31st Aug

Agenda :
-> overview of testcase
-> sample demo
-> work distribution
-> query resolve

Overview of TestCase Generation .

🧠 What Is Test Case Generation?

Test case generation is the process of creating a set of conditions or variables under which a tester determines whether a system, software, or feature works as intended.

🎯 Why It's Done:

- **Validate functionality**: Ensure the system behaves correctly.
- **Catch bugs early**: Prevent costly failures in production.
- **Ensure coverage**: Test all possible user scenarios.
- **Maintain quality**: Align with industry standards and compliance.

Examples :

🏥 Healthcare Requirement Example

Requirement: A hospital management system must allow doctors to view patient history, including prescriptions, lab results, and previous diagnoses.

✅ Test Cases:

Test Case ID	Description	Test Steps	Expected Result
HC_TC_001	Verify doctor login	Enter valid credentials	Doctor dashboard loads
HC_TC_002	Access patient history	Search patient → Click "History"	Full history is displayed
HC_TC_003	Prescription visibility	Navigate to prescriptions tab	All past prescriptions shown
HC_TC_004	Lab results format	Open lab results section	Results shown in readable format
HC_TC_005	Unauthorized access	Login as receptionist → Try accessing history	Access denied message

🚗 Automotive Requirement Example

Requirement: An in-vehicle infotainment system should allow users to connect their phone via Bluetooth and stream music.

✅ Test Cases:

Test Case ID	Description	Test Steps	Expected Result
AUTO_TC_001	Verify Bluetooth pairing	Enable Bluetooth → Search device → Pair	Device successfully paired
AUTO_TC_002	Stream music	Open music app → Play track	Audio plays through car speakers
AUTO_TC_003	Auto-reconnect	Turn off car → Turn on again	Phone reconnects automatically
AUTO_TC_004	Handle multiple devices	Pair second phone → Try streaming	Only one device streams at a time
AUTO_TC_005	Display track info	Play song → Check infotainment screen	Song title and artist displayed

Sample Demo

link:

