Python Full Stack and AI Hackathon

Problem Statements and Schedule

Teams of three will independently design, build, test, and demonstrate working applications for two manufacturing challenges. Full-stack Python and AI must be used in both phases. Faculty and mentors will provide minimal supervision, limited to clarifying requirements and resolving common environment issues.

Participation format

- Phase 1 is open to all teams. After evaluation, the top 50% qualify for Phase 2. For an odd number of teams, the qualifying count is rounded up.
- Each phase allows five hours of development. Teams choose their own architecture, task allocation, and implementation sequence within the stated requirements.
- During the presentation, one member acts as Administrator, one as Engineer, and one as Supervisor. Every member must explain their own contribution.
- Participants are expected to know the listed technologies. Environments and model access must be ready before development begins.

Schedule

Time from and To	Stage	Duration	Agenda and outcome
10 AM to 11:30 AM	Opening and briefing	1.5 Hour	Event introduction, rules, and Phase 1 problem release.
12:00 PM to 6:30 PM	Phase 1 development	6.5 hours	Independently develop the RAG manual assistant.
7:00 PM to 10:00 PM	Phase 1 evaluation	3 hours	Live demonstrations and technical questions; top 50% qualify.
10:45 PM – 11:30 PM	Phase 2 briefing	45 minutes	Release the Computer Vision problem, model, and image set to finalists.

©Blessed IT Solution Pvt Limited – All Rights Reserved

Time from and To	Stage	Duration	Agenda and outcome
12:00 AM – 7:00 AM	Phase 2 development	7 hours	Independent development of the visual inspection application.
7:45 AM – 11:15 AM	Final evaluation	3.5 hours	Live demonstrations, technical questions, and winner selection.
12:00 PM – 2:00 PM	Closing and validatory	2 hours	Results and award distribution.

Release of problem statements

The two challenge briefs are on separate pages for staged distribution. Issue the Phase 1 brief at its start, and the Phase 2 brief only to qualifying teams after the results are announced. Common technology and submission requirements appear on the final page.

Phase 1 Manufacturing Manual Assistant

Development time: 6.5 hours | Team size: 3 or 4 | Evaluation: 3 hours

Problem statement

A manufacturing organization stores equipment manuals in separate PDF files. Engineers need a faster way to find the correct instructions for a specific machine and maintenance requirement. Build a web application that retrieves relevant manual content and uses RAG to answer questions with verifiable source references.

Expected workflow

1. An Administrator logs in, registers machines, and uploads the relevant manuals with model and revision details.
2. The application extracts the text, retains PDF page references, and makes the content searchable.
3. An Engineer logs in, selects a machine, and asks a maintenance question.
4. The assistant answers using evidence from the selected machine's manuals and displays the supporting passage, manual name, revision, and page number.
5. The Engineer opens the source PDF and can revisit saved questions and answers.

©Blessed IT Solution Pvt Limited – All Rights Reserved

6. A Supervisor reviews question history and identifies unanswered queries. A simple read-only history view is sufficient.
7. **Please remember to provide access control based on the roles (Administrator, Engineer, Supervisor).**

Expected solution

- A connected browser interface, Django backend, REST APIs, and PostgreSQL or MySQL database with persistent machine, manual, and question records.
- Role-based access enforced by the backend. Engineers must not be able to upload manuals or perform administrator actions. Django admin may be used for administration.
- A working retrieval and generation pipeline that restricts evidence to the selected machine and clearly reports when the manuals contain insufficient information.
- At least two automated checks covering restricted upload access and exclusion of another machine's passages from retrieval.

Input provided

Three short text-based PDF manuals covering two fictional machines, with identifiable models and revisions. Embedding and LLM access will be available. OCR, complex diagrams, and live factory integrations are outside the required scope.

Demonstration and evaluation

Show an answerable question and its source page, a question supported only by another machine's manual, an unsupported question, restricted upload access, and persistent question history.

