# DATA PATTERNS, GREAT AI HACK-A-THON 2026

*"Migrate to AI"*

**SYNOPSIS**

## SECTION A: TEAM REGISTRATION

| | |
|---|---|
| **Team Name** | **Code Knights** |
| **Registration Date** | 04/02/2026 |
| **Leader Email** | balamurali.shiva@datapatterns.co.in |
| **Track Selection** | **Product Design AI** |

### Team Members

| # | Full Name | Department | Designation | Emp ID |
|---|---|---|---|---|
| 1 | Nithan R | SDG | Graduate Engineer Trainee SDG | 3537 |
| 2 | Fershia G Geona | HDD | System Engineer | 2919 |
| 3 | Yogeswaran S | SDG | Engineer SDG | 2823 |
| 4 | Balamurali | SDG | Sr.Engineer SDG | 2236 |
| 5 | Shivaram P B | SDG | Engineer SDG | 2857 |

## SECTION B: PROBLEM STATEMENT

### B1. What organizational problem are you solving?

Hardware design projects face critical inefficiencies in the development pipeline, from initial requirements to final software delivery. Engineering teams spend 60-80% of their time on repetitive documentation, manual component selection, compliance checking, and code generation tasks rather than innovative design work. This leads to:
- Extended project timelines (6-12 months for medium complexity designs)
- High error rates in specifications and netlists
- Inconsistent documentation quality
- Knowledge silos between hardware and software teams
- Delayed time-to-market for new products
- Manual code reviews consuming significant engineering time

### B2. Current Process & Pain Points:

**Current Process:**

1. Manual requirements gathering (1-2 weeks)
2. Component selection through datasheets (~2 weeks)
3. Manual schematic creation (3-6 weeks)
4. Hand-written specification documents (2-3 weeks)
5. Manual netlist generation from schematics (1-2 weeks)
6. PCB layout design (4-12 weeks) - **Future Scope for Automation**
7. Manual register map creation (1-2 weeks)
8. FPGA HDL implementation and testing (4-12 weeks) - **Future Scope for Automation**
9. Software driver development (4-8 weeks)
10. Manual code review and version control (1-2 weeks)
11. Manual testing and validation (4-8 weeks)

**Key Pain Points:**

- Engineers manually search through thousands of component datasheets
- Specifications written from scratch for each project (100+ page documents)
- No standardization across projects
- Netlist errors discovered late in PCB design phase
- Software team waits for hardware team to finalize specifications
- FPGA register maps created manually and prone to errors
- HDL code lacks standardization between projects
- Compliance checks done manually (RoHS, REACH, FCC, CE)
- Knowledge transfer difficult between team members
- High rework costs due to design errors
- Code reviews are manual, time-consuming, and inconsistent
- Version control conflicts due to poor documentation

**Additional Data:**

| | | | |
|---|---|---|---|
| **Time spent (hrs/week)** | - RF and Digital Engineers: 12-15 hours per engineer - Software Engineers: 12-15 hours per engineer - Total: ~30 hours/week across disciplines | **People affected** | - 15 RF Engineers - 8 Digital Engineers - 12 Software Engineers - Total: 35 people directly impacted |

| | | | |
|---|---|---|---|
| **Annual Inefficiency Cost** | **- Engineering time waste: 20,02,500/year (35 engineers × 12.5 hrs/week × 52 weeks × 885/hr average) - Rework costs: 7,50,000/year (design respins, component changes) - Delayed time-to-market: 12,00,000/year (opportunity cost) - Total: 39,52,500/year** | **Error Rate** | - ~18% design iterations due to specification errors or component mismatches - ~12% PCB respins due to netlist errors (Future Scope - will be addressed when PCB automation added) - ~25% software delays due to incomplete/incorrect specifications - ~15% code review rework due to inconsistent standards - Average: ~18% rework rate across current automation scope |

## SECTION C: PROPOSED AI SOLUTION

### C1. Solution Overview:

Hardware Pipeline is an AI-powered automation system that transforms hardware design from a manual, error-prone process into a streamlined, intelligent workflow. The system guides engineers through 8 phases (6 automated, 2 manual):

Phase 1-4 (Automated - 4 minutes):
1. Requirements capture & intelligent component selection using AI search
2. Automatic generation of 50-100 page Hardware Requirements Specification (HRS)
3. Compliance validation (RoHS, REACH, FCC, CE, Medical, Automotive, Military)
4. Logical netlist generation from block diagrams and schematic input (before PCB design)

Phase 5 (Manual - User PCB Design) - FUTURE SCOPE:
Engineers design PCB layout in their preferred EDA tool using the schematic input.
Currently out of automation scope; planned for Phase 2 development.

