# Technical Report: GetClasses System

*An Object-Oriented Transactional Application for Online Tutoring (Work in Progress)*

**Authors:**

Jhon Gonzalez, Alejandro Escobar, Sebastián Zambrano

Computer Engineering Program
Universidad Distrital Francisco José de Caldas

**Instructor:** Eng. Carlos Andrés Sierra, M.Sc.

**Date:** October 2025

**Abstract**

This technical report presents the initial version of GetClasses, an object-oriented transactional system designed to connect students and tutors through a secure, modular, and scalable software architecture. The system is being developed in Java using JavaFX as the graphical interface framework and follows a layered architecture with SOLID principles. At this stage, the project includes preliminary implementations of user registration, scheduling management, and file-based persistence. Future work will extend the system with communication, payment simulation, and review functionalities. The document details the current progress, design structure, and the roadmap for completing pending modules.

# Contents

**1 Introduction** **4**

**2 Literature Review** **5**

**3 Background** **6**

**4 Objectives** **7**
4.1 General Objective . . . . . 7
4.2 Specific Objectives . . . . . 7

**5 Scope** **8**

**6 Assumptions and Limitations** **9**
6.1 Assumptions . . . . . 9
6.2 Limitations . . . . . 9

**7 Methodology** **10**
7.1 System Architecture . . . . . 10
7.2 Class Design . . . . . 11

**8 Preliminary Results** **12**

**9 Discussion** **13**

**10 Future Work and Conclusions** **14**
10.1 Future Work . . . . . 14
10.2 Conclusions . . . . . 14

**A Mockups and Interfaces** **16**
A.1 Mockups . . . . . 16

**B Glossary** **21**

# List of Figures

7.1 UML class diagram showing inheritance and associations. . . . . 11

A.1 Login Page . . . . . 16
A.2 Register Page . . . . . 17
A.3 Main Page . . . . . 18
A.4 tutor profile Page . . . . . 18
A.5 Student Profile Page . . . . . 19
A.6 Review List Page . . . . . 19
A.7 Make Review page . . . . . 20
A.8 Chat Page . . . . . 20

# List of Tables

8.1 Preliminary testing summary . . . . . 12

# Chapter 1

# Introduction

The growing demand for flexible and remote education has motivated the development of systems that facilitate personalized tutoring experiences. GetClasses aims to provide a digital environment where students can connect with tutors, schedule sessions, and exchange information efficiently.

Currently, many existing platforms—such as Coursera or Udemy—focus on course distribution rather than real-time tutoring interactions. The objective of GetClasses is to fill this gap through a transactional system that integrates user management, scheduling, and communication tools.

This report summarizes the current state of the project, the applied methodologies, and the components that are still under development.

# Chapter 2

# Literature Review

Research on online education platforms highlights challenges such as scalability, security, and personalized user experiences. While established platforms manage large amounts of data efficiently, they often lack direct tutor-student interaction. Studies in software engineering emphasize that object-oriented programming (OOP), layered architecture, and SOLID principles are effective strategies for building modular and maintainable educational software systems [1, 2, 3].

The GetClasses project is based on these methodologies, combining theoretical foundations with a practical software prototype.

# Chapter 3

# Background

This project originated from the Object-Oriented Programming course at Universidad Distrital Francisco José de Caldas. It consolidates concepts from several academic workshops focused on conceptual design, UML modeling, inheritance and polymorphism, and SOLID-based architecture refinement.

The project uses JavaFX to develop the user interface and file-based persistence for data storage. Future iterations are expected to include a database-driven persistence layer, enhanced authentication, and integrated messaging features.

# Chapter 4

# Objectives

## 4.1 General Objective

To design and progressively implement a transactional application connecting tutors and students using object-oriented programming principles and layered software architecture.

## 4.2 Specific Objectives

- Apply OOP principles such as encapsulation, inheritance, and polymorphism.
- Design a clear and scalable class hierarchy following SOLID principles.
- Implement a JavaFX-based interface for registration and session scheduling.
- Develop file-based data persistence as an initial storage mechanism.
- Evaluate system functionality through incremental testing phases.

