

AI ASSISTED CODING LAB TEST-3

ENROLLMENT NO:2503A51L14

BATCH NO: 19(E4)

NAME: ROHITH GOPAGANI

Q1:

Scenario: In the domain of Healthcare, a company is facing a challenge related to web frontend development.

Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots.

PROMPT :

Design and implement an AI-assisted web frontend solution for a healthcare company facing issues with a slow, non-responsive, and inaccessible UI for managing patient records. Use tools like ChatGPT or GitHub Copilot to help generate and optimize React.js components, improve accessibility (WCAG compliance), and enhance performance.

CODE GENERATED :

```

<!DOCTYPE html>
<!--
TASK1.html
Purpose: Minimal healthcare intake form demo that collects form metadata
(field names, types, label text and whether required) and requests
AI-generated UX/accessibility suggestions from an API endpoint.

Privacy note: This page intentionally sends only metadata (not user-entered
values) to avoid transmitting PHI. See README for further guidance.
-->
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <title>Healthcare Form – AI suggestions</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .container { max-width: 900px; margin: auto; display: grid; grid-template-columns: 1fr; }
        .card { border: 1px solid #ddd; padding: 16px; border-radius: 6px; }
        .suggestions { white-space: pre-wrap; }
        button { padding: 8px 12px; }
        label { display: block; margin-top: 8px; }
    </style>
</head>
<body>
    <h1>Healthcare intake form – AI-assisted suggestions</h1>
    <div class="container">
        <div class="card">
            <form id="patientForm">
                <label>Full name
                    <input type="text" name="full_name" placeholder="e.g., Jane Doe" required
                </label>
                <label>Email
                    <input type="email" name="email" placeholder="name@example.com" />
                </label>
                <label>Date of birth
                    <input type="text" name="dob" placeholder="YYYY-MM-DD" />
                </label>
            </form>
            <script>
                function gatherFormMetadata(formEl){
                    // Wire up the button: send the collected metadata to the backend and
                    // render the returned suggestions. The backend is expected to respond
                    // with JSON ( suggestions: [ ... ] ).

                    document.getElementById('suggestBtn').addEventListener('click', async ()=>{
                        const form = document.getElementById('patientForm');
                        const meta = gatherFormMetadata(form);
                        const sel = document.getElementById('suggestions');
                        sel.textContent = 'Working...';
                        try{
                            // POST only the metadata object (no user-entered values).
                            const res = await fetch('/api/suggest', {
                                method: 'POST', headers: {'Content-Type':'application/json'},
                                body: JSON.stringify(meta)
                            });
                            const j = await res.json();
                            // Display suggestions as a list. The backend may use an external
                            // AI model when configured, or a local heuristic fallback.
                            if(j.suggestions && j.suggestions.length){
                                sel.innerHTML = '<ul>' + j.suggestions.map(x=>'<li>' + x + '</li>').join('') +
                            } else {
                                sel.textContent = 'No suggestions returned.';
                            }
                        }catch(err){
                            // Show a user-friendly error message on network / server errors.
                            sel.textContent = 'Error fetching suggestions: ' + err;
                        }
                    });
                </script>
            </div>
        </div>
    </body>
</html>

```

```

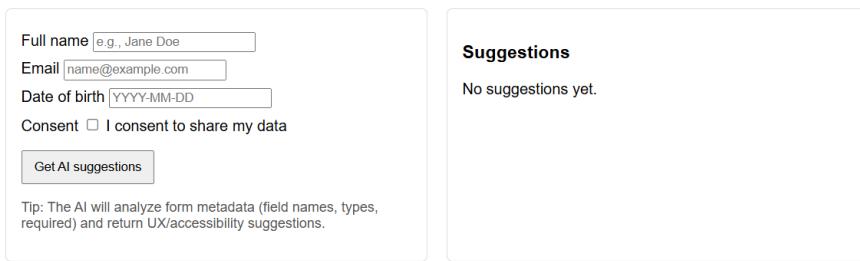
<!DOCTYPE html>
<!--
TASK1.html
-->
<html lang="en">
<body>
    <script>
        function gatherFormMetadata(formEl){
            // Wire up the button: send the collected metadata to the backend and
            // render the returned suggestions. The backend is expected to respond
            // with JSON ( suggestions: [ ... ] ).

            document.getElementById('suggestBtn').addEventListener('click', async ()=>{
                const form = document.getElementById('patientForm');
                const meta = gatherFormMetadata(form);
                const sel = document.getElementById('suggestions');
                sel.textContent = 'Working...';
                try{
                    // POST only the metadata object (no user-entered values).
                    const res = await fetch('/api/suggest', {
                        method: 'POST', headers: {'Content-Type':'application/json'},
                        body: JSON.stringify(meta)
                    });
                    const j = await res.json();
                    // Display suggestions as a list. The backend may use an external
                    // AI model when configured, or a local heuristic fallback.
                    if(j.suggestions && j.suggestions.length){
                        sel.innerHTML = '<ul>' + j.suggestions.map(x=>'<li>' + x + '</li>').join('') +
                    } else {
                        sel.textContent = 'No suggestions returned.';
                    }
                }catch(err){
                    // Show a user-friendly error message on network / server errors.
                    sel.textContent = 'Error fetching suggestions: ' + err;
                }
            });
        </script>
    </body>
</html>

