COMP7039 – Agile Processes

Interim Report

Project Title: Chess Analyzer

Team Members:
Daniel Sheehan | Cillian Houlihan | Scott Wolohan | Ciarán O'Brien

Lecturer / Product Owner: Dr Alex Vakaloudis

Date of Submission: 02/11/2025

Repository: https://github.com/KKV9/chess-analyzer

1. Introduction:

Chess Analyzer is a web based application that takes large collections of chess games and turns them into useful feedback for players. It works by focusing on analysing performance, spotting trends in play and giving feedback on how players approach different situations or openings. The goal is to help users understand their play more clearly through accurate data rather than guesswork, allowing them to hone their skills.

This project is being developed as part of our Agile Processes module using the SCRUM framework and agile methods. Our group worked in short sprints that divided up the tasks and focused on specific goals and used feedback from each stage to plan the next one while reviewing how the last one went. The work is divided into three sprints: Data input, Data Visualisation and Knowledge Extraction. This interim report will focus on the first two stages which cover data handling and visualisation.

Our project will use python and node.js for backend processing, chart.js will be used for visualisation and GitHub actions as well as our working website will be used for testing and deployment.

2. Project Objectives and Agile Methods:

The main objective of Chess Analyzer is to develop a working web platform that can import, process and visualise chess game data to provide actual meaningful insights for players. The goal is to demonstrate how following agile practices and methods helped our team to deliver software that evolved smoothly over time with a clear development path where we used feedback, communication and constant testing.

To achieve this, we broke the project down into three main sprints each of them with clear deliverables that aligned with our goals making sure that the project was done in organised phases and on time.

The sprints were as follows:

Sprint 1 (Data Input):	Develop a data input and validation system that checks and scrapes chess game data before it is used for analysis
Sprint 2 (Data Visualisation):	Create visual dashboards and charts using Chart.js that display player trends, win rates and game insights taken from data input.
Sprint 3 (Knowledge Extraction):	Implement AI-powered features using Google Generative AI to produce more advanced insights and player profile data.

The project follows the Scrum framework, with each sprint including planning, stand up meetings and calls, sprint reviews and retrospectives. This helped us as a team to stay organised when delegating work and tasks and let us spot any issues early in each sprint so we could reflect on our progress and what we thought we could have possibly done better. The backlog for our tasks was managed in Jira along with user stories, priorities and story points which allowed us to track the project over time.

We used the Agile Manifesto values [i] we covered in class as a guideline prioritising things like teamwork and adaptability and mainly focused on delivering a working product that improved and became more detailed with each sprint rather than spending too much time on documentation while stalling progress. By keeping progress visible in Jira and updating our backlog regularly we were able to react quickly when something needed to change or when we had to implement new features from the feedback we would give each other. Jira was mainly used to monitor our development progress, manage requirements and story points. As a team we also decided to use GitHub for version control while developing our software as it allowed us to collaborate easily and commit any changes to code with pull requests.

Working this way helped us make sure that the project stayed on track and within deadlines while keeping communication strong. The agile approach made it easier to plan realistic goals for each sprint and look back at everything we had accomplished at the end of each sprint before moving on to the next.

3. Team Roles and Communications Overview:

From the beginning of this project, our team had a strong focus on communication throughout the creation of the Chess Analyzer application. This was to ensure each team member was in alignment with the work needed to be done, our goals and objectives and the best agile practices. We based our work on the Scrum framework to keep everything organised while ensuring each team member contributed equally. To keep track of our progress, we used Jira to measure the *To Do / Doing / Done* section, this allowed us to keep our tasks simple while being able to visualise our progress as we move forward. Before starting each sprint, the team scheduled planning sessions where we discussed user stories, and which roles are to be assigned to each team member. During the development of the project, the team communicated daily through Discord, briefing each other on the progress made, problems that needed solving and updating our GitHub and Jira as needed. Our Scrum meetings in lab sessions gave us time to assess and review our completed work in person, update Jira and planned for the next step or sprint.

The role of Scrum Master rotated on each sprint, this gave each team member the experience of leading a sprint. Sprint 1 was led by Cillian Houlihan, the goal was to establish the project foundation by setting up the development platform, repository and creating a data import and validation process. We also used a script to reduce the size of the CSV file and get a random sampling of 200,000 games. This structure allowed us to continuously update and improve our work while aligning with best Agile practices taught in this module.

Sprint 2 was led by Daniel Sheehan, and our focus shifted to linking the back-end CSV data validation with the new front-end dashboard, improving the user interface for the Chess Analyzer application. Ciaran O'Brien, who acted as the DevOps lead looked after the deployment setup and interface to keep everything consistent across the project. We began testing local deployments using node.js and PM2, this allowed the team to keep the app live while reviewing new features. Daniel worked closely with Scott and Cillian to connect the data validation results with the newly created User Interface layer. We also conducted background research of chess analysis using Stockfish and brainstormed the creation of a simple chess game to implement for demos. Team communication improved a lot during Sprint 2, we kept in touch through Discord and Jira, which made it easier to keep updated with the workload progress and sort issues out quickly while staying on the same page between the backend and frontend development. The team also had standup meetings to discuss progress in person and brainstorm ideas, making sure everyone stayed on the same page and any blockers were dealt with quickly.

Sprint 3 will be led by Scott Wolohan. It will built on the progress made in previous sprints and will be discussed in more detail in the final report once completed.

4. Product Backlog Overview:

At the beginning of this project, our team put together a product backlog to keep track of everything that has to be completed for our Chess Analyzer App. This backlog is the main list of goals that include features and tasks needed that were discussed during the early planning phases of the project. Each of these tasks were written as a user story in the usual agile format taught to us, this format is "*As a [role], I want to [goal], so that [benefit]*." This format helps us describe what is needed on a basic level that makes sense from a developer and user point of view without overcomplicating anything.

The Product Backlog covered a range of areas like data input and validation, and creating the user interface for visualising the data, and setting up the DevOps side of things. Jira was the tool used to manage all of this, giving each story a point value and assigning to the right person based on their strengths and understanding. Once the stories were added, we labelled them by priority and decided which were most important to start with. This helped plan out the sprints and balance the work across the team.

During the planning of each sprint, we took stories from the highest priority and moved them into the active sprint. Any larger or more advanced goals were left in the backlog for later sprints. The backlog was constantly reviewed and improved, adding new stories as the project developed, existing tasks were broken down into small cycles when needed. This helped us plan, build and improve the project over time.

Each story is written from a different point of view, some of these tasks were "As a developer", "As a Scrum Master" "As a team". These different perspectives give both a technical and organisational side to the project ensuring every area was included in the backlog. This gave a clear structure and kept development balanced throughout each sprint.

In summary, the product backlog became a central part of how we managed and kept up to date with the progress of the project. It helped us stay organised, gave us a clear direction of what to focus on next, helped us communicate better as a team and ensured the project evolved in a structured and manageable way.

5. Sprint 1 Report – Data Input:

Scrum Master: Cillian Houlihan

Sprint Duration: 22/09/2025 – 29/09/2025

Sprint Goal: The goal of Sprint 1 was to develop a data input system capable of importing chess game data through a web form, validating it and preparing it for future analysis and visualisation.

Documentation:

Sprint 1 focused on setting up the foundations for our Chess Analyzer project. The main focus was to create a working data input and validation system that was able to import chess game datasets while also double checking the accuracy and if the format was correct. As a team we first used a Python script to lower the sample size from 200,000 games to 5,000 games to allow for a more manageable workload and pool of games, The second python script was responsible for validating and importing the data from each game.

Work was managed through Jira as our project management tool with each backlog item given story points and assigned fairly to a team member. We communicated through Discord calls , weekly stand up meetings and Scrum meetings in lab sessions in order to make sure that we as a team were collaborating and communicating constantly during the sprint.

