

MERN system test 01

Duration: 3–4 hours

Tech stack (Flexible)

- Backend: Node.js or Python or Java
- Database: MongoDB or RDBMS
- Frontend: React or Angular or Similar
- AI tools: Allowed (must explain logic and the approach)

What this test actually measures

- Basic logic
- Understanding of data & flow
- Ability to explain code
- Willingness to learn (AI usage allowed)
- Not judged on perfect architecture
- Not judged on advanced tech stack/tricks

Problem statement

- Build a Simple Task Tracker system when a user can
 - Create a task
 - See all tasks
 - Change task status
- Task Rules
 - Each task has a title and status
 - Status can be:
 - TODO
 - IN_PROGRESS

- DONE
- Tasks start as TODO

PART 1: Data Model

- Design one data structure for a task.
- Required Fields
 - id
 - title
 - status
 - createdAt
- Task structure (Entity or DB schema OR JS/JSON object)
- Short explanation:
 - Why you chose these fields
 - Why status is limited

PART 2: Backend API

- Required APIs
- Create task
- Get all tasks
- Update status
- Rules
 - Status must be valid
 - Use proper HTTP response status codes (200, 400)
 - Error handling
- Short explanation:
 - What each API does
 - What happens when wrong data is sent

PART 3: UI

- Design the UI
- Components
- Task List
- Create Task Form
- Status Update Button
- HTML/React/Angular components
- Fetch from backend OR use mock API
- Short explanation:
 - How data flows from backend to UI
 - Why components and what are the advantages of using components

PART 4: Simple Logic

- A task cannot go from TODO → DONE directly
- Task
- Implement this rule anywhere
- Explain why you chose that place
 - Frontend only
 - Backend only
 - Both
 - Short explanation

PART 5: AI Usage

- Did you use AI? (Yes/No)
- What did AI help with?
- One thing you understood clearly after using AI
- How much time did you gain?
- How reliable is the generated code?

PART 6: Optional Bonus (Only if time is there)

- Support to delete task
- Simple CSS (Better UI)