

CODE CRAFT SOCIETY PRESENTS

RE-INNOVATEX

Reverse Engineering Hackathon · Workflow Guide & Rulebook



Understand → Deconstruct → Reconstruct → Innovate → Demo

DATE

Thursday, October 15, 2026

TIME

9:00 AM – 5:00 PM

VENUE

**MG-505, ITM University,
Gwalior**

PARTICIPATION

**Team · Open to all branches ·
2–4 members**

REGISTRATION FEE

₹100 per team

FACULTY COORDINATOR

Dr. Geetanjali Surange

FACULTY COORDINATOR

Mr. Shubham Dhakarey

FACULTY CO-COORDINATOR

Mr. Vipin Jadon

STUDENT COORDINATOR

Garvit Mishra

Contact : +9175809 89439

STUDENT CO-COORDINATOR

Ujjwal Gupta

Contact : +91 91110 16526

RULES & REGULATIONS

Latest updates — read with this rulebook

- Registration for TechUtsav 1.0 is through the ITM University Parakh page linked on this site. The registration-form link printed in the guide and booklet is not reproduced here.
- Still to be announced by the organizers: the challenge-specific rules for AI tools and internet access, the submission format and exact Day 1 deadline, and the Day 2 start time, presentation length per team and presentation order.
- The official poster adds Round 2 details: 30 minutes of thinking / preparation time and 5 minutes of presentation time.

Part 1 · Hackathon Workflow Guide — who does what, and when

1. THE EVENT IN ONE PAGE

Re-InnovateX is a team event (2–4 members) in which teams receive a broken, incomplete or partially working system without its source code. They must follow one loop:

Understand → Deconstruct → Reconstruct → Innovate → Demo

Item	Detail
Day 1 — Challenge	15 October 2026, 9:00 AM – 5:00 PM (8 hours, offline)
Day 2 — Presentations & Evaluation	16 October 2026 (about 1 hour presentations + 1 hour evaluation and results)
Registration deadline	13 October 2026
Entry fee	₹100 per team
Prize pool	₹3,700 (1st ₹2,100 • 2nd ₹1,100 • 3rd ₹500)
Organizer	Code Craft Society, with faculty coordinators Dr. Geetanjali Surange, Mr. Shubham Dhakarey and co-coordinator Mr. Vipin Jadon
Student coordinators	Garvit Mishra (+91 75809 89439), Ujjwal Gupta (+91 91110 16526)

Day 1 phase timeline

Time	Phase	Main goal
9:00 – 11:00 AM	Phase 1 · Understand & Observe	Learn what the system does and what is broken
11:00 AM – 1:30 PM	Phase 2 · Deconstruction	Work out components, workflow, data flow, failure points
1:30 – 5:00 PM	Phase 3 · Reconstruction & Innovation	Rebuild the baseline, then improve it; prepare documentation and demo

How the three roles fit together

Stage	Organizers	Participants	Evaluators
Before the event	Design challenge, set up environment, open registration, brief judges	Form team, register, pay, prepare tools and backups	Agree on rubric, study the challenge answer key
Day 1	Brief, release package, keep order, support, monitor, collect submissions	Observe → deconstruct → reconstruct → innovate, with notes and backups	Optionally check progress; may run hidden tests on submissions
Day 2	Run schedule, arrange demos, keep time	Present, demo, answer questions	Score, question, apply tie-breaks
After	Announce results, award prizes, collect feedback	Receive feedback, certificates and prizes	Give feedback, hand in final scores

2. THE ORGANIZER'S VIEW

Organizers own the fairness, the environment and the logistics. Their job is to make sure every team gets the same challenge, the same rules, and a working place to compete.

2.1 Before the event

- 1. Open registration.** Publish the official form and the rulebook. Registration closes on 13 October. A team is registered only after the form is done and the fee is paid. Every member must be on the form.
- 2. Design the challenge package.** Prepare a broken, incomplete or malfunctioning system without original source code. Prepare the participant kit (the system plus whatever records or logs teams may use) and the organizer kit (the answer key with the true rules, a hidden test set, a scoring script, a reference solution).
- 3. Decide the challenge-specific rules** and announce them at the start: permitted tools, whether AI and the internet are allowed, submission format, and any restrictions. These override the general rulebook.
- 4. Prepare a controlled environment.** Machines or labs isolated from college infrastructure, with backups and a technical-support contact for downtime.
- 5. Brief the judges.** Share the rubric, the demo structure, the penalty framework and the answer key. Decide who runs the hidden test.
- 6. Confirm logistics.** Venue, power, internet policy, ID checks, attendance sheets, timers, printed booklets, certificates and prize money.

