Presentation of

GETCLASSES

Authors:
Jhon Gonzalez
Alejandro Escobar
Sebastian Zambrano

Índice

Introduction

The GetClasses project is an online tutoring platform that connects students with qualified tutors in a secure, intuitive, and scalable environment.

Its main goal is to optimize the processes of registration and communication between users, providing a smooth and reliable learning experience.

This project arises from the growing demand for accessible and trustworthy remote learning solutions.

Project Objetive

GetClasses aims to create an academic ecosystem where students can easily find tutors by subject, price, or location, while tutors can efficiently manage their profiles and earnings.
From a technical perspective, the system applies object-oriented programming principles such as inheritance, polymorphism, and encapsulation, ensuring that the software is modular, maintainable, and reusable.

- The main objectives include:
- Helping students find, book, and attend virtual tutoring sessions easily.
- Allowing tutors to manage their availability and earnings.
- Guaranteeing data security, reliability, and a smooth user experience across desktop environment.
- Demonstrating the practical application of OOP concepts in a real-world transactional system.

System Architecture

In GetClasses, the system's main operations are driven by user interactions. Students can create accounts, search for tutors and contact with them through integrated modules. Tutors, in turn, manage their profiles and interact with students through chat and virtual sessions. The administrator ensures compliance and resolves disputes.

The data flow of the platform is summarized as follows: users enter their data through GUI forms, which are processed and validated by the backend. Once authenticated, the system stores user information. Notifications are automatically triggered to keep users informed of bookings and reviews.

User Story

ID	User Story	Priority	Effort (hrs)	Acceptance Criteria
1	**Tutor Registration**	High	6	The tutor account is successfully created and the user is logged into the platform.
2	**Tutor Profile Picture**	Medium	3	The uploaded image appears correctly on the tutor's profile page.
3	**Tutor Search**	High	8	Only tutors matching the selected filters (subject, rate, language) are displayed in results.
4	**Tutor Listing**	High	5	The system displays a complete list of tutors showing name, subjects, and average rating.
5	**Tutor Rates Student**	Medium	3	The rating and review are recorded and associated with the student's profile.
6	**Student Rates Tutor**	Medium	3	The rating is recorded, the tutor's average is updated, and the comment is visible.

