OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Hackathon Guide Document

Theme 2026: "IoT is Bold" and "Old is Gold"

Contents

Field	Details
Document Title	Comprehensive Hackathon Guide: "PI the IoT" & "Old is Gold"
Document Version	1.0
Last Updated	December 31, 2025
Status	Final Release
Author(s)	School of Computer Science – Dr. Waqar Shahid Qureshi

Introduction

This guide provides comprehensive structure and expectations for a second semester Computing Systems (CT101) hackathon, carrying 15% of the total module grade. The hackathon is a collaborative learning experience where students apply self-learning to practical projects while developing teamwork, technical implementation, and presentation skills. By completing this hackathon, students will apply computing systems concepts to real-world projects, develop collaborative software and hardware development skills, practice requirements gathering and system design, build presentation and technical communication abilities, experience project management and team coordination, and gain hands-on experience with either IoT technologies or legacy system modernization depending on their chosen theme.

Hackathon Structure and Phases:

The hackathon is organized into four distinct phases spanning approximately 12 weeks of the academic term. Each team will maintain a **GitHub** account documenting their activities across all project phases, including deliverable documents, code repositories, and progress tracking. This version control system serves as both a project management tool and evidence of sustained effort throughout the hackathon lifecycle.

Phase 1: Team Formation & Ideation establishes the foundation for project success. Teams of 2-3 students form with a designated team lead and establish initial project direction. Deliverables include team members, workload distribution and responsibilities, a one-page project abstract (maximum 500 words) articulating the problem statement, proposed solution, and selected themes. This phase emphasizes clarity of vision and team role definition rather than extensive documentation. Teams should be formed by the end of Week 1 to allow sufficient time for team bonding and initial project ideation.

Phase 2: Functional Requirements develop deeper technical understanding. Teams produce detailed functional requirements documents spanning 4-5 pages, presenting core functional requirements, use cases, data flow diagrams, interface mockups, and preliminary testing strategy for the functional requirement. The document reflects refined understanding of the project scope. Work distribution documentation is updated to reflect any evolution in team roles as the project scope becomes clearer.

Phase 3: Implementation & Presentation represents the intensive execution period (**Week 9-10**). At the start of the week, the team leader will issue their hardware from the Teaching or Technical Staff. During a dedicated one-week period, teams finalize their implementations and prepare a demo and presentations. Deliverables consist of a working project demonstration (either live or in some cases recorded) and a 5-minute presentation-demo followed by 3-minute Q&A session delivered during the 3-hour hackathon event held in weeks 11 (Lab hours only). Individual participation and technical contribution are assessed during this phase, with each team member evaluated separately based on their demonstrated involvement and expertise.

Phase 4: Reflection & Final Report completes the learning cycle. Teams submit a comprehensive design document (8-12 pages) detailing system architecture, technical implementation decisions, encountered challenges with solutions, results achieved, and critical evaluation of outcomes. Each team member provides a final reflection covering learning outcomes, team dynamics, and actual work distribution—documenting how responsibilities evolved and what each person ultimately contributed throughout the project lifecycle. All code and documentation should be maintained on GitHub for tracking progress, with a soft copy of the final report also available on GitHub for version control and accessibility.

Use of Generative AI in the Hackathon

Generative AI tools have become valuable assistance in modern software development and technical communication. Students may leverage generative AI for specific, well-defined purposes while maintaining academic integrity and demonstrating their own understanding.

During Phase 1: Team Formation & Ideation, students should **NOT** use generative AI for brainstorming, problem exploration, or solution generation. This phase is critical for authentic ideation where teams must independently develop their problem understanding, explore solution space, and make original design decisions.

During Phase 2: Functional Requirements, students should **NOT** use generative AI for technical writing, document structure, content generation, or report composition. Teams must independently develop their functional requirements documentation, articulate use cases in their own words, and structure their technical analysis.

From Phase 3: Implementation & Presentation forward, students **may use** generative AI for code generation, debugging, and technical assistance as outlined above. All AI-assisted work must include proper attribution and disclosure.

During Phase 4: Reflection & Final Report, students should **NOT** use generative. All original analysis, results interpretation, and critical reflection must be the students' own work. You may **use** Gen AI to improve your sentences and coherence; however, this must be documented in the report.

The goal is to develop competent technical professionals who can leverage modern tools effectively during execution phases while demonstrating authentic independent thinking, problem-solving, and communication skills during foundational project phases.

