올인과외: ALL-IN-ONE Platform For Private Lesson

Seunghwan Lee, Moonhee Park, Chaeyeon Lee, and Jaeyoon Jung

capstone Team D

Abstract. Many university students work as tutors to earn extra money, but managing private lessons can be difficult. Tutors often use different tools for things like tracking tuition, scheduling classes, checking student progress, and organizing curriculum. This scattered system can make the process complicated and inefficient. Students also face problems keeping track of their schedules, assignments, and progress, which can affect their learning. At the same time, parents don't have a clear way to see how their children are doing and usually rely on incomplete updates from their kids, which can cause misunderstandings.

To solve these issues, we suggest an all-in-one web application designed specifically for private lesson management. Unlike other services that only focus on matching tutors and students, our application brings everything together into one platform. It makes managing private lessons easier and improves communication between tutors, students, and parents.

Our application includes features like a simple and easy-to-use interface, a customized tracking system for managing lessons, detailed dashboards to show progress, and a clear system for parents to monitor their child's learning. With these tools, our application aims to make private lessons more organized, effective, and transparent for everyone involved.

Keywords: Integrated Education Web, Management, Tracking, User-Friendly View

1 Introduction

Many university students work as tutors to earn extra money. However, managing private lessons is not easy. Tutors need to use different tools and apps to handle schedules, track performance, and organize lesson plans. This makes the process complicated and takes a lot of time. Students also struggle to manage their own schedules, assignments, and progress, which can make learning harder. Parents face another problem: they often don't have enough information about their child's progress and rely only on what the child tells them, which may not be complete.

To solve these problems, we created 올인과외(All-in Private Lesson), a web application that brings everything together for managing private lessons. The

goal of this project is to help tutors, students, and parents by offering a single platform that is easy to use and efficient.

Our application has three main features:

1. **Comprehensive Dashboard**: This helps students see their learning progress clearly, like score trends, assignment status, and curriculum updates. It also helps tutors and parents easily check the student's progress.
2. **Customized Tracking System**: This allows tutors to organize and manage each student's lessons in a personalized way, even if they have many students with different curricula.
3. **Easy Monitoring Interface**: Parents can quickly see the student's schedule, performance trends, and notes from the tutor, so they always know what's going on.

By combining these features, 올인과외(All-in Private Lesson) makes private lessons easier to manage, improves communication between all three groups, and helps everyone work together for better results.

1.1 Importance of Suggested Problem

Many university students work as a tutor for private lessons to earn extra income. Private lessons usually involve three different roles of people, tutor, student, and parents. Also, it includes diverse tasks to manage, such as schedules, performances, curriculum, or any notes. For schedule adjustment, the most popular service is Kakao Talk, which offers easy discussion and real-time conversation in non-face-to-face manner. For performance management, usually tutors keep in track with tools like Excel, or digital memo. When it comes to curriculum checking, it frequently requires searching the Internet for information. Even worse, since students' curricula are not standardized due to government policy, it is very confusing and complex to manage.

Like this, managing one private lesson requires at least three different services to look out for information. However, it is important to note that none of the three parties - tutors, students, and parents - can dedicate their whole time for private lessons. One of the primary purposes of private lessons is seamless management with great time efficiency, but this complex process hinders the efficiency.

Especially when the child grows up, entering adolescence, parents struggle to communicate directly with their child. Also, parents often hesitate to ask directly to the tutor due to concerns that they might place additional burden on the tutor. So, many parents have hard time of knowing exactly how the private lessons are progressing or whether their child is performing well.

To solve these problems, an all-in-one web application for private lessons' management and monitoring is extremely important.

To build our web application, we used a structured and efficient approach. Our platform is designed with three main layers: the Frontend, Backend, and Database, each with specific roles.

- **Frontend**: We used React and Vite to create a fast, interactive, and user-friendly interface. React's component-based design helped us build reusable elements like calendars and forms, while Vite ensured smooth and quick development with features like real-time updates.
- **Backend**: The backend was developed using Django, a secure and scalable web framework. It provided essential features like user authentication and data management. For data storage, we used SQLite, which integrates seamlessly with Django and supports key functionalities like schedule and student management.
- **Database**: Our database structure is optimized to handle user accounts, lesson schedules, and performance data efficiently.

