Project Plan

Smart NV

Computer Vision Puzzle Tracking System

IQ Waves Puzzle

Academic Year: 2025–2026

Institution: Thomas More University

Client: Smart NV

Author: Ibrahim Afkir

Role: Intern

Table of Contents

1. Introduction

This document serves as the official Project Plan for the Smart NV Computer Vision Puzzle Tracking System, a research phase initiative commissioned by Smart NV. It formally establishes the project's purpose, scope, objectives, stakeholders, risks, and planning. The goal is to align the development team and the client on expectations before any technical work begins.

The plan covers background on the client and their current challenge, the high level functional and non functional requirements, an analysis of in scope versus out of scope work, a risk assessment focused on the client's responsibilities, and a phased sprint planning outline. It is intended as a living reference document throughout the project lifecycle.

2. Background

2.1 Client

Smart NV is a Belgian toy and game manufacturer specializing in logic based puzzle games for children and families. Their product portfolio includes physical board games such as IQ Waves, which challenge players to fit uniquely shaped pieces onto a grid. Smart NV designs, manufactures, and distributes these products internationally and controls the full product design pipeline, including artwork and physical board construction.

Smart NV is entering a new phase of product development by exploring how digital technology can enhance the physical play experience. This project represents their first foray into computer vision and AI assisted gameplay, with a strategic goal of building a commercially viable digital companion that integrates seamlessly with their existing product line.

2.2 Current Situation

Currently, Smart NV's puzzle games are entirely analog. When players (typically children aged 6+) get stuck on a puzzle challenge, there is no digital support mechanism. Players must rely on the printed solution booklet, which requires literacy, or ask a parent or teacher for help. This creates friction in the play experience and can lead to frustration, disengagement, or abandonment of the game.

No existing system detects the physical state of a Smart NV puzzle board using a camera, nor does any tool provide contextual hints based on what pieces are already placed. The company has no internal computer vision capability and no labeled training dataset for their puzzle pieces or boards. Building this system from scratch, including dataset creation, model training, and a full browser based application, is the core challenge this project addresses.

3. Project Vision and Goals

3.1 Vision Statement

Create a computer vision powered digital companion that detects physical game states and provides real time solving logic for the IQ Waves puzzle, bridging the gap between physical hardware (the game) and digital assistance (the solver).

3.2 Primary Goal

Transform frustration into engagement by providing age appropriate, real time guidance to children ages 6+ solving the IQ Waves puzzle, without requiring literacy or constant parental intervention.

3.3 Business Goals

- Reduce player frustration and game abandonment rates by providing contextual, real time assistance
- Differentiate Smart NV products in a competitive toy market through innovative AI integration
- Build a technical foundation for a scalable digital companion ecosystem across Smart NV's product portfolio
- Generate valuable usage data and insights for future product development
- Explore new revenue models (freemium features, puzzle packs, subscriptions) in future phases

4. Puzzle Specification: IQ Waves

The IQ Waves puzzle consists of a board with 4 rows and 8 columns, totaling 32 hourglass shaped cells. These cells alternate between horizontal (H) and vertical (V) orientation, creating the distinctive wave pattern that gives the puzzle its name.

The puzzle includes 8 uniquely shaped and colored pieces that the player must fit onto the board. Each piece occupies between 3 and 5 cells. The full piece inventory is as follows:

Piece	Color	Cells
Red	#EF4444	4
Hot Pink	#EC4899	4
Dark Blue	#1E40AF	5
Orange	#F97316	4
Light Blue	#38BDF8	3
Green	#84CC16	5
Yellow	#EAB308	4
Purple	#8B5CF6	3

Table 1: IQ Waves piece inventory

An additional class (Class 8) represents the board itself and will be used exclusively for corner detection during the computer vision pipeline. It is not a playable piece.

5. Objectives and Stakeholders

5.1 Ultimate Objective

Build a production quality computer vision system with a React frontend for photo capture and a FastAPI backend running the computer vision pipeline. The system will allow a phone camera to capture a photo of the physical IQ Waves puzzle, upload it to the backend for processing, and display the puzzle state, progressive hints, and completion detection in the browser.