https://claude.ai/public/artifacts/ede5efe3-03cb-4817-b0bc-d0135c97ca94

code :

```
import React, { useState } from 'react';
import { CheckCircle, XCircle, AlertTriangle, FileText, TestTube, Shield } from
'lucide-react';

const HealthcareTestGenerationDemo = () => {
  const [currentStep, setCurrentStep] = useState(1);
  const [selectedRequirement, setSelectedRequirement] = useState(null);

  // Sample Healthcare Requirements
  const healthcareRequirements = [
    {
```

```
      id: "REQ-001",
      title: "Insulin Dosage Calculation",
      text: "The system shall calculate insulin dosage recommendations based on patient blood glucose levels, carbohydrate intake, and insulin sensitivity factor. The calculation must be accurate to within ±5% and provide warnings for potentially dangerous dosages exceeding 2 units/kg body weight.",
      domain: "Diabetes Management",
      criticality: "Safety-Critical",
      entities: ["insulin", "blood glucose", "carbohydrate intake", "dosage", "body weight"]
    },
    {
      id: "REQ-002",
      title: "Patient Data Encryption",
      text: "All patient health information (PHI) stored in the system shall be encrypted using AES-256 encryption at rest and TLS 1.3 for data in transit. The system shall maintain an audit log of all PHI access attempts with user identification, timestamp, and data accessed.",
      domain: "Data Security",
      criticality: "Security-Critical",
      entities: ["PHI", "encryption", "AES-256", "TLS 1.3", "audit log"]
    },
    {
      id: "REQ-003",
      title: "Medical Alert System",
      text: "The system shall generate real-time alerts when patient vital signs fall outside predefined normal ranges (HR: 60-100 bpm, BP: 90/60-140/90 mmHg, SpO2: >95%). Alerts must be delivered within 10 seconds and require healthcare provider acknowledgment within 5 minutes.",
      domain: "Patient Monitoring",
      criticality: "Safety-Critical",
      entities: ["vital signs", "heart rate", "blood pressure", "oxygen saturation", "alerts"]
    }
  ];

  // FDA Standards Database (Simplified)
  const fdaStandards = {
    "Software_as_Medical_Device": {
      sections: [
        {
          id: "FDA_SaMD_3.1",
          title: "Risk Categorization",
          requirements: [
            "Software influencing treatment decisions must undergo Class III validation",
            "Dosage calculation software requires extensive verification and validation",
            "Safety-critical calculations must have redundant verification
```

```
mechanisms"
          ]
        },
        {
          id: "FDA_SaMD_4.2",
          title: "Verification and Validation",
          requirements: [
            "All algorithms must be tested with boundary conditions",
            "Clinical validation required for treatment recommendations",
            "Error handling must prevent patient harm"
          ]
        }
      ]
    },
    "Cybersecurity_Guidelines": {
      sections: [
        {
          id: "FDA_Cyber_2.1",
          title: "Data Protection",
          requirements: [
            "Patient data must use validated encryption methods",
            "Access controls must be auditable",
            "Encryption keys must be managed securely"
          ]
        }
      ]
    },
    "Medical_Device_QSR": {
      sections: [
        {
          id: "FDA_QSR_7.3",
          title: "Design Controls",
          requirements: [
            "Real-time systems must meet response time requirements",
            "Alert systems must have fail-safe mechanisms",
            "Critical alerts cannot be dismissed without proper authorization"
          ]
        }
      ]
    }
  };

  // AI-Generated Test Cases (Simulated Gemini Output)
  const generateTestCases = (requirement) => {
    const testCaseTemplates = {
      "REQ-001": [
        {
          id: "TC-001-001",
          title: "Verify Insulin Dosage Calculation Accuracy",
```

```
          type: "Functional",
          priority: "High",
          preconditions: ["Patient profile with known insulin sensitivity", "Valid
blood glucose reading", "Carbohydrate intake data"],
          steps: [
            "Input patient blood glucose level: 180 mg/dL",
            "Input carbohydrate intake: 45g",
            "Input insulin sensitivity factor: 50",
            "Execute dosage calculation",
            "Verify calculated dosage"
          ],
          expected: ["Dosage calculated within ±5% accuracy", "Result: ~2.1 units",
"No warning messages for normal dosage"],
          compliance_risk: "medium"
        },
        {
          id: "TC-001-002",
          title: "Test Dangerous Dosage Warning System",
          type: "Safety",
          priority: "Critical",
          preconditions: ["Patient weight: 70kg", "System configured with safety
thresholds"],
          steps: [
            "Input parameters resulting in >2 units/kg dosage",
