Chess AI Coach- Block A Challenge

Louie Daans, Zakariae el Moumni, Nicola Ioniță
232345, 226324, 230632
Department of ADS&AI, AGM Academy
Breda University of Applied Sciences
Edirlei Soares de Lima
10 January, 2025

The following report provides a detailed, chronological analysis of our project, focusing on the successful aspects and the difficulties encountered, in addition to the lessons learned throughout the process. Our goal was ambitious: to explore the complexities of chess programming and create an artificial intelligence chess coach capable of analyzing games in incredible detail. By creating interactive content with a Large Language Model (LLM), we envisioned an artificial intelligence experience like having a personal version of the popular chess analysts like Gotham Chess. Rather than just passively watching the analysis of other people's games, users are able to engage with an AI that analyzes and explains their games in a way that feels like human interaction.

While we expected problems, we underestimated the difficulty of one of the project's fundamental elements—the chess engine itself. To address this, we broke down the bigger goal into three progressive blocks in line with the project challenges for Weeks 9 and 10:

1. *Chess Engine Creation* – Creating and modifying a chess engine to fit our project needs.
2. *Basic Chess AI Coach*—Implementing features similar to those on platforms like Chess.com, which translate positions and openings into easily understandable text.
3. *Advanced LLM Integration*—Augmenting our system with state-of-the-art LLM capabilities so that the AI can produce high-quality, engaging, and informative content.

For this block, we worked mainly towards developing the chess engine. First, we assigned each other roles in order to have a clear and defined collaboration. {link}

The project aimed at combining our passion for chess with state-of-the-art programming to lay the foundation of an artificial intelligence system that revolutionizes game analysis and understanding.

Following the start of the project, we did independent research into our various domains, complemented by team meetings where we shared updates and aligned our objectives. The research mostly focused on a few key areas:

1. Market Research and Competitor Analysis: This involved identifying the needs of different chess user groups—from casual players to competitive champions—coupled with analysis of Chess.com, Lichess, and other platforms. We assessed features such as individualized coaching, game-based learning features, and in-depth analytical functionality to find opportunities our AI chess coach could fill.

2. Chess Engine Development: We researched existing engines like Stockfish and Lc0. While Stockfish provided a good balance of performance and adaptability for integration with a neural network, the neural-network-based architecture of Lc0 was very resource-intensive. In the end, Stockfish's solid documentation and community support made it the perfect base for our project.

3. Neural Network Applications: Research on convolutional neural networks (CNNs), transformers, and reinforcement learning brought out their potential roles. CNNs were good at static board evaluations, while transformers excelled at multi-move planning and educational explanations. Combining models into an ensemble approach emerged as a promising direction for achieving strategic recommendations and personalized feedback.

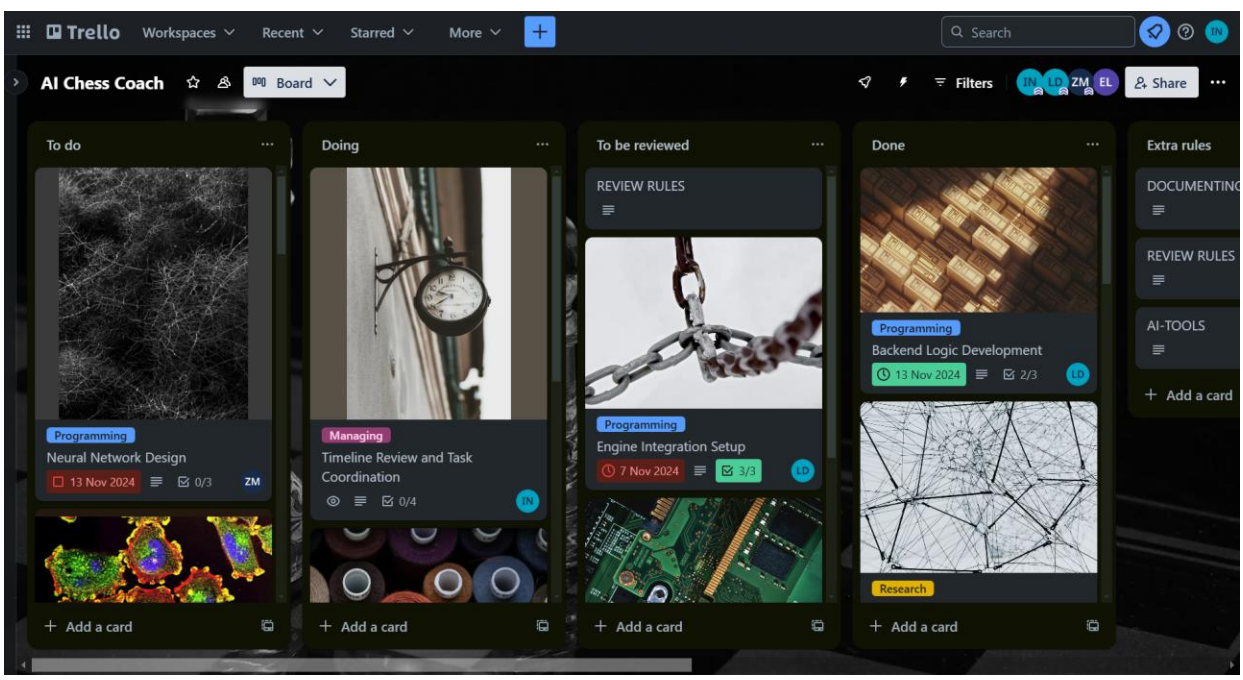
4. Computing Efficiency: We studied local and cloud-based solutions to meet the required computational demand. Though cloud computing supported intensive processing needs of this solution—for tasks like training a neural network—local setups worked out more effectively during the development and integration phase.
These focus areas, supported by our collaborative efforts, lay the groundwork for developing an innovative AI chess coach that not only analyzes games but also provides tailor-made, instructive feedback.

