GetClasses: An Object-Oriented Transactional Platform for Online Tutoring

Jhon Gonzalez
Student of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: jjgonzalezc@udistrital.edu.co
code: 20251020087

Alejandro Escobar
Student of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: aescobarb@udistrital.edu.co
code: 20251020094

Sebastián Zambrano
Student of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: szambranoh@udistrital.edu.co
code: 20251020102

***Abstract*—Online education platforms have become essential in providing accessible and personalized learning experiences. This paper introduces GetClasses, a transactional application that connects students with tutors in a secure and intuitive environment. The system applies object-oriented programming (OOP) principles and a layered architecture implemented in JavaFX. Results show a functional prototype with integrated scheduling, communication, and review modules, validating the design's scalability and maintainability.**

***Index Terms*—Object-Oriented Programming, Java, SOLID, Layered Architecture, JavaFX, Software Engineering, Online Learning**

I. Introduction

Online learning has become a major trend in recent years, requiring systems capable of connecting learners and educators efficiently and safely. Previous solutions, such as online tutoring platforms and learning management systems, often face challenges with integration, user experience, and reliability. Systems like Coursera and Udemy provide broad learning content but lack personalized tutor-student interaction and real-time scheduling. GetClasses addresses this gap by offering a transactional platform where tutors and students can interact, schedule sessions, make payments, and exchange feedback within a unified system.

The GetClasses project was developed as part of the Object-Oriented Programming course at Universidad Distrital Francisco José de Caldas. It leverages modern software engineering practices, including UML modeling, SOLID principles, and layered architecture, to ensure clear separation of concerns, maintainability, and scalability. The project aims to demonstrate how OOP-based architectures can produce modular and robust educational software.

II. Methods and Materials

The project design follows a layered architecture consisting of three main layers: presentation, business logic, and persistence. The presentation layer uses JavaFX for GUI implementation, offering a responsive interface that allows users to register, search tutors, and manage sessions. The business logic layer implements system rules, ensuring that each transaction, schedule, and interaction follows the designed workflow. The persistence layer handles data storage through file-based mechanisms, later extendable to database integration.

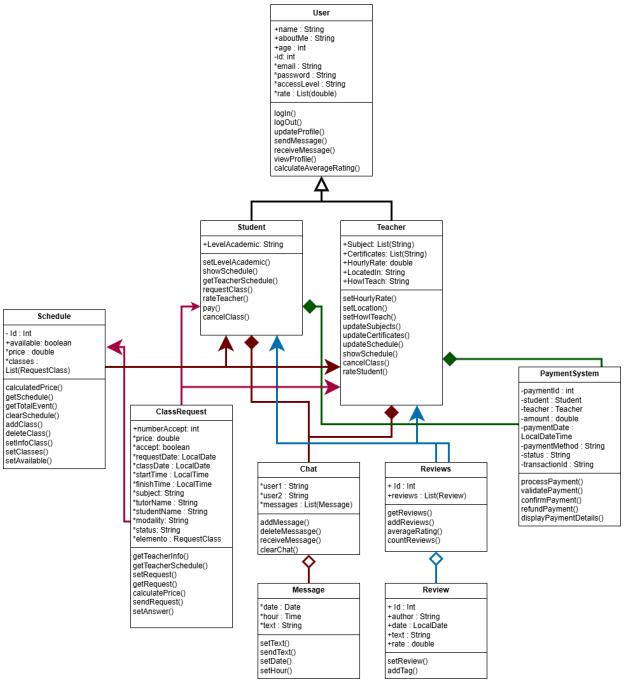

Fig. 1. UML diagram showing inheritance and interactions among system entities.

The UML class diagram (Fig. 1) defines the system structure, with a base class `User` and specialized subclasses

`StudentUser`, `TutorUser`, and `AdminUser`. Inheritance, polymorphism, and encapsulation are central to the design. The `User` class contains shared attributes (id, name, email, password) and methods (login, logout), while subclasses extend these with specific behaviors, such as class booking for students or availability management for tutors.

The GetClasses system integrates key OOP principles: encapsulation ensures secure attribute management; inheritance promotes code reuse; and polymorphism enables flexible interactions between components. SOLID principles guide the class responsibilities, reducing coupling and improving cohesion. CRC cards and user stories were used to align requirements and design decisions.

III. Results and Discussion

The first version of GetClasses provides full functionality across its core modules: registration, profile management, scheduling, chat, payment simulation, and review. Unit tests validated the correct behavior of user registration, tutor availability, and booking transactions. Integration tests confirmed interaction between modules without data loss.

Performance metrics show that typical queries (e.g., searching for tutors by subject or rate) execute in under three seconds. Usability evaluations confirmed an intuitive interface and smooth navigation. Table I summarizes the main test outcomes.

TABLE I
Summary of functional testing results

Test Module	Status	Execution Time (s)
Registration	Passed	1.8
Scheduling	Passed	2.5
Chat Module	Passed	2.0
Payment Simulation	Passed	2.7
Review System	Passed	1.9

The prototype was validated by end users who provided feedback on usability and system responsiveness. Compared with traditional online tutoring platforms, GetClasses presents a more cohesive user flow and modular design, facilitating future updates and feature additions such as group sessions or advanced analytics.

IV. Conclusions

This paper presented GetClasses, an object-oriented transactional platform that demonstrates the application of key OOP principles in real-world software. The system integrates design artifacts such as UML diagrams, CRC cards, and user stories, resulting in a modular, maintainable, and scalable solution. The use of layered architecture and SOLID principles ensures clear separation of responsibilities and system extensibility.

Future work includes implementing database persistence, adding AI-driven tutor matching, and deploying the system to a web environment. This project serves as an educational example of structured software design and development following IEEE documentation standards.

Acknowledgments

The authors acknowledge the guidance of Eng. Carlos Andrés Sierra, M.Sc., during the Object-Oriented Programming course at Universidad Distrital Francisco José de Caldas.

References

[1] I. Sommerville, *Software Engineering*, 10th ed., Pearson, 2015.
[2] K. Beck and M. Fowler, *Planning Extreme Programming*, Addison-Wesley, 2000.
[3] S. W. Ambler, *Agile Modeling: UML Diagrams and Class Design*, AgileModeling.com, 2023.
[4] IEEE Standards Association, *Guide to Software Design Documentation (IEEE 1016-2020)*, 2023.
[5] P. Coad and E. Yourdon, *Object-Oriented Design*, Prentice Hall, 1991.
[6] Overleaf, *LaTeX tutorial: Learn LaTeX step by step*, 2024.
[7] Visual Paradigm, *CRC Cards Tutorial and Examples*, 2023.