            "Execute calculation",
            "Verify warning generation",
            "Attempt to override warning"
          ],
          expected: ["Warning displayed for dosage >140 units", "Override requires
additional authentication", "Audit log entry created"],
          compliance_risk: "high"
        },
        {
          id: "TC-001-003",
          title: "Boundary Testing for Calculation Accuracy",
          type: "Performance",
          priority: "High",
          preconditions: ["Access to calculation algorithm", "Test data set
prepared"],
          steps: [
            "Test with minimum valid glucose (70 mg/dL)",
            "Test with maximum safe glucose (400 mg/dL)",
            "Test with zero carbohydrate intake",
            "Test with maximum carbohydrate intake (200g)",
            "Verify all calculations within tolerance"
          ],
          expected: ["All calculations within ±5% accuracy", "No system errors",
"Performance <1 second per calculation"],
          compliance_risk: "medium"
```

```
        }
      ],
      "REQ-002": [
        {
          id: "TC-002-001",
          title: "Verify PHI Encryption at Rest",
          type: "Security",
          priority: "Critical",
          preconditions: ["Database with PHI data", "Encryption validation tools"],
          steps: [
            "Store patient data in database",
            "Access database files directly",
            "Verify AES-256 encryption implementation",
            "Attempt to read data without decryption keys"
          ],
          expected: ["Data encrypted with AES-256", "Raw data unreadable without
keys", "Encryption key properly secured"],
          compliance_risk: "high"
        },
        {
          id: "TC-002-002",
          title: "Validate Audit Log Functionality",
          type: "Compliance",
          priority: "High",
          preconditions: ["User accounts configured", "Audit system active"],
          steps: [
            "User logs into system",
            "Access specific PHI record",
            "Modify patient data",
            "Log out of system",
            "Review audit logs"
          ],
          expected: ["All actions logged with timestamp", "User ID captured
correctly", "Data access details recorded"],
          compliance_risk: "high"
        }
      ],
      "REQ-003": [
        {
          id: "TC-003-001",
          title: "Verify Vital Signs Alert Generation",
          type: "Functional",
          priority: "Critical",
          preconditions: ["Patient monitoring active", "Normal ranges configured"],
          steps: [
            "Input HR: 45 bpm (below normal)",
            "Input BP: 200/110 mmHg (above normal)",
            "Input SpO2: 88% (below normal)",
            "Measure response time",
```

```
            "Verify alert delivery"
          ],
          expected: ["Alert generated within 10 seconds", "All three vitals
flagged", "Healthcare provider notified"],
          compliance_risk: "high"
        },
        {
          id: "TC-003-002",
          title: "Test Alert Acknowledgment System",
          type: "Safety",
          priority: "Critical",
          preconditions: ["Active alerts present", "Healthcare provider account"],
          steps: [
            "Generate critical vital signs alert",
            "Wait for 5 minutes without acknowledgment",
            "Verify escalation procedure",
            "Acknowledge alert",
            "Verify alert resolution"
          ],
          expected: ["Escalation triggered after 5 minutes", "Alert remains active
until acknowledged", "Audit trail of acknowledgment"],
          compliance_risk: "high"
        }
      ]
    };
    return testCaseTemplates[requirement.id] || [];
  };

  // FDA Compliance Validation
  const validateAgainstFDA = (testCases, requirementId) => {
    const validationResults = testCases.map(testCase => {
      let violations = [];
      let recommendations = [];

      // Apply FDA validation rules
      if (requirementId === "REQ-001") {
        // Insulin dosage - FDA SaMD Class III requirements
        if (!testCase.steps.some(step => step.includes("boundary") ||
step.includes("minimum") || step.includes("maximum"))) {
          violations.push("Missing boundary condition testing (FDA SaMD 4.2)");
        }
        if (testCase.type === "Safety" && !testCase.steps.some(step =>
step.includes("override") || step.includes("warning"))) {
          violations.push("Safety mechanisms not adequately tested (FDA QSR 7.3)");
        }
        if (!testCase.expected.some(exp => exp.includes("audit") ||
exp.includes("log"))) {
          recommendations.push("Consider adding audit trail verification");
        }
```

```
      }

      if (requirementId === "REQ-002") {
        // Data encryption - FDA Cybersecurity requirements
        if (!testCase.steps.some(step => step.includes("AES-256"))) {