2.2 Event day (15 October)

Time	Organizer action
Before 9:00	Check attendance and IDs, seat teams, confirm environments work
9:00	Short briefing: rules, restrictions, deliverables, judging. Release the challenge package
During Phase 1	Watch for questions on permitted tools; enforce observe before assuming
11:00	Announce the move to Phase 2; make any checkpoint announcements
1:30	Announce the move to Phase 3; remind teams to establish the baseline before innovating
Final hour	Remind about the submission deadline; walk through the submission package
5:00	Close submissions; confirm each team has handed in its package; secure copies

Continuous duties on Day 1

- Answer clarification questions publicly so every team hears the same answer.
- Apply the downtime policy: if an environment breaks, teams report it, organizers restore or replace the resources, and for major issues announce a fair adjustment.
- Run reasonable monitoring for integrity. Teams may later be asked to explain their reasoning.
- Handle violations using the framework: minor (warning), serious (score penalty or rejection), critical (immediate disqualification).
- If a team reports an unintended access path, thank them and fix it. The rule is that teams stop and report rather than exploit.

2.3 Day 2 (16 October)

1. Publish the presentation order and time slots.
2. Set up the demo area (projector, machine, backup laptop).
3. Run presentations on time (about 1 hour), then give judges evaluation time (about 1 hour).
4. Collect scoresheets, check totals, apply any announced tie-break, and confirm the results with faculty coordinators.
5. Announce winners and award prizes and certificates.

2.4 After the event

- Share feedback with teams, archive submissions and scoresheets, and record lessons for next year.
- Update the challenge: change seeds, rules and data so a repeat run is not leaked.

3. THE PARTICIPANT'S VIEW

Participants own the thinking, the rebuild and the explanation. The rulebook's core advice: **do not jump directly into coding.**

3.1 Before 15 October

- Form a team of 2–4 and register before 13 October; save the confirmation and the coordinator numbers.
- Pay the ₹100 team fee as instructed.
- Agree on roles (they are flexible): System Observer / Documentation Lead, Technical Analyst, Developer / Reconstruction Lead, Testing / Innovation Lead.
- Prepare tools you are allowed to use, set up shared storage and a backup routine, and be ready to work offline for 8 hours.

3.2 Day 1 — what to do in each phase

Phase	What to do	What to produce
1. Understand & Observe (9:00–11:00)	Run and interact with the system. Record inputs, outputs, screens, errors and files. Separate normal from broken behaviour. List visible modules. Form first hypotheses	An observation log and a behaviour/workflow map: a clear idea of what the system is for and an initial problem statement
2. Deconstruction (11:00–1:30)	Inspect exposed files and config. Trace workflows and data flow. Test inputs and edge cases. Compare expected with observed behaviour. Map components, dependencies and failure points	A reasoned model of how the system works, with each claim backed by evidence; facts clearly separated from assumptions
3. Reconstruction & Innovation (1:30–5:00)	Rebuild the baseline first. Test against what you observed. Back up working versions. Only then add a meaningful improvement. Write the documentation and the demo guide	A working or substantially rebuilt system, an innovation summary, documentation and a tested demo

Habits that win

- Keep a running log from minute one; do not try to recreate the story at the end.
- Every rule you claim should have at least one piece of evidence.
- Do not claim a feature you cannot demonstrate. Say honestly what does not work.
- Test normal inputs and strange ones, and make sure the innovation does not break the baseline.
- Stay inside the challenge environment. If you find a way out or a way in that was not intended, stop and tell an organizer.