# Chapter 5

# Scope

At this stage, GetClasses focuses on the implementation of core structural components, including:

- User class hierarchy (Student, Tutor, Admin).
- Basic registration and login process.
- Preliminary scheduling logic and persistence prototype.

Functionalities such as chat, payment simulation, and user reviews are planned for future versions. The current version serves as a foundation for expansion and future deployment as a full tutoring platform.

# Chapter 6

# Assumptions and Limitations

## 6.1 Assumptions

- Users are familiar with basic digital platforms.
- Tutors provide accurate subject and availability data.
- Local file storage is consistent between sessions.

## 6.2 Limitations

- Some modules are still in early stages of development.
- Communication and payment systems are not yet integrated.
- Testing results are preliminary and limited to isolated functions.
- File-based persistence restricts scalability and data integrity.

# Chapter 7

# Methodology

The project follows an iterative and incremental software development process, with continuous feedback and improvement. Each cycle includes design refinement, coding, and functional testing of individual components.

## 7.1 System Architecture

The GetClasses architecture is organized into three layers:

- **Presentation Layer:** Built with JavaFX, it manages user interaction and visualization.
- **Business Logic Layer:** Contains service classes for registration, scheduling, and system control.
- **Data Layer:** Manages file serialization and data retrieval operations.

## 7.2 Class Design

The conceptual model defines a superclass `User`, extended by subclasses `StudentUser`, `TutorUser`, and `AdminUser`. Each subclass introduces role-specific behavior, maintaining encapsulation and reusability.