Assessment Approach

Assessment is distributed across four phases spanning 12 weeks. Phase 1 (Team Formation & Ideation) comprises **10%** of the hackathon grade (1.5% of module), focusing on authentic problem definition and team organization without AI assistance. Teams document their composition, role assignments, work distribution, and submit a concise one-page project abstract. Phase 2 (Functional Requirements) carries **25%** weight (3.75% of module), requiring independent analysis and original technical writing. Teams produce a 3–5-page functional requirements document including core requirements, use cases, data flow diagrams, and mockups—demonstrating their own understanding of project scope and technical feasibility. Phase 3 (Implementation & Presentation) receives the highest weight at **35%** (5.25% of module), reflecting the critical execution period where teams deliver working projects and present findings during the 3-hour hackathon event. Phase 4 (Design Document & Reflection) accounts for **30%** (4.5% of module), requiring comprehensive technical analysis and individual reflection on learning outcomes and actual contributions.

While the hackathon is fundamentally a collaborative learning experience, individual performance is assessed separately throughout all phases. Each team member's documented work distribution in Phases 1, 2, and the final reflections in Phase 4 establishes accountability for personal contribution.

Each phase employs a consistent 1-5 point rubric scale, where a score of 5 represents exceptional work exceeding expectations with advanced understanding and original thinking, a score of 4 indicates good work meeting expectations with clear competency and sound technical decisions, a score of 3 reflects satisfactory work meeting basic requirements with adequate understanding, and scores of 2 and 1 indicate work falling short or significantly below expectations respectively. The four independent phase scores are then combined using weighted averaging to produce the overall hackathon performance score. This calculation follows the formula:

Weighted Hackathon Score = (Phase 1 score × 0.10) + (Phase 2 score × 0.25) + (Phase 3 score × 0.35) + (Phase 4 score × 0.30)

This weighted hackathon score, expressed on a 5.0 scale, is then converted to determine the hackathon's contribution to the overall module grade. The conversion formula is:

Hackathon Grade (%) = (Weighted Hackathon Score / 5.0) × 15%.

For example, if a team achieves Phase 1 (4.0/5.0), Phase 2 (3.5/5.0), Phase 3 (4.5/5.0), and Phase 4 (4.0/5.0), their weighted hackathon score calculates as (4.0 × 0.10) + (3.5 × 0.25) + (4.5 × 0.35) + (4.0 × 0.30) = 4.075/5.0. This weighted score of 4.075 converts to (4.075 / 5.0) × 15% = 12.225% contribution to the final module grade.

Any instances of improper use of generative AI or other forms of academic misconduct detected during assessment will be addressed in accordance with the university's academic integrity procedures and disciplinary policies.

Project Themes and Hardware

Theme 1: "IOT is Bold" – Raspberry Pi to an IOT device

The "IoT is Bold" track emphasizes hands-on experience with embedded systems and sensor-based applications appropriate for first-year computing students. Teams select from distinct IoT project categories to build, learn, and optimize. Each category teaches different aspects of IoT architecture: sensing environmental data, processing information, and creating intelligent responses through hardware-software integration. You can use python programming and associated libraries.

Available Hardware: 10 Raspberry Pi systems (Pi 4 models) with Sense HAT boards (Official documentation) and Camera Module v3, along with necessary USB power cables and HDMI display cables. Each team will be issued 1x Raspberry Pi kit with all required peripherals. The Raspberry Pi Sense HAT is an add-on board giving comprehensive sensing and display capabilities. Originally developed for the International Space Station's educational Astro Pi program, it includes temperature sensor, humidity sensor, barometric pressure sensor, motion and orientation sensor, 8 x 8 LED for user interface, and 5 button mini joystick.

Thematic Categories

Environmental, Remote Monitoring and Surveillance Systems: Build IoT devices that sense, log, and respond to environmental conditions using temperature, humidity, and pressure sensors. Monitor temperature/humidity; capture photos when significant changes are detected; create time-lapse sequences. Use accelerometer to detect disturbance; trigger camera capture; timestamp and store evidence photos.

Motion Detection and Interaction Systems: Build applications using accelerometers, gyroscopes, and magnetometers to detect movement, orientation, and direction. Use accelerometer to show device tilt on LED matrix; create digital spirit level; log orientation changes. Use gyroscope to detect rotation patterns; control LED animations with physical gestures; create simple gesture-based games.

Interactive Display and Gaming: Build interactive applications using LED matrix, joystick, and sensors for user engagement. Create game controlled by tilting Raspberry Pi (marble maze, balance game); display game state on LED matrix; use joystick for menu navigation.

