Project Plan

Smart NV Computer Vision Puzzle Tracking System

IQ Puzzler Pro front board, back board and Remix prototype

Field	Value
Academic Year	2025-2026
Institution	Thomas More University of Applied Sciences
Client	Smart NV
Student	Marin Janushaj
Programme	Bachelor Applied Computer Science
Company supervisor	Wouter Caeldries
Document type	Updated internship project planning
Version	2.0 - updated after final implementation

This document updates the March project plan to match the work that was actually realised during the internship. It keeps the planning structure expected for Thomas More documentation while reflecting the final scope: IQ Puzzler Pro front-board recognition, back-board recognition, synthetic-data/model work and the Remix puzzle-generation prototype.

Table of Contents

1. Introduction

This project plan defines the updated scope, objectives, technical approach, risks, planning and evaluation strategy for the Smart NV internship project. The original March planning focused on a smaller research direction. The final internship work evolved into a broader IQ Puzzler Pro companion prototype with front-board recognition, back-board recognition, solver-supported gameplay, synthetic-data generation and a Remix extension for new puzzle shapes.

The plan should be read as an up-to-date project planning document rather than as a final realisation paper. Its purpose is to explain what had to be planned, which decisions guided the work, how the work was divided into phases, and which criteria were used to judge whether the prototype was successful.

1.1 Document Goals

- Clarify the problem, target users, project scope and boundaries.
- Translate the realised internship work into an organised planning structure.
- Show how computer vision, solver logic, frontend interaction and data generation were planned together.
- Document the risks, dependencies, success criteria and future roadmap.

1.2 Document Status

This version is intentionally updated after implementation. It therefore contains both planned objectives and a realistic status view of what was achieved. This makes the planning consistent with the final project evidence instead of preserving outdated March planning assumptions from the March version.

2. Background

2.1 Client Context

Smart NV is a Belgian toy and game company known for logic-based SmartGames products. The company designs physical puzzle games where players solve challenges by placing shaped pieces onto a board. IQ Puzzler Pro was the main physical product used in this internship because it has clear board states, fixed pieces and a strong fit with solver-based digital assistance.

The business opportunity is to explore whether digital technology can support a physical puzzle without replacing the physical play experience. A companion app can help a player when they are stuck, validate a board state and make the product more accessible for children who may not want to read a solution booklet.

2.2 Current Situation

The physical game is normally played without digital support. When a player becomes stuck, the only official support is the printed booklet or help from a parent, teacher or supervisor. This can interrupt play, especially for younger players. The company also did not already have a trained dataset, a computer-vision pipeline or a working solver-connected application for this exact puzzle workflow.

2.3 Project Opportunity

The project opportunity was to build a web-first proof of concept that can connect a real puzzle board to digital logic. The application should take a photo, analyse the board, reconstruct a logical 55-cell state, and then use deterministic puzzle logic to validate, solve or guide the player. Later in the internship, the same solver idea also supported a Remix prototype that generates new playable 55-cell shapes.

3. Project Vision and Goals

3.1 Vision Statement

Create a computer-vision-powered digital companion for IQ Puzzler Pro that recognises physical board states and gives solver-backed feedback while keeping the physical puzzle experience central.

3.2 Primary Goal

Build an explainable prototype where phone photos, model output, grid mapping, solver validation, hints and manual correction work together. The goal is not only to detect pieces visually, but to turn visual evidence into a board state that can be trusted by puzzle logic.

3.3 Business and Learning Goals

- Reduce player frustration by providing contextual hints instead of only complete printed solutions.
- Demonstrate whether Smart NV puzzle products can be supported by a digital companion.
- Create a reusable foundation for later product experiments: phone capture, model training, solver integration and dashboard interaction.
- Explore new puzzle content through Remix, where validated 55-cell shapes can become playable screens, booklets or STL exports.
- Build technical learning in computer vision, Blender synthetic-data generation, React dashboards, FastAPI backends and exact-cover solving.

4. Puzzle and Product Specification

4.1 IQ Puzzler Pro

IQ Puzzler Pro contains twelve coloured pieces made of connected balls. The relevant digital representation is a 55-cell board state. The front side is a rectangular 5 by 11 board. The back side uses the same pieces but has a different square-diamond active layout. A valid solution fills the active cells exactly once with no overlap and no empty holes.