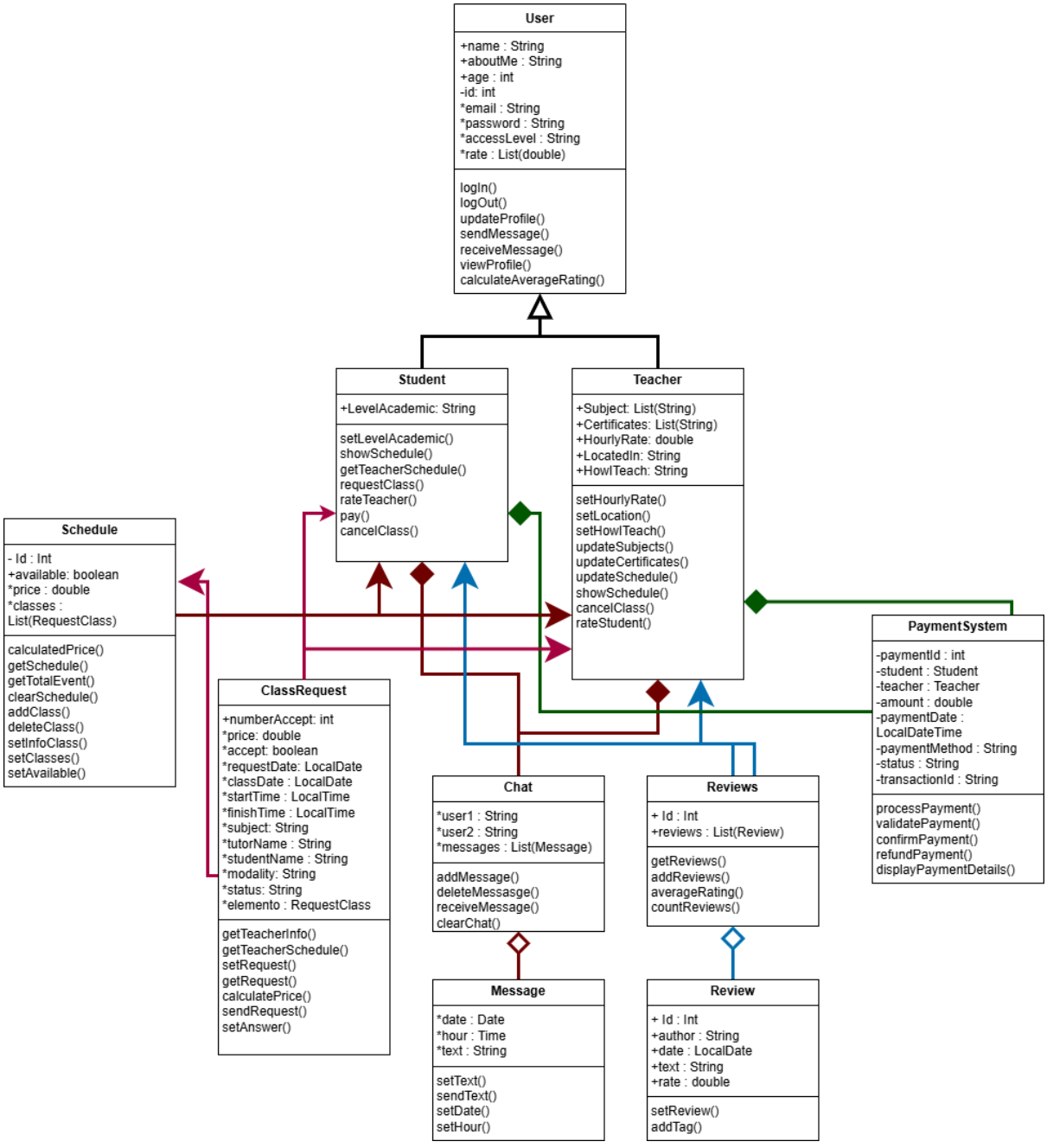


Figure 7.1: UML class diagram showing inheritance and associations.

# Chapter 8

# Preliminary Results

At this stage, only partial testing has been conducted on user registration and file-based persistence. The following table summarizes the preliminary outcomes:

Table 8.1: Preliminary testing summary

| **Test Module** | **Status** | **Description** |
|---|---|---|
| User Registration | In Progress | Users can register and store data in files. |
| Login Function | In Progress | Partial functionality; missing validation layer. |
| Review System | In Progress | Initial structure created; functionality under development. |
| Scheduling System | In Progress | To be implemented in the next iteration. |
| Chat Module | Pending | Not yet developed. |
| Payment Simulation | Pending | Planned for final stage. |

Initial results indicate that the system correctly stores and retrieves user data using file serialization. No major errors have been detected in these early modules.

# Chapter 9

# Discussion

The development of GetClasses demonstrates how object-oriented principles can be applied progressively in a real-world educational context. Even though several modules remain under construction, the current design ensures modularity, maintainability, and future scalability.

Challenges include managing dependencies between layers and implementing a more efficient persistence model. Once database integration and user communication are completed, the system will achieve a more realistic and functional structure.

# Chapter 10

# Future Work and Conclusions

## 10.1 Future Work

- Implement full scheduling, chat, and payment modules.
- Replace file-based persistence with a relational or NoSQL database.
- Add user interface improvements using advanced JavaFX components.
- Conduct complete unit and integration testing.
- Explore web or mobile deployment options.

## 10.2 Conclusions

Although GetClasses is still under active development, the project demonstrates solid progress in design and foundational architecture. It successfully applies OOP and SOLID principles to create a maintainable software structure. Future implementations will expand its features and provide a comprehensive platform for online tutoring.

# Bibliography


[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] P. Coad and E. Yourdon, *Object-Oriented Design*, Prentice Hall, 1991.

[5] IEEE Standards Association, *Guide to Software Design Documentation (IEEE 1016-2020)*, 2023.