During the development process, we faced challenges, especially since many team members were beginners. For instance, the frontend team initially struggled to match the designs from Figma, and the backend team had to assist in resolving layout issues. Through teamwork and role-sharing, we overcame these difficulties and successfully integrated the frontend and backend.

In terms of implementation, our system offers:

1. An **individual tracking system** for personalized lesson management.
2. A **user-friendly interface** that simplifies navigation for tutors, students, and parents.
3. Real-time synchronization of data for better communication and efficiency.

1.2 Achievements with Empirical Evaluations

Our platform successfully met its primary objectives. Teachers can now easily manage schedules, track student progress, and monitor performance through an intuitive interface. Students benefit from tools that help them organize their learning, and parents gain better visibility into their child's lessons. While some parts, like the parent page, require further development, the system provides a strong foundation for seamless private lesson management and an improved user experience for all roles.

2 Design for the Proposed Service

2.1 Overall Architecture

Our platform is divided into three main layers: Frontend, Backend, and Database, with clear responsibilities and interactions between them. The frontend sends requests (e.g., schedule updates or progress data) to the backend through APIs.

The backend processes these requests, interacts with the database, and returns the necessary responses to the frontend. The frontend dynamically updates the user interface based on the responses to provide real-time feedback to users. Our database schema is illustrated in the diagram below:

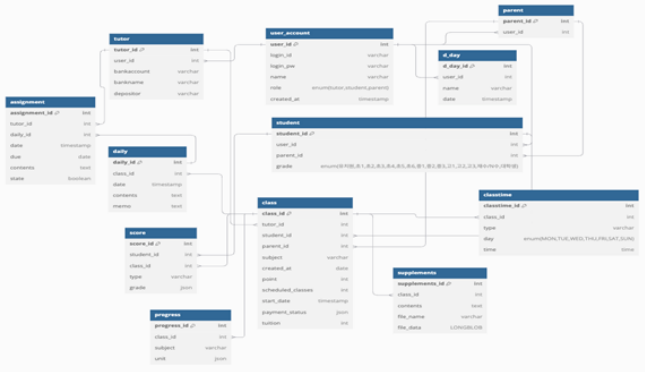

Fig. 1. Database Schema

2.2 Core skill and Reasoning

Frontend Our team selected React and Vite as the core technologies for developing the frontend of the integrated tutoring platform. React provided a component-based architecture, allowing us to efficiently manage complex user interfaces. This approach enabled the creation of reusable UI elements, such as buttons, calendars, and forms, improving both maintainability and scalability. By leveraging React's state and props, we effectively synchronized data and implemented various features, including schedule management, student management, and teacher management. Additionally, React's virtual DOM significantly enhanced the application's performance by updating only the necessary parts of the UI, ensuring a fast and smooth user experience. React's extensive system also facilitated the implementation of routing, dynamic styling, and other essential functionalities, simplifying the development process and boosting productivity. We also utilized Vite alongside React to establish a fast and efficient development environment. Vite's rapid build speed and Hot Module Replacement (HMR) functionality allowed us to reflect code changes in real-time, significantly reducing development time.

Backend For the backend development of our integrated tutoring platform, we used Django and SQLite. Django, a web framework based on the well-known Python programming language, allowed us to rapidly develop secure and scalable applications. Its built-in functionalities, such as user authentication and database management, made it an ideal choice for building applications like our platform. These features enabled us to efficiently implement key backend functionalities, including schedule management, student management, teacher management, user account management, and data synchronization. Additionally, Django's Object-Relational Mapping (ORM) system simplified database operations, allowing us to model data and execute queries effectively. For the database, we chose SQLite, which provided a lightweight and simple solution for managing

data during the development and testing phases. Its serverless architecture required no additional configuration and integrated seamlessly with Django, allowing us to reliably manage user accounts, schedules, student data, teacher data, and session information. SQLite's portability and quick setup further accelerated our development process. The combination of Django and SQLite enabled seamless integration with the React-based frontend, supporting essential features such as schedule management, student management, and teacher management in a reliable and efficient manner. This setup provided a strong foundation for building a scalable, secure, and high-performance platform.

2.3 Challenges

Our team primarily consists of beginners, which led to many challenges and trial-and-error processes during the project. In particular, the frontend team, with no prior experience in frontend development, struggled significantly to replicate the Figma design accurately. For example, during the initial stages, the components failed to fully utilize the page, and certain sections, such as the left panel, deviated considerably from the original design. These issues stemmed from the team's unfamiliarity with handling HTML, CSS, and React components, resulting in slower progress and mounting frustration. To overcome these challenges, the backend team stepped in to assist with layout adjustments while also working on linking the API and database. This allowed the frontend team to focus on developing and refining new components based on the design specifications.