Role 1- Project Manager

- **Task 1: Trello Board Management**

On the first day of the project, I set up the Trello board to manage the structure, tasks, and deadlines efficiently. Having learned about Trello during Block D of my first year, I have since used it extensively for both academic and personal projects. For this project, I created multiple lists such as "TO DO," "DOING," "TO BE REVIEWED," and "DONE" to maintain clarity and ensure smooth task tracking. Each card on the board was labeled according to its nature—such as "Research," "Programming,“, “Managing” or “Learning"—and assigned to specific team members. I also added UX elements like task descriptions, checklists, and deadlines to provide better clarity for each assignment. The board was updated after our daily meetings to reflect the current progress and adjustments, making it a living tool for project management. A critical aspect of this task involved breaking down larger tasks into smaller, more manageable chunks and coordinating team efforts to ensure a consistent workflow. While the initial structure was completed on Monday, updates and refinements were necessary every few days to accommodate evolving project needs.

Example of cards: 1, 2, 3, 4, 5.



Before the start of the project, we had a meeting to discuss the “rules” of uploading and reviewing. Firstly, we used OneDrive for uploading the files and add the link into the Trello card for that specific task. In order to have a document reviewed, you are supposed to move your card in the “TO BE REVIEWED” list. After that, the other teammates check the file and comment “reviewed”. Only after getting the comments, we are supposed to upload the files into the GitHub repository.

To keep everything neat, before using branches in GitHub, we wanted to refreshen our memory about this topic: https://trello.com/c/WWwTxdKC.

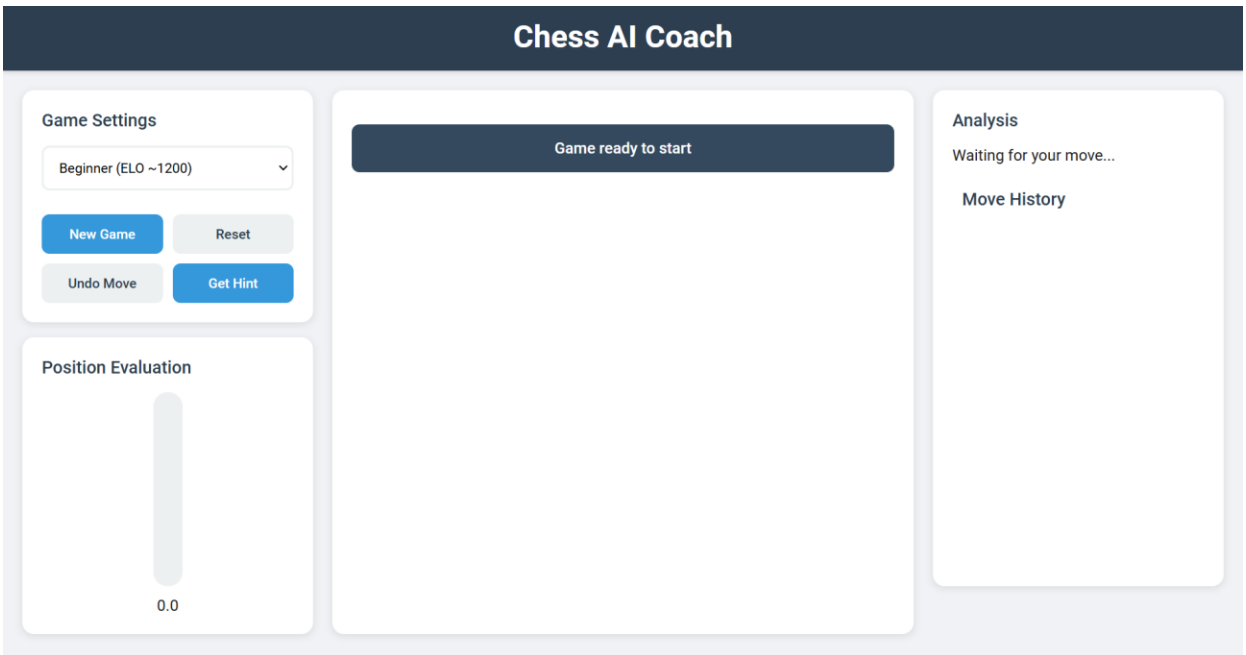
- **Task 2: Market Research**

Over the second and third days of the project, I conducted extensive market research to analyze competitors, understand consumer behavior, and evaluate the technical feasibility of our project. This research focused on identifying gaps in existing platforms like Chess.com, Lichess, and Play Magnus. For example, Chess.com has robust community engagement but lacks personalized AI-driven coaching features, while Lichess provides free unlimited analysis but lacks polish and advanced features. These insights highlighted opportunities for our project to differentiate itself by offering tailored AI coaching and gamified learning features. The process involved reviewing competitor platforms, exploring market reports, and synthesizing findings into actionable conclusions. By Wednesday, I had compiled the key findings, which shaped our project's direction.