Element	Planning Implication
Front board	5 by 11 rectangular grid; suited to perspective warp and regular grid mapping.
Back board	55 active wells in a square-diamond arrangement; required a separate board mask and dashboard renderer.
Pieces	Twelve fixed shapes with rotations; all placement and hints depend on exact piece geometry.
Board state	55-character logical representation used by detection, dashboards, validation, solving and hints.
Solver	Exact-cover/backtracking logic checks whether a partial state can still be completed.
Remix	Custom 55-cell masks can become new challenges if the solver can tile them.

4.2 Piece and Class Planning

The model and solver use stable class names and piece identifiers so that detections can be converted into the same identifiers used by the digital board. The front and back workflows share the same physical pieces but require different board-localisation and mapping logic.

Class Group	Purpose
Piece classes 0-9, A and B	Twelve physical pieces, each with its own shape and colour.
Board or tray class	Segmentation class used to localise the board/tray surface and support perspective mapping.
Empty cells	Cells represented as empty in the board state until a piece is placed or detected.
Hint cells	Cells returned by the solver for progressive hint display and optional apply-to-board actions.

5. Objectives and Stakeholders

5.1 Ultimate Objective

Deliver a working research prototype for IQ Puzzler Pro that supports phone capture, AI-assisted board reconstruction, solver validation, hints, solving, manual correction, debug review and Remix puzzle generation.

5.2 Functional Requirements

- Photo Capture (P1): browser phone clients allow the user to capture or upload photos for the front and back boards.
- Front Detection (P2): the backend uses YOLO v26 nano segmentation and OpenCV mapping to reconstruct the front 5 by 11 board.
- Back Detection (P3): the backend maps detections onto the back board's square-diamond 55-cell coordinate system.
- Dashboard Apply Flow (P4): model output is shown as a candidate state before the user applies it to the playable board.
- Manual Correction (P5): dashboards allow selection, rotation, placement and removal of pieces when detection is uncertain.
- Validation and Solving (P6): solver endpoints check whether a current board state is valid and can return a complete solution.
- Progressive Hints (P7): the app provides three hint levels, from light guidance to exact placement.
- Debug Review (P8): every relevant detection path produces inspectable evidence such as original image, masks, warp, grid and final labels.
- Synthetic Data and Fine-Tuning (P9): Blender renders and Roboflow annotations support model training and real-photo adaptation.
- Remix (P10): users can play generated masks, validate custom 55-cell shapes, export booklets and generate STL output for physical experiments.

5.3 Non-Functional Requirements

Requirement	Planning Target
Responsiveness	Use a lightweight model and backend processing so the photo-to-result loop remains usable.
Explainability	Keep debug output for model masks, board warp, grid mapping and final state.
Usability	Provide manual correction because camera-based detection cannot be perfect in all real-world conditions.
Privacy	Avoid collecting personal information; local debug images are development artefacts and should be removed or disabled for production use.
Portability	Use web-first React/Vite clients and a FastAPI backend so the prototype can run locally and be packaged for handover.
Maintainability	Separate phone clients, dashboards, backend routes, model helpers, board logic and solver code.

5.4 Stakeholders

Stakeholder	Role	Value or Responsibility
Smart NV	Client and product owner	Provides product context and evaluates whether the prototype supports physical play.
Wouter Caeldries	Company supervisor	Provides guidance, feedback and practical direction during the internship.
Marin Janushaj	Intern and developer	Builds the prototype, training workflow, solver integration, documentation and final delivery.
Thomas More	Academic institution	Evaluates project planning, realisation quality, methodology and reflection.
Children and families	Target users	Use the companion to receive hints and continue playing independently.
Classmates / internship peers	Peer support	Discuss blockers and help validate assumptions during difficult technical phases.