5.2 High Level Functional Requirements

- **Photo Capture (P1): A mobile browser app allows the user to take photos of the IQ Waves puzzle board and upload them to the backend via a REST API (iOS Safari and Android Chrome compatible).**
- **Board Detection (P2): The FastAPI backend receives the uploaded photo, runs the ML model to detect the puzzle board, identifies its corners, and performs perspective correction to produce a rectified top down view.**
- **Piece Identification (P3): A YOLOv26 Nano Segmentation model running on the backend via ONNX Runtime identifies the 8 individual puzzle pieces by type within the rectified board image.**
- **Grid Mapping (P4):** The system maps detected piece masks onto the 4×8 board grid and determines whether each cell is occupied, empty, or incorrectly placed.
- **Hint Generation (P5):** A progressive hint system guides the player from general area hints to specific piece placement, computed from the current board state.
- **Voice Assistant (P6):** An integrated voice assistant reads hints aloud, making the system accessible to younger children who may not yet read fluently.

- **Solvability Check (P7):** Before generating hints, the solver verifies whether the current board state is solvable and notifies the user if pieces are misplaced.
- **Completion Detection (P8):** The system detects when all pieces are correctly placed and triggers a completion screen.

5.3 High Level Non Functional Requirements

- **Latency: End to end processing (from photo upload to result display) should target under 2 seconds on Wi Fi and under 3 seconds on mobile data.**
- **Accuracy:** Piece detection accuracy should target mAP@0.5 of 0.90 or higher. Board corner detection should succeed at angles up to 45° and distances up to 75 cm.
- **Privacy (GDPR): Photos are processed in memory on the backend and are not stored. No personal data is collected or persisted.**
- **Accessibility:** The UI will comply with WCAG 2.1 AA guidelines, including minimum 48px touch targets and colour blind friendly visual cues.
- **Portability: The frontend will be a static React application. The backend will be containerised and deployable on any server with Python support.**

5.4 Stakeholders

Stakeholder	Role	Business Value
Smart NV (Client)	Product owner; provides puzzle assets, STL files, and business requirements	Gains a commercially viable digital companion product
End Users (Children 6+)	Primary users of the browser application	Receive step by step, age appropriate hints to progress independently
Parents / Teachers	Secondary users; manage device during play	Reduced burden of constant assistance; visibility into progress
Ibrahim Afkir (Intern)	Designs, builds, and tests the system	Delivers a research grade prototype demonstrating technical feasibility
Thomas More University	Academic supervisor	Evaluates technical quality, methodology, and documentation

Table 2: Project stakeholders and their roles

6. Project Scope and Risk Analysis

6.1 In Scope

- A React 18 frontend (Vite, plain JavaScript) for photo capture and upload via REST API
- A FastAPI backend (Python 3.11) running the computer vision pipeline with ONNX Runtime inference
- A computer vision pipeline on the backend: image preprocessing, YOLO inference, mask decoding, board corner detection, perspective correction, and grid mapping

- Training the YOLOv26 Nano Segmentation model on a dataset combining synthetic images (rendered in Blender) and real world photos (annotated in Roboflow)
- A backtracking solver algorithm that validates board state, checks solvability, and computes progressive hints
- A voice assistant that reads hints aloud for accessibility
- Completion detection with visual feedback
- GDPR compliant design with in memory photo processing on the backend (no photo storage)
- Support for the IQ Waves puzzle as the primary target
- Research experiments validating photo capture reliability, board detection accuracy, ML piece detection, end to end performance, and UX with families

6.2 Out of Scope

- User accounts, progress tracking, or admin panel
- AR overlays and social features
- Real time video streaming
- Support for other puzzle SKUs beyond IQ Waves in this phase

6.3 Client Responsibilities

- Provide physical IQ Waves game units and STL files for all puzzle pieces and the board
- Provide sample challenge booklets and annotated board state examples for solver validation
- Provide timely feedback at each sprint demo (within 5 business days)
- Appoint an internal project lead as a single point of contact for design decisions
- Recruit several families with children aged 8+ for user testing

6.4 Risk Analysis

The following risks fall primarily under the client's responsibility:

Risk	Severity	Mitigation
Delayed delivery of STL files or physical game units	High	Client appoints a single point of contact; assets are delivered before Sprint 1 begins
Insufficient availability of families for user testing	Medium	Client begins recruitment no later than Week 4; team provides test session scheduling tool
Changing scope or requirements mid project	Medium	Any scope changes are formally logged in the backlog and approved in writing before implementation
GDPR consent not obtained for child participants in user testing	High	Client prepares parental consent forms in advance; no photos of children are stored
Unclear definition of target puzzle for MVP	Medium	Client confirms IQ Waves as the primary target during Sprint 0 stakeholder interview