Sprint Backlog:

ID	User Story	Story Points	Assigned Member	Acceptance Criteria
CA-1	As a developer, I need to decide how many games from the CSV file are needed to complete and display the visualisation of the data.	2	Cillian Houlihan	Research the optimal amount of games to provide accurate sampling for data display.
CA-2	As a developer, I want the system to validate imported data (e.g., check missing values, invalid moves) so that I can trust the dataset.	4	Scott Wolohan	Create a script that validates imported data and identifies empty or invalid fields.
CA-3	As a developer, I want parsed data stored in a structured format so that later visualisation and analysis are possible.	3	Ciarán O'Brien	Finish research and decide on the optimal way to parse data using the web form.
CA-4	As a developer, I need to use a script to reduce the size of the CSV file and get a	4	Cillian Houlihan	Complete the Python script that successfully generates a reduced dataset.

	random sampling of the 200,000 games.			
CA-5	As a developer, I want GitHub repo setup so as to exchange and verify team code.	1	Daniel Sheehan	Complete repository setup and ensure all team members have access.
CA-6	As a group, decide the optimal way to parse the data (Python/SQL/Excel).	2	Daniel Sheehan	Outline the pros and cons of each method and record the team's final decision.
CA-7	As Scrum Master, I want a record kept of meetings held and to make sure workload is balanced.	2	Cillian Houlihan	Keep accurate records of meetings and workload distribution
CA-8	As Scrum Master, I will write up the Sprint 1 report with help from team members.	4	Cillian Houlihan	Complete full Sprint 1 report as outlined.
CA-9	As a developer, I want to do background research on how to complete the sprint goal by looking at similar projects.	3	Scott Wolohan	Complete background research using GitHub and W3Schools to inform sprint tasks.

Sprint Tracking: Ideal vs Actual Work done

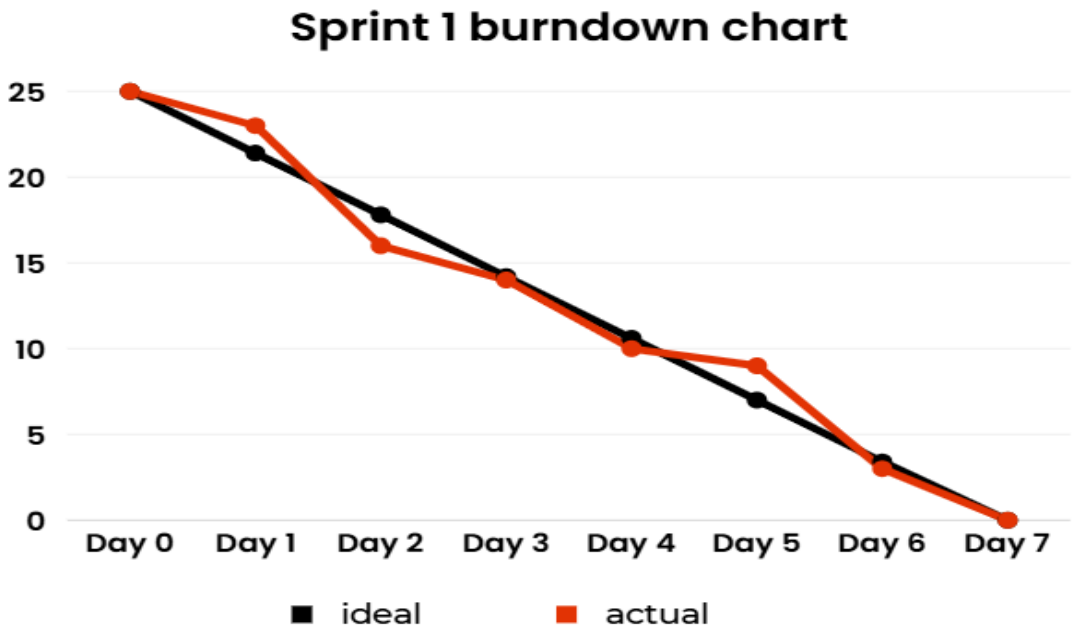
The table below compares the team's actual progress against the ideal sprint plan. We managed to complete early tasks like setup and research ahead of schedule however scripting and validation took much longer which slowed the sprint down a bit. Our progress picked up again as the team moved on to finalising and testing the sampling and validation scripts.

Overall as a team we managed to keep a good pace throughout the sprint and managed to complete all 25 story points on time. We stayed on track without any major issues and roadblocks showing that we planned everything out effectively and worked well at a consistent level.

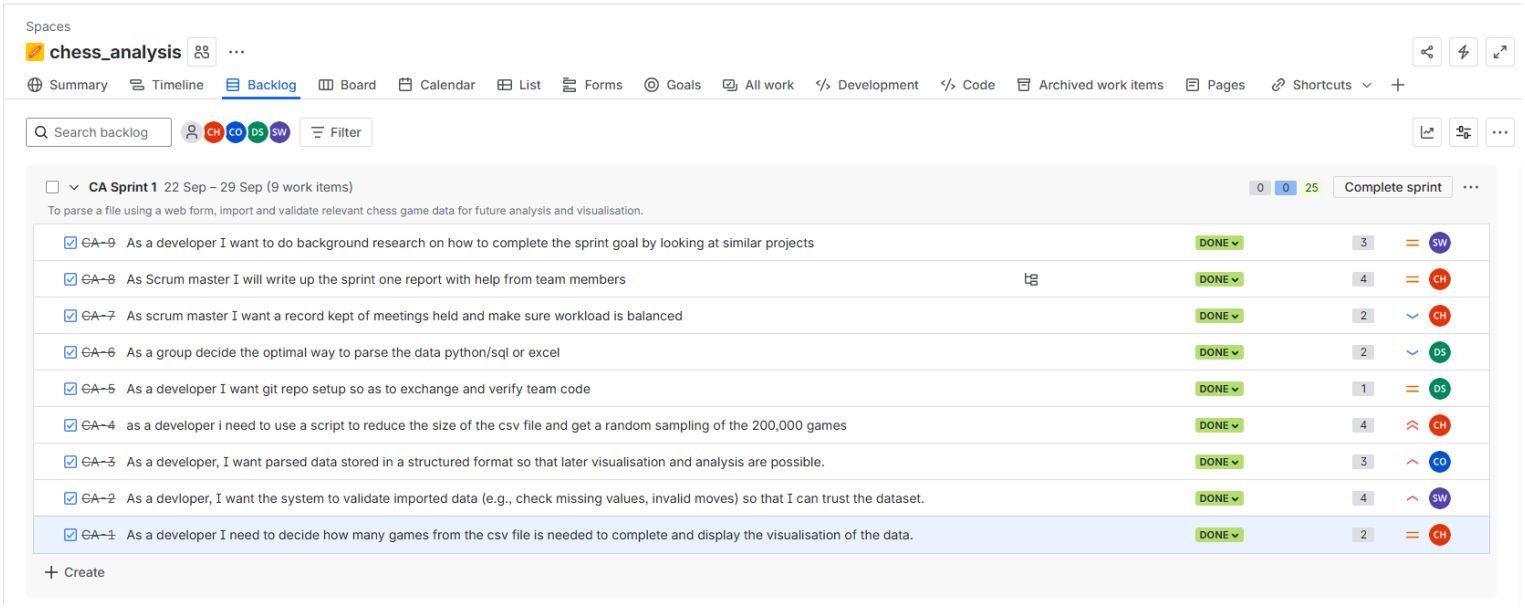
SCRUM	Story points (hrs)	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
CA-1	2	1				1		
CA-2	4						4	
CA-3	3							3
CA-4	4				4			
CA-5	1	1						
CA-6	2			2				
CA-7	2		2					
CA-8	4		2				2	
CA-9	3		3					
Total	25	2	7	2	4	1	6	3
	Actual	23	16	14	10	9	3	0
	Ideal	21.4	17.8	14.2	10.6	7	3.4	0