3.3 Submission (end of Day 1)

1. The reconstructed system, plus the repository or package in the format announced.
2. Technical documentation: System Understanding, Failure Analysis, Deconstruction, Reconstruction, Testing, Innovation, Limitations, Demo Guide.
3. A short innovation summary, clearly separate from the baseline.
4. Final check using the submission checklist, then submit before the deadline and keep a backup.

3.4 Day 2 — presentation

Follow the recommended structure: what we observed → the key failure → our analysis and reconstruction → demo of the rebuilt functionality → testing and edge cases → innovation → limitations. **Every member must be able to answer questions about the whole solution, not only their own part.**

3.5 Rules to remember

- No original source code through unauthorized means, no leaked solutions, no outside help beyond what is allowed.
- AI tools and the internet only if the challenge instructions permit them.
- Do not touch other teams' work, college systems, or external sites.

4. THE EVALUATOR'S VIEW

Evaluators own fair, evidence-based scoring. The key idea is that they judge understanding and reasoning, not only whether the screen looks right.

4.1 Before the event

- Read the rulebook, the challenge and the answer key so that “correct behaviour” is clear.

- Agree on how to score each criterion and how to treat partial solutions.
- Decide how the hidden test will be run and who runs it.

4.2 Scoring rubric

Criterion	Weight	What to look for
Understanding	20%	Accurate observations of inputs, outputs and expected behaviour; clarity on what the system is for
Technical Analysis	20%	Depth and correctness of deconstruction: components, workflow, dependencies, data flow, failure points; evidence-backed
Reconstruction	30%	Correct, stable, tested rebuild of the required behaviour; verified during the demo
Innovation	20%	Meaningful, demonstrable improvements beyond the baseline (features, performance, UX, accessibility, security, AI or automation, validation)
Documentation & Demo	10%	Clear reasoning, test evidence, reproducible demo steps, quality of explanation

4.3 The evaluation flow

- 1. Baseline check.** Verify the required functionality in the demo. If a hidden test batch is used, run it and record the result.
- 2. Evidence check.** Does the team's reconstruction follow from its observations? Were rules inferred from evidence, or guessed?
- 3. Questions.** Ask technical questions of different members. A team that cannot explain its own solution should lose marks, whatever the demo looked like.
- 4. Innovation, scored separately.** Judge it on its own, after confirming that the baseline still works.
- 5. Documentation.** Check that observations, hypotheses, implementation and innovation are kept apart.
- 6. Consistency.** Apply the same criteria to every team.

4.4 Red flags to watch for

- Claims that cannot be demonstrated, or features that fail live.
- Work that looks copied, or a solution only one member understands.
- Signs of prohibited practices: obtaining source code, using leaked solutions, outside collaboration, tampering with monitoring.

Suspected violations follow the penalty framework: minor (warning or small score impact), serious (score penalty, rejection or disqualification), critical (immediate disqualification and escalation).

4.5 Ties and final decision

If scores tie, the sequence is: reconstruction functionality → technical analysis → innovation → documentation and demo quality, unless a different tie-break is announced before judging. The judges' and organizers' decision is final.

5. THE WHOLE EVENT AS ONE FLOW

Step	Who leads	Output
1. Registration	Organizers + participants	Verified team list and fees
2. Challenge prepared	Organizers	Participant kit, organizer kit, hidden test, rubric
3. Judge briefing	Organizers + evaluators	Shared understanding of the answer key and scoring
4. Briefing and release (9:00)	Organizers	Rules and challenge package in teams' hands
5. Phase 1: Understand	Participants	Observation log, behaviour map
6. Phase 2: Deconstruct	Participants	Evidence-based model of the system
7. Phase 3: Reconstruct and innovate	Participants	Working rebuild, innovation, documentation
8. Submission (5:00)	Participants → Organizers	Secured packages
9. Hidden test (optional)	Evaluators	Objective baseline score
10. Presentations	Participants + evaluators	Demos and question-and-answer
11. Scoring and tie-breaks	Evaluators + organizers	Final ranking
12. Results and prizes	Organizers	Winners, certificates, feedback

Part 2 • Participant Booklet — the detailed rules

1. EVENT AT A GLANCE

The system is broken. The ₹3,700 prize is waiting. Can your team fix it?