# Appendix A

# Mockups and Interfaces

## A.1 Mockups

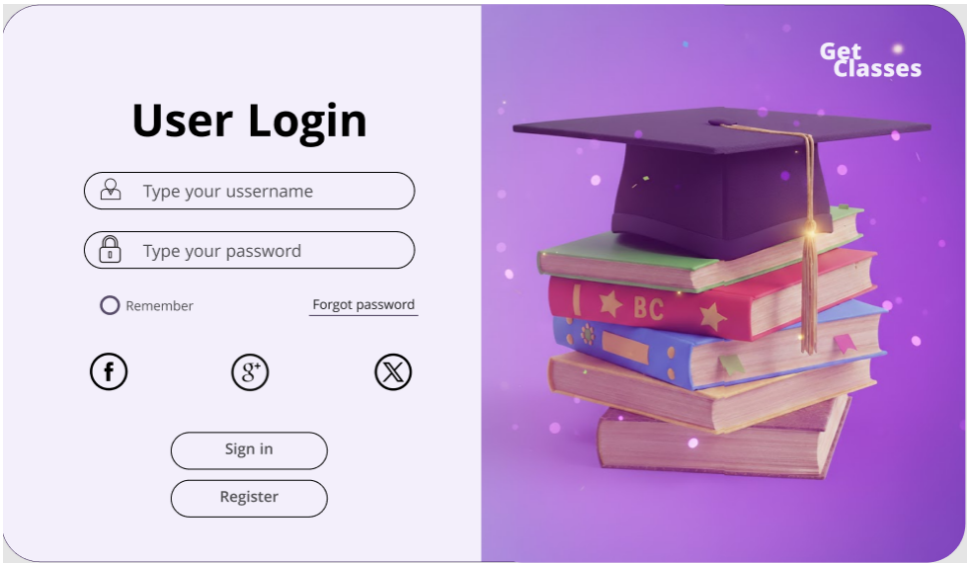


Figure A.1: Login Page

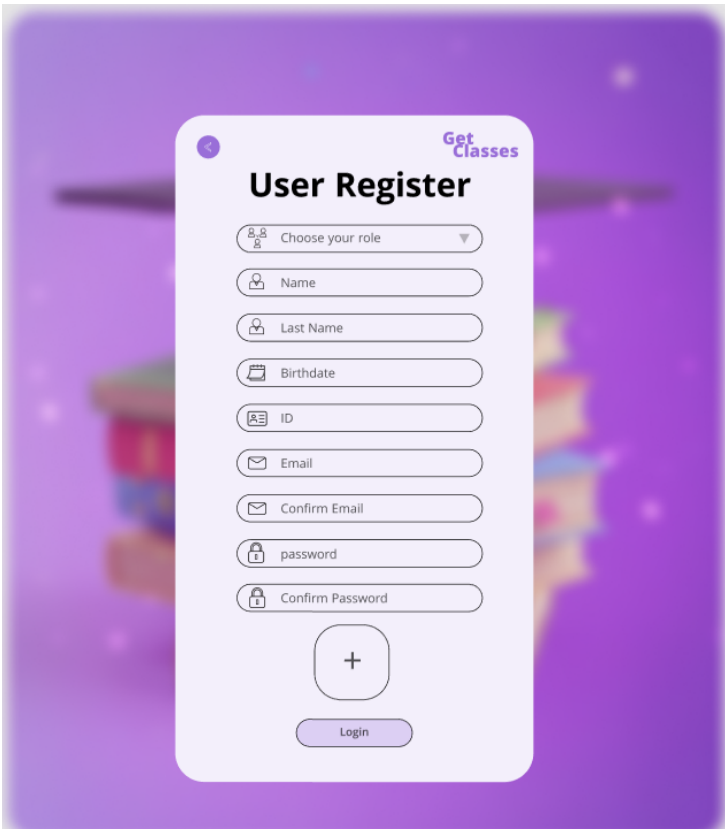