Submit the working version at the five-hour deadline. The top 50% advance based on Phase 1 scores.

©Blessed IT Solution Pvt Limited – All Rights Reserved

Phase 2 AI Visual Quality Inspection

Development time: 7 hours | Qualifying teams only | Evaluation: 3 hours proposed

Problem statement

A manufacturing organization wants to screen product images for visible defects and enable supervisors to verify the results. Build a web application that uses Computer Vision to classify an uploaded product image as Defective or Normal and maintains a record of each inspection and review.

Expected workflow

1. An Administrator registers the supported product type.
2. An Engineer logs in and uploads a JPG or PNG image for inspection.
3. The Python backend validates and preprocesses the image, runs the supplied model, and displays the predicted class and model confidence score.
4. The application saves the image reference, product, prediction, score, model version, user, and timestamp.
5. The Engineer uploads a small image batch; background processing displays queued, processing, completed, or failed status.
6. A Supervisor reviews inspection history, filters results, and confirms or corrects a prediction with an optional comment. Original model predictions remain available alongside human reviews.

Expected solution

- A full-stack inspection workflow with browser screens, Django REST APIs, persistent relational records, and server-side role permissions.
- Real image inference using the supplied preprocessing and class mapping; Celery and Redis for batch processing, with readable failure messages.
- Automated checks for invalid image rejection, supervisor-only review access, and result persistence.
- A confusion matrix and defect recall on the supplied labeled samples, with an explanation of defective items incorrectly classified as Normal.

Input provided and scope

At the Phase 2 release, we will supply a task-specific pretrained classifier, model version, class labels, preprocessing requirements, and a small, labeled sample set. The runtime will be installed beforehand. Use one product type and image classification. You do not need to train from scratch, use bounding boxes, or use live camera feeds.

Teams may reuse Phase 1 login, database connections, API setup, and interface layout. They must implement the new inspection workflow during Phase 2. Model confidence is a prediction score, not a guarantee of correctness.

Demonstration and evaluation

Demonstrate all three roles, single and batch uploads, a saved inspection, and supervisor correction. Judges will also use unseen images and check invalid uploads and restricted access.

©Blessed IT Solution Pvt Limited – All Rights Reserved

Technology and Submission Requirements

Technology stack

Component	Technology	Applies to
Frontend	HTML, CSS, JavaScript, Bootstrap	Both phases
Python backend	Django	Both phases
REST APIs	Django REST Framework	Both phases
Relational database	PostgreSQL or MySQL	Both phases
Authentication and authorization	Django authentication and permissions	Both phases
File handling	Python and Django	Both phases
Document processing	PyMuPDF and Python	Phase 1
AI and RAG	Embedding model, vector store, and LLM	Phase 1
Computer Vision	Supplied classifier and compatible Python inference runtime	Phase 2
Background processing	Celery and Redis	Phase 2
Testing	Django tests or pytest	Both phases

Independent work and support

Teams own the design, coding, integration, testing, and time management. Supervisors will clarify requirements and address common infrastructure issues; they will not provide a step-by-step implementation. Identical input packs and access conditions apply to all teams. The college will announce the policy for AI coding assistants and reused code before the event.

Submission at the end of each phase

- A frozen repository commit or timestamped ZIP containing source code, dependencies, and database migrations.
- A README with setup and run instructions, demonstration credentials, and a short architecture explanation. Keep real API keys and secrets out of the submission.
- Test results and a working local application for a live demonstration. For Phase 2, include model details, worker startup instructions, sample-set metrics, and a note of reused Phase 1 components.

©Blessed IT Solution Pvt Limited – All Rights Reserved



Presentation and final selection

Each member must take part in the demonstration and answer questions about their contribution. A suggested slot is 12 minutes per team: 7 minutes for demonstration, 3 minutes for questions, and 2 minutes for scoring and transition. Parallel panels may be used to fit the evaluation window.

©Blessed IT Solution Pvt Limited – All Rights Reserved