Sprint 1 Burndown Chart:

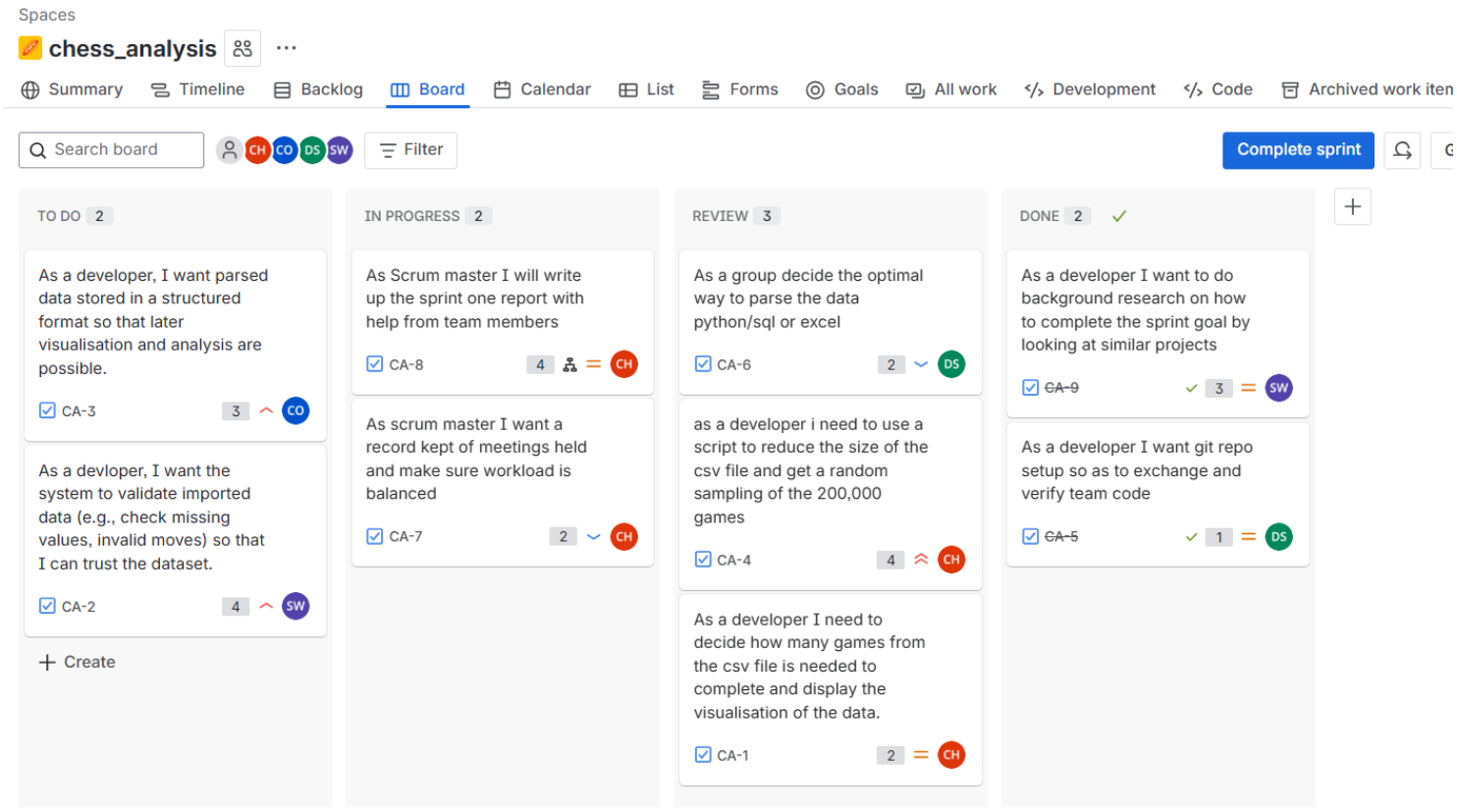
The burndown chart shows steady progress across the 7 day period of sprint 1, we managed to stay close to the ideal line other than the minor delays we encountered mid sprint however we recovered quickly with the team finishing all tasks by the 7th day deadline. This showed us that we coordinated well and managed tasks calmly as a team.



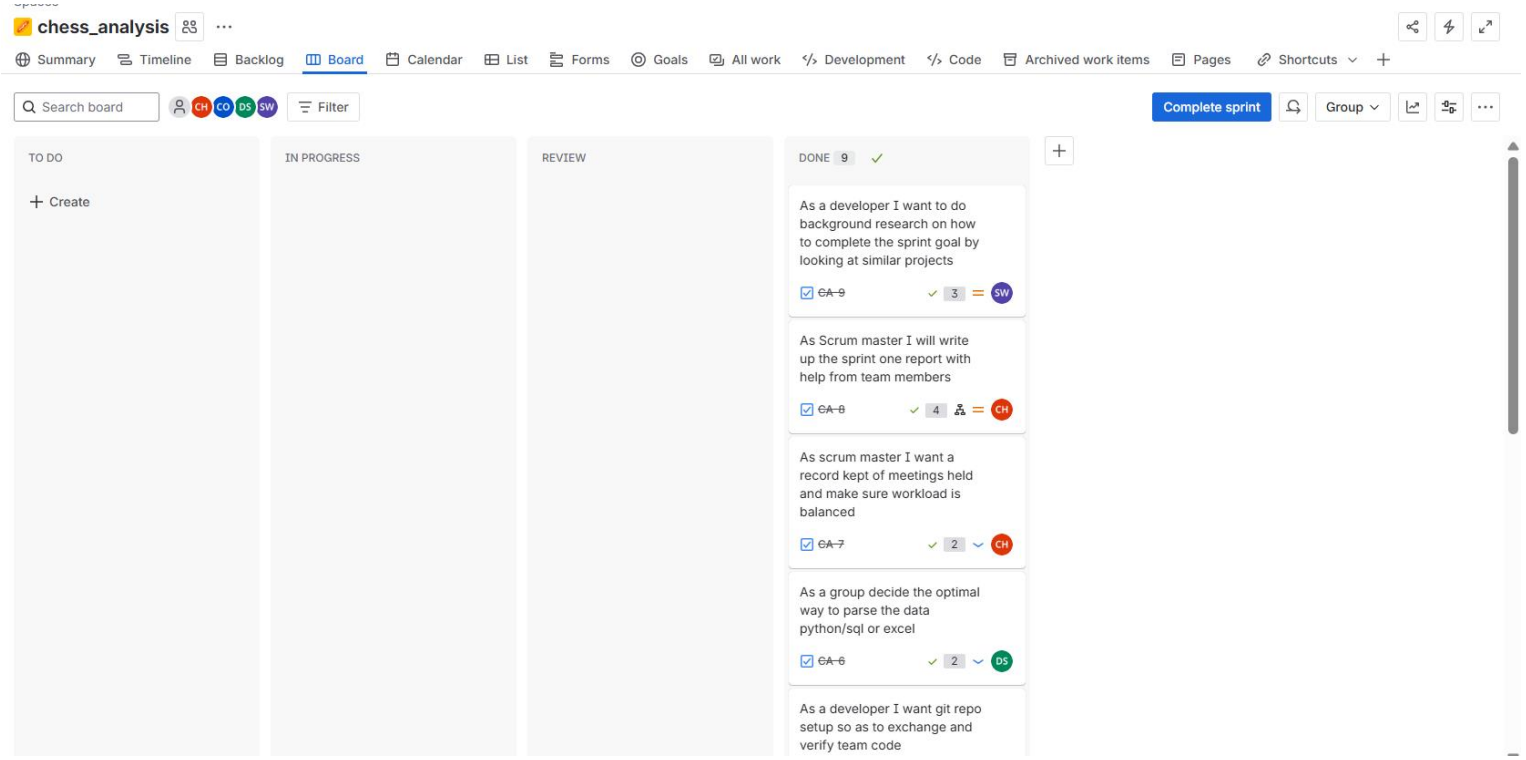
Kanban Board Overview:



Kanban Board Mid Sprint:



Kanban Board at End of Sprint 1:



Sprint 1 Retrospective:

Positives:	• Completed 100% of Sprint tasks on time • Strong communication on Discord and met frequently in person to iron out issues • GitHub Repository and validation system were fully functional by the end of sprint
Negatives:	• We were unfamiliar with extracting data from the CSV file due to the large volume of games (200,000), it would not open in Excel as a result of this • At beginning of Sprint 1 the group only consisted of 2 members • Some user stories were added late in Jira making progress tracking confusing at the start
Solutions and Improvements:	• In our initial prototype we improved CSV formatting by lowering the sample size from 200,000 chess games to 5,000 games with a python script to avoid future issues when importing data. • After several debates we decided to switch to user based uploads where the user could upload their own chess data provided the data did not exceed 2mbs for flexibility.

