🎯 Interview Preparation: TaskMaster ToDo App

Project Overview

TaskMaster is a modern, responsive To-Do List application built with **React 19** and **Vite**, featuring dark/light themes, task categorization, and local storage persistence.

🎤 Personal Introduction (Tell Me About Yourself)

Sample Answer

> *"Good morning, and thank you for the opportunity. My name is **Pruthiviraj Sahu**, and I'm currently pursuing my **B.Tech in Information Technology** at ITER, Bhubaneswar, with a **CGPA of 9.3**.*
>
> *I enjoy working on real-world projects that solve problems. For example, I've built applications like a **Weather App**, **TaskMaster To-Do List**, **Simon Game**, and recently an **AlumniConnect platform** during a hackathon, which focused on alumni engagement and mentorship. I've worked with technologies like **Java, Spring Boot, React, and SQL**, and I'm passionate about applying them to practical solutions.*
>
> *Earlier this year, I experienced a health challenge — I had a brain stroke that temporarily affected my balance and caused some slurred speech. However, with treatment and consistent recovery efforts, I've been regaining my strength and I continue to stay fully committed to my learning and career goals.*
>
> *What keeps me motivated is my ability to adapt and push forward. I believe this **resilience**, combined with my technical foundation, will help me grow and contribute effectively."*

Why This Works

- ✅ **Leads with strengths** — education, CGPA, projects
- ✅ **Shows technical depth** — mentions specific tech stack
- ✅ **Addresses health transparently** — prevents surprises if they notice speech
- ✅ **Ends positively** — focuses on resilience and commitment

Likely Follow-Up Questions

Question	How to Answer
"Tell me more about your health situation"	"I had a stroke in [month], which affected my balance and speech temporarily. I've been undergoing therapy and I'm recovering well. It hasn't impacted my ability to code or learn."
"How do you manage your recovery with studies?"	"I prioritize my health appointments and plan my study schedule around them. It's taught me better time management."
"Are you fully fit to work?"	"Yes, I can work regular hours. My recovery is progressing well and my cognitive abilities are unaffected."

Tips for Delivery

- 🎯 Maintain eye contact and speak with confidence
- 🎯 Keep the health mention brief (10-15 seconds max)
- 🎯 Smile and show enthusiasm when discussing projects
- 🎯 Practice this intro until it feels natural (not memorized)

🛠️ Tech Stack & Why They Matter

Technology	Purpose	Interview Talking Point
React 19	UI library	Latest version with improved hooks

Vite	Build tool	Faster HMR than Create React App
Tailwind CSS	Styling	Utility-first CSS framework
Context API	State management	Theme management without Redux
localStorage	Data persistence	Client-side storage solution

📁 Project Architecture

```
src/
├── App.jsx          # Main component (state + task logic)
├── main.jsx         # Entry point
├── index.css        # Global styles + animations
├── context/
│   └── ThemeContext.jsx  # Theme provider + hook
└── components/
    ├── Header.jsx   # Navigation + theme toggle
    └── Footer.jsx   # Social links
```

🔥 Key Features You Built

1. CRUD Operations

- ✅ **Create**: Add tasks with categories
- ✅ **Read**: Display tasks with filtering/sorting
- ✅ **Update**: Inline task editing
- ✅ **Delete**: Remove tasks with animations

2. State Management with Hooks

```
// Lazy initialization with localStorage
const [tasks, setTasks] = useState(() => {
  const saved = localStorage.getItem(STORAGE_KEY);
  return saved ? JSON.parse(saved) : DEFAULT_TASKS;
});
```

3. Theme Context (Light/Dark Mode)

```
// Custom hook pattern
export function useTheme() {
  const context = useContext(ThemeContext);
  if (!context) {
    throw new Error('useTheme must be used within a ThemeProvider');
  }
  return context;
}
```

4. Data Persistence

```
// Sync to localStorage on every change
useEffect(() => {
```

```
  localStorage.setItem(STORAGE_KEY, JSON.stringify(tasks));
}, [tasks]);
```

5. Filtering & Sorting

```
const processedTasks = tasks
  .filter((task) => {
    const matchesSearch = task.title.toLowerCase()
      .includes(searchTerm.toLowerCase());
    const matchesCategory = filterCategory === 'All'
      || task.category === filterCategory;
    return matchesSearch && matchesCategory;
  })
  .sort((a, b) => {
    if (sortBy === 'Newest') return b.id - a.id;
    if (sortBy === 'Oldest') return a.id - b.id;
    if (sortBy === 'A-Z') return a.title.localeCompare(b.title);
    return 0;
  });
```

💬 Common Interview Questions & Answers

React Fundamentals

Q: What is Virtual DOM and how does React use it?

