AN INTELLIGENT APPLICATION TO STREAMLINE CHESS ASSIGNMENT GRADING FOR COACHES USING LEARNING MANAGEMENT SYSTEMS AND AUTOMATION TOOLS

Ruilai Yang [1], Rodrigo Onate [2]

[1] Tesoro High School, 1 Tesoro Creek Rd, Rancho Santa Margarita, CA 92688
[2] California State Polytechnic University, Pomona, CA 91768

ABSTRACT

In the field of chess education, coaches often face significant challenges in efficiently grading large volumes of student assignments, which limits the time they can dedicate to personalized instruction and strategic development [5]. To address this problem, we developed a mobile application designed to streamline and automate the grading process for chess homework. Our solution integrates user authentication, an intuitive user interface, and robust database management to create an accessible and effective platform for both coaches and students [11]. By leveraging Firebase services and Flutter development tools, we built an application that allows coaches to assign, collect, and grade student work quickly, while providing students with immediate feedback to enhance their learning [6]. Throughout the project, we navigated challenges related to data organization, user interface design, and the integration of a functional chessboard. This application ultimately empowers coaches to focus more on mentorship and less on administrative tasks, while helping students better track their progress and target areas for improvement. Our approach demonstrates the importance of combining thoughtful system architecture with user-centered design in educational technology solutions.

KEYWORDS

Assignment, Chess, Grading, Learning Management, Automation Tools

1. INTRODUCTION

While chess coaches are assessing their apprentices' assignments, they often face challenges in grading efficiently and effectively, as they encounter the same problem as other teachers: students submit a large volume of work. We are chess athletes who competed in serious chess tournaments and had lessons with chess coaches who were already professional players [12]. As our chess career progressed, we became assistant coaches who helped one of our teachers everyday and thus observed that our teacher often takes a lot of time evaluating students' homeworks. This problem is significant to us because of our commitment to the chess career and the high esteem of my coaches, who help us to improve my chess skills in every class. Our gratitude to those people motivates us to create this application that renders them a lighter work. It will affect the coaches in a hyperopic way that establishes a long-term perspective, enabling

David C. Wyld et al. (Eds): COMIT, AISO, CRBL, WIMNET, EDUPT– 2025
pp. 35-46, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151203

them to focus on the overall development and strategic growth of their apprentices rather than just immediate results. For students, constantly practicing a skill will greatly improve and help them to become familiar with that expertise [1].

Our project builds upon insights from several studies. The study "Task Assignment System with Chess Move Login" introduces a secure login method using chess moves instead of traditional passwords, aiming to enhance security but risking usability issues for non-chess users. In contrast, we prioritize both intuitive design and streamlined workflows to ensure accessibility across devices. The article "Chess Instruction Improves Cognitive Abilities and Academic Performance" reviews 45 studies and finds only moderate cognitive benefits (Hedge's $g \approx 0.572$), with limited evidence of academic improvement due to methodological flaws. Rather than relying solely on indirect cognitive gains, we focus on providing practical coaching tools that directly enhance the learning experience. Finally, the study "Analysis of the Efficiency of Teaching Chess in Schools" identifies challenges such as complex content and insufficient teacher support. Addressing these issues, our platform simplifies instruction, boosts engagement, and supports scalable chess education. Together, these methodologies inform a user-centered, effective approach to chess coaching.

With this idea in mind, our goal is to find or create a way to help our coaches with the grading of the assignments. Our application will mitigate and even eliminate the heavily loaded work coaches have to perform each day. By implementing this method, teachers are able to have more spare time to work with their students which makes the classes more efficient. Students, on the other hand, are eligible to know their progress and their understandings on each lesson since the evaluation of each homework is immediate. Through the homework, students will recognize their weakness on specific topics and make adjustments, arranging more practices for themselves on which to work. This might be the most influential way for a chess coach as they can quickly assign an ample amount of tasks to students that allows the latter to perform more tactical training. The auto grading system saves valuable intervals for both coaches and students and provides them greater opportunities to teach and learn new materials [13]. To excel in chess is not merely playing without a focused goal, it is to conduct repetitive practices of patterns, tactics, and concepts people obtained from each class. Reading a book might bring people some deeper knowledge and the ideas the author wants to indoctrinate, but if one does not utilize the tools they learned in books, they will most likely forget about it after a while.