Velocity Achieved:

As a team we completed all 25 story points that were planned for sprint 1. We worked consistently across the week, and everyone managed to handle their project roles well. By the end of the sprint all stories were finished on time without any tasks carried over to Sprint 2.

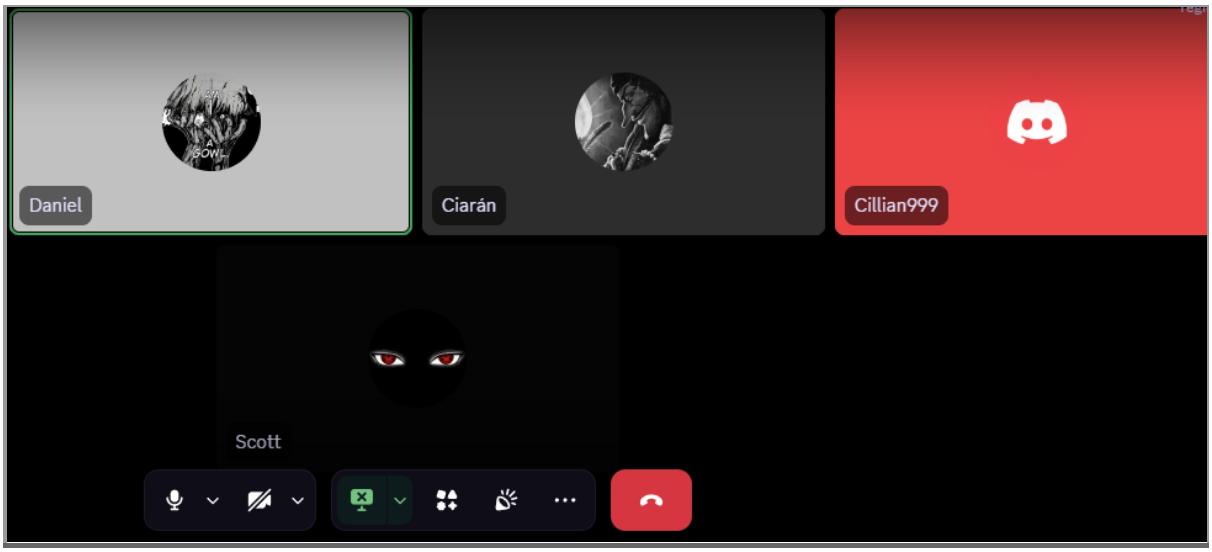
25/25 velocity.

Sprint Summary:

Sprint 1 focused on setting up the main structure of our project and making sure that we had a working data import and validation process. We ran into issues early on with the large dataset and small team size, but these were solved quickly once all members were on board. Reducing the CSV sample size improved the validation process and sped up our overall development and allowed us to stay on track. By the end of the sprint , the system for uploading and validating chess data was complete and ready to use for Sprint 2.

Evidence of Communication:

Regular Discord Calls whilst working together



We also met in weekly labs to discuss progress and held stand up meetings to discuss and monitor progress.

Sprint 1 Deliverables:

https://github.com/KKV9/chess-analyzer/blob/master/scripts/validate.py

Validate.py:

Once file is uploaded clicking the validate button runs the script which validates data and if headings (black and white) are correct, if it is a .csv file and does not exceed file size limit.

```python
#!/usr/bin/env python3
"""
Simple Validation Script for Chess Analyzer App
-----------------------------------------------
Checks that data/data.csv exists, can be read, and contains required columns.
"""

import os
import csv

FILE_PATH = "data/data.csv"
REQUIRED_COLUMNS = ["Black", "White"]  # Required column names

def main():
    # 1. Check if file exists
    if not os.path.exists(FILE_PATH):
        print(f"❌ File not found: {FILE_PATH}")
        print("Please upload a CSV file using the upload form.")
        return

    print(f"✅ Found file: {FILE_PATH}")

    # 2. Try reading first few lines
    try:
        with open(FILE_PATH, newline='', encoding='utf-8') as f:
            reader = csv.reader(f)
            header = next(reader, None)
            first_row = next(reader, None)

            if not header:
                print("❌ CSV file is empty or missing a header row.")
                return
            else:
                print(f"✅ Header columns: {header}")

            # 3. Check for required columns
            missing_columns = []
            for col in REQUIRED_COLUMNS:
                if col not in header:
                    missing_columns.append(col)

            if missing_columns:
                print(f"❌ Missing required columns: {', '.join(missing_columns)}")
                print(f"   Required columns are: {', '.join(REQUIRED_COLUMNS)}")
                return
            else:
                print(f"✅ All required columns present: {', '.join(REQUIRED_COLUMNS)}")

            if not first_row:
                print("⚠️  CSV file has no data rows.")
            else:
                print(f"✅ First data row: {first_row}")

                # Count total rows
                row_count = 1
                for _ in reader:
                    row_count += 1
                print(f"✅ Total data rows: {row_count}")
```

Server.js and Multer:

Node package which allows user to upload their game file.

```
// server.js
require('dotenv').config(); // Load environment variables from .env file

const express = require("express");
const { exec } = require("child_process");
const path = require("path");
const multer = require("multer");
const fs = require("fs");

const app = express();
const PORT = process.env.PORT || 3000;

// Configure multer for file uploads
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        const uploadDir = path.join(__dirname, "data");
        // Create data directory if it doesn't exist
        if (!fs.existsSync(uploadDir)) {
            fs.mkdirSync(uploadDir, { recursive: true });
        }
        cb(null, uploadDir);
    },
    filename: (req, file, cb) => {
        // Save as data.csv, replacing any existing file
        cb(null, "data.csv");
    }
});

const upload = multer({
    storage: storage,
    limits: {
        fileSize: 2 * 1024 * 1024 // 2MB limit
    },
    fileFilter: (req, file, cb) => {
        // Only accept CSV files
        if (file.mimetype === 'text/csv' || file.originalname.endsWith('.csv')) {
            cb(null, true);
        } else {
            cb(new Error('Only CSV files are allowed'));
        }
    }
});

// Middleware to parse JSON bodies
app.use(express.json());

// Serve static files from 'public' folder
app.use(express.static("public"));

// Endpoint to upload CSV file
app.post("/upload-csv", upload.single('csvFile'), (req, res) => {
    if (!req.file) {
```

6. Sprint 2 Report – Data Visualisation:

Scrum Master: Daniel Sheehan

Sprint Duration: 06/10/2025 – 20/10/2025 (3 weeks duration)

Sprint Goal: The goal of Sprint 2 was to create a working front end dashboard that could display the validated chess data in a visual format like a graph. We wanted users to see player trends and win rates by colour in an easy to digest way.

Documentation:

Sprint 2 focused on connecting the validated data from sprint 1 to a front end dashboard that was visible to users. Using HTML, CSS and Chart.js the team created visualisations that showed win rate trends and performance insights. We designed the interface to be clean and responsive, and GitHub Actions was configured to automatically deploy updates to the AWS EC2 server. Weekly lab meetings and Discord calls let us discuss our progress and how we were going to carry this out.