> *The Virtual DOM is a lightweight JavaScript representation of the actual DOM. React creates a virtual copy, compares it with the previous version (diffing), and only updates the changed parts in the real DOM (reconciliation). This makes updates efficient.*

Q: Explain the difference between `useState` and `useEffect`.

> - `useState` *manages component state and triggers re-renders when state changes*
> - `useEffect` *handles side effects like API calls, localStorage, subscriptions*
>
> *In my project: I use* `useState` *for tasks, newTask, searchTerm, etc., and* `useEffect` *to persist tasks to localStorage.*

Q: Why did you use lazy initialization in useState?

```
const [tasks, setTasks] = useState(() => {
  const saved = localStorage.getItem(STORAGE_KEY);
  return saved ? JSON.parse(saved) : DEFAULT_TASKS;
});
```

> *Lazy initialization ensures localStorage is only read once on mount, not on every re-render. Passing a function instead of a value prevents unnecessary computation during subsequent renders.*

Context API & State Management

Q: Why use Context API instead of prop drilling?

> *Context provides a way to share values (like theme) across the entire component tree without passing props at every level. In TaskMaster, the theme needs to be accessible in Header, App, and Footer - prop drilling would be cumbersome.*

Q: When would you use Redux instead of Context?

> *Redux for:*

> - *Complex state with many reducers*
> - *Middleware needs (logging, async actions)*
> - *Time-travel debugging*
>
> *Context for:*
>
> - *Simple global state (theme, user preferences)*
> - *Smaller apps with fewer state updates*

Q: Explain the custom hook pattern in your ThemeContext.

> *I created `useTheme()` to:*
>
> 1. *Abstract away `useContext(ThemeContext)` calls*
> 2. *Add error handling if used outside Provider*
> 3. *Provide a cleaner API for components*

Component Design

Q: How do you handle form submissions in React?

```
const addTask = (e) => {
  e.preventDefault(); // Prevent page refresh
  if (newTask.trim() === '') return;

  // Duplicate check
  if (tasks.some(t => t.title.toLowerCase() === newTask.trim().toLowerCase())) {
    alert('Task already exists');
    return;
  }

  setTasks((prev) => [...prev, {
    id: Date.now(),
    title: newTask.trim(),
    completed: false
  }]);
  setNewTask('');
};
```

Q: Why use `Date.now()` for IDs?

> *It's a simple way to generate unique IDs in a client-only app. For production with a backend, I'd use UUIDs or server-generated IDs to avoid collisions.*

Q: How do you handle conditional rendering?

```
{processedTasks.length === 0 ? (
  <div className="empty-state">No tasks</div>
) : (
  processedTasks.map((task) => <TaskRow key={task.id} {...task} />)
)}
```

Styling & CSS

Q: How does your dark/light mode work?

> 1. *ThemeContext stores `isDark` state in localStorage*

> 2. `useEffect` toggles `light-mode` class on `<html>` element
> 3. CSS uses `html.light-mode` selector to override dark theme styles
>
> *This approach avoids CSS-in-JS overhead and leverages CSS cascade.*

Q: Explain the mesh gradient animation.

```
background:
  radial-gradient(ellipse at 0% 0%, rgba(124, 58, 237, 0.5) 0%, transparent 50%),
  radial-gradient(ellipse at 100% 100%, rgba(236, 72, 153, 0.3) 0%, transparent 50%);
animation: meshGradientDark 15s ease infinite;
```

> *Multiple overlapping radial gradients animated with `background-position` create a fluid, dynamic effect without JavaScript.*

Performance & Best Practices

Q: How do you optimize React performance?