          violations.push("Encryption algorithm not explicitly verified (FDA Cyber
2.1)");
        }
        if (testCase.type === "Security" && !testCase.steps.some(step =>
step.includes("without decryption keys"))) {
          violations.push("Inadequate encryption verification testing");
        }
      }

      if (requirementId === "REQ-003") {
        // Alert system - FDA QSR Design Controls
        if (!testCase.expected.some(exp => exp.includes("10 seconds") ||
exp.includes("response time"))) {
          violations.push("Real-time requirement not verified (FDA QSR 7.3)");
        }
        if (testCase.type === "Safety" && !testCase.steps.some(step =>
step.includes("escalation"))) {
          violations.push("Fail-safe mechanism not tested (FDA QSR 7.3)");
        }
      }

      return {
        ...testCase,
        fda_compliant: violations.length === 0,
        violations,
        recommendations
      };
    });

    return validationResults;
  };

  // Filter compliant test cases
  const getCompliantTestCases = (validatedCases) => {
    return validatedCases.filter(tc => tc.fda_compliant);
  };

  const handleRequirementSelect = (req) => {
    setSelectedRequirement(req);
    setCurrentStep(2);
  };

  const generatedTestCases = selectedRequirement ?
generateTestCases(selectedRequirement) : [];
```

```
  const validatedTestCases = selectedRequirement ?
validateAgainstFDA(generatedTestCases, selectedRequirement.id) : [];
  const compliantTestCases = getCompliantTestCases(validatedTestCases);

  return (
    <div className="max-w-6xl mx-auto p-6 bg-gray-50 min-h-screen">
      <div className="bg-white rounded-lg shadow-lg p-6 mb-6">
        <h1 className="text-3xl font-bold text-gray-800 mb-2 flex items-center">
          <TestTube className="mr-3 text-blue-600" />
          Healthcare AI Test Generation Demo
        </h1>
        <p className="text-gray-600">Step-by-step demonstration of requirement
parsing, test generation, and FDA compliance validation</p>
      </div>

      {/* Step Indicator */}
      <div className="flex justify-center mb-8">
        <div className="flex items-center space-x-4">
          {[1, 2, 3, 4].map((step) => (
            <div key={step} className="flex items-center">
              <div className={`w-8 h-8 rounded-full flex items-center justify-center
${
                currentStep >= step ? 'bg-blue-600 text-white' : 'bg-gray-300 text-
gray-600'
              }`}>
                {step}
              </div>
              {step < 4 && <div className={`w-16 h-1 ${currentStep > step ? 'bg-
blue-600' : 'bg-gray-300'}`} />}
            </div>
          ))}
        </div>
      </div>

      {/* Step 1: Requirement Selection */}
      {currentStep >= 1 && (
        <div className="bg-white rounded-lg shadow-lg p-6 mb-6">
          <h2 className="text-xl font-semibold mb-4 flex items-center">
            <FileText className="mr-2 text-blue-600" />
            Step 1: Healthcare Software Requirements
          </h2>
          <div className="grid md:grid-cols-3 gap-4">
            {healthcareRequirements.map((req) => (
              <div
                key={req.id}
                className={`p-4 border rounded-lg cursor-pointer transition-all ${
                  selectedRequirement?.id === req.id
                    ? 'border-blue-500 bg-blue-50'
                    : 'border-gray-200 hover:border-gray-300'
```

```
                }`}
                onClick={() => handleRequirementSelect(req)}
              >
                <h3 className="font-semibold text-gray-800">{req.id}: {req.title}
</h3>
                <p className="text-sm text-gray-600 mt-2">{req.domain}</p>
                <div className={`inline-block px-2 py-1 rounded text-xs mt-2 ${
                  req.criticality === 'Safety-Critical' ? 'bg-red-100 text-red-800'
: 'bg-yellow-100 text-yellow-800'
                }`}>
                  {req.criticality}
                </div>
                <p className="text-sm text-gray-700 mt-2 line-clamp-3">{req.text}
</p>
              </div>
            ))}
          </div>
        </div>
      )}

      {/* Step 2: AI Test Case Generation */}
      {currentStep >= 2 && selectedRequirement && (
        <div className="bg-white rounded-lg shadow-lg p-6 mb-6">
          <h2 className="text-xl font-semibold mb-4 flex items-center">
            <TestTube className="mr-2 text-green-600" />
            Step 2: AI-Generated Test Cases
          </h2>
          <div className="mb-4 p-4 bg-blue-50 rounded-lg">
            <h3 className="font-semibold">Selected Requirement:
{selectedRequirement.title}</h3>
            <p className="text-sm text-gray-700 mt-2">{selectedRequirement.text}</p>
          </div>

          <div className="space-y-4">
            {generatedTestCases.map((testCase, index) => (
              <div key={testCase.id} className="border rounded-lg p-4">
                <div className="flex justify-between items-start mb-3">