Data Logging and Analysis Systems: Build systems that collect sensor data over extended periods for analysis and insight generation. Collect all sensor readings every 5 minutes; store in structured format (CSV); generate statistical summaries and graphs.

Theme 2: "Old is Gold" – Legacy Server Systems and Infrastructure

The "Old is Gold" track emphasizes hands-on experience with different server architectures and their appropriate use cases in institutional settings. Teams select one of six distinct web server types to build, learn, and optimize. Each category teaches different architectural principles, performance considerations, and real-world deployment challenges. You can choose any of the following theme or use your own theme.

Available Hardware: 20 older PC systems suitable for server deployment with flexible network infrastructure on request. Each team will be issued 1x Desktop with required power cables.

Thematic Categories:

Web Server Type 1: Serves pre-built HTML, CSS, and client-side JavaScript files directly from disk. Content is identical for all visitors and does not change based on user interaction, e.g., website for each year of CS undergraduate and graduate cohort

Web Server Type 2: Generates page content in real-time based on database queries and user requests. Different users see different content, e.g., Student Assignment Submission & Grading System.

Web Server Type 3: Streams audio and video content to multiple simultaneous users. Handles format conversion, adaptive quality adjustment, and access control.

Web Server Type 4: Provides searchable access to large collections of documents and data. Implements advanced filtering, faceted navigation, and rapid content discovery.

Web Server Type 5: Manages multiplayer game sessions with real-time player synchronization. Handles player authentication, game state updates, matchmaking, and persistent progression.

Web Server Type 6: Executes complex business logic and manages intricate multi-step workflows. Handles high concurrency, transaction management, and integration of multiple systems, e.g. Simplified Student Information Management System

Web Server Type 7: A chat server manages real-time message exchange between multiple clients connected simultaneously. Uses persistent connections (WebSocket) instead of traditional HTTP polling for instantaneous message delivery, e.g., "Campus Community Chat Platform".

Project Submission Requirements

Submission 1: Project Abstract (Due: End of Week 4)

Purpose: Establish project direction and secure theme/hardware allocation

Format: 1-page document (500 words maximum)

Required Elements:

- Project title and team name
- Chosen theme ("PI the IoT" or "Old is Gold")
- Problem statement (what challenge or opportunity are you addressing?)
- Proposed solution (high-level overview)
- Team member names and initial role assignments

Submission 2: Functional Requirements Document (Due: Mid-Semester, Week 6-7)

Purpose: Define detailed project specifications and success criteria

Format: 4-5 pages with rubric included

Required Sections:

Section	Length	Content
1. Introduction	0.5 pages	Project context, motivation, target users
2. Functional Requirements	1.5-2 pages	Detailed list of features, use cases
3. Non-Functional Requirements	1 page	Performance, security, scalability, reliability
4. Technical Architecture	0.5-1 page	System design, hardware/software components
5. Success Criteria	0.5 page	Measurable metrics for project success
6. Risk Analysis	0.5 page	Identified risks and mitigation strategies

Table 1: Functional Requirements Document Structure

Phase-2 Functional Requirement Document Rubric:

Criterion	**Excellent (5)**	**Good (4)**	**Satisfactory (3)**	**Needs Work (2)**	**Poor (1)**
Requirement Clarity	All requirements are specific, measurable, and testable	Most requirements clear with minor ambiguity	Requirements present but some lack specificity	Many requirements unclear or vague	Requirements poorly defined or missing
Completeness	All functional and non-functional requirements documented	Most categories covered with minor gaps	Some requirements documented; gaps exist	Significant gaps in documentation	Incomplete or minimal documentation
Technical Feasibility	Project scope is well-matched to timeline and team capability	Generally feasible with minor concerns	Mostly feasible with some challenges	Some feasibility concerns not addressed	Scope unclear or unrealistic
Architecture Design	Clear, well-reasoned system design with appropriate technology choices	Logical design with mostly appropriate technology	Basic design present with some tech choices unexplained	Design lacks clarity; tech choices not justified	No clear architecture provided

Submission Details: PDF format submitted via GitHub link and canvas document submission.