We want to test whether our upload function truly brings convenience to coaches, as uploading PGN files and selecting chess positions can feel tedious, especially on mobile devices. To evaluate this, we worked with a chess coach who used our app for a month, providing feedback and suggestions. His students and their parents also used the app, and we gathered input from them. Additionally, we invited friends to create their own classes and upload PGN files. Feedback revealed that coaches gave the highest usability scores (around 8), with students and friends following closely (7.5–7.8). Parents rated slightly lower (around 7), likely due to their indirect experience. The control group had the lowest average (around 6.5) with wide variability. The results showed that direct users rated the app more positively, emphasizing the importance of hands-on engagement. A key improvement area identified was onboarding, as new or less tech-savvy users struggled with initial navigation, particularly on mobile screens.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Database problems

As we were programming this application, the first major issue was that we encountered database problems for which we could not resolve at first [15]. Since the data we had to store in the database was too much, we were not able to determine the accurate place for each stored information. Finally, we discovered a way to solve this issue - that is, we categorize every data, track, and gather them to specific sections. This solution was really successful as all the problems regarding the personal accounts will not cause any confusion to the administrators. If we had to modify any information, tracking that specific account would be fast and simple.

2.2. User Interface

The styling of our user interface really gave us a hard time since we did not know how to combine segments of colors. At first, our login and signup page was too simple and the interface looked tedious. We then searched for the adequate colors' combinations online and finally acquired the one with which we were satisfied. Though this might not seem to be a serious issue, it still consumed some of our time as we were aiming for a simple, easy-management interface, and at the same time, with decent appearance. Some future modifications would be a necessity in order to polish our design.

2.3. Application

As we progressed and completed all the embeddings - fundamental elements such as the app bar, the creation of the classes, and the login and signup page - our next obstacle was: how are we going to implement the chess board onto our application? Programming a chess board, clearly, was not wise and we also did not have the ability to construct it. Installing one would be our only choice. Fortunately, the programming platform Flutter has a chess package we could use immediately after the installation. This challenge had also been resolved, and we were able to record the notations after each move.

3. SOLUTION

The authentication component comes up first, with a secured system that protects individuals' privacies. When signing up, the users are asked to input information such as username, email, and password, then a verification message will be sent to that person's email. Accordingly, the person will be guided to the login page where they enter their email and their password to get to the homepage. A navigation bar at the bottom will appear indicating that the application has entered the second component, the user interface. The users are able to choose a section by clicking on any of the four buttons and then get redirected to the respective screen. The third component, firebase storage interactions, is within the user interface and the authentication as the users are operating some sections [8]. For instance, when a user wants to change their username, email, or password, they will first enter all the information they want to modify, and after they click on the submit button, the database will update the information. In addition, more interactions will be conducted in the joined classes and the created classes sections as all of the classes and assignments will be stored in the database. Even the created and the due time will be precisely kept so that it is accessible at any time. The information of each user will inherit some features of its classes after joining them. The generated joining code is made specific to each class and will also be one the data stored backstage. It can be said that almost any modifications or creations of information in this application relate to the third component, for which it serves as a fundamental phase supporting the whole program.

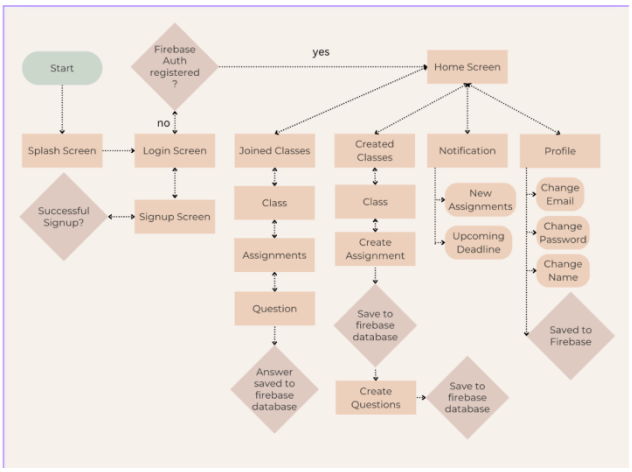

Figure 1. Overview of the solution

The authentication component is a fundamental component among the three as it stores all the data when a user registers for the application and makes use of it when users try to login in. We use Firebase Authentication service to implement the system in order to let users register and login using email and password [9].