Sprint Backlog:

ID	User Story	Story Points	Assigned Member	Acceptance Criteria
CA-12	As Scrum master, I want to maintain and create a successful log of project progression.	2	Daniel Sheehan	Filled and complete log
CA-14	As scrum master, assigned roles for this sprint to create a balanced workload.	1	Daniel Sheehan	Assign every team member has a dedicated role.
CA-15	As a developer, I will research the background of chess analysis such as Stockfish.	2	Scott Wolohan	Conduct sufficient background information to gain insight into how stockfish works and how we could use it to develop our project.
CA-16	As a user, I want to see an animated chess game demo.	3	Ciaran O'Brien	Complete a chess demo
CA-17	As a team, decide on what visualisations to display.	2	Cillian Houlihan	Confirm the visualisation method e.g. chart etc

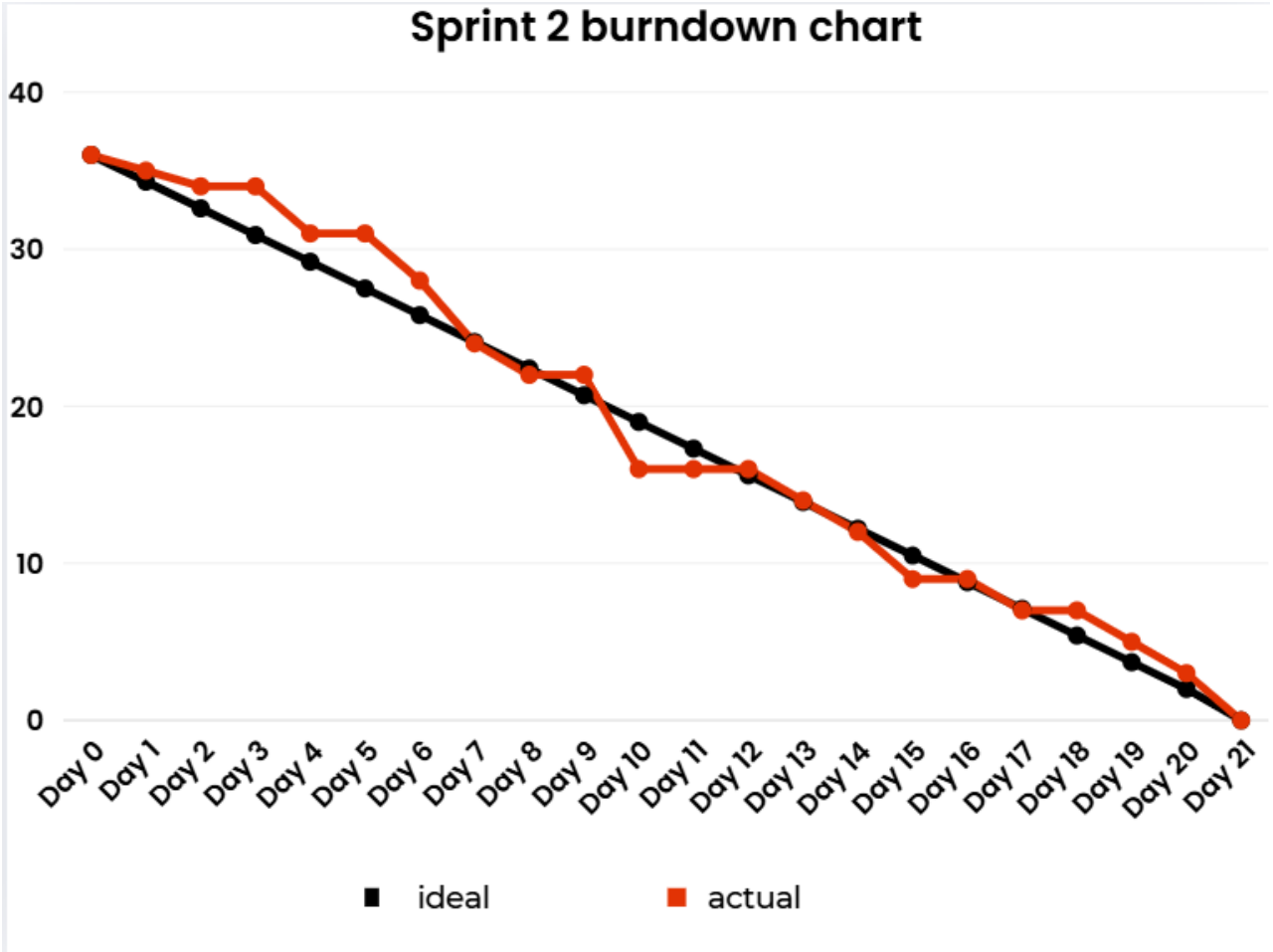
CA-18	As a user, I want an enhanced UI with animations using css, html and server.js	4	Ciaran O' Brien	Develop a easy to access and a nice looking UI.
CA-19	As a development team, we want to complete the interim report	4	Daniel Sheehan	sprint 1 and 2 complete. plus additional areas e.g. introduction
CA-20	As a team member finish the write up for sprint 1 including all charts and information required.	4	Cillian Houlihan	sprint 1 and 2 complete. plus additional areas e.g. introduction
CA-21	As a team member finish the write up for sprint 2 including all charts and information required.	4	Scott Wolohan	complete sprint 2 write up and burndown chart.
CA-22	As a team member, I want to combine the 2 sprint reports and finish additional areas for the interim report.	4	Scott Wolohan	combine all areas of the report.
CA-23	As a developer, create a python script to analyse win-rate distributions.	4	Ciaran O' Brien	develop a script to gather the correct data
CA-24	As a user, I want to be able to easily view the data in a visual format	2	Cillian Houlihan	be able to visually see the data from the csv file.

Sprint Tracking: Ideal vs Actual Work done:

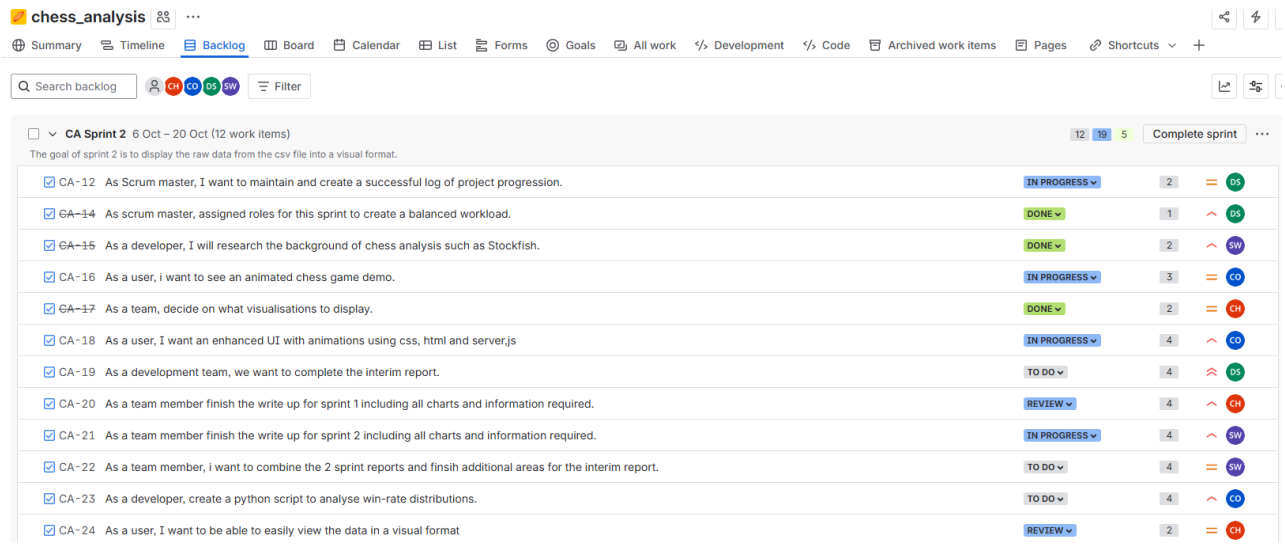
The table shows our steady progress across the 21 day sprint. Early tasks such as research, UI setup and report writing were completed quickly but the integration and chart visualisation work took slightly longer mid sprint. Once deployment testing started we were fully caught up and finished all 36 story points by the last day of the sprint. Overall we stayed consistent throughout the second sprint.