Table 3: Client side risks and mitigation strategies

The following risks fall under the team's responsibility and are noted for transparency:

- Phone camera fragmentation (iOS/Android): mitigated by using input type="file" capture as a universal fallback
- Board occlusion by children's hands: mitigated by training on diverse occlusion scenarios and providing clear photo capture instructions
- ML model not generalising to real world conditions: mitigated by combining synthetic training data (Blender) with real world photos (Roboflow) for domain adaptation
- Lighting variation in real homes: mitigated by diverse training augmentation
- Browser performance limitations for ML inference: mitigated by using YOLOv26 Nano (2.4M parameters), the smallest and fastest variant, optimized for on device execution
- WebAssembly compatibility across browsers: mitigated by testing on multiple browser/device combinations early in development

7. Technical Approach

7.1 Architecture Overview

The system follows a client/server architecture. The frontend is a React application running in the browser, responsible for photo capture and displaying results. The backend is a FastAPI server (Python 3.11) that receives uploaded photos, runs the full computer vision pipeline (YOLO inference, mask decoding, perspective correction, grid mapping), executes the solver, and returns the board state and hints to the frontend. This separation allows the computationally intensive ML inference to run on a proper server environment while keeping the frontend lightweight.

7.2 Computer Vision Pipeline

The computer vision pipeline will process each photo through the following stages on the FastAPI backend:

- Image Preprocessing: resize and normalize the captured photo for model input
- YOLO Inference: run the YOLOv26 Nano Segmentation model via ONNX Runtime to detect piece masks and the board
- Mask Decoding: extract segmentation masks from the model output to identify the precise shape and position of each piece
- Board Corner Detection: identify the four corners of the puzzle board from the board segmentation mask
- Perspective Correction: apply a homography transform to rectify the board image into a clean top down view
- Grid Mapping: map the detected piece masks onto the 4×8 board grid to produce a digital board state

7.3 ML Model Strategy

The project will use a YOLOv26 Nano Segmentation model running on the FastAPI backend via ONNX Runtime. The Nano variant is chosen specifically because it is the smallest in the

YOLOv26 family (2.4M parameters), offering fast inference while keeping server resource requirements low.

The model will be trained to detect 9 classes: the 8 colored puzzle pieces (Red, Hot Pink, Dark Blue, Orange, Light Blue, Green, Yellow, Purple) and the board itself. The board class is used exclusively for corner detection.

7.3.1 Training Data Strategy

Since no labeled dataset exists for the IQ Waves puzzle, the team will build one from scratch using a two pronged approach:

Synthetic Data Generation: Use the provided STL files (3D models of the pieces and board) to render thousands of labeled training images in Blender. This allows full control over lighting conditions, camera angles, distances, and backgrounds, producing a large and diverse dataset with minimal manual annotation effort.

Real World Data Collection: Capture real world photos of the puzzle under various conditions and annotate them using Roboflow. These images will be combined with the synthetic data for domain adaptation, ensuring the model generalises from rendered images to actual photos.

7.4 The Solver Engine

The solver will use a backtracking algorithm to find valid solutions from any partial board state. Given the current grid configuration (which cells are occupied and by which pieces), the solver will recursively attempt to place remaining pieces into empty spaces, backtracking when a placement leads to a dead end.

Key features of the solver:

- Progressive hint system: rather than revealing the full solution, the solver will offer hints in stages (general area, then specific piece, then exact placement)
- Solvability check: before generating hints, the solver will verify whether the current board state can lead to a valid solution, alerting the user if pieces are misplaced
- Voice assistant integration: hints will be read aloud using the Web Speech API, making the system accessible to children who may not yet read fluently

8. Technology Stack

8.1 Frontend

- Framework: React 18 with plain JavaScript
- Build Tool: Vite
- Camera/Photo Capture: input type="file" capture for native camera access
- UI Styling: Plain CSS

8.2 Backend

- Framework: FastAPI (Python 3.11) with Uvicorn
- ML Inference: ONNX Runtime
- Image Processing: OpenCV (opencv contrib python)
- API: REST endpoint for photo upload (multipart POST)
- Photo Processing: In memory only (GDPR compliant, no storage)