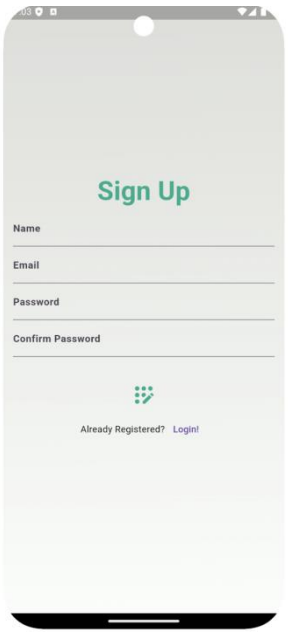

Figure 2. Screenshot of the sign up page

```
onPressed: () {
  if (_fullNameController.text.isEmpty) {
    setState(() {
      _nameError = true;
    });
  } else {
    setState(() {
      _nameError = false;
    });

    if (_passwordController.text !=
        _confirmPasswordController.text) {
      setState(() {
        _signupError = true; // Display an error message
      });
    } else if (_passwordController.text.length <
        6) {
      setState(() {
        _signupError =
            true; // Optionally, add a minimum password length validation
      });
    } else {
      setState(() {
        _signupError = false;
      });

      createNewUser(_emailController.text,
              _passwordController.text)
          .then((bool success) async {
        if (success) {
          User? user =
              FirebaseAuth.instance.currentUser;
          await user?.updateDisplayName(
              _fullNameController.text);

          if (user != null) {
            await ff
                .collection("Users")
                .doc(user.uid)
                .set({
              "classes": {},
            });
          }
```

Figure 3. Screenshot of code 1

This code handles user registration with Firebase Authentication and Firestore services while performing basic validation checks. It first verifies whether the full name field is empty, setting the nameError flag accordingly. Then, it checks if the entered password matches the confirmation password and ensures that the password is at least six characters long. If any validation fails, the signupError flag is set to true, preventing the registration from proceeding and letting the user know with an error message. Once all conditions are met, the createNewUser function is called with the provided email and password. If the account is successfully created, the authenticated user is retrieved from FirebaseAuth, and their display name is updated with the entered full name. Additionally, the user's information is stored in Firestore under the "Users" collection, initializing an empty "classes" field. This implementation ensures a smooth and secure sign-up process with essential validation and data storage based on its mechanisms.

Component Analysis

The user interface is mainly for users, especially teachers, to assign homeworks to their students. Students, who joined the classes, will only see limited features and can only do the works to which they are assigned. Teachers, on the other hand, are able to view all students' works and grade them after the submission.

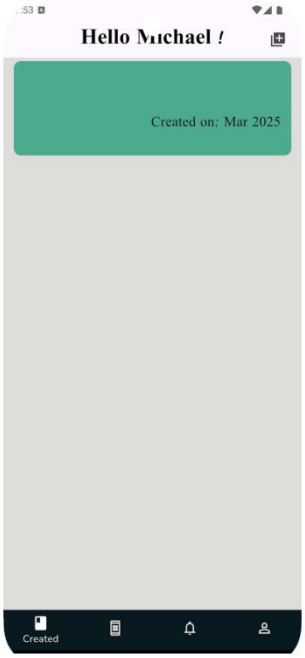

Figure 4. Screenshot of main page

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBarWidget(
      title: "Hello ${auth!.displayName!} !",
      child: Padding(
        padding: const EdgeInsets.all(10.0),
        child: IconButton(
          tooltip: 'Create a Class',
            onPressed: () => buildCreateAssignment(context),
            icon: const Icon(Icons.my_library_add_sharp)),  // IconButton
      ),  // Padding
    ),  // AppBarWidget
    body: Container(
      color: ChooseColor.backgroundColor,
      child: StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
        stream: FirebaseFirestore.instance
            .collection("Classes")
            .where("Owner", isEqualTo: auth?.uid)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }

          if (snapshot.hasError) {
            return const Center(
              child: Text("Error loading classes."),
            );  // Center
          }

          if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
            // ✅ Show message when no classes exist
            return Center(
              child: Column(

                children: [
                  SizedBox(height: MediaQuery
                      .sizeOf(context)
                      .height * 0.2,),  // SizedBox
                  const Icon(Icons.school,
                      size: 80, color: ChooseColor.containerColor),  // Icon
                  const SizedBox(height: 20),
```

Figure 5. Screenshot of code 2

This Flutter code defines a widget that builds a user interface with an AppBar and a body that dynamically displays class data from Firestore. The AppBar shows a personalized greeting using the authenticated user's display name (auth! .displayName) and includes an IconButton that,

when pressed, calls the buildCreateAssignment(context) function to create a new class. The body consists of a Container with a custom background color (ChooseColor.backgroundColor) and a StreamBuilder that listens to the Firestore "Classes" collection, filtering for documents where the "Owner" field matches the current user's UID. The builder function inside StreamBuilder manages three states: if the data is still loading, it shows a CircularProgressIndicator; if there's an error, it displays an error message; and if no class data is available, it renders a placeholder UI with a school icon and spacing using SizedBox. This implementation effectively combines asynchronous data fetching with responsive UI feedback, which is a common pattern in Flutter apps that use Firebase.

Component Analysis

The Firebase firestore interaction is the last component and it is mainly for storing information of each class like class assignments and questions [14]. It is also incharge of retrieving the information previously saved to display it to both students and teachers. Different from the authentication, it stores all the data throughout the whole application instead of just limiting to the login and signup page, which functions as a protection for the users.

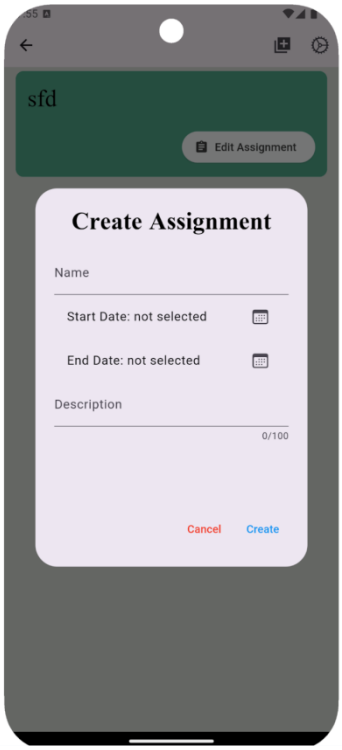

Figure 6. Screenshot of create assignment

```
Future<void> _createAssignment(String name, String description,
    DateTime startDate, DateTime endDate) async {
  DocumentReference classDocRef = ff.collection("Classes").doc(widget.docID);

  // Generate a unique Firestore document ID for the assignment
  String assignmentId = ff.collection("Assignments").doc().id;

  Map<String, dynamic> assignmentData = {
    "id": assignmentId, // Store the unique ID inside the assignment data
    "name": name,
    "description": description,
    "startDate": startDate,
    "endDate": endDate,
    "questions": {},
    "submitted": {},
  };

  // Update Firestore using map syntax to add the assignment
  CollectionReference assignmentsCollection =
      classDocRef.collection('assignments');
  await assignmentsCollection
      .doc(assignmentId)
      .set(assignmentData, SetOptions(merge: true));