SCRUM	Story points (hrs)	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
CA-12	2		1																			1
CA-14	1	1																				
CA-15	2				1		1															
CA-16	3										2					1						
CA-17	2								2													
CA-18	4										2					2						
CA-19	4													2							2	
CA-20	4							4														
CA-21	4																	2		2		
CA-22	4														2							2
CA-23	4						2				2											
CA-24	2				2																	
totals	36	1	1	0	3	0	3	4	2	0	6	0	0	2	2	3	0	2	0	2	2	3
actual	36	35	34	34	31	31	28	24	22	22	16	16	16	14	12	9	9	7	7	5	3	0
ideal	36	34.3	32.6	30.9	29.2	27.5	25.8	24.1	22.4	20.7	19	17.3	15.6	13.9	12.2	10.5	8.8	7.1	5.4	3.7	2	0
36/21=1.7		-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7	-1.7

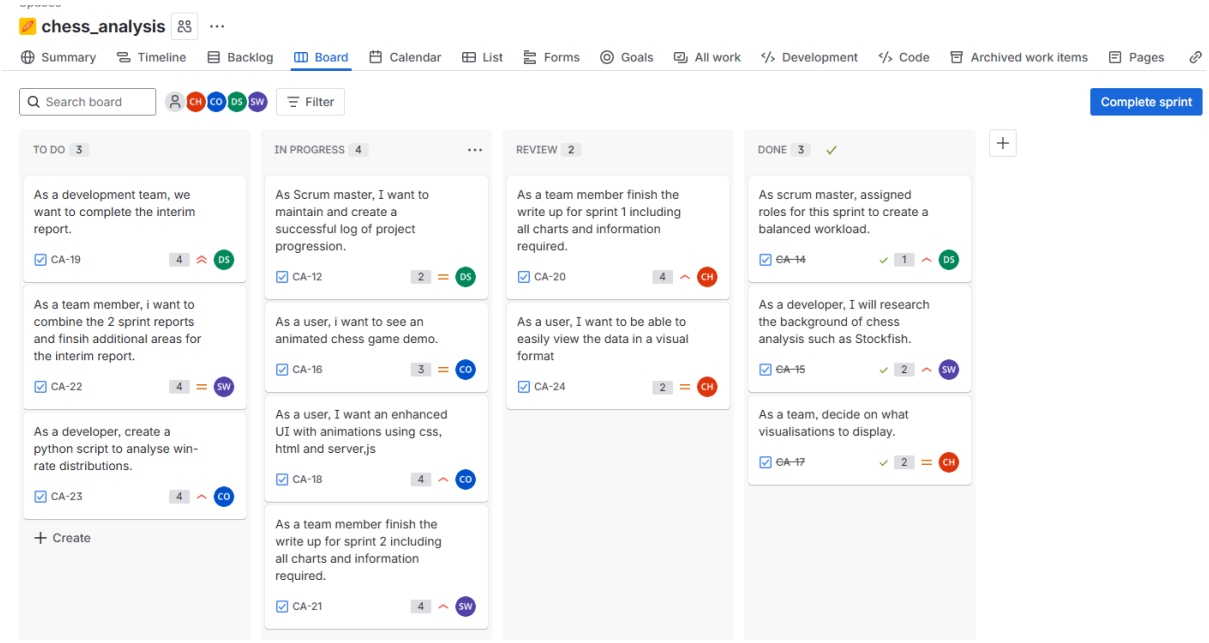
Sprint 2 Burndown Chart:



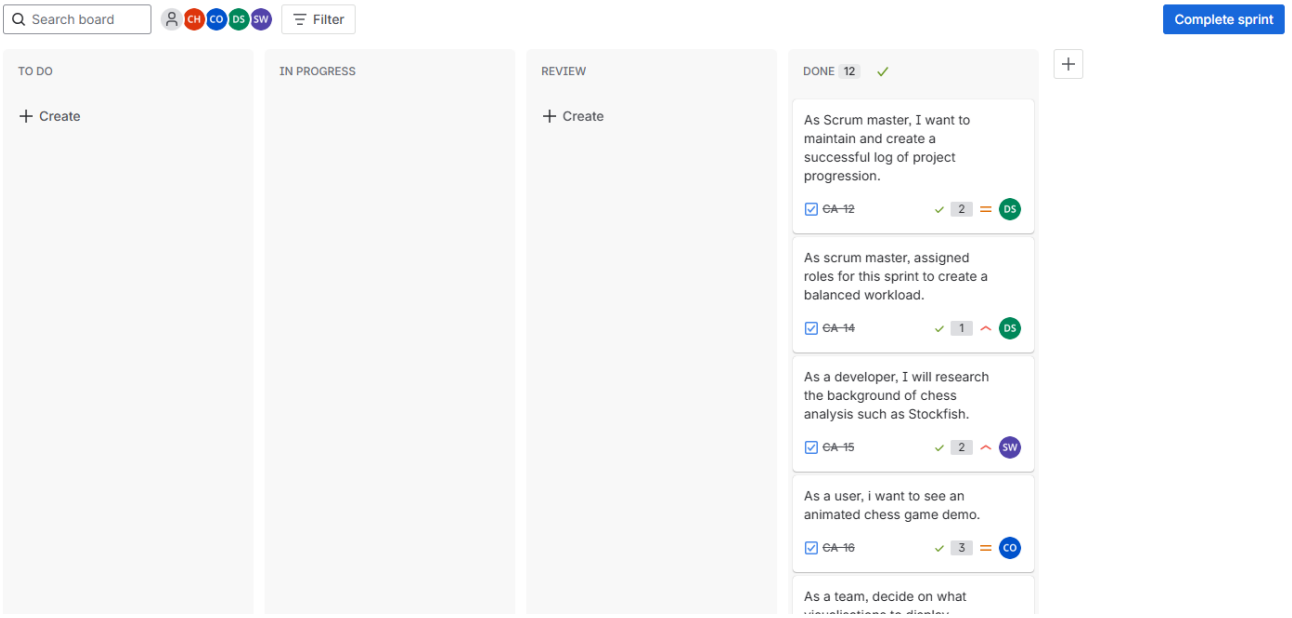
Kanban Board Overview:



Kanban Board Mid Sprint:



Kanban Board at End of Sprint 2:



Sprint 2 Retrospective:

Positives:	• All tasks from Sprint 2 were finished on time. • Chart.js dashboard worked smoothly and displayed the data correctly. • Team communication and roles were clear from the start.
Negatives:	• We learnt from the mistakes in Sprint 1 so things went smooth and no major issues were encountered. • Minor disagreements about what way to display the data (decided on a chart)
Solutions and Improvements:	• UI was smoother and more modern in design, Making it more attractive to users. • We cleaned up the CSS layout to make charts responsive and consistent.

Velocity Achieved:

As a team we completed all 36 story points, staying consistent throughout the sprint and keeping good momentum. Progress was steady from start to finish as we knew what each of us had to do.