                  <h4 className="font-semibold text-gray-800">{testCase.id}:
{testCase.title}</h4>
                  <div className="flex space-x-2">
                    <span className={`px-2 py-1 rounded text-xs ${
                      testCase.type === 'Safety' ? 'bg-red-100 text-red-800' :
                      testCase.type === 'Security' ? 'bg-purple-100 text-purple-800'
:
                      testCase.type === 'Functional' ? 'bg-green-100 text-green-800'
:
                      'bg-blue-100 text-blue-800'
                    }`}>
                      {testCase.type}
```

```
                    </span>
                    <span className={`px-2 py-1 rounded text-xs ${
                      testCase.priority === 'Critical' ? 'bg-red-100 text-red-800' :
'bg-orange-100 text-orange-800'
                    }`}>
                      {testCase.priority}
                    </span>
                  </div>
                </div>

                <div className="grid md:grid-cols-3 gap-4 text-sm">
                  <div>
                    <h5 className="font-medium text-gray-700">Preconditions:</h5>
                    <ul className="text-gray-600 mt-1">
                      {testCase.preconditions.map((pre, i) => (
                        <li key={i}>• {pre}</li>
                      ))}
                    </ul>
                  </div>

                  <div>
                    <h5 className="font-medium text-gray-700">Test Steps:</h5>
                    <ol className="text-gray-600 mt-1">
                      {testCase.steps.map((step, i) => (
                        <li key={i}>{i+1}. {step}</li>
                      ))}
                    </ol>
                  </div>

                  <div>
                    <h5 className="font-medium text-gray-700">Expected Results:</h5>
                    <ul className="text-gray-600 mt-1">
                      {testCase.expected.map((exp, i) => (
                        <li key={i}>• {exp}</li>
                      ))}
                    </ul>
                  </div>
                </div>
              </div>
            ))}
          </div>

          <button
            onClick={() => setCurrentStep(3)}
            className="mt-4 bg-blue-600 text-white px-6 py-2 rounded-lg hover:bg-
blue-700 transition-colors"
          >
            Validate Against FDA Standards →
          </button>
```

```
        </div>
      )}

      {/* Step 3: FDA Validation */}
      {currentStep >= 3 && selectedRequirement && (
        <div className="bg-white rounded-lg shadow-lg p-6 mb-6">
          <h2 className="text-xl font-semibold mb-4 flex items-center">
            <Shield className="mr-2 text-red-600" />
            Step 3: FDA Compliance Validation
          </h2>

          <div className="space-y-4">
            {validatedTestCases.map((testCase) => (
              <div key={testCase.id} className="border rounded-lg p-4">
                <div className="flex justify-between items-start mb-3">
                  <h4 className="font-semibold">{testCase.id}: {testCase.title}</h4>
                  <div className="flex items-center">
                    {testCase.fda_compliant ? (
                      <CheckCircle className="text-green-600" size={24} />
                    ) : (
                      <XCircle className="text-red-600" size={24} />
                    )}
                  </div>
                </div>

                {testCase.violations.length > 0 && (
                  <div className="bg-red-50 border border-red-200 rounded-lg p-3 mb-
3">
                    <h5 className="font-medium text-red-800 flex items-center">
                      <XCircle className="mr-2" size={16} />
                      FDA Violations:
                    </h5>
                    <ul className="text-red-700 text-sm mt-2">
                      {testCase.violations.map((violation, i) => (
                        <li key={i}>• {violation}</li>
                      ))}
                    </ul>
                  </div>
                )}

                {testCase.recommendations.length > 0 && (
                  <div className="bg-yellow-50 border border-yellow-200 rounded-lg
p-3">
                    <h5 className="font-medium text-yellow-800 flex items-center">
                      <AlertTriangle className="mr-2" size={16} />
                      Recommendations:
                    </h5>
                    <ul className="text-yellow-700 text-sm mt-2">
                      {testCase.recommendations.map((rec, i) => (
```

```
                        <li key={i}>• {rec}</li>
                      ))}
                    </ul>
                  </div>
                )}
              </div>
            ))}
          </div>

          <button
            onClick={() => setCurrentStep(4)}
            className="mt-4 bg-green-600 text-white px-6 py-2 rounded-lg hover:bg-