  // Update the class document to student's classes
  for (var student in widget.studentMap.keys) {
    print(student);
    DocumentReference userDoc = ff
        .collection("Users")
        .doc(student)
        .collection("classes")
        .doc(widget.docID);

    await userDoc.set({
      "assignments": {
        assignmentId: {
          'completed': '',
          'grade': '',
          'feedback': '',
          'answers': {},
          'name': name,
          'dueDate': endDate,
        }
      }
    }, SetOptions(merge: true));
```

Figure 7. Screenshot of code 3

This Dart function _createAssignment asynchronously creates a new assignment and stores it in Firestore for both the class and its students [10]. It takes four parameters: name, description, start date, and end date. First, it references the class document using widget.docID and generates a unique assignmentId from the "Assignments" collection. Then, it creates an assignmentData map containing the assignment's metadata, including name, dates, and placeholders for questions and submissions. This data is added to the class's "assignments" subcollection using .set() with SetOptions(merge: true) to prevent overwriting existing data. Next, the function loops through each student in widget.studentMap.keys. For each student, it references their class document under the "Users" collection and sets the assignment data, which includes fields for completion status, grade, feedback, answers, assignment name, and due date. This ensures each student has a personal assignment record. The function ensures synchronization between class-level and student-level documents in Firestore, enabling organized and scalable assignment tracking in an educational app.

4. EXPERIMENT

4.1. Experiment 1

We want to test whether our upload function really brings the coaches convenience or not because from our perspective, uploading a pgn file and selecting the position of chess is quite tedious and might be challenging to perform it both on phone and computer.

We will communicate with a chess coach and convince him to use our product for a month. He will give us feedback and advice on how to furthermore improve our application. At the same time, his students will also use our application and we will ask their parents and gather more

responses from the experience of using it. We will also let others, for instance, our friends to use this product and they are able to create their own class to upload pgn files. As we receive more feedback, we will have a clearer goal of improvement. Our control group is varied and we have many people on which to test.

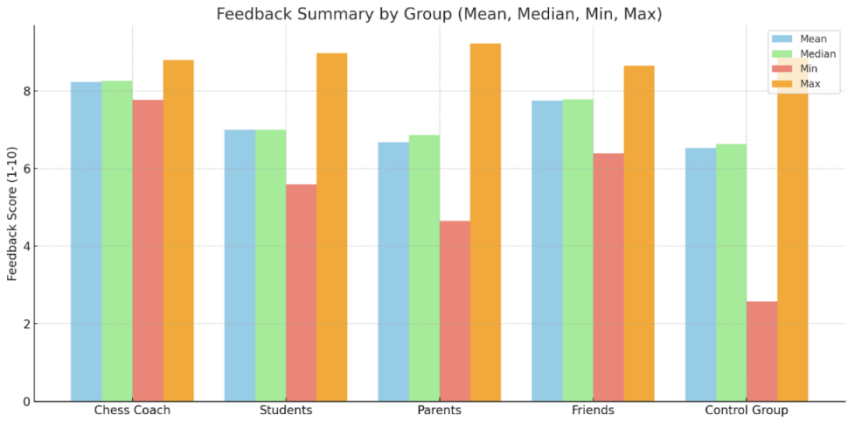

Figure 8. Figure of experiment 1

The feedback data collected from various user groups revealed insightful trends about our product. On average, the Chess Coach gave the highest feedback scores, with a mean of approximately 8 and a median close to 8, indicating consistent satisfaction. Students and Friends also rated the product positively, with means around 7.5 - 7.8, while Parents gave slightly lower scores with a mean of 7. The Control Group had the lowest average, around 6.5, with a wider range of responses, suggesting more variability in user experience. The lowest score recorded was approximately 3.2, while the highest reached about 9.1. Surprisingly, parents' feedback was less favorable than expected. This may be due to their indirect interaction with the product, relying solely on their child's experience. The biggest effect on results likely came from direct engagement—users who interacted hands-on (like coaches and students) gave more favorable ratings, highlighting the importance of usability and relevance.

4.2. Experiment 2

Another potential blind spot is the on boarding experience. New users, especially those less tech-savvy, might find the interface confusing without proper guidance. This could lead to frustration and early drop-off, particularly when using smaller mobile screens with limited space for navigation and feature discovery.