For instance, the backend team worked on dynamically linking the UI of the left section with API data, while the frontend team concentrated on creating the primary interface components on the right. This division of roles and collaborative effort enabled us to gradually resolve the issues. As a result, all APIs and the database for the teacher's page and student's page are fully integrated, and the platform closely aligns with the initial Figma design. Through this process, we learned the importance of team collaboration and effective role distribution, allowing us to complement each other's skills and work more efficiently.

3 Implementation

3.1 Frontend Implementation

Technology Stack and Development Environment The frontend of the application was developed using React, a robust JavaScript library for building user interfaces, in combination with Vite, a build tool designed for rapid development. React's component-based architecture facilitated the decomposition of complex views into smaller, reusable components, improving maintainability and scalability. Vite, on the other hand, accelerated the development cycle through its Hot Module Replacement (HMR) capabilities, enabling immediate reflection of code changes without full page reloads.

Architectural Considerations The UI was structured into logically distinct components, each representing specific functionalities such as lesson scheduling calendars, performance charts, and user dashboards. By adhering to a modular design pattern, the team could manage state and data flow more effectively. State management was primarily handled using React's built-in state and props mechanisms, ensuring a clear and predictable data flow.

Routing and Navigation The React Router library was integrated to facilitate seamless navigation between different user roles (tutor, student, and parent). Dedicated routes were implemented for each role-specific dashboard, allowing users to access relevant functionalities without friction. This routing architecture ensured a clear separation of concerns and improved the overall user experience.

User Interface Development Design specifications provided via Figma were meticulously translated into code using HTML, CSS, and JavaScript. Initial challenges, such as layout inconsistencies and difficulty reproducing complex visual elements, were resolved through iterative refinement and close collaboration within the team. Over time, the UI converged to closely match the intended design while maintaining responsiveness and accessibility.

Performance Considerations React's Virtual DOM mechanism minimized unnecessary re-renders, enhancing performance by updating only modified components. Code splitting and lazy loading were employed to improve initial load times, ensuring that users only downloaded the necessary resources when required.

