International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Chess Academy Management System and Tournament Analyser

Tharun SU, Shanthiv B, Dr. Kiran Kumar
BCA, BMS College of Commerce and Management, Bengaluru, Karnataka, India

Abstract

Chess academies and tournament organisers often manage player details, registrations, pairings, round results and performance records through separate spreadsheets or paper documents. Such fragmented workflows can cause duplicated entries, delayed updates and difficulty in reviewing a player's progress across rounds. This paper presents the design of the Chess Academy Management System and Tournament Analyser, a web-based application that centralises tournament information and supports more consistent event administration. The system uses a Python Flask application, a relational database accessed through SQLAlchemy, and a browser interface built with HTML, CSS, Bootstrap and JavaScript. Its tournament workflow maintains player records, manages event participation, records round results, derives standings from stored results and presents performance information to organisers and players. The architecture separates presentation, application logic and data persistence and applies role-aware access control. The paper explains the problem context, objectives, architecture, data design, principal workflows and implementation technologies, and defines a functional test plan. Planned test cases are kept distinct from executed tests, so that no result is claimed without evidence. The system is presented as an administrative and analysis aid, not as a replacement for official federation systems. Verified rating integration, additional pairing formats, richer reporting and engine-based analysis are identified as future work.

Keywords: Chess tournament management, Flask, SQLAlchemy, tournament standings, performance analysis

1. Introduction

Chess academies combine regular coaching with the organisation of tournaments. Each tournament depends on accurate player information, registration records, round pairings, result entry and standings. When these tasks are carried out on paper sheets or in independent spreadsheets, the same information must often be typed in several places, and any correction has to be repeated in each of them. The result is extra administrative work and a weak record of who played in which event and with what outcome.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

A web-based system can bring these tasks together and give authorised users access to a single set of records. An analysis component adds value when it organises stored results and presents performance information, so that every tournament contributes to a continuing record and is not treated as an isolated event. This paper describes the Chess Academy Management System and Tournament Analyser, a modular application designed around chess academy and tournament workflows. Its contribution is a focused and extensible implementation in which player, event and result records are linked, so that standings and player-level performance views are derived from one body of stored data.

2. Related Work

Several kinds of resource support chess activity, but they serve different purposes. Online platforms such as Lichess [6] and Chess.com [7] are primarily player-facing services that offer online play and access to game records. Federation resources, such as the FIDE Handbook [5], set out the regulations that govern organised and rated play, including rules on how tournaments are conducted.

For that kind of local administration, general-purpose spreadsheets remain a common choice because they are familiar and flexible. Their weakness is that registrations, pairings, results and standings tend to be held in separate sheets or files, so consistency depends on the care of the person editing them. A purpose-built web application can keep these records in one relational database and apply validation and access control in a way that a spreadsheet cannot easily enforce. The present system is therefore positioned as an administrative and analysis aid that complements federation and online services by organising an organiser's own records. It is not intended to replace official federation systems or their rating processes.

The design follows common web-development practice, in which a server-side application enforces business rules, a relational database stores structured records, and a browser interface handles interaction. Flask [1] is a lightweight Python web framework that supports this separation. SQLAlchemy [2] maps Python classes to database tables and can work with different database back ends. MySQL [3] provides a relational database for deployment, and Bootstrap [4] supplies components for a responsive interface.

3. Problem Statement

Manual or fragmented tournament administration causes several practical problems. Player details may be duplicated across files. Registrations and results may be kept in separate documents. Standings may have to be recalculated by hand after each round, and corrections

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

are difficult to trace. Historical performance is not available in a consolidated view. These problems consume organisers' time and increase the risk of inconsistent information. The problem addressed in this paper is the lack of a single, structured workflow for maintaining tournament data and reviewing results consistently.

4. Objectives and Intended Users

The objectives of the study are as follows:

- To design a centralised web application for maintaining player and tournament information.
- To provide structured workflows for tournament registration and round-result recording.
- To calculate and present standings from the results entered into the system.
- To provide a foundation for reviewing player participation and performance across rounds and events.
- To apply a maintainable architecture with database-backed persistence and access control.
- To identify further improvements, including external rating integration, advanced analytics and expanded reporting.

The system is intended for two groups of users. Tournament organisers and academy staff create events, register participants and enter results. Players and coaches review standings and performance information.

5. Scope of the Study

The study concentrates on the software design and the tournament-oriented workflows of the application: player and event records, registration, round-result recording, standings and performance review. The system has a browser-based interface and a Flask backend with relational data persistence. Wider academy administration functions, namely coach management, batch management, attendance and fee collection, are outside the scope of the current build and are not presented as completed features. Automatic synchronisation with FIDE or state-association rating databases and chess-engine analysis are treated as future extensions, unless they are separately integrated and verified.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

6. Proposed System

The proposed system organises tournament information in a single application. An authorised organiser can maintain player and event records, register participants, record round outcomes and review the resulting standings. The application keeps interface handling, business logic and database persistence apart, which improves maintainability. Role-aware access control is applied, where configured, so that sensitive administrative actions are not available to every user.

