REACT – PROBLEM STATEMENTS

*NAME – **DABHI HET BHARATBHAI***
*ENROLLMENT NO – **2303051057012***
*DEPARTMENT - **CSE***

TASKS	*PROBLEM-STATEMENTS*
1	Setup Tasks: **Set Up the Project:** – Initialize a React project using Vite or Create React App. – Create a folder named `components` for reusable components.
2	Building a Simple Portfolio Page: **Create a Simple JSX Element:** – Create a `Header.jsx` component that displays your name, a tagline (e.g., "Aspiring Software Engineer"), and a navigation bar with links to sections like About, Projects, and Contact.
3	**Create a Complex JSX Element:** – Extend the portfolio header to include a `div` with: – A profile picture. – Social media icons (LinkedIn, GitHub) as clickable links.
4	**Add Comments in JSX:** – Comment on different parts of your `Header.jsx` explaining what each section does.
5	**Render HTML Elements to the DOM:** – Render the `Header` component to the DOM using `ReactDOM.render` or `createRoot`.
6	Building a Project Showcase: **Define an HTML Class in JSX:** – Create a `ProjectCard.jsx` component with `className` styling to display: ▫ A project title. ▫ A brief description. ▫ A "View More" button.
7	**Learn About Self-Closing JSX Tags:** – Use an `img` tag to display a project screenshot in the `ProjectCard` component.
8	Building a Weather App: **Create a Stateless Functional Component:** – Build a `WeatherInfo.jsx` component that takes props like `city`, `temperature`, and `description` and displays them.
9	**Create a React Component:** – Create a `WeatherApp.jsx` component that fetches data from a weather API and passes it to `WeatherInfo`.

10	**Use React to Render Nested Components:** – Render multiple `WeatherInfo` components for different cities inside `WeatherApp`.
11	Student Attendance Tracker: **Create a Stateful Component:** – Build a `AttendanceTracker.jsx` component to track the attendance of students in a class.
12	**Render State in the User Interface:** – Display the total number of students present and absent.
13	**Set State with this.setState:** – Add a button to mark a student as present or absent and update the state.
14	Building a To-Do List App: **Bind 'this' to a Class Method:** – Create a `TodoApp.jsx` class component where `this` is bound to methods for adding and removing tasks.
15	**Use State to Toggle an Element:** – Add a toggle button to mark tasks as completed or not.
16	**Write a Simple Counter:** – Display the total number of tasks and the number of completed tasks.
17	Building a Student Feedback Form: **Create a Controlled Form:** – Build a `FeedbackForm.jsx` that captures name, email, and feedback message.
18	**Pass State as Props to Child Components:** – Pass the captured feedback to a `FeedbackCard.jsx` component that displays feedback entries.
19	Building a Library Management System: **Pass Props to a Stateless Functional Component:** – Create a `BookCard.jsx` component that displays details of a book (title, author, and availability status).
20	**Pass an Array as Props:** – Create a `Library.jsx` component that takes an array of books and renders a list of `BookCard` components.
21	**Use Default Props:** – Set default props for `BookCard` to handle cases where a book title or author is missing.
22	**Override Default Props:** – Pass custom book details to override default props.
23	Building a Blogging Platform: **Use PropTypes to Define the Props You Expect:**

	– Add PropTypes to a `BlogPost.jsx` component to validate props like `title`, `content`, and `author`.
24	**Create a Controlled Input:** – Create a form to add a new blog post, controlling the input fields with state.
25	**Pass State as Props to Child Components:** – Pass the blog data to a `BlogList.jsx` component to display all posts.
26	Building a Real-Time Chat App: **Create a Component with Composition:** – Build a `ChatApp.jsx` that uses components like `ChatHeader`, `MessageList`, and `MessageInput`.
27	**Compose React Components:** – Use composition to display different chats in a single `ChatApp` interface.
28	**Create a Stateful Component:** – Store chat messages in the state and update them as new messages are sent.
29	**Render State in the User Interface Another Way:** – Use a `map` function to dynamically render messages in the `MessageList` component.

Capstone Project Idea
Build a Complete Student Dashboard:

- **Features:**

1	A profile section with a student's details.
2	Attendance tracking.
3	A to-do list for assignments.
4	A weather widget showing real-time weather.
5	A blog section for sharing thoughts or academic updates.
6	A feedback form for collecting teacher or peer feedback.

- **Components:**
  - **`ProfileCard.jsx`:** Displays student details.
  - **`AttendanceTracker.jsx`:** Tracks attendance.
  - **`TodoApp.jsx`:** Manages a list of tasks.
  - **`WeatherApp.jsx`:** Displays weather.
  - **`FeedbackForm.jsx`:** Handles feedback submission.
  - **`BlogList.jsx`:** Displays blog posts.

- **Skills Applied:**
  - JSX rendering.
  - Props and state.
  - Controlled components.
  - Component composition.