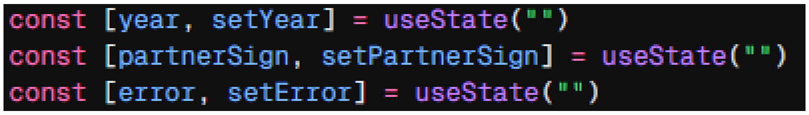
```
// State for UI control
const [isModalOpen, setIsModalOpen] = useState(false)
const [formKey, setFormKey] = useState(0)
const [error, setError] = useState("")
```

Figure 4. useState Rendering for UI control

These state variables manage UI-related aspects:

- **isModalOpen:** A boolean to control the visibility of a modal *(likely for displaying results)*
- **formKey:** Used for forcing re-renders of a form component, useful for resetting forms
- **error:** Stores error messages for display to the user

2. State Management in **inputYear.jsx**



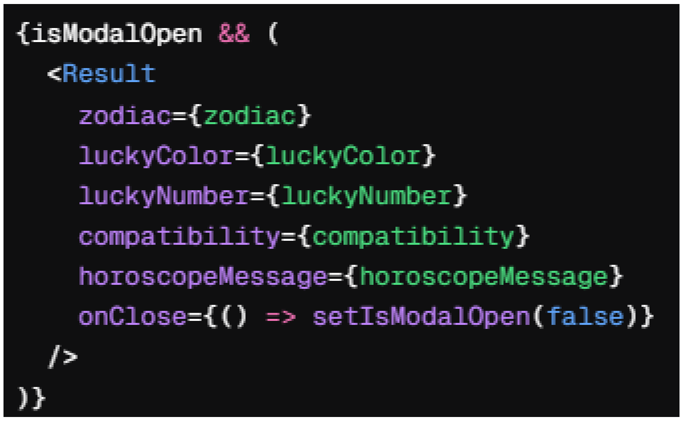
```
const [year, setYear] = useState("")
const [partnerSign, setPartnerSign] = useState("")
const [error, setError] = useState("")
```

Figure 5. useState Rendering for user input

These state variables manage the user input of Birth Year:

- **year:** This state variable will store the user's input year
- **partnerSign:** This state will hold the zodiac sign of a partner or for compatibility checking
- **error:** This state will store any error messages that need to be displayed to the user

3. Conditional Rendering in **App.jsx**



```
{isModalOpen && (
  <Result
    zodiac={zodiac}
    luckyColor={luckyColor}
    luckyNumber={luckyNumber}
    compatibility={compatibility}
    horoscopeMessage={horoscopeMessage}
    onClose={() => setIsModalOpen(false)}
  />
)}
```

Figure 6. Conditional Rendering of Modal

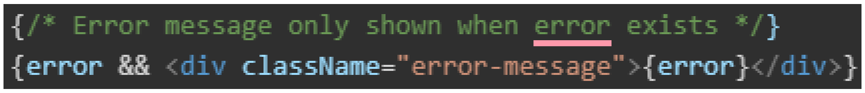
As discussed recently, each *state variable* is initialized with an empty string, except for **isModalOpen** which is a boolean initialized to **false** as shown in Figure 4.

If **isModalOpen** is **true**, the expression after **&&,** in this case, the **<Result … />** component, will be rendered. If **false**, nothing will be rendered.

zodiac, **luckyColor**, **luckyNumber**, **compatibility**, **horoscopeMessage** are the state variables being passed as *props* to the **<Result … />** component. They contain the zodiac information to be displayed.

onClose is a function prop that, when called, will set **isModalOpen** to **false**, effectively *closing* the modal.

4. Conditional Rendering in **inputYear.jsx**

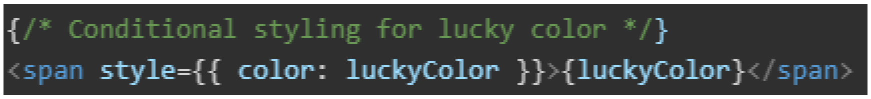


```
{/* Error message only shown when error exists */}
{error && <div className="error-message">{error}</div>}
```

Figure 7. Conditional Rendering to display an error message when a user is inputting his/her Birth Year

As shown in Figure 5, **error** is a variable that holds an error message. If **error** is *truthy* (**not null**, **undefined**, **false**, **0**, **''**, **etc.**), the second expression, **<div className="error-message">{error}</div>**, is evaluated and rendered. If **error** is *falsy*, nothing is rendered.

5. Conditional Rendering in **displayZodiac.jsx**



```
{/* Conditional styling for lucky color */}
<span style={{ color: luckyColor }}>{luckyColor}</span>
```

Figure 8. Conditional Rendering to change the color values in **luckyColor** variable

As shown in Figure 3, the **luckyColor** variable holds the lucky color for the zodiac. It dynamically changes the text color based on the **luckyColor** prop. If **luckyColor** is "red", the text appears in *red*. If it's "blue", the text appears in *blue*.

C. THE INNOVATIONS IN THE DESIGN

Compatibility Checker: With just one click, users can instantly discover their most compatible zodiac sign, helping them gain insights into potential romantic connections. This feature enhances their understanding of astrological compatibility, providing guidance on love, relationships, and emotional dynamics based on zodiac traits. By exploring these connections, users can make more informed decisions about their relationships and deepen their appreciation for astrology's role in love and attraction.

Lucky Numbers & Colors: This feature allows users to uncover their lucky numbers and colors based on their zodiac sign's unique traits and astrological influences. Lucky numbers are often associated with fortune, success, and positive energy, helping users make confident decisions in various aspects of life, such as choosing important dates, making investments, or selecting lottery numbers. Similarly, lucky colors are believed to enhance mood, attract good vibes, and align with an individual's personal energy, influencing everything from fashion choices to workspace aesthetics. By integrating these elements, users can gain a deeper connection with their zodiac sign and harness its potential for good luck and positivity in their daily lives.

~End~