Item	Details
Event	Re-InnovateX — Reverse Engineering Hackathon
Organizer	Code Craft Society
Tagline	Break • Analyze • Rebuild • Innovate
Day 1 — Challenge	15 October 2026, 9:00 AM – 5:00 PM (8 hours, offline)
Day 2 — Presentations & Evaluation	16 October 2026
Team size	2–4 members
Entry fee	₹100 per team (per team, not per participant)
Prize pool	₹3,700
Mode	Offline
Registration deadline	13 October 2026

Read first: Challenge-specific instructions announced on the event day (permitted tools, AI and internet use, submission format, timings) take precedence over this booklet wherever they differ.

2. EVENT OVERVIEW

2.1 Concept

Re-InnovateX is a reverse engineering hackathon. Teams are given a broken, incomplete, malfunctioning, or partially functional system, software, application, or program package — not its original source code. Teams first understand the observable behaviour, then deconstruct how it works, then reconstruct, repair, or reproduce the required functionality, and finally introduce meaningful improvements beyond the reconstructed baseline. The event tests technical observation, logical reasoning, debugging, system analysis, experimentation, documentation, implementation, and the ability to work under a fixed deadline.

2.2 Core principle

Given System → Understand → Deconstruct → Reconstruct → Innovate → Demo. Do not jump directly into coding. Teams must show that their implementation is based on systematic observation and technical analysis of the provided challenge.

2.3 Reverse Engineering Hackathon vs Normal Hackathon

Aspect	Normal Hackathon	Reverse Engineering Hackathon
Starting point	Problem statement or theme	Broken / incomplete / malfunctioning system
First task	Ideation and solution planning	Observation and behaviour discovery
Core process	Ideate → Develop → Test → Demo	Understand → Deconstruct → Reconstruct → Innovate → Demo
Technical focus	Build a solution	Infer, analyse, repair/rebuild, then improve
Debugging	Mostly refinement/testing	Central to system analysis and reconstruction
Innovation	Usually starts with the proposed solution	Comes after reconstruction and extends the baseline
Outcome	Working prototype/solution	Reconstructed system + documentation + innovation

2.4 Objectives

- Understand a system from its observable behaviour.
- Identify components, dependencies, workflows, data flow, and failure points.
- Develop evidence-backed hypotheses about how the system works.
- Reconstruct or repair the required functionality, then improve it meaningfully.
- Demonstrate a working result and explain the reasoning behind it.
- Document the reverse-engineering process clearly enough for judges to evaluate it.
- Build teamwork, task allocation, and ethical reverse-engineering practice under time pressure.

3. SCHEDULE

Day 1 — 15 October 2026 · 8-Hour Challenge

Time	Phase	Primary activities
9:00 AM – 11:00 AM	Phase 1 · Understand & Observe	Interact with the system, map behaviour, record inputs/outputs, identify issues and document them.
11:00 AM – 1:30 PM	Phase 2 · Deconstruction	Analyse components, workflow, dependencies, data flow, and failure points.
1:30 PM – 5:00 PM	Phase 3 · Reconstruction & Innovation	Implementation, integration, debugging, testing, and meaningful innovation; prepare documentation and demo.

Day 2 — 16 October 2026 · Presentations & Evaluation

Duration	Stage	Activities
1 hour	Presentations	Team presentations, demonstrations, and judge questions.
1 hour	Evaluation & Results	Scoring, evaluation, feedback, and final results.

The detailed presentation sequence and reporting instructions will be communicated by the organizers.

Phase transition rules

- Teams must follow the official phase timings.
- Work may continue only within the permissions and instructions communicated by the organizers.
- Phase 1 is primarily for observation; Phase 2 for technical deconstruction; Phase 3 for reconstruction, innovation, and final preparation.
- Organizers may announce operational instructions or checkpoints during the event.

Suggested time management

- Maintain an observation log from the very beginning.
- Do not spend excessive time on internal details unnecessary for the required behaviour.
- Create a minimum viable reconstructed baseline before major innovation work.
- Reserve time for testing and documentation; keep backups at reasonable checkpoints.

- Do not leave packaging until the final minutes.

4. REGISTRATION & PARTICIPATION

4.1 Registration

The last date to register is 13 October 2026. Registration is complete only after the required registration process and fee payment are completed. Payment instructions and verification requirements will be communicated through the official registration process or by the organizers.

