THOMAS MORE

Internship at Smart NV

Reflection

Marin Janushaj
Student Bachelor Applied Computer Science

2025 – 2026

Table of contents

1. Introduction

This reflection looks back on my internship project at Smart NV. The project was about building and improving a computer-vision puzzle tracking system for IQ Puzzler Pro. It included the front-board workflow, the back-board workflow and an extra Remix prototype for generating and exporting new playable puzzle boards. The goal of this reflection is to describe what I achieved, what the project means for Smart NV and the users, what still remains to be done, and what I learned personally during the internship.

The reflection is divided into two main parts. The substantive reflection focuses on the project itself: the delivered functionality, the value for the client, the current state and future advice. The personal reflection focuses on my own growth, the technical and non-technical problems I faced, and the way the internship changed how I work.

2. Substantive reflection

During my internship at Smart NV I worked on a computer-vision puzzle tracking system for IQ Puzzler Pro. The project focused on recognising physical puzzle boards from photos and converting those detections into a digital board state that could be used by the application. I worked on both the front board and the back board, because they have different shapes, different board geometry and different detection problems. I also worked on the Remix prototype, where generated puzzle shapes could be tested digitally and exported as STL files for 3D printing.

The main result is a working prototype that connects several parts of the pipeline: image upload or photo capture, YOLO v26 nano segmentation, board and piece detection, board-state reconstruction, validation, hints and solving. The decision to use a nano model was important because the application should remain realistic for phone-oriented usage. A larger model could be heavier and slower, while a small model loads faster and keeps future mobile deployment more practical.

For the data part I used two approaches. Real photos were labelled and exported through Roboflow, while Blender scripts were used to generate synthetic training images with automatic labels. The synthetic data was useful because it made it possible to create many board states, rotations, lighting conditions and backgrounds without manually photographing every situation. The real images were still needed because the final model has to work on actual phone photos, not only on clean rendered images.

One of the most difficult technical parts was the step after detection. A good YOLO mask does not automatically mean that the pieces will be placed correctly on the digital board. The application also needs robust geometry: the board has to be mapped to the correct grid, each detected piece must be assigned to the right cells, and the front-board and back-board coordinate systems must be treated differently. This taught me that a computer-vision system is not only the model. The surrounding logic, debug tools and user correction flow are just as important.

The project creates value for Smart NV because it shows how a physical puzzle can be supported by a digital tool. Instead of manually entering every piece, a user can start from a photo and let the app reconstruct the board state. The solver and hint system can then help the player continue. The hints are progressive: the first levels give support without immediately giving away the full answer, while the final level can show the placement more directly. This makes the application useful as a learning and assistance tool rather than only as a solution generator.

The Remix part added another direction to the project. During the AI Jam on 21 May I worked with Bart Van Assche, while Ibrahim and Wouter worked together in another group. The first idea was more digital, but after feedback that a purely digital game would not be as enjoyable as a real physical product, the concept moved toward printable puzzle boards. The STL export was an important proof that the software could produce something physical, with well depth and spacing calculated so real pieces could fit.

The project is not fully finished as a production product yet. It is a strong prototype and technical foundation, but it still needs more real-world testing before it can be released to users. The most important remaining work is to collect more real photos, especially difficult cases with different lighting, phone angles, nearly full boards and partly hidden pieces. The model and board-mapping pipeline should also be evaluated with fixed metrics, such as board detection success, piece classification accuracy and final board-state accuracy.

The main points I would advise Smart NV to continue with are:

- expand the real-photo dataset for both front-board and back-board situations;
- continue improving the Blender synthetic data so it matches real camera photos more closely;
- test the model, geometry mapping and user interface separately so failures can be diagnosed correctly;
- keep the model small unless there is a clear reason to move away from the phone-oriented nano model;
- test the hint, solve and manual correction flows with real users;
- decide whether Remix should become a product direction, a design tool or only a research prototype.

At this stage I would describe the result as an internal prototype used for testing, debugging and demonstration. It proves the main concept, but a final version should be hardened with more data, better evaluation and user testing.

3. Personal reflection

This internship meant a lot to me because it showed me the difference between a school project and a project in a real company environment. In school, the assignment is usually more clearly defined. During this internship, the problem changed as I learned more about it. At first I thought the hardest part would be training a model, but later I realised that the full system around the model was just as important: data, geometry, debugging, user interface, solver logic and product decisions.

I learned many technical skills during the project. I improved my understanding of full-stack development, image processing, YOLO segmentation, dataset preparation, Roboflow, Google Colab training, Blender scripting, synthetic data generation and solver logic. Blender was especially challenging because I had not used it before for synthetic data generation. At the beginning it slowed me down, but later it became one of the most important parts of the project.

I also learned to think more like a systems developer. The application was not one isolated feature. It was a pipeline where every step influenced the next step: photo capture, model inference, mask selection, board transformation, grid mapping, piece placement, validation, hints and user correction. If one part was slightly wrong, the final board state could be wrong. Because of that, I learned to debug step by step instead of changing many things at once.

One of my biggest personal difficulties was time management. The office was more than one hour away, and combining travel, development, documentation and learning new technologies was not always easy. There were weeks where I felt stuck on the same issue, especially when the detected pieces did not sit correctly on the digital board. Those weeks were frustrating, but they also taught me persistence.

When I was stuck, I handled it by reading the code again, checking debug outputs, testing new approaches and asking for help. The company helped me with this. My company supervisor, Wouter Caeldries, treated us very well. He did not apply pressure in a negative way, and he was kind and supportive when things were difficult. That made it easier to ask questions. My classmates Ibrahim Afkir and Abdul Salam Aldabik also helped me when I was blocked, because talking through a problem often made it clearer.

The internship also made me more open to working in an open environment. Before this, I was less used to showing unfinished work or explaining why something did not work yet. During the internship I learned that this is part of professional work. If I explain the problem clearly, other people can help and the project moves forward faster.

Another lesson was that a technically possible idea is not automatically a good product idea. In the Remix project, the direction changed after feedback that a fully digital version of the game would not be enjoyable enough for users. That feedback helped me understand that user experience and product value must guide the technical decisions. The stronger idea was not only to generate a digital board, but to make something that could become a real printed puzzle.

I grew most in persistence, independence and communication. At the start, when something failed, I sometimes wanted to fix it quickly without fully understanding the cause. Later I learned to isolate the problem first: is it the dataset, the model, the mask, the coordinate mapping, the solver or the interface? This made me more structured and more professional in how I approach technical problems.

Looking back, I am proud of the progress I made. The project was difficult because it combined AI, synthetic data, physical game geometry, software development and user experience. There were moments where I was stuck for a long time, but those moments taught me the most. I now understand better that a developer does not only write code. A developer also has to understand the problem, communicate clearly, test honestly, document decisions and think about whether the solution is usable.

In conclusion, this internship helped me grow both technically and personally. I became more confident with unfamiliar technologies, more open to collaboration and more aware of the connection between technical decisions and real user value. I will take these lessons with me into future projects.