Table 1 summarises the functional modules of the system and the purpose of each.

Table 1. Functional modules of the system

Module	Purpose
Player records	Store the details that identify each participant, so that one player can be reused across events.
Tournament records	Define each event and hold the information needed to administer it.
Registration	Associate players with events and avoid unintended duplicate entries.
Round and result management	Record pairing and round information and the outcome of each game as it finishes.
Standings	Present points and positions derived from the recorded results and the configured scoring rules.
Performance review	Present stored results in a form that organisers, coaches and players can review across rounds and events.
Authentication and access control	Restrict sensitive administrative actions to authorised users, where configured.

7. System Architecture

The application follows a three-layer logical architecture, shown in Figure 1.

- Presentation layer: HTML templates, CSS, Bootstrap, Font Awesome and JavaScript provide the pages, forms, tables and visual summaries shown in the browser.
- Application layer: Python and Flask routes process requests, validate submitted data, apply tournament rules and coordinate authentication and access control.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

- Data layer: SQLAlchemy models and a relational database store player, tournament, registration and result information. MySQL is intended for deployment, with SQLite available for local development or testing where configured.

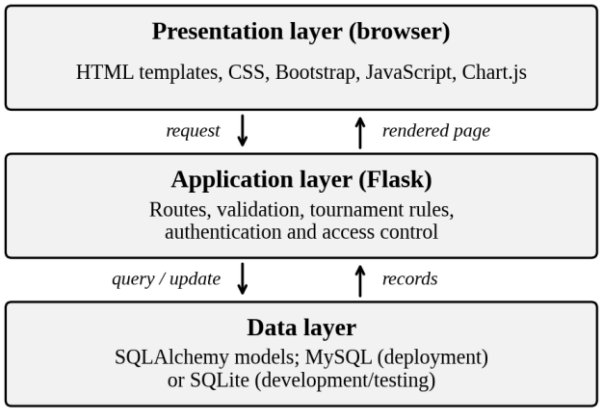


Figure 1. Three-layer architecture of the system

A typical request begins when a user submits a form or opens a tournament page. The Flask route validates the request and calls the relevant application logic. SQLAlchemy reads or updates the database, and the server returns a rendered page that shows the current records. Separating these responsibilities keeps data operations consistent and makes individual modules easier to test.

8. Methodology

8.1 Development Approach

An iterative development approach was used. The core administrative problem and the expected users were identified first. The main entities and workflows were then translated into application routes, database models and page templates. The interface and the server-side logic were developed together, with validation and access checks added to the relevant operations. The application is then reviewed through the workflow-oriented functional tests described in Section 10. Iteration allows defects and usability gaps to be corrected before further capabilities are added.

8.2 Data and Workflow Design

The logical data model centres on four concepts, summarised in Table 2. Keeping them separate reduces duplication and allows the same player to take part in several events. Validation is applied to prevent incomplete or contradictory entries wherever possible.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Table 2. Core concepts of the logical data model

Concept	Role in the model
Player	Identifies a participant. One player record can be linked to many events.
Tournament	Defines an event for which participants register and results are recorded.
Participation	Associates a player with a tournament, so that registration is recorded once and not repeated.
Result	Stores the outcome of a round or pairing, and is the data from which standings are derived.

8.3 Tournament Processing

The tournament workflow is shown in Figure 2. The organiser creates or selects an event, registers participants, enters pairing and round information according to the event workflow, and records results as games finish. Standings are then derived from the results stored in the database and displayed for review.

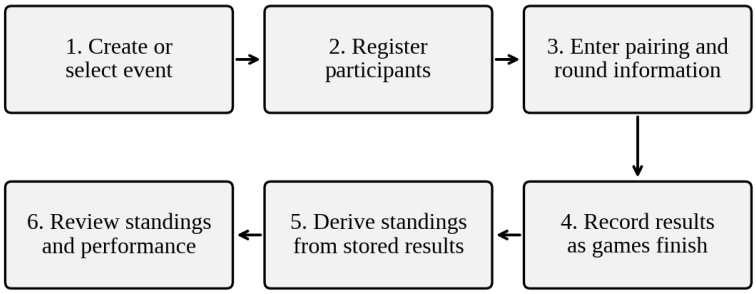


Figure 2. Tournament processing workflow

The correctness of any pairing method or tie-break calculation depends on the rules implemented and on the data entered. Organisers should therefore check the event settings and the official regulations, for example those in the FIDE Handbook [5], before using generated outputs for a rated or formally sanctioned event.

9. Implementation Technologies

Table 3 lists the technologies used and the purpose of each.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Table 3. Technologies used and their purpose

Technology	Purpose
Python	Primary programming language for application logic.
Flask [1]	Web framework, routing and application structure.
SQLAlchemy [2]	Object-relational mapping and database interaction.
MySQL [3] / SQLite	Relational persistence for deployment and for local development or testing, respectively, where configured.
HTML, CSS, Bootstrap [4]	Page structure, styling and responsive layout.
JavaScript and Chart.js	Client-side interaction and visual summaries where implemented.
Flask-Login and Werkzeug	Session-based authentication support and secure password hashing where configured.
Git	Version control and change tracking.