4.2 Team composition

- Each team must have 2 to 4 members; every member must be registered through the official process.
- Participants are responsible for accurate team details at registration.
- Team changes after registration require organizer approval.
- Providing false registration or team information where it materially affects eligibility is grounds for disqualification.

4.3 Entry fee

₹100 per team, charged for the complete team irrespective of whether it has 2, 3, or 4 members.

4.4 Prizes

Position	Prize
1st Prize	₹2,100
2nd Prize	₹1,100
3rd Prize	₹500
Total prize pool	₹3,700

The top three teams receive the cash prizes above. Organizers may also provide certificates, recognition, or other rewards as announced separately.

4.5 Attendance

- Registered members must report according to event instructions and carry any identification or materials requested.

- Late arrival, absence, or early departure may affect a team's ability to complete the challenge; organizers make the final decision.
- Be ready to work offline for the full 8-hour challenge.

4.6 Team responsibility

Each team is responsible for its own work, files, documentation, backups, and submission. Do not rely on a single device or storage location.

4.7 Team roles (suggested, flexible)

Role	Responsibility
System Observer / Documentation Lead	Records behaviour, inputs, outputs, errors, and hypotheses.
Technical Analyst	Studies components, dependencies, workflow, data flow, and failure points.
Developer / Reconstruction Lead	Implements or repairs the required functionality.
Testing / Innovation Lead	Validates behaviour, tests edge cases, and develops improvements.

Smaller teams may combine roles. Every member should understand enough of the final work to explain it during evaluation.

5. THE CHALLENGE PACKAGE

5.1 What may be provided

- A broken or incomplete application, or a partially functional software package.
- A malfunctioning module or workflow; missing, incomplete, or unclear components.
- Observable inputs, outputs, files, interfaces, logs, or errors; data and configuration.
- Challenge-specific instructions and technical restrictions.
- Only the information and access explicitly provided by the organizers. The exact environment, permitted tools, and submission format may be announced at the start of the event.

5.2 What will not be provided

- The original source code of the challenge system (unless explicitly announced).
- A complete explanation of the internal implementation.
- Any confidential internal material not explicitly released by the organizers.

- Access to systems or information outside the challenge. Do not attempt to obtain source code or confidential material through unauthorized means.

5.3 Phase 1 — Understand & Observe

Objective: determine what the system does before attempting to rebuild it.

- Run and interact with the system using permitted methods.
- Record observable inputs, outputs, screens, responses, files, logs, and errors.
- Identify normal versus broken behaviour; list visible modules and functional areas.
- Form initial hypotheses from evidence; maintain an observation log.
- Create a behaviour/workflow map.

Expected outcome: a clear functional understanding and an initial problem statement.

5.4 Phase 2 — Deconstruction

Objective: infer how the major parts interact and where failures or gaps occur.

- Inspect files and configuration exposed by the challenge; trace workflows and data movement.
- Test different inputs and edge cases; compare expected and observed behaviour.
- Use permitted debugging, inspection, testing, and analysis tools.
- Map components, dependencies, and failure points; separate observed facts from assumptions.

Expected outcome: a technically reasoned deconstruction that is a basis for reconstruction.

5.5 Phase 3 — Reconstruction & Innovation

Objective: rebuild, repair, reproduce, or implement the required functionality using the team's own work.

- Implement and integrate components based on documented findings.
- Test reconstructed behaviour against observations using representative cases; fix major errors.
- Prepare a working build or package, a clear demonstration, and the final documentation.

Expected outcome: a working or substantially reconstructed solution with an explanation of the approach and its limitations.

Baseline first: Establish the required reconstructed functionality before relying on the innovation layer.

5.6 Innovation layer

Innovation must go beyond simple reproduction and be technically meaningful and demonstrable. Examples:

- New feature
- Performance or efficiency improvement
- UI/UX improvement
- Accessibility improvement
- Security or reliability improvement
- Relevant AI or automation
- Better validation, error handling, monitoring, or maintainability

Clearly distinguish reconstructed baseline functionality from your own innovation, and make sure innovation does not break the baseline.