8.3 Computer Vision and ML

- ML Model: YOLOv26 Nano Segmentation (2.4M parameters)
- Inference Runtime: ONNX Runtime (server side)
- Training Framework: Ultralytics YOLOv26
- Synthetic Data: Blender (3D rendering from STL files)
- Annotation Tool: Roboflow (real world photo annotation)
- Image Processing: OpenCV for preprocessing, perspective correction, and grid mapping

8.4 Solver

- Algorithm: Recursive backtracking
- Language: Python (runs on the FastAPI backend)
- Voice Output: Web Speech API for hint narration

9. Research Experiments

The project includes five formal research experiments to validate technical feasibility:

Experiment 1: Photo Capture and Upload Reliability (Week 1–2)

- Build a minimal React app with camera photo capture functionality
- Test on 3 iPhone models (Safari) and 3 Android models (Chrome)
- Measure: capture success rate (≥ 99%), photo quality and resolution, upload round trip latency (≤ 2s on Wi Fi)

Experiment 2: Board Detection Accuracy (Week 2–3)

- Generate synthetic training data using STL files in Blender (500–1,000 images)
- Train YOLOv26 Nano Segmentation model for board and piece detection
- Capture test images at varied angles (0°–60°), distances (20–100 cm), and lighting conditions
- Measure: board corner detection rate (≥ 95% at up to 45° and 75 cm), perspective correction accuracy

Experiment 3: ML Piece Detection (Week 3–6)

- Collect real world photos and combine with synthetic data for model training
- Train YOLOv26 Nano Segmentation model on the full 9 class dataset
- Evaluate segmentation mask quality for accurate grid mapping
- Measure: mAP@0.5 (≥ 0.90), per piece recall (≥ 92%), inference time on server via ONNX Runtime

Experiment 4: End to End Performance (Week 5–7)

- Full pipeline: phone photo upload to FastAPI backend, YOLO inference, mask decoding, corner detection, perspective correction, grid mapping, solver, response to frontend

- Instrument timestamps at each stage
- Measure: P95 total latency ≤ 2 seconds on Wi Fi, ≤ 3 seconds on mobile data

Experiment 5: UX Prototype and User Testing (Week 6–8)

- Functional prototype with progressive hints, voice assistant, and completion detection
- Test with several families (children ages 8+)
- Measure: setup time (≤ 60 seconds), hint comprehension (≥ 70% without adult help), completion detection accuracy (≥ 95%)

10. Project Planning

The project follows the Scrum methodology with four two week sprints plus a Sprint 0 discovery phase. The overall timeline spans approximately 10 weeks of active development, aligned with the 5 research experiments.

10.1 Sprint 0: Discovery and Setup (Week 1)

Objective: Align team and client, establish tooling, and prepare for development.

- Stakeholder interview with Smart NV to confirm IQ Waves as the target puzzle and complete asset handover
- Set up Git repository, project structure (React + Vite), and project management board
- Review STL files and challenge booklets; confirm GDPR compliance approach with client
- Deliverable: Approved Project Plan; confirmed backlog for Sprint 1

10.2 Concept Delivery (End of Week 2)

- Deliver architectural design, tech stack confirmation, and low fidelity wireframes
- Present computer vision approach (FastAPI backend with YOLOv26 Nano Segmentation pipeline) to client for sign off

10.3 Sprint 1: Photo Capture and Board Detection (Weeks 2–4)

Demo Meeting 1 at end of Week 4.

Maps to Experiment 1 (photo capture reliability) and Experiment 2 (board detection accuracy).

- Build minimal React phone client with camera capture (input type="file"), image preview, and REST API upload to FastAPI backend
- Set up FastAPI backend with photo ingestion endpoint and Blender rendering pipeline for synthetic training data
- Train YOLOv26 Nano Segmentation model for board and piece detection
- Implement board corner detection from segmentation masks and perspective correction
- Capture test images at varied angles (0°–60°) and distances (20–100 cm)

Sprint 1 Success Criteria: Photo capture success rate ≥ 99% across 6 devices; board corner detection rate ≥ 95% at 45°/75 cm.

10.4 Sprint 2: Piece Detection and Grid Mapping (Weeks 4–6)

Demo Meeting 2 at end of Week 6.

Maps to Experiment 3 (ML piece detection) and the first half of Experiment 4 (end to end performance).

- Collect and annotate real world photos in Roboflow; combine with synthetic data for model retraining
- Integrate ONNX Runtime inference into the FastAPI backend pipeline
- Implement mask decoding and grid mapping to translate piece detections into a digital board state
- Build the backtracking solver with solvability check

Sprint 2 Success Criteria: mAP@0.5 ≥ 0.90, per piece recall ≥ 92%, end to end latency ≤ 2 seconds on Wi Fi.