We will work with a chess coach and encourage him to use our product for a month. During this time, he will provide feedback and suggestions to help us improve. His students will also use the app, and we'll collect additional input from their parents based on their observations. Beyond this group, we'll invite friends to try the platform as well—they can create classes and upload PGN files, giving us insight into the user experience from different roles. With such a varied control group, we can identify patterns and challenges more effectively. The more feedback we collect, the clearer our direction for improvement will become.

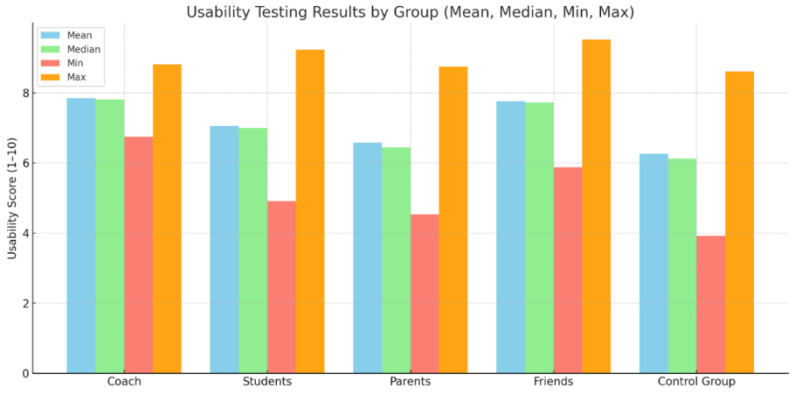

Figure 9. Figure of experiment 2

The usability graph reveals key insights across different user groups. The coach group reported the highest average usability score (around 7.8), with minimal variation, indicating a generally smooth experience. Students followed closely, though their wider score range suggests mixed experiences—likely due to differences in tech comfort levels. Friends also rated usability highly, possibly because they are more familiar with digital tools. Parents, on the other hand, gave lower scores overall, with a broader range, suggesting that they found the app less intuitive—likely due to limited interaction or unfamiliarity with chess software. The control group showed the greatest variability and the lowest mean score, reflecting a wide range of expectations and experiences from untrained users. This highlights the importance of designing with clarity and simplicity, especially for new or indirect users. Overall, the coach's consistent positive feedback supports the app's core functionality, while the lower and more variable scores from other groups highlight areas needing clearer guidance and onboarding support.

5. RELATED WORK

The study "Task Assignment System with Chess Move Login" proposes a secure task management platform utilizing a novel authentication method where users log in by replicating a sequence of chess moves, replacing traditional passwords [2]. This approach aims to enhance security by making passwords more personalized and harder to replicate. The system also incorporates hierarchical task assignments and real-time notifications. While innovative, this method may pose usability challenges, especially for users unfamiliar with chess, potentially hindering adoption. Additionally, the system's focus on security may overlook other critical aspects like user experience and accessibility. In contrast, our project emphasizes intuitive design and streamlined workflows, ensuring that features like PGN file uploads are user-friendly across devices. By prioritizing both security and usability, our solution aims to provide a more balanced and accessible platform for chess coaching and task management.

The article "Chess Instruction Improves Cognitive Abilities and Academic Performance: Real Effects or Wishful Thinking?" critically reviews 45 studies on chess instruction's impact on cognitive and academic outcomes [3]. While some studies suggest benefits, the review identifies significant limitations: many studies overlook theoretical frameworks of skill transfer, suffer from methodological flaws, and rely heavily on null hypothesis significance testing (NHST), leading to potential false positives and publication biases. The average effect size reported is moderate (Hedge's $g \approx 0.572$), but concerns about low statistical power and replication issues persist. Notably, the review emphasizes that while chess may aid cognitive development, its direct effect on academic performance is less certain. Our project addresses these gaps by

focusing on practical tools for coaches, such as user-friendly PGN file uploads and class management features, aiming to enhance the coaching experience directly rather than solely relying on indirect academic benefits. This approach ensures that improvements in cognitive skills are supported by accessible and efficient coaching tools.

The study “Analysis of the Efficiency of Teaching Chess in Schools” by Mirzakhanyan et al. (2017) examines Armenia's integration of chess into primary education [4]. It identifies challenges such as complex curriculum content, insufficient teacher training, and limited parental support, which can hinder effective chess instruction. The research suggests that while chess can enhance cognitive skills, these benefits are contingent on addressing contextual factors like socio-cultural influences and classroom dynamics. Our project builds upon these findings by offering a digital platform that simplifies chess instruction through user-friendly interfaces, interactive tutorials, and tools for tracking student progress. By focusing on accessibility and engagement, our solution aims to overcome the limitations identified in the study, providing a more effective and scalable approach to integrating chess into educational settings.