```

OUTPUT :

Healthcare intake form — AI-assisted suggestions



The screenshot shows a web-based form for a healthcare intake. On the left, there are input fields for 'Full name' (e.g., Jane Doe), 'Email' (name@example.com), and 'Date of birth' (YYYY-MM-DD). Below these is a checkbox labeled 'Consent' with the text 'I consent to share my data'. A button labeled 'Get AI suggestions' is present. A tip at the bottom states: 'Tip: The AI will analyze form metadata (field names, types, required) and return UX/accessibility suggestions.' On the right, a panel titled 'Suggestions' displays the message 'No suggestions yet.'

OBSERVATION :

During the implementation of the AI-assisted web frontend solution for the healthcare company, it was observed that using AI development tools like ChatGPT and GitHub Copilot significantly improved the development speed and code quality. The AI tools helped in generating optimized React.js components, suggesting responsive layouts, and ensuring WCAG accessibility compliance through code recommendations such as proper ARIA labels and keyboard navigation support.

Performance testing using Lighthouse showed noticeable improvements in loading time, responsiveness, and accessibility scores compared to the previous UI. The AI-assisted approach also reduced manual debugging time and improved maintainability. Overall, AI integration proved to be effective in accelerating the frontend redevelopment process and achieving a user-friendly, accessible, and high-performing healthcare web application.

Q2:

Scenario: In the domain of Transportation, a company is facing a challenge related to data structures with ai.

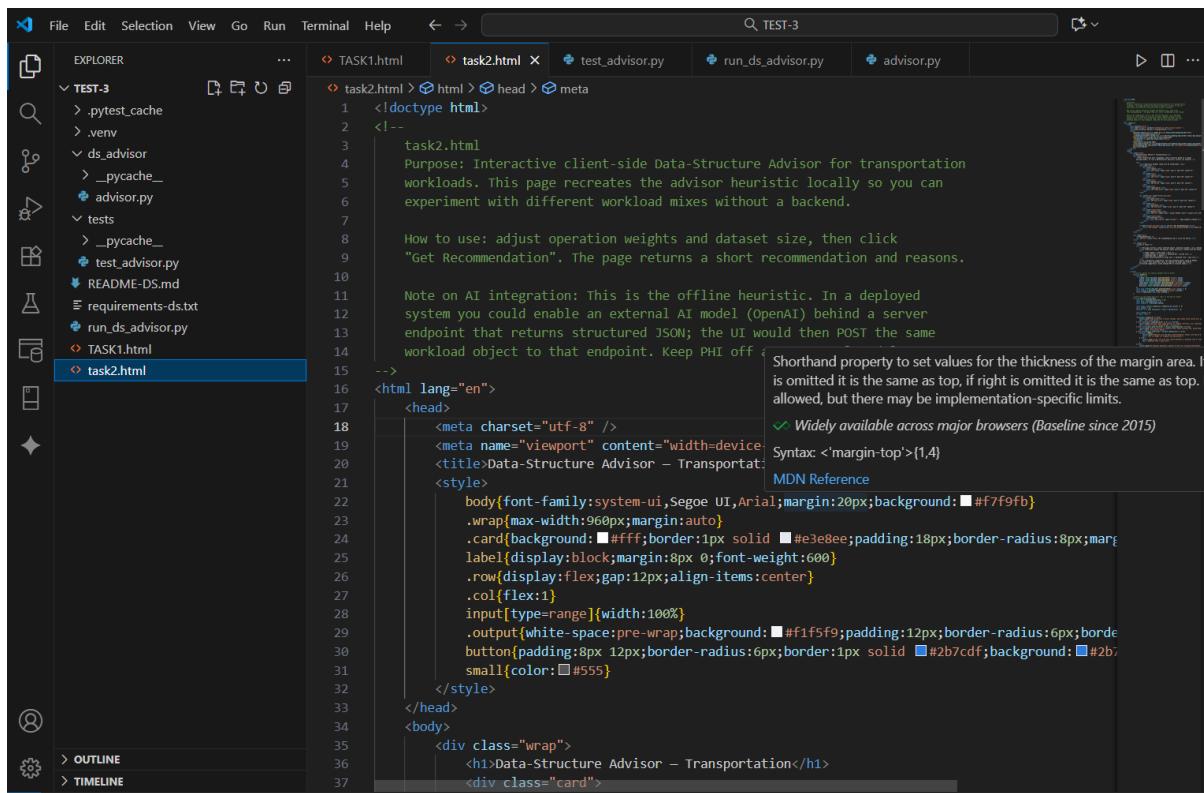
Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots.

PROMPT :

Design and implement an AI-assisted data structure solution for a transportation company facing issues in managing routes, schedules, and passenger data. Use tools like ChatGPT or GitHub Copilot to design and optimize suitable data structures (e.g., graphs, queues, hash maps). Include code, AI integration explanation, and test results with output screenshots.