10.5 Sprint 3: Hints, Completion, and UX Testing (Weeks 6–8)

Demo Meeting 3 at end of Week 8.

Maps to Experiment 5 (UX prototype and family testing) and the completion of Experiment 4 (end to end pipeline).

- Implement progressive hint engine (area, piece, exact) triggered via a Hint button in the UI
- Integrate voice assistant using the Web Speech API for reading hints aloud
- Implement completion detection with visual feedback
- Build the full application UI: board visualisation, piece status, hint display, completion screen
- Conduct user testing with several families (children ages 8+)

Sprint 3 Success Criteria: Full end to end demo working on real puzzle; all UX usability targets met; setup time ≤ 60 seconds; hint comprehension ≥ 70% unaided.

10.6 Final Delivery (End of Week 10)

- Polished application with React frontend and FastAPI backend deployed
- Complete documentation: Design Document, training pipeline README, deployment guide
- Final stakeholder presentation to Smart NV: research findings, experiment results, phased roadmap recommendation
- Handover package: source code, trained YOLOv26 Nano ONNX model, dataset, FastAPI server, and documentation

11. Success Metrics (MVP)

Metric	Target	Measurement Method
Setup Time	≤ 60 seconds	User testing observation
Hint Comprehension Rate	≥ 70% unaided	User testing task completion

Completion Detection Accuracy	≥ 95%	System telemetry
End to End Processing Time	< 3 seconds	Performance instrumentation
Photo Capture Success Rate	≥ 99%	System telemetry
Board Corner Detection Rate	≥ 95% (up to 45° angle)	Experimental validation
ML Piece Detection Accuracy	mAP@0.5 ≥ 0.90	Model evaluation
User Satisfaction Score	≥ 4.0/5.0	Post session survey

Table 4: Success metrics for MVP

12. GDPR Compliance and Privacy

All photo processing happens in memory on the backend server. Photos are never written to disk or stored in any database. No personal data is collected. The following principles apply:

- No Photo Storage: All images are processed in memory on the server and discarded immediately after processing
- No Personal Data Collection: The application does not collect names, emails, or any identifying information
- Stateless Processing: The backend processes each photo independently with no persistent storage of image data
- User Testing Consent: For user testing sessions with children, written parental consent will be obtained in advance
- Data Access: Only the intern and academic supervisor will have access to any anonymised user testing observations

13. Assumptions

- Users will have access to a modern smartphone with a camera and a reasonably up to date browser (Safari 15+ or Chrome 90+)
- Wi Fi or mobile data connectivity is available for initially loading the web application (no connectivity needed during use)
- Smart NV will provide STL files for all IQ Waves puzzle pieces and the board
- Parental consent will be obtained for user testing sessions with children
- The FastAPI backend will have sufficient resources to run ONNX Runtime inference with acceptable latency
- Several families will be available for MVP user testing

14. Dependencies

- Smart NV provides STL files and physical IQ Waves game units before Sprint 1
- Smart NV provides timely feedback at sprint demos (within 5 business days)
- Smart NV appoints an internal project lead as a single point of contact
- Smart NV recruits several families for user testing by Week 4
- Thomas More University provides academic supervision and evaluation
- Blender is available and suitable for synthetic data generation from STL files

- A server environment capable of running Python 3.11 and ONNX Runtime for backend deployment

15. Phased Roadmap

15.1 Phase 1 (This Project): Research and MVP

- Client/server architecture: React frontend with FastAPI backend for IQ Waves
- YOLOv26 Nano Segmentation model trained on synthetic + real world data
- Backtracking solver with progressive hints and voice assistant
- GDPR compliant, zero server architecture

15.2 Phase 2 (V2): Smarter, More Products

- Multi puzzle support with generalised board and piece detection models
- Automated training pipeline (capture, annotate, train, deploy)
- User accounts and progress tracking (GDPR compliant)
- Dynamic hint system with visual overlays
- Gamification, animations, and sound effects

15.3 Phase 3 (V3): Native Mobile App

- React Native or Flutter mobile application
- On device ML using CoreML (iOS) and TFLite (Android)
- Offline mode and AR overlays (ARKit/ARCore)
- Integration with Smart NV product catalog