Stakeholder	Level of Influence	Level of Interest	Primary Interests/Concerns	Engagement Strategy
Project Team (Developers, PM, etc.)	High	High	Project success, milestones, technical challenges	Involve in decision-making, regular updates, feedback for alignment
Investors	High	Medium	ROI, market position, revenue generation	Regular reports on progress, milestones, and financial alignment
Primary Users (Chess Enthusiasts, etc.)	Medium	High	User experience, educational value, accuracy	Engage via feedback, beta testing, and surveys for feature input
Educational Institutions and Clubs	Medium	Medium	Learning resources, accessibility, teaching integration	Provide demos, training resources, explore partnership opportunities
Secondary Users (Developers, Researchers)	Low	Medium	Data access, technical documentation, integration	Access to documentation, development tools, and research collaboration
Competitors	Medium	Low	Market trends, technology advancements	Monitor strategies, limit direct engagement, stay informed on trends
Regulatory Bodies	High	Low	Data privacy, AI compliance, ethical data use	Ensure compliance, prepare for audits, maintain transparency
Marketing and Sales Teams	Medium	Medium	Market reach, user acquisition, promotional resources	Collaborate on messaging, provide project updates for marketing
Community Influencers and Content Creators	Low	Medium	Content collaboration, engagement, audience insights	Regular updates, early access for review, partnerships for co-promotion

- **Bonus Task**

Additionally, although frontend development was not part of my initial responsibilities, I decided to explore this area as a personal challenge. After struggling to understand how to use the HTML and CSS files, I managed to get this skill. I also explored existing chess project repositories on GitHub, I managed to create a basic interface. However, I encountered significant difficulties with getting the chess table to work properly and decided to step back and refocus on my assigned tasks after spending two days on this.



- **Task 3: Research on Computing Efficiency**

On Saturday of the first week, I focused on researching ways to optimize computational efficiency for the project. This involved comparing the advantages and disadvantages of cloud services such as AWS and Google Cloud versus local computing. Cloud computing was appealing due to its scalability and cost-efficiency for neural network training, but it raised concerns about latency, dependency on stable internet connections, and data privacy. Local computing, on the other hand, provided a cost-effective solution for initial development and testing phases, especially for smaller tasks that didn't require intensive processing power. The research helped the team decide on a hybrid approach, leveraging local resources for development and cloud computing for resource-heavy tasks. This work was completed within the scheduled five hours and documented for the team's reference.

- **Task 5: Code Review and Team Reflection**

The final day of Week 1 was dedicated to reviewing the project's progress and preparing for the second week. During our team meeting, Louie presented his work on the chess engine and discussed various integration options. I updated the Trello board with a new card documenting the meeting and added a checklist to track our agreed-upon tasks for Week 2. The meeting also included a reflective discussion on challenges and successes, ensuring that everyone was aligned moving forward. After the meeting, I provided detailed feedback on my teammates' research and code, which helped refine their work and ensured consistency across the project.

- **Bonus Task**

Despite the initial challenges, taking a break was helpful, the frontend design came together, thanks to some timely help and insightful explanations from Louie. He provided the foundational code for the chessboard, complete with all the rules and mechanics, which served as a solid starting point for my work. With his guidance, I was able to grasp the complexities of integrating the chessboard's functionality, ensuring that it was interactive and adhered to the game's rules seamlessly.

Building on this foundation, I designed a sleek and intuitive interface, adding buttons, layouts, and other elements that enhanced the user experience. The process of bringing the frontend to life was both exhilarating and rewarding, especially as each component started to function as intended. Louie's support and the collaborative effort made the challenges feel more manageable, and seeing the final result—a functional and interactive chessboard—was satisfying.

Somehow, I can't find the final version of it.

- **Bonus Task**

On Tuesday of the second week, I focused on one of the new tasks from the Sunday meeting: Implementing your own bot into Lichess. This involved researching the Lichess API, understanding how bots are registered and integrated, and preparing our chess engine to interact seamlessly with the platform. While the process was challenging due to the technical depth of the API and the need to ensure compatibility with our engine, it provided valuable insights into bot functionality and potential areas for improvement in future iterations.

- **Task 5: Code Review**

On Wednesday, I conducted a thorough code review of Louie's work. My goal was to understand how the chess engine and its components were structured and implemented. I analyzed the logic, checked for any inconsistencies, and ensured alignment with our project goals. This review also helped me identify areas where the integration with other components, such as the frontend and neural network, could be streamlined. Additionally, it strengthened my understanding of the backend functionality and how it tied into the overall system.

- **Final deliverables**

From Thursday to Friday, I dedicated my time to working on the project deliverable—the final report. I documented the project's progress, including market research findings, team roles, technical decisions, and challenges encountered. I also updated the Trello board to ensure all tasks, deadlines, and milestones were accurately reflected. This involved creating cards for completed tasks, adding detailed descriptions to ongoing ones, and marking items that required follow-up discussions. The Trello updates ensured that the team had a clear overview of the project's current state and next steps.