Crc Cards

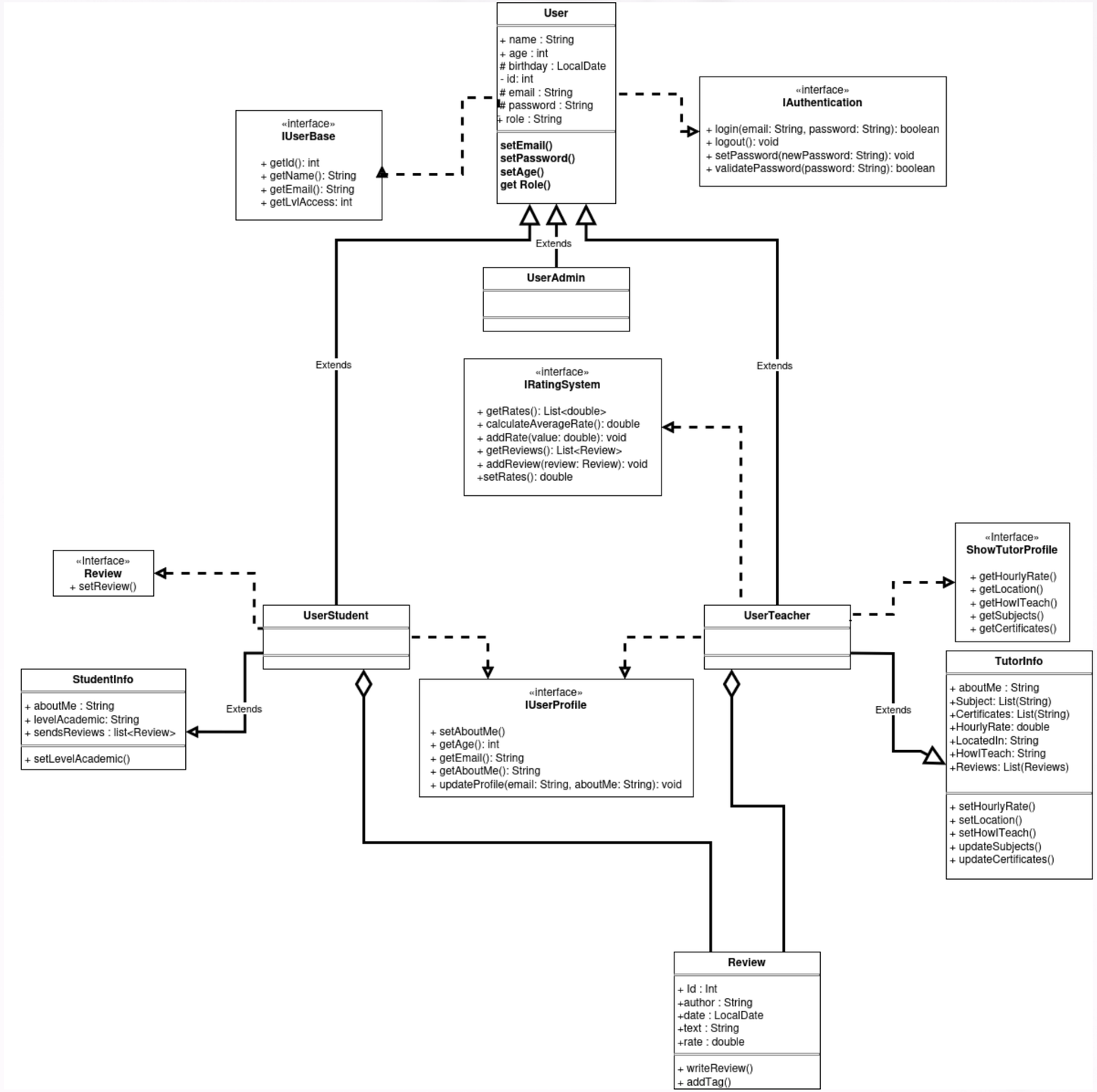
Class	Responsibility	Collaborators
User	Representing the general data and behaviors of a user within the system (e.g., ID, Email, Authentication).	UserAdmin UserStudent UserTeacher IAuthentication IDataBase
TutorInfo	Manage and store the specific data that only the tutor should have (e.g., Hourly Rate, Certificates, Subject/Areas).	UserTeacher IBillingSystem IShowTutorProfile
StudentInfo	Manage and store the specific data that only the student should have (e.g., Academic Level, List of Reviews sent).	UserStudent Review
Review	Represent a specific rating instance; store the review details (author, date, text, rate).	IBillingSystem UserTeacher UserStudent

Technical Design (UML Diagram)

The following UML class diagram represents the conceptual structure of the GetClasses system. It illustrates inheritance relationships, attributes, methods, and associations between main entities.

The diagram highlights class hierarchies where the base class User acts as the parent of specialized subclasses StudentUser and TutorUser. Each subclass extends the base behavior to meet specific user needs, demonstrating OOP principles such as inheritance, polymorphism, and encapsulation.

Technical Design (UML Diagram)



Requirements Documentation

Functional Requirements

1. Registration and login for students and tutors.
2. Profile management including experience, degrees, and rate.
3. Tutor search with filters by subject, rate, language, and country.
4. Virtual classes with video call, screen sharing, and whiteboard.
5. Reviews and ratings after sessions.
6. Dispute management by the administrator.

Non-Functional Requirements

- ·Performance: The system must respond to search queries in less than two seconds to ensure smooth navigation and usability. It should support at least 25 concurrent users without degradation in performance.
- Usability: A clean and intuitive interface accessible from Desktop Platform (JavaFX)
- Security: Data encryption, credential protection, and tutor identity verification to ensure user trust and compliance with PCI DSS standards.
- Availability: 99% uptime is required for service
- reliability, supported by automated daily backups and error recovery systems.
- Flexibility: The architecture should allow integration of future features without major structural changes.

Conclusions

The project successfully demonstrated that a layered monolithic architecture with JavaFX is a robust solution for academic environments, minimizing infrastructure costs while maintaining high performance.While the current desktop version fulfills immediate needs, the modular design allows for a smooth transition to a distributed web architecture and database integration in future development phases.

THANKS