Submission 3: Design Document (Due: After Hackathon Presentation, Week 12-14)

Purpose: Document implementation decisions, results, and learning reflection

Format: Comprehensive document with multiple sections

Required Sections:

1. **Methodology** (2 pages): Development approach, timeline, team collaboration methods, agile practices used
2. **Detailed Implementation** (2-5 pages): Technical decisions, code architecture, hardware integration details, database schema (if applicable), API designs
3. **Challenges and Solutions** (1 page): Problems encountered, debugging approaches, how team overcame obstacles
4. **Results and Demonstration** (2 pages): What works, feature demonstrations, performance metrics, user testing feedback
5. **Reflection and Learning** (1-1.5 pages): Key learnings, skills developed, what you would do differently, future improvements

Submission Details: PDF format, 8 -12 pages total, include screenshots/diagrams, and GitHub link

Phase 4: Design Document Rubric (Post-Event Reflection)

Criterion	Weight	Excellent (5)	Good (4)	Satisfactory (3)	Needs Work (2)	Poor (1)
Technical Documentation	25%	Detailed, well-organized; all decisions explained; diagrams clear	Good documentation; most decisions explained; adequate diagrams	Basic documentation; some gaps; adequate detail	Incomplete documentation; missing explanations	Minimal or missing documentation
Implementation Details	20%	Thorough explanation of approach; code samples or pseudocode; clear design rationale	Clear explanation of approach; mostly justified decisions	Adequate explanation; some justification	Lacks detail in implementation	Implementation not explained
Problem-Solving Narrative	20%	Detailed challenges; creative solutions; learning extraction evident	Clear challenges and solutions; some learning reflected	Basic problem-solving narrative; limited reflection	Superficial discussion; minimal learning	No reflection on challenges
Results and Testing	20%	Comprehensive results; thorough testing documented; metrics provided	Good results; testing described; basic metrics	Adequate results; limited testing documentation	Results unclear; minimal testing evidence	No results provided
Reflection and Learning	15%	Deep reflection; significant learnings articulated; thoughtful future improvements	Good reflection; clear learnings; reasonable improvements	Adequate reflection; some learnings stated	Superficial reflection; limited insight	No meaningful reflection

GitHub Repository Structure

```
├── README.md (project overview, setup guide, team members)
├── requirements.txt (Python dependencies)
├── src/ (source code)
├── docs/ (design documents, diagrams)
├── hardware-setup/ (GPIO diagrams, wiring guides)
├── data/ (sample data, logs)
└── CHANGELOG.md (version history, progress updates)
```

Deadline	**Week**	**Deliverable**	**Format**
End of Week 4	4	Project Abstract	1 page, 500 words
Mid-week 6-7	6-7	Functional Requirements	4-5 pages
Mid-week 8	8	Hardware Checkout	In-person
Week 10 (TBA)	10	Hackathon Presentation + GitHub	Live event + demo
End of Week 12	12	Hardware Return	In-person
End of Week 14	14	Final Report	8-12 pages + reflections

Hackathon Presentation Event (Week 10)

Event Format: 3-Hour Intensive Showcase

Schedule

Time Block	Duration	Activity
0:00-0:30	30 min	Doors open, final setup, tech check, Final development/integration and feature completion
0:30-3:50	200 min	Team presentations (5 min per team + 3 min Q&A)
3:50-4:00	10 min	Judging deliberation and winner announcement
4:00+	30 min	Prize distribution and celebration with refreshments

Phase 3: Hackathon Presentation Rubric

Criterion	Weight	Exemplary (5)	Proficient (4)	Developing (3)	Beginning (2)	Incomplete (1)
Demonstration Quality	40%	Fully functional, impressive demo; smoothly handles unexpected issues	Working demo of core features; minor glitches handled well	Core features demonstrated; some issues occur	Limited demonstration; significant issues	No meaningful demo
Presentation Clarity	30%	Compelling narrative; excellent pacing; audience fully engaged	Clear explanation; good pacing; audience engaged	Adequate explanation; some pacing issues	Unclear explanation; poor pacing	Incoherent or missing
Technical Communication	20%	Expertly explains technical approach; judges understand implementation	Clearly explains approach; judges understand basics	Explains approach but some technical details unclear	Limited technical explanation	No technical explanation
Time Management	10%	Precisely fits 3-minute window; excellent flow	Uses time well; minimal pauses	Time used appropriately with minor issues	Significantly over/under time	Severely over/under time

Table 2: Hackathon Event Timeline

Presentation Requirements

Each team receives **8 minutes total** (5 minutes presentation + 3 minutes questions):

Judging Panel

Composition: 4-5 judges including:

- Faculty members with IoT/systems expertise
- Faculty member with business/impact focus
- GTA representative

Judge Responsibilities: Evaluate presentations using standardized rubric, ask clarifying questions, provide constructive feedback