- **Thoughts**

Throughout this project, I thoroughly enjoyed my role as the project manager, overseeing the course of the project and ensuring that all tasks were aligned with our objectives. From setting up the Trello board to conducting code reviews, researching key areas, and documenting our progress, this role allowed me to take responsibility for the overall direction of the project while fostering collaboration within the team. It was gratifying to see how our combined efforts came together to create a cohesive and well-organized workflow. However, the role was not without its challenges. Balancing technical contributions with organizational responsibilities required careful time management and prioritization. At times, I felt the pressure of coordinating multiple moving parts while diving into technical areas like frontend development and Lichess integration, which were outside my initial responsibilities. Despite these challenges, I found immense satisfaction in problem-solving and learning new skills along the way.This experience reinforced my appreciation for the importance of clear communication, adaptability, and teamwork in ensuring a project's success. It also strengthened my confidence in leading projects and managing tasks effectively, preparing me for future challenges in similar roles. Overall, being a project manager allowed me to grow both as a leader and a contributor, making this journey a truly rewarding experience.

For the creation of the running app and engine, things got more complicated than I thought. After doing some research on different approaches, I started discovering the Pychess library on Python, which used a Stockfish wrap and had an already integrated GUI, but it quickly frustrated me that I didn't understand anything. The goal was not to just create something that works but that we don't understand, since this would not allow us to achieve our final goal, which is developing the AI-chess-coach. I thus decided to learn everything from scratch, fully in Python, to understand deeper.

Approach 1 -> Create chess engine and GUI from scratch using Python and Pygame.

Thanks to this playlist made by Eddie Sharick, I was able to learn a lot, coded along, and created a fully playable (against myself) chess game in Python. (code)

All rules are set (except pawn promotion, which for GUI simplicity purposes is set to queen automatically).
The problem is that there is 0 engine here, and thus no evaluation, no analysis mode. Neither less, this helped me understand better how the rules work programmatically.

Then, I continued doing some research on how to create an engine from scratch, to learn how Stockfish was created, an created a very basic (stupid) rule-based engine based on the chess-pieces values (hard-coded), using an optimised Min-Max-algorithm. (code)

The bot is very easy to beat, it doesn't consider position, double pawns, king's safety, and much other important factors that should be considered. Also, for computational purposes, the depth isn't very deep here.

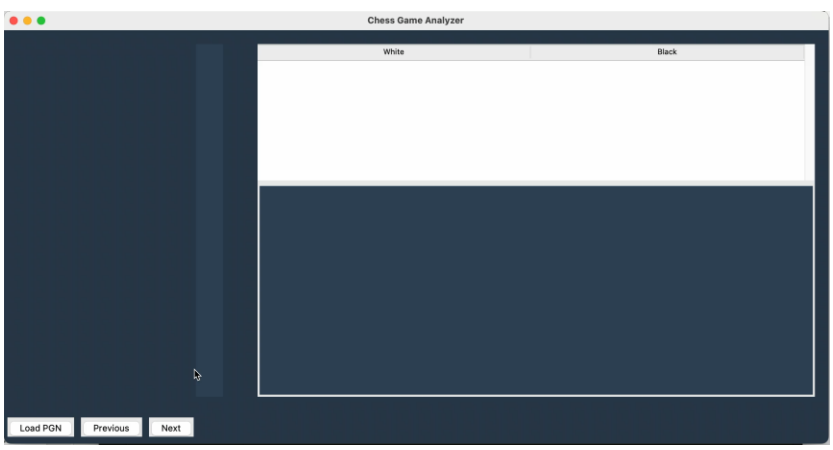
The code of the final version of this approach unfortunately got lost, I can't find it anymore, but I implemented positional scores, which helped a lot. I would say our bot was at 1000 Elo. Now, what was bothering me was that we would need more computer power to continue getting closer to Stockfish performance, also more time. So, since Stockfish is opensource and I now understood how it worked, I wanted to implement it into my game logic, but that's where things got very complicated. Stockfish follows severe rules that must be respected in how your chess game works. It would take too much time to change everything, and I didn't want to slow down the team more, so I decided to finally use the Stockfish wrapper in Python and try to create our analysis tool. The goal is that you can import a game and then iterate trough it with evaluations and best moves. You should also be able to just play against yourself with evaluations and best moves attached, ideally export FEN.

Approach 2 -> Create the analysis tool using Stockfish engine Python wrap.

The code here is a bit messy, but the goal is that there are 4 different modes. Human-vs-human, Human-vs-Bot, Bot-vs-Bot (to generate training data perhaps) and Analyse-game. Here is an example of Human-vs-Bot:

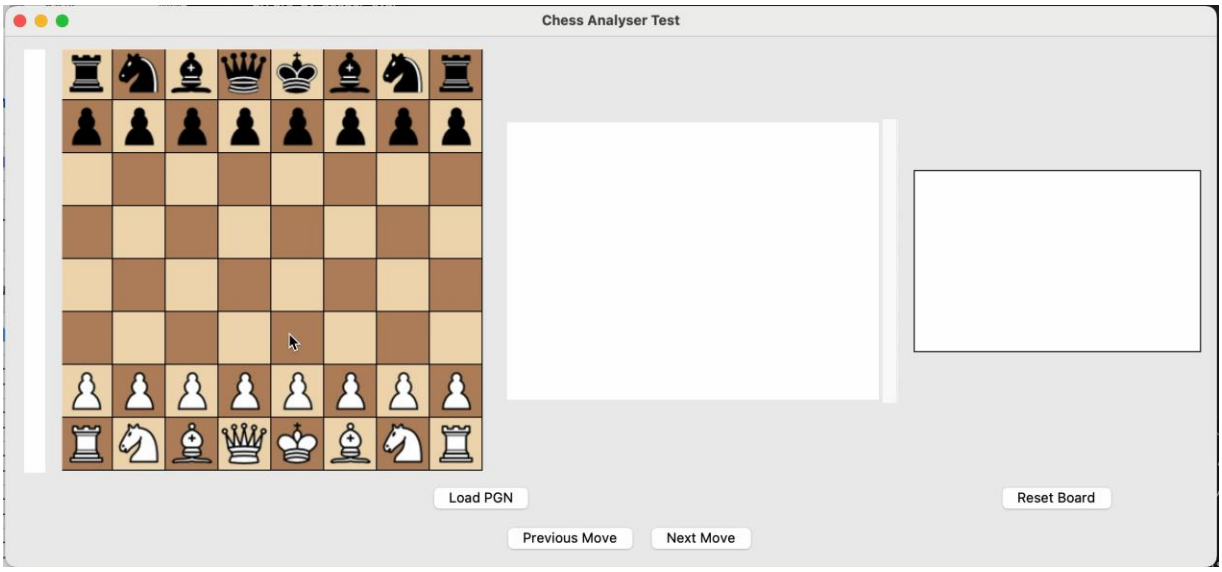
Stockfish is way better, but still can be improved, here we see the evaluation bar also, but not best moves

Analyse-game:



Eve, though the evaluation bar is flipping, the idea here is that we can import games and analyse

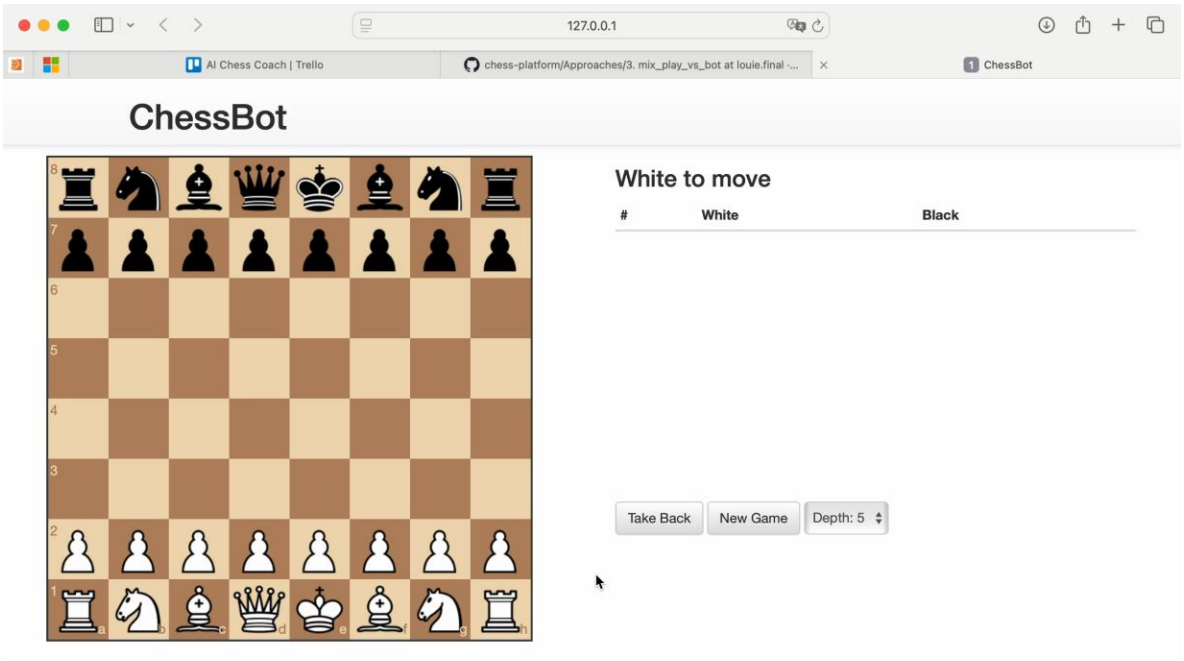
I then further changed things a bit, tried to clean it up, but this screwed up a lot of things. The important thing here is that I managed to give the best moves. (code)