6. RULES, TOOLS & TECHNICAL BOUNDARIES

6.1 Permitted technical work

Tools that are explicitly permitted by the organizers and appropriate for inspecting, testing, debugging, analysing, and reconstructing the challenge — for example development environments, debuggers, testing utilities, file inspection tools, and logging/monitoring tools, when allowed by the challenge instructions.

6.2 Source code restriction

- No original source code is provided unless explicitly announced; infer behaviour from the challenge package.
- Do not obtain source code through unauthorized access, credential theft, or exploitation.

6.3 Controlled environment

- Remain inside the challenge environment and within permissions announced by organizers.
- Do not attack, damage, or probe unrelated systems — including college infrastructure, organizer accounts, other teams, external websites, or public services.

- Challenge-specific restrictions are mandatory.

6.4 Ethical conduct

- Use lawful and permitted methods only.
- No leaked solutions, stolen credentials, or confidential information.
- Do not damage challenge infrastructure or another team's work; do not impersonate participants.

6.5 External assistance, AI tools & internet

- Final work must be the team's own work produced during the event; unauthorized collaboration with non-team members is prohibited.
- AI tools may be used only if permitted for the specific challenge. Teams remain responsible for understanding, testing, documenting, and defending any AI-assisted work.
- Internet and external-resource access depend on challenge-specific instructions.
- If a tool, service, AI assistant, online resource, library, package, or collaborator is restricted, the restriction is binding. When in doubt, ask an organizer first.

6.6 General rules

- Participants must be registered members of their declared team; teams must have 2–4 members unless organizers announce otherwise.
- Use only permitted challenge resources and complete all submissions before the announced deadline (late submissions may be rejected or penalized).
- The final submission must represent the team's own work.
- Maintain professional conduct; follow venue, lab, and institutional rules.
- Do not interfere with another team's work, environment, device, repository, or submission.
- Follow instructions from organizers, judges, faculty coordinators, and authorized technical staff. Their decisions are final and binding.

7. PROHIBITED ACTIVITIES & PENALTIES

7.1 Prohibited activities

- Unauthorized access to systems, accounts, devices, networks, repositories, or services.
- Credential theft, phishing, password attacks, or impersonation.

- Attacking unrelated college or public infrastructure; destructive or denial-of-service activity.
- Attempting to obtain or leak the original source code through unauthorized means.
- Using leaked solutions, another team's private work, or confidential organizer material; sharing private team solutions during the event.
- Tampering with monitoring, submission, or judging infrastructure.
- Submitting substantially copied or another team's work as original.
- Harassment, intimidation, abusive behaviour, or disruption.
- Any activity expressly prohibited in challenge-specific instructions.

7.2 Grounds for disqualification

- Violation of the no-unauthorized-source-code rule; unauthorized access to systems or data.
- Use of leaked solutions or another team's confidential work; substantially plagiarized or stolen work.
- Deliberate interference with the challenge environment or other teams.
- Violation of a challenge-specific rule; false registration or team information.

7.3 Penalty framework

Category	Examples	Possible action
Minor	Procedural non-compliance, incomplete documentation, avoidable format issue.	Warning, correction request, or limited score impact.
Serious	Unauthorized collaboration, prohibited assistance, interference with another team.	Score penalty, submission rejection, or disqualification depending on severity.
Critical	Unauthorized external access, destructive activity, credential theft, deliberate infrastructure attack, or serious misconduct.	Immediate disqualification and escalation under event/institutional procedures.

The response depends on the nature, evidence, impact, and severity of the violation. Unauthorized access or attempts to compromise systems outside the challenge are not part of the competition.

Found an unintended access path? If your team discovers a potentially sensitive or unintended access path, stop and report it to the organizers rather than exploiting it.

7.4 Monitoring & fair play

- Organizers may use reasonable monitoring mechanisms to maintain integrity; comply with any monitoring requirements announced before or during the event.
- Submissions may be inspected for compliance; teams may be asked to explain their reconstruction reasoning or innovation; suspicious activity may be reviewed.