Phase 6 (Automated - 40 seconds):
6. Glue Logic Requirement (GLR) generation with I/O specifications

Phase 7 (Manual - FPGA Implementation) - FUTURE SCOPE:
7. FPGA HDL implementation and register map creation
- Currently manual: Engineers write Verilog/VHDL code
- Currently manual: Create Register Description Table (RDT)
- Currently manual: Define Programming Sequence (PSQ)
- Planned for Phase 2: Automated HDL generation from requirements
- Planned for Phase 2: Automatic register map generation
- Note: This phase is required for FPGA-based designs only

Phase 8 (Automated - 60 seconds):
8. Automatic software generation (C/C++ drivers, Qt GUI, tests, documentation) with:
- Automated code review using AI-powered static analysis
- Automatic version control integration with Git
- Code quality scoring and recommendations
- MISRA-C compliance checking (for embedded C)
- Security vulnerability scanning
- Automated test generation and coverage analysis

Key Innovation:The system generates the logical netlist BEFORE PCB design, providing engineers with a validated starting point. This eliminates the traditional workflow where netlists are extracted from schematics, reducing errors by 85%. The GLR (Phase 6) provides complete I/O specifications for FPGA implementation (Phase 7 - currently manual, future automation), ensuring seamless hardware-software integration. Additionally, automated code review and version control integration eliminate 90% of manual review time.

The AI assistant engages in natural conversation to understand requirements, suggests optimal components with 2-3 alternatives, performs complex calculations (RF link budgets, power analysis, timing), and generates production-ready documentation and code automatically. All generated code is automatically reviewed for quality, security, and standards compliance before delivery.

## C2. AI Technology Selection:

RAG (Document Search), Machine Learning, AI Agents / Automation, NLP / Text Analytics

## C3. Tools & Frameworks:

**Tools:**

| | |
|---|---|
| **AI/ML Framework:** | - LangChain for agent orchestration and RAG pipeline - Claude API (Anthropic Sonnet 4.5) - primary reasoning engine - Vector database (Chroma/Pinecone) for component datasheet embeddings - Embedding models: text-embedding-3-large for semantic search - GPT-4/Codellama/GLM-4 as fallback models |
| **LLM Model:** | - Git for version control with automated commit messages - GitHub/GitLab API integration via n8n - SonarQube for code quality metrics - Semgrep for security vulnerability scanning - Clang-Tidy for C/C++ static analysis - MISRA-C compliance checker - Pytest for automated test generation |
| **UI/Frontend:** | - React with TypeScript for web interface - Material-UI component library - Streamlit for rapid prototyping and internal dashboards - Integration with n8n webhooks for real-time updates |
| **Other Tools:** | - Python-docx for Word (.docx) documents - OpenPyXL for Excel (.xlsx) spreadsheets - ReportLab for PDF generation - Graphviz/Draw.io for block diagrams - Jinja2 for template-based document generation - NetworkX for netlist graph analysis and connectivity validation - PySpice for circuit simulation - NumPy/SciPy for RF calculations (link budget, S-parameters) - SymPy for symbolic mathematics (impedance matching) |

**Frameworks:**

Self-created/custom-built, Can run air-gapped/locally

## SECTION D: PQCDSI BUSINESS IMPACT (15% OF SCORE)

| Metric | Description | Example Format | Your Estimate |
|---|---|---|---|
| **P (30%)** | Productivity - Hours saved | *5 hrs/week x 3 people = 780 hrs/yr* | 1,365 hours/year (35 engineers × 12.5 hrs/week × 50% reduction × 52 weeks × 0.6 automation coverage) |
| **Q (20%)** | Quality - Defect/error reduction | *Reduces errors by 30%* | 85% reduction in specification/netlist errors (from 18% rework to 3%); 90% reduction in code review time |
| **C (15%)** | Cost - Direct savings (?) | *Prevents Rs.2L excess stock* | 29,50,000/year (20L labor + 7.5L rework + 2L faster time-to-market) |
| **D (15%)** | Delivery - Time reduction | *2 days to 2 hours* | 55% faster project completion for automated phases (requirements to software delivery) |
| **S (5%)** | Safety - Risk mitigation | *Predicts failure 48hrs early* | 100% compliance validation before fabrication (RoHS, REACH, FCC, CE, Medical, etc.) + automated security scanning |
| **I (10%)** | Innovation - Collaboration | *Cross-team knowledge sharing* | Unified workflow across RF, digital, and software teams with automated code review knowledge base. System learns from each project |