6. Scope and Risk Analysis

6.1 In Scope

- React/Vite phone clients for front-board and back-board photo capture/upload.
- FastAPI backend routes for detection, solver actions, hints and Remix endpoints.
- YOLO v26 nano segmentation model direction for board and piece masks.
- OpenCV-based board localisation, perspective correction and grid mapping.
- Front dashboard and back dashboard with manual correction, validate, hint, solve and clear flows.
- Exact-cover/backtracking solvers for board validation, solving and hint generation.
- Blender synthetic-data generation scripts and Roboflow real-photo fine-tuning workflow.
- Debug pages and saved detection artefacts for explaining failures.
- Remix Daily, Themes, Shape Studio, Challenge Booklet, hinting and STL export prototype.
- Documentation and handover package for Smart NV and academic evaluation.

6.2 Out of Scope

- Commercial production release or App Store / Play Store publication.
- Native mobile app implementation; web-first clients were used during the internship.
- Real-time video streaming and augmented-reality overlays.
- User accounts, long-term progress tracking, payments or subscriptions.
- Full large-scale user testing with families; the internship focused on prototype validation and technical feasibility.
- A separate confidential labelling/training side task, which is intentionally not described in this project plan.

6.3 Risk Analysis

Risk	Severity	Mitigation / Actual Response
Model does not generalise from synthetic images to real photos	High	Combine Blender synthetic data with Roboflow real photos; use debug failures to guide new examples.
YOLO mask is good but board placement is wrong	High	Inspect warp/grid/final-label debug artefacts and improve geometric mapping separately from model training.
Back-board coordinate system is misunderstood	High	Standardise the back board as a square-diamond 55-cell model across detector, dashboard and solver.
Large model is too slow or heavy for phone-oriented usage	Medium	Plan around YOLO v26 nano because it is small, loads quickly and keeps future phone/edge deployment realistic.
Manual correction is too hard for users	Medium	Add selectable pieces, rotation controls, remove actions, apply detected state and progressive hints.
Blender and synthetic data take longer than expected	Medium	Use scripts and calibration constants to automate generation instead of manually producing every image.
Scope expands beyond recognition into new product ideas	Medium	Treat Remix as an extension/prototype with separate scope and clear deliverables.
Confidential information appears in public documentation	High	Keep confidential side work generic and do not name protected internal projects.

7. Technical Approach

7.1 Architecture Overview

The system uses a web-first client/server architecture. Phone clients handle capture and upload. The FastAPI backend performs model inference, computer-vision processing, board-state reconstruction and solver calls. Dashboards display the current board, candidate detections, solver results and manual correction controls. Remix uses separate frontend pages and backend solver/export logic for generated masks.

Layer	Responsibility
Phone clients	Capture/upload photos for front and back boards; show result or review state.
Backend API	Receive images, run detection, call solvers and return structured board results.
Computer vision	YOLO segmentation, board mask extraction, perspective warp, grid mapping and debug artefacts.
Game logic	Piece definitions, rotations, board masks, exact-cover solving, validation and hints.
Dashboards	Apply detected state, correct pieces manually, validate, hint and solve.
Remix	Generate/validate custom masks, play challenges, export booklets and STL files.

7.2 Computer-Vision Pipeline

- Image input: phone photo or gallery upload is sent to the backend.
- YOLO v26 nano segmentation: model detects board/tray and piece masks.
- Board localisation: the board mask or contour is used to estimate board position.
- Perspective correction: OpenCV homography creates a flatter board view.
- Grid mapping: detected masks are translated into exact cells.
- Shape-constrained placement: piece definitions constrain how detections become legal cells.
- Solver validation: the board state is checked before it is treated as trustworthy.
- Debug output: original, annotated, warped, grid and labelled views are saved for inspection.

7.3 Model Strategy: YOLO v26 Nano

YOLO v26 nano segmentation was chosen as the practical target model because the companion workflow is phone-driven. Even when inference runs on the backend during the prototype, the user experience still depends on fast loading and fast response after a photo is submitted. A small nano model also keeps the project aligned with possible future on-device or edge deployment, where memory and startup time matter more than in a desktop-only demo.

A larger model could provide higher accuracy in some difficult cases, but the planning decision was to start with the smallest useful segmentation model and strengthen the rest of the pipeline with geometry, solver validation and manual correction. This is why the project combines AI evidence with deterministic checks instead of relying only on model confidence.