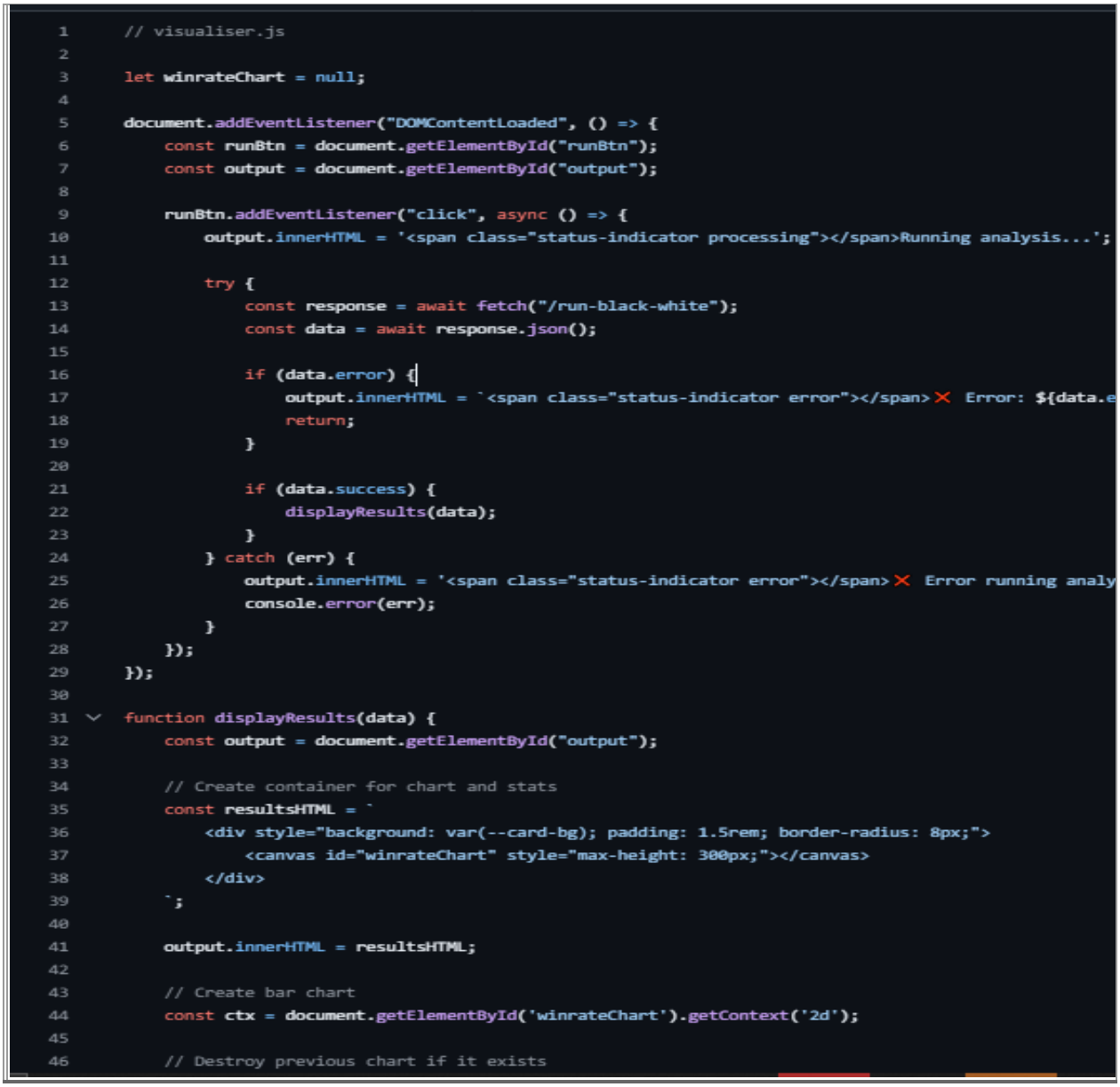
36/36 velocity.

Sprint Summary:

Sprint 2 was successful in linking the validation and upload process from sprint 1 to a front-end dashboard that can cleanly display the data through a graphic (chart). By the end of the sprint, we had refined the site's UI and had successfully shown data in a way that's neat and readable. We are now fully prepared to move onto sprint 3.

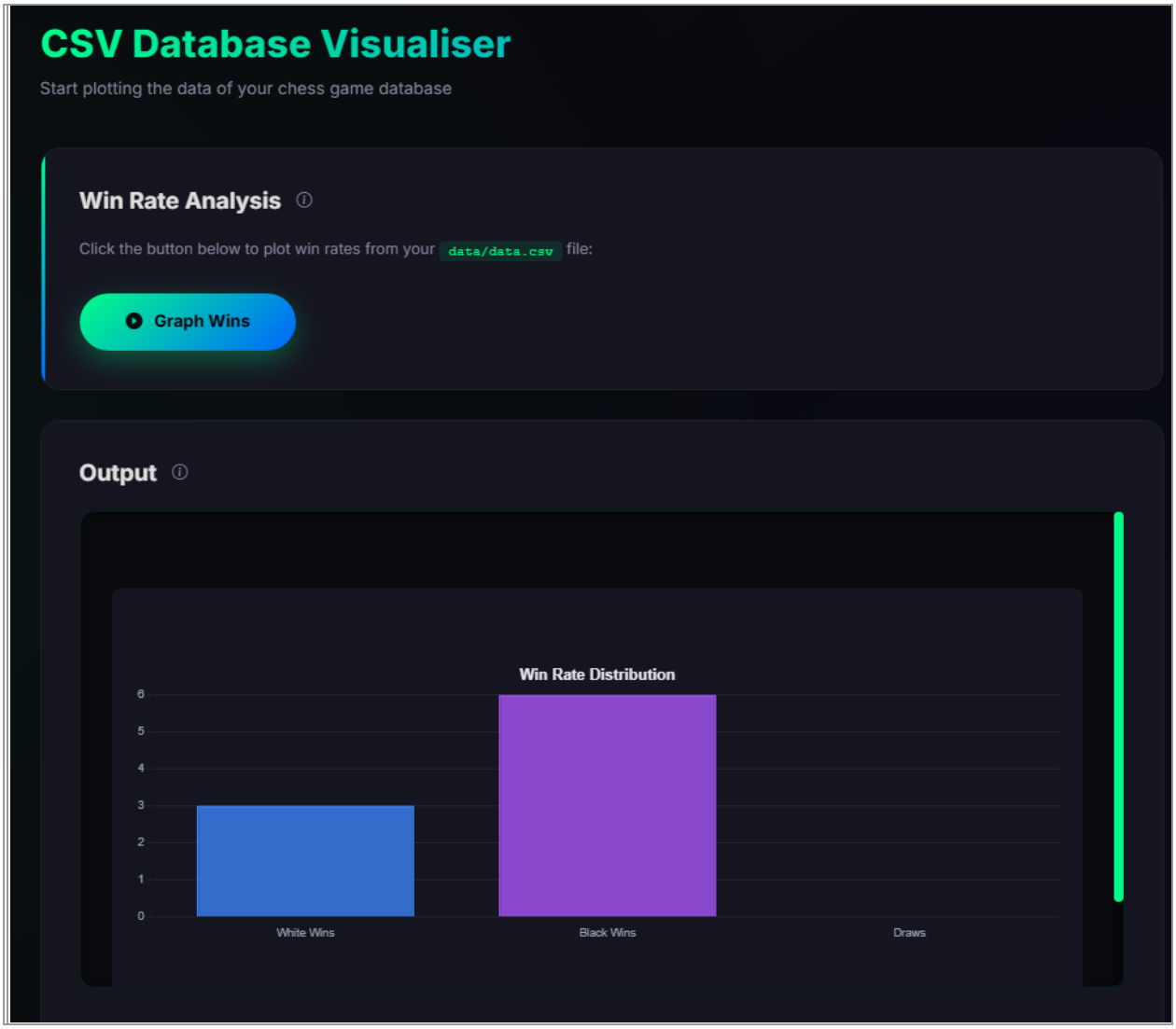
Sprint 2 Deliverables:

https://github.com/KKV9/chess-analyzer/blob/master/public/js/visualiser.js



```
// visualiser.js

let winrateChart = null;

document.addEventListener("DOMContentLoaded", () => {
    const runBtn = document.getElementById("runBtn");
    const output = document.getElementById("output");

    runBtn.addEventListener("click", async () => {
        output.innerHTML = '<span class="status-indicator processing"></span>Running analysis...';

        try {
            const response = await fetch("/run-black-white");
            const data = await response.json();

            if (data.error) {
                output.innerHTML = `<span class="status-indicator error"></span>❌ Error: ${data.e
                return;
            }

            if (data.success) {
                displayResults(data);
            }
        } catch (err) {
            output.innerHTML = '<span class="status-indicator error"></span>❌ Error running analy
            console.error(err);
        }
    });
});

function displayResults(data) {
    const output = document.getElementById("output");

    // Create container for chart and stats
    const resultsHTML = `
        <div style="background: var(--card-bg); padding: 1.5rem; border-radius: 8px;">
            <canvas id="winrateChart" style="max-height: 300px;"></canvas>
        </div>
    `;

    output.innerHTML = resultsHTML;

    // Create bar chart
    const ctx = document.getElementById('winrateChart').getContext('2d');

    // Destroy previous chart if it exists
```


CHESS
ANALYZER
Master Your Chess Game
Analyze, validate, and elevate your chess strategy with our powerful suite of AI-driven tools
Get Started ↓


CSV Database Visualiser
Start plotting the data of your chess game database
Win Rate Analysis
Click the button below to plot win rates from your data/data.csv file:
Graph Wins
Output
Win Rate Distribution
6
5
4
3
2
1
0
White Wins
Black Wins
Draws

7. DevOps and Integration:

Initial Repository Setup:	We started by creating a simple git repository and uploading it to GitHub. The initial commit (67e2e24, Oct 6, 202**5**) had some sample data in CSV format along with the index.html page. This formed the bedrock of our project. We set up a Node.js server with Express.js, creating the server.js file and package.json to keep track of dependencies. The server.js served our project and contained API endpoints for executing Python scripts and data analysis. Each team member configured their local development environment to mirror production. We standardized on Node.js 14+ and Python 3.10 for consistency. For Python dependencies, we implemented virtual environments (venv) for each developer, with venv/ added to .gitignore. Running "python3 -m venv venv" followed by source "venv/bin/activate and pip install -r scripts/requirements.txt" became our standard setup procedure.
Server Infrastructure and Hosting:	On the host, we set up port forwarding on ports 443 (HTTPS), 80 (HTTP), and 22 (SSH). This was required to host the website on both secure and insecure modes. The HTTPS was not necessary but perhaps the user didn't want their private chess data to be intercepted by hackers. It also saved us the hassle of clicking insecure mode whenever accessing the webpage through Chrome. SSL certificates were obtained through Let's Encrypt for free encryption and this was configured with certbot automatically. We set up our domain and DNS using Cloudflare. The domain (chess-app.kkv9.ovh) was configured to point to our EC2 instance's IP. We used an Nginx reverse proxy to route our web app running on port 3000 to our domain.
Amazon EC2 Backend Configuration:	On the backend, we set up and configured an Ubuntu 22.04 LTS server on Amazon EC2. This was a simple process that we all had lots of experience with. We chose a t3.micro instance deployed in the eu-north-1 region. The instance was set up with a static IP to ensure the server's public IP remained constant. We needed to set up SSH in order for GitHub Actions to be able to talk to the server. We created an SSH key for administrator access and a separate key to be used by GitHub Actions for CD (Continuous Delivery). This separation followed security best practices - if the GitHub Actions key were compromised, it wouldn't give an attacker full admin access. The GitHub Actions key was stored as a repository secret in GitHub's encrypted secrets storage. The server configuration involved installing Node.js, npm, Nginx, and PM2 (Process Manager 2). PM2 was chosen as our process manager because it provides automatic restarts on crashes and zero-downtime reloads when deploying updates. We configured PM2 with an

	ecosystem.config.js file that specified our application settings, including environment variables, memory limits, and restart policies.
Continuous Delivery Pipeline:	We created the deploy.yml file in our GitHub repository's .github/workflows directory to automate deployment. The trigger was configured as on push: branches: - master, which ensures that GitHub Actions triggers whenever we push new code to the master branch. This allows us to have a continuous delivery workflow so that the server stays up to date automatically.
Python Integration and Environment Management:	One tricky aspect was ensuring Python virtual environment was setup properly. We needed to do this because ubuntu won't allow us to install python modules using pip.
Monitoring and Security:	PM2 provides built-in monitoring through pm2 monit and pm2 logs for debugging.

8. Testing and Verification:

Testing and verification has taken place throughout the development of the Chess Analyzer project as opposed to being left until the end. Each sprint required us to test new features and confirm that they are meeting the goals of the user stories before being marked as complete on Jira. This helps us to spot issues early and to improve the application as we progress. Testing and verifying also ensures smooth user interaction and application stability.

A python script called 'validate.py' was developed as part of the verification process. Its purpose is to ensure the data from the CSV file exists, can be opened and contains valid headers and data rows. This verifies the dataset before analysis begins and prevents errors caused by badly formatted data. Once our verification tool has ran and the file passes verification, it is then safe to use later in our other scripts such as the win rate script.

The win rate analysis script (wins.py) uses the validated data.csv file as its main input. The data file contains thousands of chess games. The script analyses each row of the dataset and checks the "result column", which contains the outcomes in a format such as '1-0, 0-1, ½-½'. From this data, the script then determines if white or black has won or if the match was ended in a draw. After counting the total, it calculates the percentage of each result type and outputs the results into a structured JSON format. By automating the win rate calculations, the script ensures that the data collected from the chess games is accurate, consistent, and ready for visualisation.

The Chess Analyzer App is deployed on Amazon Web Services (AWS). AWS lets us test and verify the system works in real-world conditions outside of local hosting. This is deployment verification and production testing.

PM2 was used to test and verify the apps stability and uptime. It continuously monitors the Node.js server, showing if anything crashes or fails. After each sprint deployment PM2 checked that new features didn't crash the server and could stay running smoothly under load. This is operational verification, confirming the software runs reliably in its deployed environment.

9. Interim Reflection and Next Steps:

As a development team we have successfully completed the first two sprints, building a strong foundation for our chess analyzer.

Sprint one's focus was on importing the data from a csv file using a web form and validating the data. Using python scripts, we implemented parsing and random sampling that allowed the dataset to be handled effectively. This sprint gave us a sense of how we work as a team, and it also allowed us to identify any weaknesses in our planning. This lets us improve our methods and redeploy improved agile methods with accurate user stories. This further highlighted the need for good and clear communication as much as the need for correct code.

Sprint two changed our focus to data visualization as well as the user's personal experience. Our approach to planning had greatly improved by this sprint, we learned to refine our user stories, use Jira more accurately and set achievable goals. This was further improved by employing the burndown chart to monitor our progress. As well as the weekly stand-up meetings grew more natural switching to more in-depth discussions on issues in the project and what else needs to be done to successfully meet the requirements. The switch from raw data to visualization taught us the importance of incremental progress. Instead of trying to build all the features at once we divided the work up into small achievable parts. This mindset reduced the teams over all pressure and helped us maintain steady progress throughout the sprint.

The next steps for our team to complete is to move towards insight generation. Sprint 3 will focus on integrating machine learning and large language model capabilities to spot patterns in the existing data and give the user strategic help. Implementing AI assisted querying such as Gemini Ai to produce personalized player profiles. A new scrum master will oversee this sprint facilitating planning, stand ups and role designation as previously done in the other sprints. The attention will stay on keeping up the Velocity of the project and making sure the deliverables remain in line with the user stories. By continuing to apply short feedback cycles, regular reviews, and incremental testing, the team aims to deliver sprint 3 successfully.