green-700 transition-colors"
          >
            Show Final Compliant Test Cases →
          </button>
        </div>
      )}

      {/* Step 4: Final Compliant Test Cases */}
      {currentStep >= 4 && selectedRequirement && (
        <div className="bg-white rounded-lg shadow-lg p-6">
          <h2 className="text-xl font-semibold mb-4 flex items-center">
            <CheckCircle className="mr-2 text-green-600" />
            Step 4: Final FDA-Compliant Test Cases
          </h2>

          <div className="mb-4 p-4 bg-green-50 rounded-lg">
            <div className="flex justify-between items-center">
              <span className="font-medium text-green-800">
                Compliance Summary for {selectedRequirement.title}
              </span>
              <span className="text-green-700">
                {compliantTestCases.length}/{validatedTestCases.length} test cases
compliant
              </span>
            </div>
          </div>

          {compliantTestCases.length === 0 ? (
            <div className="text-center p-8 text-gray-500">
              <AlertTriangle className="mx-auto mb-4" size={48} />
              <p>No test cases meet FDA compliance requirements.</p>
              <p>Test cases need refinement based on violations identified above.
</p>
            </div>
          ) : (
            <div className="space-y-4">
              {compliantTestCases.map((testCase) => (
```

```
                <div key={testCase.id} className="border-2 border-green-200 rounded-
lg p-4 bg-green-50">
                  <div className="flex justify-between items-start mb-3">
                    <h4 className="font-semibold text-gray-800">{testCase.id}:
{testCase.title}</h4>
                    <div className="flex items-center space-x-2">
                      <CheckCircle className="text-green-600" size={20} />
                      <span className="text-sm text-green-700 font-medium">FDA
Compliant</span>
                    </div>
                  </div>

                  <div className="grid md:grid-cols-3 gap-4 text-sm">
                    <div>
                      <h5 className="font-medium text-gray-700">Test Steps:</h5>
                      <ol className="text-gray-600 mt-1">
                        {testCase.steps.map((step, i) => (
                          <li key={i}>{i+1}. {step}</li>
                        ))}
                      </ol>
                    </div>

                    <div>
                      <h5 className="font-medium text-gray-700">Expected Results:
</h5>
                      <ul className="text-gray-600 mt-1">
                        {testCase.expected.map((exp, i) => (
                          <li key={i}>• {exp}</li>
                        ))}
                      </ul>
                    </div>

                    <div>
                      <h5 className="font-medium text-gray-700">Compliance Notes:
</h5>
                      <div className="text-gray-600 mt-1">
                        <p>✅ Meets FDA requirements</p>
                        <p>✅ Traceability maintained</p>
                        <p>✅ Safety mechanisms tested</p>
                      </div>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          )}

          {compliantTestCases.length > 0 && (
            <div className="mt-6 p-4 bg-blue-50 rounded-lg">
```

```
              <h3 className="font-semibold text-blue-800 mb-2">Generated Artifacts:
</h3>
              <div className="grid md:grid-cols-2 gap-4 text-sm">
                <div>
                  <h4 className="font-medium">Traceability Matrix:</h4>
                  <p className="text-gray-600">{selectedRequirement.id} →
{compliantTestCases.length} compliant test cases</p>
                </div>
                <div>
                  <h4 className="font-medium">Regulatory Coverage:</h4>
                  <p className="text-gray-600">FDA SaMD, QSR Design Controls
validated</p>
                </div>
              </div>
            </div>
          )}
        </div>
      )}

      {/* Controls */}
      <div className="mt-6 flex justify-center space-x-4">
        <button
          onClick={() => setCurrentStep(1)}
          className="px-4 py-2 border border-gray-300 rounded-lg hover:bg-gray-50
transition-colors"
        >
          Reset Demo
        </button>
        <button
          onClick={() => {

setSelectedRequirement(healthcareRequirements[(healthcareRequirements.findIndex(r =>
r.id === selectedRequirement?.id) + 1) % healthcareRequirements.length]);
            setCurrentStep(2);
          }}
          disabled={!selectedRequirement}
          className="px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700
transition-colors disabled:opacity-50 disabled:cursor-not-allowed"
        >
          Try Next Requirement
        </button>
      </div>
    </div>
  );
};

export default HealthcareTestGenerationDemo;
```