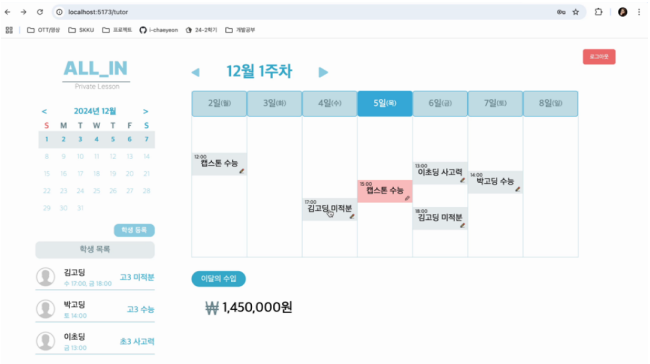

Fig. 2. Tutor Webpage

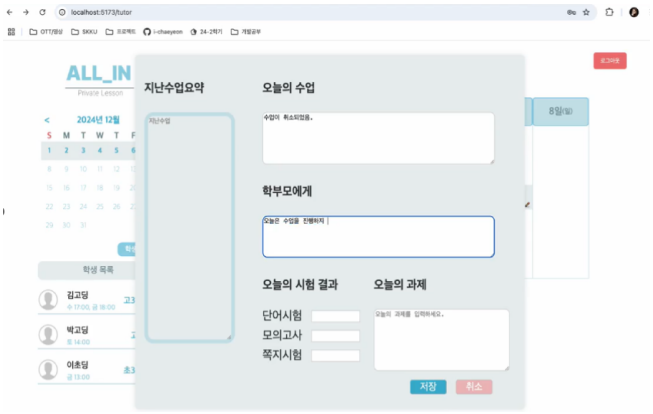

Fig. 3. Previous Lesson Summary Webpage

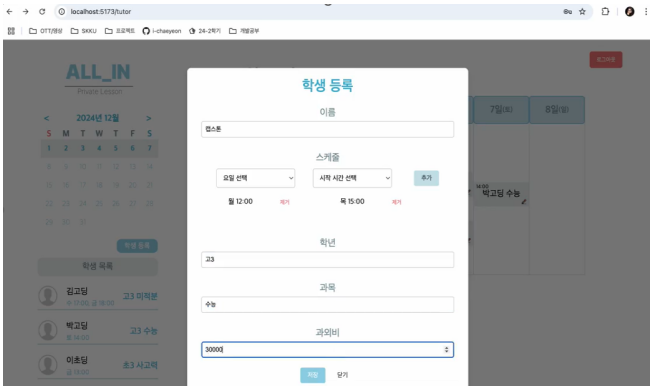

Fig. 4. Student Registration Webpage

3.2 Backend Implementation

Technology Stack and Framework Selection The backend was developed using the Django web framework, chosen for its robust feature set, including a built-in authentication system, an Object-Relational Mapping (ORM) layer, and strong security features. SQLite was initially selected as the database for development and testing due to its simplicity and ease of setup. This choice facilitated rapid prototyping and validation of key features.

Data Modeling and ORM Utilization Data models were defined to represent users, schedules, performance metrics, and related entities. Django's ORM allowed developers to interact with the database using Python objects, eliminating the need to write raw SQL. This abstraction not only improved code readability and maintainability but also accelerated development by simplifying data manipulation and schema updates.

User Authentication and Authorization Django's built-in authentication system was leveraged to implement role-based access control. Tutors, students, and parents were assigned appropriate permissions, ensuring that only authorized individuals could view or modify specific data. This security measure enforced data integrity and protected sensitive information from unauthorized access.

RESTful API Development A set of RESTful APIs was created using the Django REST Framework (DRF) to facilitate communication between the frontend and backend. Endpoints were established for managing schedules, retrieving student performance data, and handling teacher information. The APIs returned JSON responses, enabling the frontend to dynamically update the UI without page reloads, thereby improving responsiveness and user satisfaction.

Scalability and Future Enhancements While SQLite was sufficient for initial testing and small-scale deployment, architectural decisions were made with scalability in mind. The team considered future migrations to more robust databases such as PostgreSQL or MySQL. Additionally, indexing and query optimization strategies were identified as potential next steps to ensure the backend can handle increased load and complexity over time.

3.3 Integration of Frontend and Backend

Data Flow and Synchronization The frontend communicated with the backend via RESTful APIs. User actions such as adding a new lesson, updating student performance records, or reviewing schedules triggered asynchronous requests. The backend processed these requests, updated the database accordingly, and returned JSON-formatted responses. Upon receiving responses, the frontend seamlessly updated its state, ensuring that users always saw up-to-date information.

3.4 Continuous Collaboration and Iterative Refinement

Close collaboration between frontend and backend teams was critical. Backend engineers assisted in adjusting layout and resolving data consistency issues, while frontend developers provided feedback on API usability and performance. Regular testing and debugging sessions identified layout discrepancies, data mismatches, and performance bottlenecks. These sessions facilitated prompt issue resolution and guided incremental improvements.

4 Limitaitons and Discussions

Despite the successful implementation of the core functionalities, our platform still has several limitations and areas that warrant further development and refinement. Addressing these concerns will not only enhance user experience but also ensure the platform's long-term viability and scalability.

4.1 Limited Scope of Parent Interface

While the tutor and student pages have achieved a reasonable level of maturity, the parent interface requires additional features to provide more comprehensive insights. Currently, parents have a limited view of their child's progress, lesson schedules, and performance data. Future enhancements could include more detailed progress reports, interactive charts that show performance trends over time, and communication channels that allow parents to easily reach out to tutors with minimal friction. By expanding these capabilities, we can foster a more collaborative environment and ensure that parents remain well-informed without placing an undue burden on tutors.

4.2 Scalability Constraints

The current use of SQLite as the database and the chosen architecture may face limitations as the platform grows. SQLite is suitable for development and small-scale deployment but might not handle larger user bases or more complex query loads efficiently. As the number of tutors, students, and parents increases, migrating to a more scalable database solution (e.g., PostgreSQL or MySQL) and considering cloud-based hosting will likely become necessary. This transition would ensure better performance, increased data integrity, and improved availability for a potentially global user base.