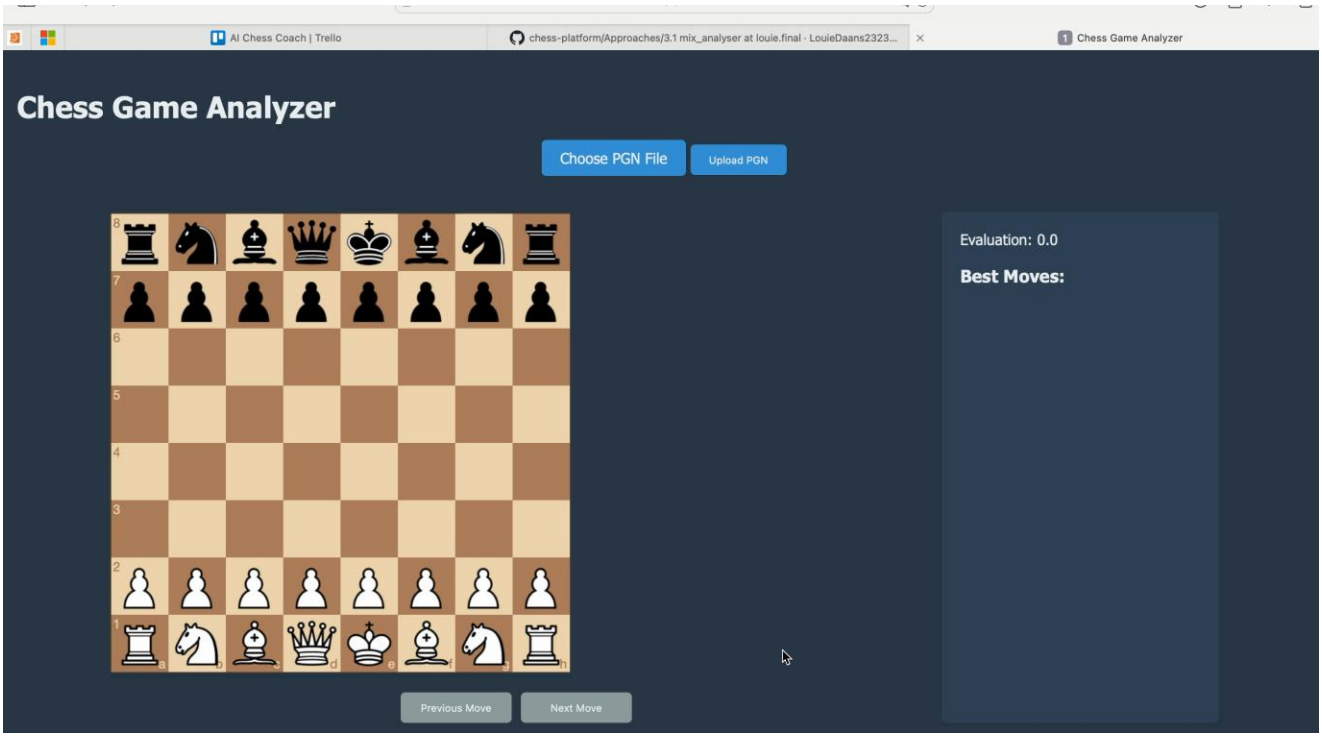
It was still a mess. I wanted to gain more freedom on the frontend, and this was complicated using only Python, I thus decided to try a new approach, using Python, JavaScript, HTML and CSS.

Approach 3 -> Using other programming languages, making it a web app.

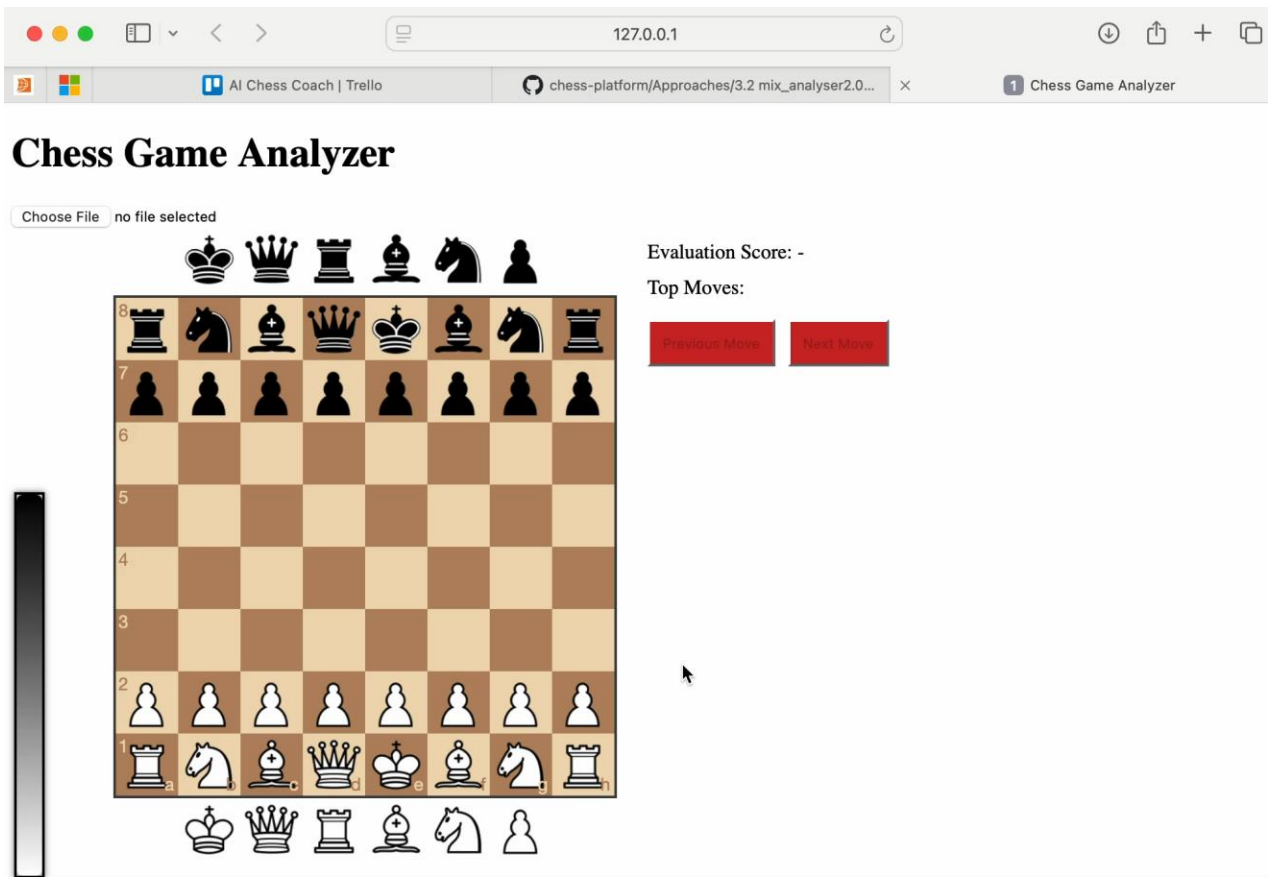
Here again, to get familiar with JavaScript and frontend, which I had to learn, it was important to first create something playable, where I could play against Stockfish. (code)