Work Distribution

Subham & Jasmeet

1. Healthcare Domain Knowledge Base

Objective: Create structured repository of regulatory and domain knowledge

Tasks:

- **Regulatory Framework Mapping**
  - FDA guidance documents parsing
  - IEC 62304 standard requirements extraction
  - ISO 14971 risk management guidelines
  - HIPAA compliance checkpoints
- **Medical Entity Recognition**
  - Drug names, dosages, medical devices
  - Patient data categories (PII, PHI)
  - Clinical workflows and procedures
  - Safety-critical requirements identification

Data Structures:

```
{
  "regulatory_standards": {
    "FDA_510k": {...},
    "IEC_62304": {...},
    "ISO_14971": {...}
  },
  "medical_entities": {
    "medications": [...],
    "devices": [...],
    "procedures": [...]
  },
  "compliance_mappings": [...]
}
```

2. Structured Prompt Templates

Objective: Create robust, context-aware prompts for requirement analysis

Core Templates:

A. Requirement Classification Prompt

```
CLASSIFICATION_PROMPT = """
You are a healthcare software requirements analyst. Analyze the following
requirement and classify it:

REQUIREMENT: {requirement_text}
DOCUMENT_CONTEXT: {document_metadata}
REGULATORY_CONTEXT: {applicable_standards}

Extract and classify:
1. REQUIREMENT_TYPE: [Functional, Non-functional, Safety, Security, Performance]
2. CRITICALITY: [Safety-critical, Mission-critical, Important, Nice-to-have]
3. REGULATORY_SCOPE: [FDA, IEC62304, HIPAA, ISO14971, None]
4. MEDICAL_ENTITIES: [List relevant medical concepts, devices, procedures]
5. TESTABILITY: [Easily testable, Requires special setup, Subjective, Non-testable]

Output as structured JSON.
"""
```

B. Entity Extraction Prompt

```
ENTITY_EXTRACTION_PROMPT = """
Extract healthcare-specific entities from this requirement:

REQUIREMENT: {requirement_text}

Identify:
- MEDICAL_DEVICES: [Specific devices mentioned]
- PATIENT_DATA: [Types of patient information]
- CLINICAL_WORKFLOWS: [Medical procedures or processes]
- DOSAGE_INFORMATION: [Drug dosages, timing, administration]
- SAFETY_CONSTRAINTS: [Risk mitigation requirements]
- PERFORMANCE_METRICS: [Measurable criteria]

Format as structured JSON with confidence scores.
"""
```

C. Compliance Mapping Prompt

```
COMPLIANCE_PROMPT = """
Map this requirement to regulatory standards:

REQUIREMENT: {requirement_text}
AVAILABLE_STANDARDS: {regulatory_knowledge_base}

For each applicable standard:
1. STANDARD_NAME: [FDA 510k, IEC 62304, etc.]
2. RELEVANT_SECTIONS: [Specific clauses/sections]
```

```
3. COMPLIANCE_LEVEL: [Must comply, Should comply, Optional]
4. VALIDATION_APPROACH: [How to verify compliance]

Output structured mapping with traceability links.
"""
```

Hemanth

Understand , evaluate and propose integration flow and validation strategies for the enterprise QA/ Google services integration

1. IMPO , what and how healthcare enterprise use to generate and manage testcases : COMPLETE IDEA

2. TOOLS

ALM (Application Lifecycle Management)

- *Jira, Azure DevOps, Polarion*: Track requirements, defects, and workflows.
- Research: How they link with test cases and CI/CD tools.

Test Management

- *TestRail, qTest, Zephyr*: Create, execute, and report test cases.
- Research: Integration with ALM and automation tools.

Requirements Management

- *DOORS, Helix RM*: Handle requirement baselining, traceability.
- Research: How they sync with test and defect tracking.

CI/CD Tools

- *Jenkins, GitLab, Azure Pipelines*: Automate build, test, deploy.
- Research: Triggering test runs and publishing results to QA tools.

🧪 QA Framework Components

1. Requirement-to-Test Traceability

- Ensure every requirement maps to test cases.
- Research: RTM formats and auto-generation.

2. Regulatory Compliance

- Align with standards like FDA 21 CFR Part 820, ISO 13485.
- Research: Audit trails, validation workflows.

3. Test Case Completeness

- Score based on coverage, priority, and risk.
- Research: Dashboards and metrics in tools.

4. Domain Expert Review

- Set up review workflows for test and requirement validation.
- Research: Approval flows and feedback loops

SUBIR + Ritu

1. Research about how current test generation tools look and feel .

2. what is the issue with them

3. 2-3 Must have features

4. 1-2 Unique features for us

5. Tech - Stack selection and proceed .

Notes

1. Explore evaluator optimization from langgraph
2. ADK & langgraph are kinda equivalent