7.4 Data Strategy

Data Source	Purpose	Planning Notes
Blender synthetic images	Scale the dataset with automatic labels	Randomise camera, lighting, backgrounds, board fill level and piece combinations.
Roboflow real photos	Reduce synthetic-to-real gap	Annotate real board and piece examples from phone photos.
Debug failures	Drive improvements	Use failed runs to decide whether to fix model data, board localisation, grid mapping or solver assumptions.
Colab training	Train/fine-tune model	Use GPU training workflow and export usable model artefacts for the backend.

7.5 Solver and Hint Engine

The solver is a deterministic exact-cover/backtracking component. It receives a 55-cell board state and checks whether the empty cells can be covered by the remaining pieces without overlap. It is used for validation, complete solving and progressive hints.

- Level 1 hint: identify a useful next piece without revealing exact cells.
- Level 2 hint: reveal a starting cell or smaller target area.
- Level 3 hint: reveal the exact placement and allow applying it to the board.

7.6 Remix Extension

Remix extends the original product idea by using the solver on arbitrary 55-cell masks. A shape can become playable only if the solver can tile it with the twelve IQ Puzzler pieces. The extension includes daily shapes, themed challenge lists, Shape Studio, generated booklets and STL export for physical prototyping.

8. Technology Stack

Area	Technology	Reason for Selection
Frontend	React, Vite, TypeScript/JavaScript	Fast web-first development for phone clients, dashboards and Remix pages.
Dashboard rendering	Konva / canvas-style board rendering	Needed interactive board drawing, piece placement and visual hints.
Backend	Python, FastAPI, Uvicorn	Simple API layer for image upload, detection, solver actions and Remix endpoints.
Computer vision	OpenCV, NumPy	Board localisation, perspective warp, grid mapping and image debug artefacts.
ML model	YOLO v26 nano segmentation	Small, fast-loading model suitable for phone-driven workflows and future edge/on-device direction.
Training	Blender, Roboflow, Colab, Ultralytics workflow	Combine synthetic scale with annotated real photos and GPU fine-tuning.
Solver	Python exact-cover/backtracking search	Deterministic validation, solving and hint generation.
Exports	PDF/booklet generation and STL export	Turn Remix from a digital prototype into printable or physical puzzle output.

9. Research Experiments and Validation Plan

The project was evaluated through practical experiments rather than only through one final metric. The most important validation was whether the full chain could turn real photos into board states that the solver could validate and the user could correct.

Experiment	Validation Question	Evidence or Measurement
Photo capture	Can phone/gallery input provide usable images?	Front and back phone clients, upload status, review screens.
Front-board detection	Can a real photo become a 5 by 11 board state?	Debug outputs: original, YOLO overlay, warp, grid and final labels.
Back-board detection	Can the square-diamond board be mapped consistently?	Back debug view, mapped state, solver quality flags and dashboard correction.
Solver integration	Can validation, hints and solve actions use the detected/current board?	Validate, three-level hints and solve flows on front and back boards.
Synthetic/real data	Can generated data and real annotations support training?	Blender scripts, Roboflow packages, Colab training/fine-tuning workflow.
Remix	Can new 55-cell shapes become playable/printable?	Shape validation, daily/theme play, booklet generation and STL print test.

9.1 Success Metrics

Metric	Target	Status / Validation Method
Photo-to-dashboard flow	User can capture/upload and see a processed result	Implemented for front and back phone workflows.
Board-state reconstruction	Detected board becomes a 55-cell state	Validated through debug views and dashboard apply flow.
Solver validation	Partial states can be checked for solvability	Implemented through Validate endpoints and dashboard feedback.
Hint system	Three progressive levels	Implemented for front and back boards; Remix supports solver hints for masks.
Model performance	Small enough for responsive workflow	YOLO v26 nano selected for fast loading and smaller model size; exact exported metrics should be added from Colab logs if required.
Manual recovery	User can correct model mistakes	Dashboards support piece selection, rotation, placement, removal and clear/revert flows.

Remix physical proof	STL output fits real pieces	Printed prototype showed the generated board could hold the physical pieces.

10. Project Planning

The internship work was planned as an iterative research-and-build process. The project did not move in a perfectly linear way because model training, board mapping and UI usability influenced each other. The sprint structure below describes the updated plan that matches how the final project should be understood.