## D2. ROI CALCULATION

| | |
|---|---|
| **Annual Time Savings:** | **Annual Cost Savings:** |
| **Implementation Cost (Est.):** | **First Year ROI:** **- ROI = (Benefits - Cost) / Cost × 100 - ROI = (43,02,500 - 27,00,000) / 27,00,000 × 100 - ROI = 59.4% (first year) - ROI = 860% (ongoing years: 43.02L savings / 5L operating cost)%** |

## SECTION E: DATA SECURITY & COMPLIANCE (5% OF SCORE)

### E1. Data Sources

Synthetic/sample data only, Anonymized real data, Publicly available data

### E2. Synthetic Data Strategy & Deployment

Air-gapped capable, On-premise deployment

### E3. Compliance Checklist

NO live production data used, NO internal project details shared, NO employee PII shared, All IP belongs to Data Patterns Ltd., Air-gapped environment path ready, NO classified/export-controlled data, I will use only generic/sanitized data for demonstration purposes

## SECTION F: INNOVATION & CREATIVITY (25 POINTS)

### What makes your solution unique?

*How is this a novel application to YOUR specific domain problem?*

1. Pre-PCB Netlist Generation: Unlike traditional EDA tools that extract netlists FROM schematics, Hardware Pipeline generates logical netlists from block diagrams and schematic input BEFORE PCB design. This is a paradigm shift - engineers get a validated, AI-reviewed connectivity map before investing weeks in layout.
2. Unified Hardware-Software Pipeline: First tool to bridge the complete gap from initial concept to working software drivers with automated code review. Most tools stop at hardware design; we continue through to Qt applications, test suites, and automated quality assurance.
3. Universal Design Scope: Handles RF/wireless, motor control, power electronics, industrial control, sensor systems, and high-speed digital in ONE workflow. Existing tools are specialized (RF tools, motor control tools, etc.). Our AI adapts to any hardware domain.
4. Conversational Design Interface: Natural language interaction ("Design an RF amplifier with 40dBm output") instead of complex CAD interfaces. The AI asks clarifying questions and guides engineers like an experienced mentor.
5. Instant Compliance Validation: Real-time checking against RoHS, REACH, FCC, CE, Medical (IEC 60601), Automotive (ISO 26262), and Military standards with cost estimates. No other tool provides this breadth.
6. Auto-Generated Documentation: 50-100 page specifications, datasheets, compliance reports, validation checklists - all generated in under 60 seconds. This typically takes weeks of manual work.
7. Intelligent Component Selection: AI searches 500K+ components, suggests 2-3 optimal alternatives with trade-offs (cost vs. performance), checks availability, lifecycle status, and second sources automatically.
8. Built-in Design Knowledge: Encodes decades of best practices (grounding strategies, thermal management, signal integrity, EMI/EMC) and applies them automatically to prevent common errors.
9. Automated Code Review & Version Control: First hardware design tool with integrated AI-powered code review that:
- Checks MISRA-C compliance for embedded C code
- Performs security vulnerability scanning
- Validates coding standards automatically
- Generates Git commits with meaningful messages
- Tracks code quality metrics over time
- Suggests improvements based on best practices
- Reduces manual review time by 90%
10. Intelligent GLR Generation: Automatically creates Gate-Level Requirements with complete I/O specifications, voltage levels, drive strengths, and timing requirements. This GLR becomes the critical bridge between hardware design (Phase 6) and FPGA implementation (Phase 7), eliminating 80% of specification errors that typically occur at this interface.
11. Incremental Automation Strategy: Focuses on highest-value, lowest-risk automation first (documentation, code generation, component selection, GLR) while leaving complex tasks (PCB layout, FPGA HDL) as future scope. This accelerates ROI and reduces implementation risk.
12. Low-Code Workflow Integration (n8n + Playwright + AntiGravity): First hardware design tool built on modern low-code architecture:
- n8n: Visual workflow builder enables rapid iteration and customization without heavy coding
- Playwright: Reliable browser automation eliminates manual datasheet hunting across thousands of manufacturer websites
- AntiGravity: AI-powered IDE integration provides real-time code quality visualization during generation
- This stack enables non-developers to modify workflows while maintaining enterprise-grade reliability
- Self-hosted deployment ensures IP protection and air-gapped operation
- 10x faster development compared to traditional Python-only approaches

Domain-Specific Innovation:

- For RF: Automatic matching network design, link budget calculations, harmonic analysis
- For Motor Control: FOC algorithms, current sensing optimization, protection circuits
- For Power: Loop compensation, stability analysis, efficiency optimization
- For Digital: Timing analysis, signal integrity, DDR memory interfaces
- For Software: Automated driver generation with error handling, test coverage, and quality assurance

## How does this differ from existing solutions?

*What new value does your AI approach bring compared to current methods?*

vs. Traditional EDA Tools (Altium, Xpedition, KiCad):
- Traditional: Manual component selection, manual schematic capture, manual documentation
- Hardware Pipeline: AI-driven component recommendation, auto-generated specifications, logical netlist from block diagrams
- Value Add: 55% faster project completion (for automated phases), 85% fewer specification errors

vs. PLM/Requirements Tools (Jama, Polarion):
- Traditional: Text-based requirements, no component intelligence, no automatic design generation
- Hardware Pipeline: AI understands hardware context, suggests components, generates complete design from requirements
- Value Add: Seamless transition from requirements to implementation, no information loss

vs. Component Search Engines (DigiKey, Mouser, Octopart, SnapEDA):
- Traditional: Manual search, no design context, no validation
- Hardware Pipeline: AI suggests optimal components for specific application, checks compatibility, validates against design constraints
- Value Add: 10x faster component selection with better optimization

vs. Documentation Tools (Confluence, SharePoint):
- Traditional: Manual document creation, templates need customization, no validation
- Hardware Pipeline: Auto-generated specifications with calculations verified, compliance built-in, consistent format
- Value Add: 95% time reduction in documentation, zero format inconsistencies

vs. Code Generators (vendor HAL libraries, STM32CubeMX):
- Traditional: Generic drivers, no application context, manual integration required, no code review
- Hardware Pipeline: Application-specific drivers with automated code review, complete with error handling, tests, documentation, and quality scoring
- Value Add: Production-ready, reviewed code in seconds vs. days of development and manual review

vs. Code Review Tools (SonarQube, Gerrit, GitHub PR reviews):
- Traditional: Separate tool, manual configuration, reactive (reviews after code is written), no hardware context
- Hardware Pipeline: Integrated into generation pipeline, proactive (reviews as code is generated), understands hardware-software interaction, learns from design patterns
- Value Add: Zero setup time, hardware-aware review rules, instant feedback

vs. Version Control Systems (Git, SVN):
- Traditional: Manual commit messages, no automation, requires developer discipline
- Hardware Pipeline: Auto-generated meaningful commit messages, automatic branching strategy, tracks design evolution with hardware context
- Value Add: Perfect version history, no missed commits, design traceability

vs. Workflow Automation Platforms (Zapier, Make, n8n standalone):
- Traditional: Generic automation, no hardware domain knowledge, requires building from scratch, limited AI

integration, cloud-only
- Hardware Pipeline: Hardware-specific workflows built on n8n, pre-configured 8-phase pipeline, deep Claude API integration, self-hosted/air-gapped capable
- Value Add: 90% pre-built vs. starting from scratch, hardware expertise embedded, one-click deployment, on-premise security

vs. Browser Automation Tools (Selenium, Puppeteer, standalone Playwright):
- Traditional: Brittle scripts requiring constant maintenance, 60-70% reliability, manual selector updates when websites change
- Hardware Pipeline: Playwright with AI-guided selectors integrated into n8n, self-healing automation, 95% reliability, parallel execution across 100+ manufacturer sites
- Value Add: Zero maintenance for website changes, 5x faster datasheet collection, automatic retry logic

## SECTION G: PRESENTATION PLAN

**Demo Format**

| | |
|---|---|
| **How will you demonstrate?** | Live working demo, Recorded demo + explanation, Slides with screenshots |
| **Resources Required** | Hardware: - Laptop with projector connection (HDMI) - Internet connection (for component database API and Git integration) - Backup: Local database for offline demo Software: - Web browser (Chrome/Firefox) - PDF viewer for document preview - Excel for spreadsheet preview - VS Code for code preview (with code review annotations) - Git GUI client (GitKraken or similar) for version control visualization Data: - Sample design requirements (motor controller, RF system, digital controller) - Synthetic component database (10K parts cached locally) - Pre-loaded compliance rules - Sample Git repository for version control demo Backup Plan: - Recorded video demo (if live demo fails) - Screenshots of key screens including code review results - Printed samples of generated documents and code review reports - Offline mode with cached API responses |
| **Estimated Demo Duration** | 30 minutes |