Intel AI for Manufacturing – Project Report

1. Project Overview

a. Project Title

Packaging Quality Inspection using AI & Computer Vision

b. Project Description

This project aims to automate the inspection process on packaging lines by identifying defective products using an AI model. Manual inspections are slow, inconsistent, and prone to human error. By deploying an AI-driven system through a web interface, the project ensures faster, more accurate quality assurance, reducing operational costs and product returns.

c. Timeline

- Week 1: Problem definition & dataset preparation
- Week 2: Model training via Teachable Machine
- Week 3: Frontend development with React + Vite
- Week 4: Integration, testing, and Vercel deployment

d. Benefits

- Reduces human error in QA
- Speeds up inspection processes
- Minimizes waste and recalls
- Offers easy-to-use web interface

e. Team Members

- Bhavya Shah (Solo Developer)

f. Risks

- Limited dataset diversity may reduce model generalization
- Teachable Machine model not fine-tuned for edge cases
- Lack of real-time backend inference could limit scalability

2. Objectives

a. Primary Objective

To develop a web-based AI solution that identifies packaging defects using image classification.

b. Secondary Objectives

- Enable easy image upload and defect detection
- Integrate Teachable Machine model using TensorFlow.js
- Ensure compatibility with low-resource systems (client-side execution)

c. Measurable Goals

- Achieve ≥90% accuracy on validation data
- Image upload + prediction under 5 seconds
- Deploy frontend with no build errors on Vercel

---

3. Methodology

a. Approach

Agile, iterative development with separate phases for training, UI development, and deployment.

b. Phases

1. Dataset preparation
2. Model training using Teachable Machine (image classification)
3. React + Vite frontend development
4. Integration and cloud deployment via Vercel

c. Deliverables

- Teachable Machine .json model
- React-based UI for image upload
- Hosted web app on Vercel

d. Testing and Quality Assurance

- Manual testing with known good/defective images
- Model validation using Teachable Machine test mode
- Cross-browser frontend testing for layout & performance

e. Risk Management

- Used client-side inference to avoid server dependency
- Simplified model architecture for faster predictions
- Used confidence threshold to flag uncertain predictions

---

4. Technologies Used

a. Programming Languages

- JavaScript (React), TypeScript, Python (for dataset formatting)

b. Development Frameworks

- Vite (Frontend bundler), TensorFlow.js (Teachable Machine deployment)

c. Database Management Systems

- Not applicable (no backend used)

d. Development Tools

- Visual Studio Code, GitHub, Teachable Machine, Vercel

e. Testing Tools

- Manual image validation
- Teachable Machine's built-in evaluation UI

f. Cloud Services

- **Vercel** for frontend deployment
- **Teachable Machine** (Google) for cloud-hosted model

g. Security

- No user data collected
- Client-side only inference ensures data privacy

h. APIs and Web Services

- **Teachable Machine TensorFlow.js Model API**
  https://teachablemachine.withgoogle.com/models/wwG4ipELH/model.json

---

5. Results

a. Key Metrics

- Model Accuracy: ~91%
- Average Inference Time: ~1.2 seconds
- UI Load Time: <3 seconds
- Model Size: ~1 MB (optimized for web)

b. ROI

- Saves inspection labor cost
- Consistent defect detection

- Easily replicable for multiple production lines

---

6. Conclusion

a. Recap the Project

An AI-powered web app for packaging quality inspection using image classification, Teachable Machine model, and a Vercel-hosted React UI.

b. Key Takeaways

- Teachable Machine is fast for prototyping
- Frontend-only AI model improves accessibility
- Web deployment allows platform independence

c. Future Plans

- Add webcam/video support for live inspections
- Migrate to a Streamlit or FastAPI backend for real-time API inference
- Collect more data and retrain for higher accuracy

d. Successes and Challenges

Successes: Model trained and deployed, frontend live with prediction
Challenges: Limited data variety, no backend inference

---

7. Project Specifics

a. Project URL

https://quality-inspect-visual.vercel.app

b. GitHub URL

https://github.com/Bhavya227/Quality-inspect-visual

c. Collab/Notebook URL

Not applicable – model trained using Teachable Machine.

d. Dataset URL

Internal dataset uploaded to Teachable Machine.

e. Model URL

Teachable Machine Model – TensorFlow.js JSON

Note: If the deployed site asks you for a model URL, paste this:
https://teachablemachine.withgoogle.com/models/wwG4ipELH/model.json