> 1. ***Lazy initialization*** *- prevents expensive operations on every render*
> 2. ***Functional updates*** *- `setTasks(prev => [...prev, task])` ensures latest state*
> 3. ***Key prop*** *- unique keys help React identify which items changed*
> 4. ***Event delegation*** *- using `stopPropagation` to prevent bubbling issues*

Q: What accessibility features did you implement?

> - `aria-labels` *on icon buttons*
> - *Semantic HTML (* `<header>` *,* `<form>` *)*
> - *Keyboard navigation (Enter to save, Escape to cancel editing)*
> - *Sufficient color contrast*

🌐 MERN Stack Interview Questions

What is MERN Stack?

Component	Technology	Purpose
M	MongoDB	NoSQL database (document-based)
E	Express.js	Backend web framework for Node.js
R	React	Frontend UI library
N	Node.js	JavaScript runtime for server-side

MongoDB Questions

Q: What is MongoDB and why use it?

> *MongoDB is a NoSQL, document-oriented database that stores data in flexible JSON-like documents (BSON). Unlike SQL databases with rigid tables, MongoDB allows dynamic schemas, making it ideal for agile development.*
>
> *Benefits: Horizontal scaling, flexible schema, fast reads, great for hierarchical data.*

Q: What is the difference between SQL and NoSQL?

SQL (MySQL, PostgreSQL)	NoSQL (MongoDB)
Tables with rows/columns	Collections with documents

Fixed schema	Flexible schema
JOIN operations	Embedded documents or references
ACID transactions	BASE (Eventually consistent)
Vertical scaling	Horizontal scaling

Q: Explain MongoDB CRUD operations.

```
// CREATE
db.tasks.insertOne({ title: "Learn MERN", completed: false });

// READ
db.tasks.find({ completed: false });
db.tasks.findOne({ _id: ObjectId("...") });

// UPDATE
db.tasks.updateOne({ _id: ObjectId("...") }, { $set: { completed: true } });

// DELETE
db.tasks.deleteOne({ _id: ObjectId("...") });
```

Q: What are MongoDB indexes?

> *Indexes improve query performance by creating a searchable structure. Without indexes, MongoDB scans every document (collection scan).*

```
db.tasks.createIndex({ title: 1 }); // Ascending index on title
db.tasks.createIndex({ category: 1, createdAt: -1 }); // Compound index
```

Q: What is Mongoose?

> *Mongoose is an ODM (Object Data Modeling) library for MongoDB and Node.js. It provides schema validation, middleware, and easier query building.*

```
const taskSchema = new mongoose.Schema({
  title: { type: String, required: true },
  completed: { type: Boolean, default: false },
  category: { type: String, enum: ['Personal', 'Work', 'Urgent'] }
});
const Task = mongoose.model('Task', taskSchema);
```

Express.js Questions

Q: What is Express.js?

> *Express is a minimal, flexible Node.js web application framework that provides robust features for building APIs and web applications.*

Q: Explain Express middleware.

> *Middleware functions have access to request (`req`), response (`res`), and next middleware (`next`). They can execute code, modify req/res, end the request cycle, or call the next middleware.*

```
// Custom logging middleware
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
```

```
  next(); // Pass control to next middleware
});

// Built-in middleware
app.use(express.json()); // Parse JSON body
app.use(cors()); // Enable CORS
```

Q: How do you create a REST API with Express?

```
const express = require('express');
const app = express();
app.use(express.json());

// GET all tasks
app.get('/api/tasks', async (req, res) => {
  const tasks = await Task.find();
  res.json(tasks);
});

// POST new task
app.post('/api/tasks', async (req, res) => {
  const task = new Task(req.body);
  await task.save();
  res.status(201).json(task);
});

// PUT update task
app.put('/api/tasks/:id', async (req, res) => {
  const task = await Task.findByIdAndUpdate(req.params.id, req.body, { new: true });
  res.json(task);
});

// DELETE task
app.delete('/api/tasks/:id', async (req, res) => {
  await Task.findByIdAndDelete(req.params.id);
  res.status(204).send();
});

app.listen(5000, () => console.log('Server running on port 5000'));
```

Q: What are HTTP status codes you commonly use?