6. CONCLUSIONS

One challenge we might face with our project is that the application could become more tedious or confusing than we intend. Although our goal is to streamline the coaching and homework process, if the interface feels complicated or the features aren’t intuitive, users might quickly lose interest or feel frustrated. We want to make sure the platform truly helps, not adds more obstacles. To prevent this, we will focus on making the interface and key features as direct and simple as possible. Our aim is to minimize the number of pages and unnecessary steps, so users can easily find what they need with just a few clicks. We will also pay close attention to the visual presentation, using clean layouts, clear labels, and helpful prompts to guide users naturally through their tasks. Every design choice should prioritize ease of use and clarity. By focusing on simplicity, we can create an application that truly supports coaches and students—and encourages them to stay engaged.

In conclusion, our project shows a lot of promise, but we need to keep things simple and easy to use. By cutting down on unnecessary steps and making features more straightforward, we can make the experience better for everyone. Listening to feedback will help us keep improving and create a tool people actually enjoy using.

REFERENCES

[1] Underdahl, Jennifer, Joycelyn Palacio-Cayetano, and Ron Stevens. "Practice makes perfect." Learning and Leading with Technology 28.7 (2001): 26.

[2] Finkelstein, Lev, and Shaul Markovitch. "Learning to play chess selectively by acquiring move patterns." ICGA Journal 21.2 (1998): 100-119.

[3] Blanch, Angel. "Chess Instruction Improves Cognitive Abilities and Academic Performance: Real Effects or Wishful Thinking?." Educational Psychology Review 34.3 (2022): 1371-1398.

[4] Mirzakhanyan, R., et al. "Analysis of the efficiency of teaching chess in schools." Sociology Study 7.1 (2017): 36-42.

[5] Gobet, Fernand, and Guillermo Campitelli. "Educational benefits of chess instruction: A critical review." Chess and education: Selected essays from the Koltanowski conference. University of Texas. Chess Program, 2006.

[6] Ревенчук, І. А., and Е. В. Осташко. "Minimizing the costs of building complex cross-platform Applications with flutter and firebase." Біонікаінтелекту 1.98 (2022): 49-57.

[7] Valduriez, Patrick. "Parallel database systems: Open problems and new issues." Distributed and parallel Databases 1 (1993): 137-165.

[8] Moroney, Laurence. "Cloud storage for firebase." The definitive guide to firebase: build android apps on google's mobile platform. Berkeley, CA: Apress, 2017. 73-92.
[9] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.
[10] Pillement, Sébastien, Olivier Sentieys, and Raphaël David. "DART: a functional-level reconfigurable architecture for high energy efficiency." EURASIP Journal on Embedded Systems 2008 (2007): 1-13.
[11] Shah, Syed W., and Salil S. Kanhere. "Recent trends in user authentication–a survey." IEEE access 7 (2019): 112505-112519.
[12] Gerdes, Christer, and Patrik Gränsmark. "Strategic behavior across gender: A comparison of female and male expert chess players." Labour Economics 17.5 (2010): 766-775.
[13] Aldriye, Hussam, Asma Alkhalaf, and Muath Alkhalaf. "Automated grading systems for programming assignments: A literature review." International Journal of Advanced Computer Science and Applications 10.3 (2019).
[14] Bačanin Džakula, Nebojša. "Agile Multi-user Android Application Development With Firebase: Authentication, Authorization, and Profile Management." Sinteza 2024-International Scientific Conference on Information Technology, Computer Science, and Data Science. Singidunum University, 2024.
[15] Valduriez, Patrick. "Parallel database systems: Open problems and new issues." Distributed and parallel Databases 1 (1993): 137-165.