7.5 Technical issue / downtime policy

- If a challenge–environment issue occurs, immediately inform authorized organizers or technical support.
- Report with a clear description and screenshots/logs where possible.
- Do not independently modify infrastructure or repeatedly perform risky actions after a system–level failure.
- Organizers may restore or replace resources and, for major issues, announce an appropriate adjustment.

7.6 Disputes & clarifications

Raise questions about rules, permitted tools, challenge interpretation, or technical boundaries with the organizers as early as possible. Organizer instructions given during the event may clarify or supplement these guidelines.

8. SUBMISSION & DOCUMENTATION

8.1 Required deliverables

- Working reconstructed / repaired / reproduced system.
- Source repository or submission package as specified in the challenge instructions.
- Technical documentation covering observations, analysis, reconstruction, and testing; evidence of testing and important findings.
- Innovation summary describing improvements beyond the baseline.
- Final presentation / demo.

8.2 Recommended documentation structure

Section	What to include
System Understanding	Observed inputs/outputs, workflows, and initial findings.
Failure Analysis	Issues, incomplete functionality, failure points, and evidence.
Deconstruction	Components, dependencies, workflow, data flow, and reasoning.

Section	What to include
Reconstruction	What was rebuilt/repared and the implementation approach.
Testing	Test cases, results, fixes, and stability checks.
Innovation	New features or improvements beyond the baseline.
Limitations	Known limitations and assumptions.
Demo Guide	Steps for judges to reproduce key functionality.

8.3 Deadline

All submissions must be completed before the official deadline announced during the event. Late submissions may not be accepted unless an organizer explicitly grants an exception.

8.4 Testing guidelines

- Test normal workflows and expected inputs, and invalid/unexpected inputs where relevant.
- Compare reconstructed outputs with observed behaviour; test integrated modules, not only isolated functions.
- Record important bugs and fixes; demonstrate meaningful test evidence.
- Ensure innovation does not break the baseline.

8.5 Documentation & demo guidelines

- Keep documentation concise but technically meaningful; use diagrams, flowcharts, screenshots, tables, logs, or test evidence.
- Distinguish observation, hypothesis, implementation, and innovation.
- Do not claim functionality that cannot be demonstrated; identify a non-working feature honestly rather than presenting it as complete.
- All members should understand the overall solution.
- Best practice: keep regular backups, save working versions, and document major discoveries as you progress.

9. EVALUATION & JUDGING

9.1 Criteria and weightage

Criterion	Weight	What will be considered
Understanding	20%	Accuracy of observations, inputs/outputs, expected behaviour, and initial analysis.
Technical Analysis	20%	Depth of deconstruction: components, workflow, dependencies, data flow, failure-point analysis.
Reconstruction	30%	Correctness, functionality, stability, testing, and successful implementation of required behaviour.
Innovation	20%	Meaningful improvements, additional functionality, optimization, usability, security, accessibility, AI/automation.
Documentation & Demo	10%	Clarity of documentation, reasoning, testing evidence, presentation, and final demonstration.

9.2 What judges look for

- How accurately the team understood the system's observable behaviour.
- Whether the deconstruction is logical, evidence-based, and technically sound.
- How closely the reconstruction matches intended or reasonably inferred behaviour.
- Whether the team tested its implementation rather than relying on assumptions.
- How clearly the team explains technical decisions, discoveries, and limitations.
- How effectively the team used time and resources; creativity and decisions under pressure.
- Understanding, not only visual appearance; innovation is evaluated separately from the baseline.

9.3 Tie-breaking

If scores are tied, organizers may use this sequence: reconstruction functionality, technical analysis, innovation, then documentation/demo quality. Any specific tie-break announced before judging takes precedence.

9.4 Final decision

The decision of the organizers and judges regarding evaluation, ranking, and awards is final and binding, subject to the event's announced procedures.

9.5 Day 2 presentation & demo structure

On 16 October, teams present and may be questioned by judges. All team members should be able to explain their contribution and the overall approach. Recommended flow:

1. Explain what the team observed initially.
2. Show the key failure, missing behaviour, or incomplete workflow.
3. Explain the analysis and reconstruction approach.
4. Demonstrate the reconstructed functionality.
5. Show testing evidence or important edge cases.
6. Demonstrate innovation.
7. State important limitations and future improvements.