Then, I managed to create a chess analyser that was cleaner than before and more importantly, on web.
(code)



After long hours of reworking the thing, I managed to have a version that was ok, but the evaluation bar was still impossible to implement, I abandoned it. (code)



Here we had a long time of reflection, I lost a lot of time trying things out, learning programming languages, and we didn't finish enough time to finish it up. There were two options, finish fast but without making it easier to proceed in our project, so finishing without our final goal in mind, or not being able to finish in time, but learn instead so that we can achieve our main goal in a smarter and more flexible way.
We did research again, and we found out that all our competitors use C++ and JavaScript, and that those two are mainly used for Chess, not Python, so we would have to master those two languages to get to our final goal.

I tried some last approach, going deeper into JavaScript. But it didn't work as expected (code)

We learned a lot from all of this. Personally, these are the most important things I learned from this project:

- Attacking a project from scratch is harder than we imagine. All the little details, all the research, the lack of a 'mentor' who generally knows how to approach projects, makes it hard.

- Teamwork. Me trying to fix problems on my side, even though checking in with the team, makes it difficult for us all to unite. I learned that if we lack project management, if we lack a clear path where we know what to do, we should not have roles, but instead all work on the same thing together.

- I learned a lot about data formats, know exactly what programming is behind chess.

- My code was not clean and structured. Realising this, and how much it affects understanding your work, taught me to be cleaner next time, also for my team.

- I learned JavaScript well, also some basic HTML and CSS

Role 3: Data Engineer/Scientist (Data Collection, Model Creation)

As the data engineer I was responsible for the gathering, processing and cleaning of the data along with the research and creation of multiple neural networks to create an AI model that is capable of accomplishing the goals that were set for this project. To achieve this, I carried out some key tasks that were already discussed at the beginning of the project (Planning) each contributing to the success of the project.

Task 1: Research on the best approach for neural networks

(Basic Research findings)

The first step I took on the first day of the project was to start researching all the topics that are related to the problem at hand. In the beginning I researched mainly the best kind approach for a neural network, I found many different approaches that have potential use.
The first approach I studied was using **CNN's,** which is a good option for evaluating chess positions, but it is not great for teaching, which is the main topic for the project.

The second approach was by using **Transformer Models**; These type of models are great for multiple move planning given their ability to predict multiple moves ahead, this is great for a technical game like chess where this quality could be used to learn how to make piece sacrifices in the game or make a move that serves a purpose later in the game. Additionally, Transformer models can be used in NLP (Natural language processing) which can be helpful in generating tips and instructions for the players to level up their chess skills and knowledge.

The third approach was using **Deep Q-Learning**, which is good for decision making through trial and error, the model learns by playing against itself and to use the data from the game to train itself again.

Finally, after further research and a discussion with the other team members we have reached a decision of using an **Ensemble model**, which is a model that holds 2 models combined, our choice of models was using a **Transformer model** (for their ability to predict multiple moves ahead and for the ability to generate text to the user) and a **CNN** model for board evaluation (understanding static positions) .

Task 2: Data gathering and cleaning

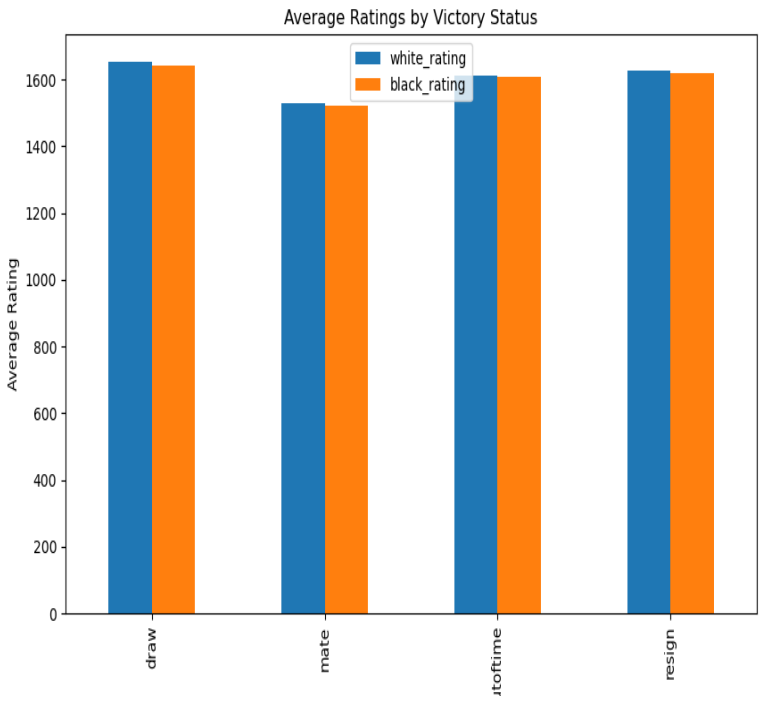
To ensure data accuracy, consistency and readiness for the Exploratory Data Analysis, all the data in every dataset was collected to be cleaned and preprocessed. This involved removing missing values and duplicates, validating FEN strings format and consistency across datasets, standardizing the formats as much as possible for easier merging between datasets.
Each dataset was extracted from different chess platforms (e.g., Lichess, Chess.com) to ensure diversity , therefore each dataset will serve a different purpose such as training models on openings or evaluating gameplay . The table below provides an overview of the datasets including their sources, a small description and the number of rows available.