Hardware Checkout Process

- Teams sign hardware checkout forms. Hardware assigned mid-Week 8 after functional requirement documentation. Designated hardware owner per team (responsible for safekeeping). Return by end of Week 12 with inspection

Prizes and Recognition

1. **Best Bold and Gold** – Technical excellence, innovation, and impact
2. **Best Bold Implementation** – Creative hardware use and sensor integration
3. **Best Gold Implementation** – Effective infrastructure or system design

Reference Documents and Resources

1. https://www.raspberrypi.com/documentation/
2. https://www.raspberrypi.com/documentation/
3. https://sense-hat.readthedocs.io/en/latest/
4. https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat
5. https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf
6. https://www.geeksforgeeks.org/postgresql/postgresql-tutorial/
7. https://www.w3schools.com/html/
8. https://www.w3schools.com/css/
9. https://www.w3schools.com/js/
10. https://socket.io/get-started/chat
11. https://talent500.com/blog/implementing-a-real-time-chat-application-using-websockets-and-socket-io/
12. https://realpython.com/tutorials/flask/

FREQUENTLY ASKED QUESTIONS (FAQ)

Q1: How do I form a team?

Team formation happens during Week 1 of the hackathon. Identify 1-2 other students to form a team of 2-3 people. Complete Abstract Form (Due: End of Week 1), that along with other project idea also includes team name, member names/IDs, primary contact, theme selection, role assignments. Create a shared GitHub repo for code, documentation, and work logs

Q2: What's the ideal team composition?

Teams consist of 2-3 students. For 3-person teams divide the role of code development, architecture, hardware or database, system testing, reports, presentations, and GitHub documentation. You will have at least one Team lead.

Q3: Can I change my team after Week 1?

Changes allowed in teams before Week 6. Changes after Week 6-8 are possible but discouraged, requires instructor approval. After Week 8, NO team changes are allowed except for medical/emergency circumstances only.

Q4: What if a team member stops participating?

Follow escalation process: (1) Team addresses it directly (2) GTA intervention if unresolved after 1 week (3) Instructor intervention if still unresolved. Remaining team continues with reduced scope proportionally.

Q5: How do I choose between "IoT is Bold" and "Old is Gold"?

IoT is Bold is best for students who enjoy hardware, sensors, real-time data processing, and Python with embedded systems. Old is gold suits those who prefer database design, SQL, web interfaces, system modernization, and multi-technology architecture. Decide by Friday of Week 1.

Q6: Can everyone do "PI the IoT"? What if too many teams want the same theme?

NO - Limited hardware constrains theme selection.

Theme	Hardware Available	Max Teams	Students
PI the IoT	10-15 Raspberry Pi kits (BOLD)	15 teams MAXIMUM	2 – 3 Students
Old is Gold	20 PC systems	15 teams MAXIMUM	2 – 3 Students

Q7: Can my team change themes after registering?

Theme changes allowed before abstract submission (Week 1-5) with no penalty. After Week 5, NO theme changes are allowed except if PI kit becomes available and instructor approves.

Q8: When do I get my hardware? Can I keep it longer?

Hardware checkout: Mid-Week 8. Available for use: Weeks 8-12. Return deadline: End of Week 12. Start coding before getting hardware using simulators if required (Weeks 1-7). Final integration with actual hardware occurs Weeks 8-10.

Q9: What happens if my hardware breaks?

Damage assessment varies: No damage ($0), minor cosmetic ($0), repairable or damaged (Will be decided by the school CTO). Teams are responsible during checkout period. Normal wear covered. Report damage within 24 hours with photos.

Q10: Can I use my own hardware instead of checking out from school?

IoT is Bold Track: You may use your own Raspberry Pi 4 but must check out official Sense HAT board from school. Old is Gold Track: Only use the provided PC. Notify instructor by end of Week 1 if using personal sensors.

Q11: What if my abstract gets rejected?

Abstracts rarely get rejected; they receive feedback. Outcomes: Approved (proceed to Phase 2), Approved with Feedback (incorporate feedback, no resubmission), Revision Requested (rewrite within 48 hours), Rejected (choose different project, resubmit within 3 days).

Q12: How is my grade determined?

Four phases weighed: Phase 1 Abstract (10% - end Week 4), Phase 2 Requirements (25% - mid-week 6-7), Phase 3 Presentation (35% - Week 10), Phase 4 Design Document (30% - end Week 14). All scored 1-5 scale. Hackathon = 15% of overall module grade.