By acknowledging and addressing these limitations, we can move forward with a clearer roadmap for future developments. Each challenge presents an opportunity for improvement, and through careful planning, user feedback, and continuous iteration, the platform can evolve into a fully integrated and comprehensive ecosystem for private lesson management. Ultimately, striving for scalability, enhanced features, and greater inclusivity will help ensure that tutors, students, and parents reap the full benefits of a unified and transparent education management system.

5 Related Work

5.1 클래스팅(ClassTing)

클래스팅(ClassTing) enhances user convenience by allowing sign-ups not only through standard registration but also via Google, Kakao, and Naver accounts. Inspired by this, our website also offers multiple login options, including standard registration. 클래스팅(ClassTing) enables users to export grade statistics in Excel format and track progress, helping students gain a sense of accomplishment. Similarly, our web application allows tutors to input grades and students to view progress through graphs. However, 클래스팅(ClassTing)includes features like photo and video sharing between teachers and parents, which may be unnecessary. Instead, our platform allows tutors to leave simple notes for parents. Additionally, the platform supports communication of essential details such as lesson progress, assignments, and notes relevant to students.

5.2 수업기록노트(Class Record Note)

This app allows users to easily record class times, tuition fees, and homework, as well as check weekly schedules. For students managing multiple private lessons, tuition fees can sometimes cause confusion. Our platform addresses this by enabling tutors to manage tuition details on the tutor page.

For those juggling multiple lessons, our integrated weekly calendar helps manage schedules. Using color coding, users can distinguish between upcoming lessons, canceled classes, and completed lessons. Similarly, our platform supports homework management, allowing students to check their assignments.

5.3 레슨북(Lesson Book)

레슨북(Lesson Book)offers features for managing student schedules in monthly, weekly, and daily formats. In our web application, tutors can manage schedules and important events like exams using weekly and monthly calendars. While 레슨북(Lesson Book)은 includes more complex features like lesson promotion and tutor-student matching, we have chosen to exclude these to focus on creating an efficient, streamlined platform. Our system aims to address challenges tutors face, such as managing multiple lessons and ensures nothing is overlooked.

5.4 클래스노트(Class Note)

클래스노트(Class Note) supports attaching photos and videos and restricts message transmission during late hours. However, most university students who tutor tend to organize their schedules late at night, making this feature unnecessary.

클래스노트(Class Note) also includes features like medication request forms, consent forms, and meal plans, which are irrelevant for private tutoring. We excluded such features and instead focused on essential functionalities, such as managing assignments, announcements, and lesson schedules.

5.5 Other Apps

Popular tutoring apps like *Olbaleun* **올바른 과외(Olbaleun Private Tutoring)** and **김과외(Kim Tutoring)**focus primarily on tutor-student matching rather than schedule management.

Our platform allows users to log in using various methods such as Google, Kakao, and Naver, and it is designed to cater to students, parents, and tutors with distinct roles and functionalities. Tutors can manage individual tutoring sessions and schedules for multiple students through a calendar and keep track of the tuition fees. They can share announcements and grades directly on the platform, eliminating the need for personal communication with students or parents. Students can organize their schedules even when attending multiple tutoring sessions, instantly view their grades through graphs, and conveniently check their classes and assignments all in one place. Parents can track tuition payments, monitor their child's progress, and review grades at a glance, without

requiring face-to-face meetings or direct communication with their child or tutor. This platform helps tutors, students, and parents reduce unnecessary personal communication by centralizing management in one place. It provides convenience and efficiency by allowing users to view all relevant information.

6 Conclusion

In conclusion, we developed a web platform to minimize inconvenience for tutors, students, and parents during private tutoring sessions. Our platform provides several features tailored to each user group. For tutors, it offers an integrated calendar, the ability to generate multiple student reports and tools to manage diverse curricula for each student. For students, the platform includes an integrated calendar, progress reports, personal achievement tracking, assignment and supplement management, and a reward system. Finally, for parents, it provides a quick-look calendar, tuition payment reminders, and access to their child's progress reports. Our platform ensures tutors can effectively manage multiple lessons without missing schedules or details, students can stay updated on assignments and announcements, and parents can monitor their child's tutoring progress without needing direct communication or meetings. This creates a seamless and efficient web solution for private tutoring management.

References

1. 클래스팅(Classting) Homepage, https://www.classting.com/en, last accessed 2024/10/03
2. 올바른과외 Homepage, http://m.eduallbarun.com/, last accessed 2023/10/03
3. 김과외 Homepage, https://kimstudy.com/, last accessed 2023/10/03