10. RESPONSIBILITIES

Participants

- Read and follow the rulebook and challenge instructions; use the environment responsibly.
- Maintain ethical conduct and respect other teams.
- Keep work organized and backed up; maintain evidence of analysis and testing.
- Submit on time; be available for judge questions and technical verification.

Organizers

- Provide the challenge package and a controlled environment; communicate rules, restrictions, deliverables, and criteria.
- Provide reasonable event-level technical support; collect and secure submissions.
- Coordinate judging and demonstrations; communicate changes clearly; maintain fair-play procedures.

Faculty coordination

Faculty coordination provides institutional supervision and support, including event-level discipline, safety, and coordination with the organizing body.

Participant code of conduct

- Be respectful and professional toward participants, organizers, faculty, volunteers, and judges.

- Do not intentionally disrupt or tamper with another team's files, devices, accounts, or materials.
- Follow organizer instructions promptly during the challenge and evaluation.

11. PARTICIPANT CHECKLISTS

Before 15 October

- Form a team of 2–4 members.
- Complete registration before 13 October 2026.
- Pay / complete the ₹100 per-team entry requirement as instructed.
- Save registration details and organizer contact numbers.
- Be ready to work offline for the full 8-hour challenge.

Phase 1 — Understand

- Explore before assuming.
- Record inputs, outputs, and visible behaviour.
- Identify what works and what is broken.
- Form testable hypotheses.

Phase 2 — Deconstruct

- Map components and workflows.
- Trace dependencies and data flow.
- Identify failure points and technical gaps.
- Keep evidence and notes for reconstruction.

Phase 3 — Reconstruct & Innovate

- Reconstruct the required functionality first.
- Test representative cases; keep backups of working versions.
- Add innovation without breaking the baseline.
- Prepare documentation and demonstration; submit before the deadline.

Final submission

- Reconstructed / repaired system works.
- Required repository or package is included.
- Documentation is complete, with observations, analysis, and testing evidence.

- Innovation is clearly separated from baseline.
- Demo flow has been tested; team details are correct.
- Submitted before the deadline; backup copy retained.

For the 16 October presentation

- Know what the original system did.
- Explain how you deconstructed it.
- Demonstrate what you reconstructed and your innovation.
- Explain key decisions and limitations.
- Ensure every team member can answer questions.

12. FREQUENTLY ASKED QUESTIONS

Q. Are we given source code? No, not by default. The standard format is a designated broken/incomplete challenge package. Any exception will be announced.

Q. Do we have to reproduce everything? Teams reconstruct the required functionality identified through the challenge. Exact scope is defined by the challenge package and briefing.

Q. Can we innovate before reconstruction is complete? Planning is fine, but the expected approach is to establish the required baseline before final innovation.

Q. Can we use AI? Only if permitted for the challenge. Specific AI restrictions take precedence.

Q. Can we use the internet? Internet and external-resource access depend on challenge-specific instructions.

Q. What if the environment breaks? Report it immediately to authorized support. Do not perform unauthorized infrastructure intervention.

Q. Is documentation important? Yes. Documentation and demo quality are part of evaluation and show the reasoning behind the reconstruction.

Q. Is the fee per person? No. The entry fee is per team, regardless of team size.

FINAL EVENT PRINCIPLES

Principle	Meaning
Observe before assuming	Use evidence from the system instead of guessing.

Principle	Meaning
Understand before rebuilding	Build a clear model of the system first.
Reconstruct before innovating	Establish the required baseline, then improve it.
Test what you build	Support the demo with verification.
Document your reasoning	The process matters as well as the output.
Stay within the boundary	Reverse engineer the provided challenge, not unrelated systems.
Compete fairly	Protect event integrity and respect other teams.

The goal is not simply to make something work. It is to understand why it works, reconstruct it from evidence, improve it meaningfully, and demonstrate the engineering process.

TechUtsav 1.0 2026 · ITM University Gwalior

Register: <https://parakh.itmuniversity.ac.in/pay/45f4d813-62af-4630-87af-990a2f7d6aa3>

Source: Re-InnovateX Hackathon Workflow Guide and Participant Booklet (Code Craft Society).