Phase	Focus	Main Deliverables
Sprint 0 - Discovery and setup	Understand IQ Puzzler Pro, assets, rules, piece shapes and technical baseline	Repository setup, board-state representation, initial architecture, risk list.
Sprint 1 - Front capture and first detection	Build phone capture and backend image flow for the front board	Front phone client, upload endpoint, first YOLO/OpenCV debug outputs.
Sprint 2 - Front dashboard and solver	Turn detected/current state into an interactive playable board	Front dashboard, apply state, validate, solve and progressive hints.
Sprint 3 - Data generation and training	Create training data and prepare model improvement loop	Blender synthetic-data scripts, Roboflow annotations, Colab training package.
Sprint 4 - Back-board workflow	Adapt the system to the back side's square-diamond board	Back phone client, back debug view, back dashboard, solver-compatible mapping.
Sprint 5 - Robustness and correction	Fix real-photo failures and improve usability	Manual correction flow, better debug review, refined board mapping and hints.
Sprint 6 - Remix extension	Explore generated puzzle shapes and physical output	Daily/themes/studio/play views, booklet export, STL export and printed prototype.
Final delivery	Prepare project handover and academic documentation	Clean repository structure, documentation, realisation paper, project plan and final presentation material.

10.1 Milestones

- First working photo upload to backend.
- First front-board debug output with model overlay, warp and grid.
- First solver-connected front dashboard with validate, hint and solve.
- First Blender-generated synthetic dataset and Colab training package.
- First back-board detection result that uses the square-diamond board representation.
- First end-to-end manual correction workflow for both board sides.
- First Remix custom shape validated by the solver.
- First STL-generated physical board tested with real pieces.

11. GDPR, Privacy and Data Handling

The prototype is a research and development system. For production, photos should be processed without storing identifiable data. During development, debug artefacts were useful and sometimes necessary, but they should be treated as local technical evidence rather than user data for long-term storage.

- No user accounts or personal profiles are required for the prototype.
- Real photos used for model improvement should avoid faces and personal identifying details.
- Development debug images should be stored locally, reviewed only for technical purposes and deleted when no longer needed.
- If children participate in formal testing, parental consent and school/company privacy rules must be followed.

12. Assumptions and Dependencies

12.1 Assumptions

- Users have access to a modern smartphone browser and a laptop/desktop browser for dashboard testing.
- The backend has enough resources to run YOLO v26 nano inference and OpenCV mapping responsively.
- The physical board and pieces are available for real-photo testing.
- Model quality can be improved iteratively through additional real photos and synthetic data.
- Solver definitions correctly match the physical piece shapes.

12.2 Dependencies

- Smart NV provides physical products, product context and feedback.
- Blender remains available for synthetic-data generation and geometry calibration.
- Roboflow or equivalent annotation tooling remains available for real-photo datasets.
- Colab/GPU training access remains available for model training and fine-tuning.
- Thomas More project deadlines define the documentation and presentation schedule.

13. Phased Roadmap

13.1 Phase 1 - Internship Prototype

- Front and back board photo workflows.
- YOLO v26 nano segmentation with OpenCV mapping.
- Solver validation, hints, solve actions and manual correction.
- Synthetic-data and real-photo training workflow.
- Remix proof of concept with booklet and STL output.

13.2 Phase 2 - Product Hardening

- Collect more real photos in difficult lighting, backgrounds and angles.
- Export final training metrics and compare model variants if accuracy becomes more important than size.
- Improve mobile usability and reduce manual correction friction.
- Add a production privacy mode that avoids persistent debug photo storage.

13.3 Phase 3 - Future Product Direction

- Investigate native or on-device deployment using CoreML/TFLite if Smart NV wants a mobile app.
- Evaluate Remix with real players before considering commercialisation.
- Explore additional SmartGames products only after the IQ Puzzler Pro pipeline is stable.

14. References

SmartGames - IQ Puzzler Pro product page: https://www.smartgames.eu/
Ultralytics documentation: https://docs.ultralytics.com/
FastAPI documentation: https://fastapi.tiangolo.com/
React documentation: https://react.dev/
Vite documentation: https://vite.dev/
OpenCV documentation: https://docs.opencv.org/
Blender Python API: https://docs.blender.org/api/current/
Roboflow documentation: https://docs.roboflow.com/