Flask suits an application of this size because routes, models and templates can be extended one module at a time. Using MySQL for deployment and SQLite for development or testing lets the same models be exercised locally first. Where configured, administrative actions require an authenticated session and passwords are stored as hashes, which matters because the system holds personal information about players.

10. Evaluation and Discussion

10.1 Planned Test Cases

Functional testing checks whether each workflow behaves as expected for valid and invalid inputs. Table 4 defines the planned test cases and the behaviour expected in each. The expected behaviour is a specification and is not a result.

Table 4. Planned functional test cases and expected behaviour

ID	Test case	Action	Expected behaviour
T1	Authentication	Log in with valid and with invalid credentials.	Valid credentials permit access; invalid credentials are rejected.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

ID	Test case	Action	Expected behaviour
T2	Player record	Create a player with valid details, then retrieve it.	The record is saved and displayed correctly.
T3	Tournament record	Create an event with valid details.	The event is saved and displayed.
T4	Registration	Register a player for an event, then repeat the registration.	The participant is linked to the event and no unintended duplicate is created.
T5	Result entry	Save a round result and reload the page.	The result remains available after reload.
T6	Standings	Enter several results and view the standings.	Points and positions reflect the recorded results and the configured rules.
T7	Input validation	Submit forms with missing or invalid values.	Invalid entries are rejected with understandable feedback.
T8	Access control	Attempt restricted actions as an unauthorised user.	Restricted administrative actions are denied.

10.2 Test Execution Record

This paper does not report executed test results. Table 5 is the record in which the outcome of each test is to be entered, with Pass or Fail, once the test has been run on the final version of the application. A case that has not been run must be marked as not executed, and a failed case must be recorded together with the defect found and the result of retesting after correction.

Table 5. Test execution record (to be completed from actual test runs)

ID	Executed (Yes/No)	Observed result	Outcome (Pass/Fail)	Defect / retest note
T1				
T2				
T3				
T4				

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

ID	Executed (Yes/No)	Observed result	Outcome (Pass/Fail)	Defect / retest note
T5				
T6				
T7				
T8				

10.3 Discussion

The principal contribution of the system is the consolidation of tournament information into one structured digital workflow. Storing player, event and result records together can reduce repeated data entry and make information easier to retrieve. Deriving standings from stored results can reduce reliance on manual calculation, provided that the implemented scoring and tie-break rules are correct.

The usefulness of the system depends on data quality, correct configuration and usability during live events. Incorrectly entered results lead to incorrect standings, and software-generated pairings should not be assumed to meet every federation or tournament requirement without validation. The application is therefore treated as an administrative aid, and no claim of official certification is made. No claims about accuracy, speed or reduced tournament duration are made, because none have been measured.

11. Limitations

- The reliability of standings depends on accurate participant, pairing and result data.
- Official rating updates and federation-ID validation need verified external data sources and are not assumed to be available automatically.
- Chess-engine analysis, live public standings and mobile applications would require additional development.
- Performance, scalability and usability have not been measured in a large-scale controlled study.
- Executed functional test outcomes are not reported in this paper (Section 10.2).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

12. Future Enhancements

- Verified integration with official rating and federation data sources, subject to access and applicable rules.
- Additional pairing formats and independently validated tie-break calculations.
- Export of pairings, standings and results to PDF, Excel or CSV.
- PGN import and chess-engine-based game analysis.
- Player dashboards with historical event summaries and progress trends.
- Automated backups, audit logs, improved accessibility and deployment monitoring.

13. Conclusion

This paper presented the Chess Academy Management System and Tournament Analyser, a web-based application for organising tournament records and supporting result-based performance review. The system combines a Flask application, SQLAlchemy-based data access and a browser interface to provide a modular foundation for player, event and result workflows. Centralised records make tournament information easier to maintain and retrieve, and standings derived from stored results can support organisers when the scoring logic and the entered data have been validated. The system should be evaluated against the functional test plan and the rules of the intended event before operational use. Future work can extend the platform with verified rating integration, richer reports and deeper game analysis.

References

[1] Pallets Projects, “Flask Documentation.” [Online]. Available: https://flask.palletsprojects.com/

[2] SQLAlchemy, “SQLAlchemy Documentation.” [Online]. Available: https://docs.sqlalchemy.org/

[3] Oracle, “MySQL 8.0 Reference Manual.” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[4] Bootstrap, “Bootstrap Documentation.” [Online]. Available: https://getbootstrap.com/docs/

[5] FIDE, “FIDE Handbook.” [Online]. Available: https://handbook.fide.com/

[6] Lichess, “Lichess.org.” [Online]. Available: https://lichess.org/

[7] Chess.com, “Chess.com.” [Online]. Available: https://www.chess.com/

(This work is licensed under a Creative Commons Attribution 4.0 International License.)