Dataset	Source	Description	Number of rows
Chess Game Dataset (Lichess)	link	A dataset of full chess games collected using the Lichess API from a selection of users on the site Lichess.org.	20,000
High-Elo Games Chess Opening Dataset	link	This small dataset of high Elo players a great dataset to focus the training of the model on openings, and try the approach of training the model on the different stages of the game.	1750
60,000+ Chess Game Dataset	link	The dataset is sourced from Chess.com API, it provides detailed information about chess games, including board states in PGN and FEN format.	60,000

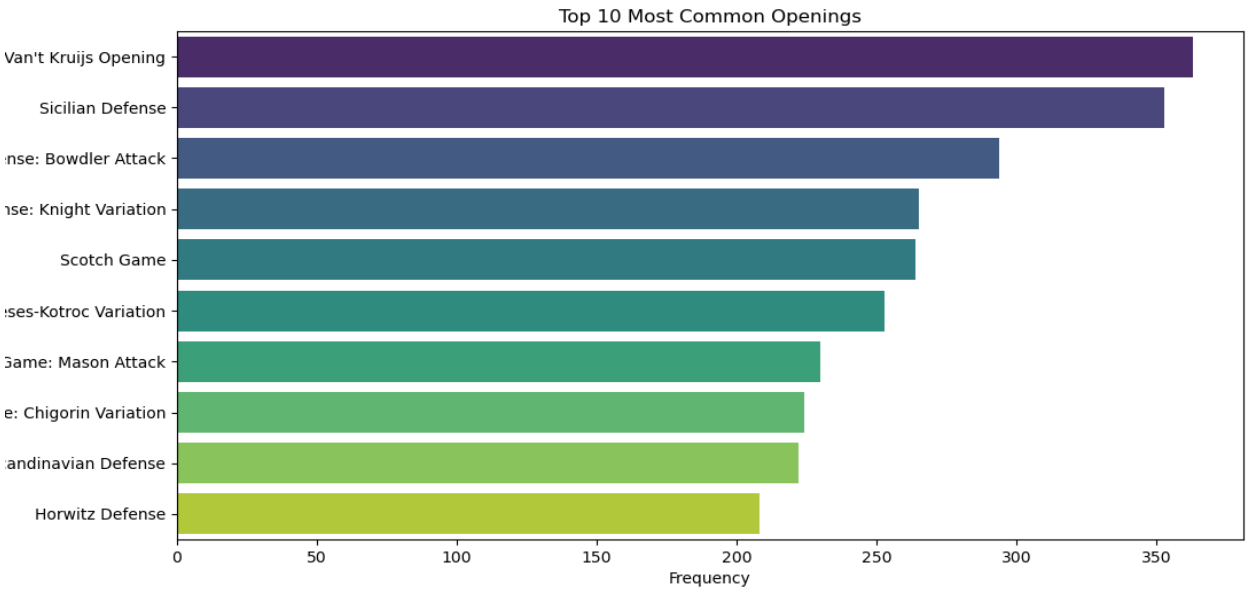
•Exploratory data analysis

In the exploratory data analysis (EDA) of the chess dataset, I examined several important aspects, such as the distribution of victory status across different ELO rating categories, the comparison of winning rates between white and black players, and the frequency of various openings. These visualizations are made for the porpuse of having a deeper understanding of the data and trying to find the patterns and correlations that are present in it.

By creating a grouped bar chart, I illustrated the frequency of each victory status across the average of player ratings for both white and black, indicating that white players had a slightly higher chance of winning. This is a well-known phenomenon in the chess community, as the player who makes the first move (white) enjoys a slight strategic advantage.



Following that, I investigated the most common chess openings in the dataset, with the Van't Kruijs Opening and the Sicilian Defense emerging as the top two. This information shows the strategies employed across various player levels.



• Challenges

- One of the first challenges that faced me was finding suitable and diverse data to clean and train the model on, but many datasets do not include all the moves but only include the other information about the chess games.
- During the data leaning some datasets were too big to even open on VSC, so I had to abandon those datasets and only keep the datasets mentioned earlier.
- The cleaning process had many issues from inconsistent formats of some columns throughout the dataset itself, and there were some missing and duplicated values.

• Next steps

• *Exploring more data:*

For the next steps we will need to have some endgame data and more data in general to increase the possibility of the best performance possible by the model.

• *Model training:*

After gathering and preprocessing the data, the models that were agreed on which are a **CNN**(Convolutional neural network) model and a **Transformer** model, these models were chosen for their great potential in solving the issue at hand, therefore they will be combined to create an **Ensemble** model.

• *Iterating and improving:*

After the training process is finished, the performance of the model would be monitored, and based on the performance of the model, the model should be iterated yb changing the hyperparameters and experimenting with different batch sizes, epochs, different optimizers, different loss functions and different model architectures. This will guarantee the best possible performance from the data that's available.