Code	Meaning	When to Use
200	OK	Successful GET/PUT
201	Created	Successful POST
204	No Content	Successful DELETE
400	Bad Request	Invalid input
401	Unauthorized	Missing/invalid auth
404	Not Found	Resource doesn't exist
500	Server Error	Internal error

Node.js Questions

Q: What is Node.js?

> *Node.js is a JavaScript runtime built on Chrome's V8 engine that allows JavaScript to run server-side. It uses an event-driven, non-blocking I/O model.*

Q: Explain the Event Loop in Node.js.

> *The Event Loop handles asynchronous operations. It continuously checks the call stack and callback queue. When the stack is empty, it moves callbacks from the queue to the stack for execution.*
>
> *Phases: Timers → Pending → Idle → Poll → Check → Close*

Q: What is the difference between `require` and `import`?

CommonJS (`require`)	ES Modules (`import`)
Synchronous loading	Asynchronous loading
`const x = require('x')`	`import x from 'x'`
`module.exports = x`	`export default x`
Node.js default	Modern standard

Q: How do you handle errors in Node.js?

```
// Try-catch for async/await
app.get('/api/tasks', async (req, res) => {
  try {
    const tasks = await Task.find();
    res.json(tasks);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

// Error handling middleware
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ message: 'Something went wrong!' });
});
```

Connecting MERN Stack

Q: How do you connect React to Express backend?

```
// React (frontend) - using fetch
const fetchTasks = async () => {
  const response = await fetch('http://localhost:5000/api/tasks');
  const data = await response.json();
  setTasks(data);
};

// Or using Axios
import axios from 'axios';
```

```
const fetchTasks = async () => {
  const { data } = await axios.get('/api/tasks');
  setTasks(data);
};
```

Q: What is CORS and why is it needed?

> *CORS (Cross-Origin Resource Sharing) is a security feature that blocks requests from different origins by default. If React runs on `Localhost:3000` and Express on `Localhost:5000`, you need CORS.*

```
const cors = require('cors');
app.use(cors({ origin: 'http://localhost:3000' }));
```

Q: How would you add a backend to your TaskMaster app?

> *1. Set up Express server with MongoDB connection*
> *2. Create Task model with Mongoose*
> *3. Build REST API endpoints (GET, POST, PUT, DELETE)*
> *4. Replace localStorage with API calls using fetch/axios*
> *5. Add authentication (JWT) for user-specific tasks*

🎨 Design Decisions to Highlight

1. **Glassmorphism Header**: `backdrop-filter: blur(20px)` creates modern frosted-glass effect
2. **Animated Title**: Gradient text with shimmer animation draws attention
3. **Mobile-First Approach**:
   - Touch targets ≥44px
   - `font-size: 16px` to prevent iOS zoom
   - Responsive breakpoints
4. **Micro-interactions**:
   - Hover scale effects on buttons
   - Slide-in/out animations for tasks

🚀 How to Explain Your Project

30-Second Elevator Pitch

> *"I built TaskMaster, a modern To-Do application using React 19 and Vite. It features full CRUD operations, category-based organization, real-time search and sorting, plus a dark/light theme that persists to localStorage. I focused on clean code architecture by using Context API for theme management and implementing responsive design with smooth animations for a premium user experience."*

Technical Deep Dive Points

1. **React Hooks**: Demonstrate understanding of useState, useEffect, useContext
2. **State Architecture**: Single source of truth in App.jsx, derived state for filtering
3. **Persistence Strategy**: localStorage with lazy initialization
4. **CSS Architecture**: Utility classes + custom CSS with theme inheritance

---

❓ Questions YOU Should Ask

1. "What does your typical code review process look like?"
2. "How do you balance shipping features quickly vs. code quality?"

3. "What's your tech stack and any plans to modernize?"
4. "How does the team handle on-call and production issues?"

📝 Quick Reference Card

Concept	Your Implementation
State Management	React Context + useState
Side Effects	useEffect for localStorage
Styling	Tailwind CSS + Custom CSS
Build Tool	Vite (fast HMR)
Framework	React 19
Persistence	localStorage
Theme	Dark/Light with CSS classes
Responsive	Mobile-first with breakpoints

Good luck with your interview! 🍀