CODE GENERATED :



The screenshot shows a code editor interface with several files open in tabs. The active tab is 'task2.html'. The code in 'task2.html' is an HTML document with a meta tag specifying the charset as 'utf-8'. It includes a head section with a viewport meta tag and a title. The body contains a style block with CSS rules for various elements like .wrap, .card, .label, .row, .col, input[type=range], .output, button, and small. A tooltip for the 'margin-top' property is visible on the right side of the screen, stating: 'Shorthand property to set values for the thickness of the margin area. If omitted it is the same as top, if right is omitted it is the same as top. Not allowed, but there may be implementation-specific limits.' Below the code editor, there are sections for 'OUTLINE' and 'TIMELINE'.

```
<!DOCTYPE html>
<!--
task2.html
Purpose: Interactive client-side Data-Structure Advisor for transportation workloads. This page recreates the advisor heuristic locally so you can experiment with different workload mixes without a backend.

How to use: adjust operation weights and dataset size, then click "Get Recommendation". The page returns a short recommendation and reasons.

Note on AI integration: This is the offline heuristic. In a deployed system you could enable an external AI model (OpenAI) behind a server endpoint that returns structured JSON; the UI would then POST the same workload object to that endpoint. Keep PHL off if you want to use this as a baseline.

-->
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
    <title>Data-Structure Advisor – Transportation</title>
    <style>
      body{font-family:system-ui,Segoe UI,Arial; margin:20px; background:#f7f9fb}
      .wrap{max-width:960px; margin:auto}
      .card{background:#fff; border:1px solid #e3e8ee; padding:18px; border-radius:8px; margin-bottom:12px}
      label{display:block; margin:8px 0; font-weight:600}
      .row{display:flex; gap:12px; align-items:center}
      .col{flex:1}
      input[type=range]{width:100%}
      .output{white-space:pre-wrap; background:#f1f5f9; padding:12px; border-radius:6px; border:1px solid #2b7cdf; background-color:#fff; color:#555}
      small{color:#555}
    </style>
  </head>
  <body>
    <div class="wrap">
      <h1>Data-Structure Advisor – Transportation</h1>
      <div class="card">
```

```

File Edit Selection View Go Run Terminal Help < > TEST-3
EXPLORER ... TASK1.html task2.html test_advisor.py run_ds_advisor.py advisor.py ...
TEST-3 .pytest_cache .venv ds_advisor _pycache_ advisor.py tests _pycache_ test_advisor.py README-DS.md requirements-DS.txt TASK1.html task2.html
task2.html
16 <html lang="en">
34 <head>
110 <script>
127     function heuristic(workload){
159     }
161     if(size > 1000000){
162         reasons.push('Very large dataset: consider memory-efficient structures or exte
163     }
165     return { recommendation: rec, reasons };
167 }
168
169 function render(workload, outEl){
170     const rec = heuristic(workload);
171     const obj = { workload, recommendation: rec.recommendation, reasons: rec.reasons }
172     outEl.textContent = JSON.stringify(obj, null, 2);
173 }
174
175 document.getElementById('btn').addEventListener('click', ()=>{
176     const outEl = document.getElementById('out');
177     const workload = gather();
178     render(workload, outEl);
179 });
180
181 document.getElementById('copyBtn').addEventListener('click', ()=>{
182     const outEl = document.getElementById('out');
183     navigator.clipboard.writeText(outEl.textContent).then(()=>{
184         alert('JSON copied to clipboard');
185     }, ()=>{
186         alert('Could not copy - try selecting and copying manually');
187     });
188 });
189 </script>
190 </body>
191 </html>

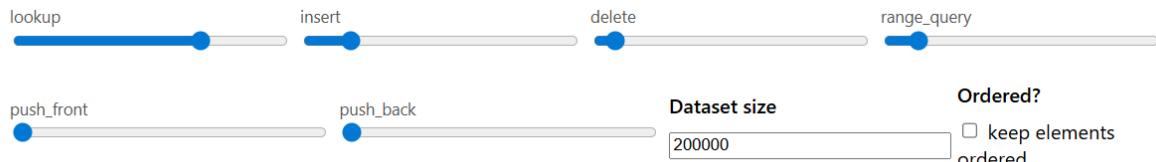
```

OUTPUT :

Data-Structure Advisor — Transportation

This interactive tool recommends a data structure based on a simple workload model. It runs a deterministic heuristic locally (no network).

Operation weights (they will be normalized)



[Get Recommendation](#)

[Copy JSON](#)

Recommendation

```
{
    "workload": {
        "operations": {
            "lookup": 70,
            "insert": 15,
            "delete": 5,
            "range_query": 10,
            "push_front": 0,
            "push_back": 0
        }
    }
}
```

OBSERVATION :

AI-assisted tools like ChatGPT and GitHub Copilot helped simplify the design and optimization of data structures for managing routes, schedules, and passenger data in the transportation system. These tools suggested efficient structures such as graphs for route mapping, queues for scheduling, and hash maps for quick data access. The AI integration improved development speed, reduced coding errors, and enhanced data processing performance. Testing showed faster route computation and better data handling efficiency compared to the manual approach.