Figure A.2: Register Page

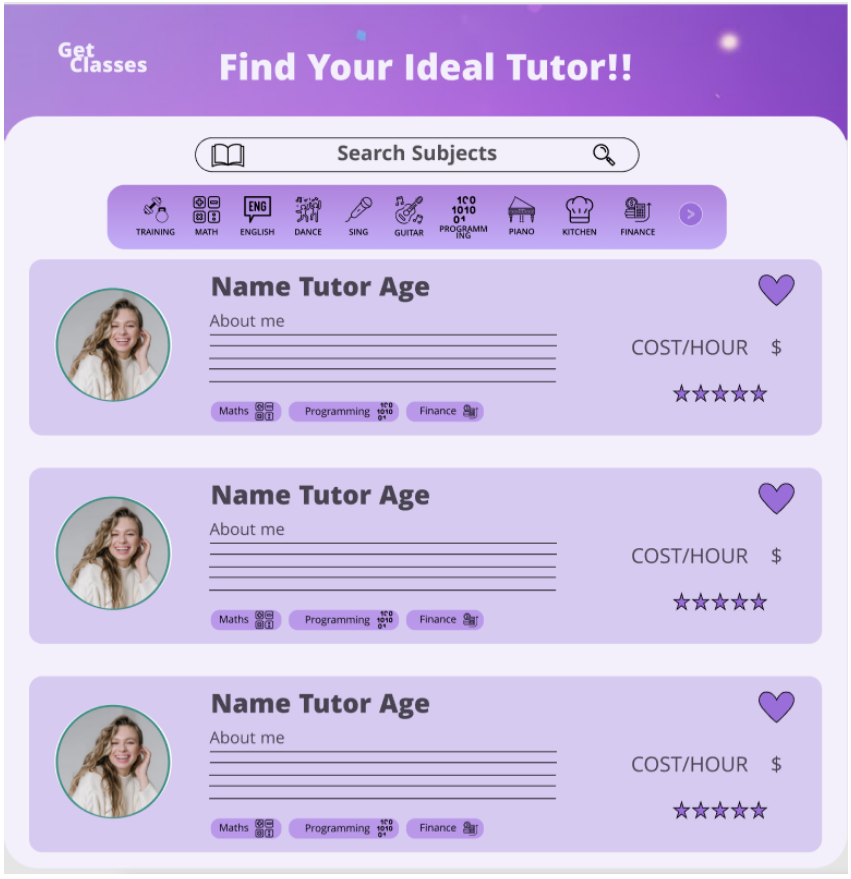


Figure A.3: Main Page

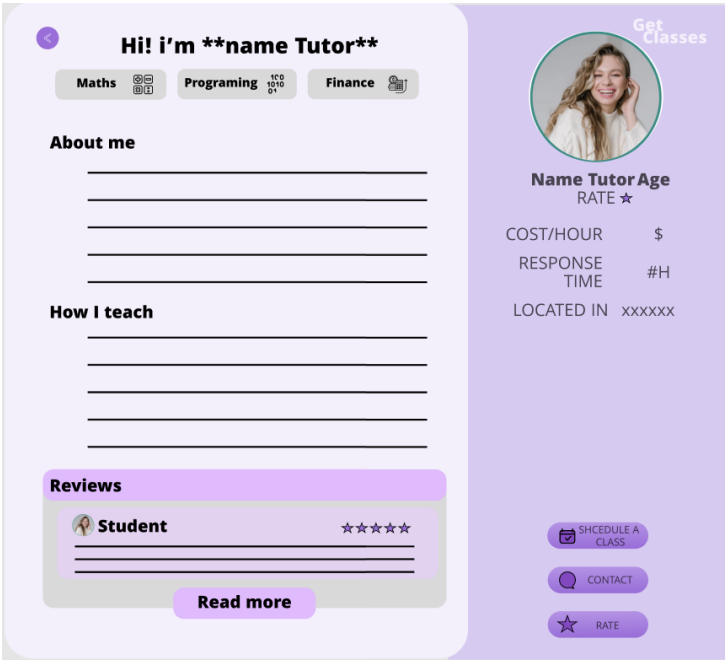


Figure A.4: tutor profile Page

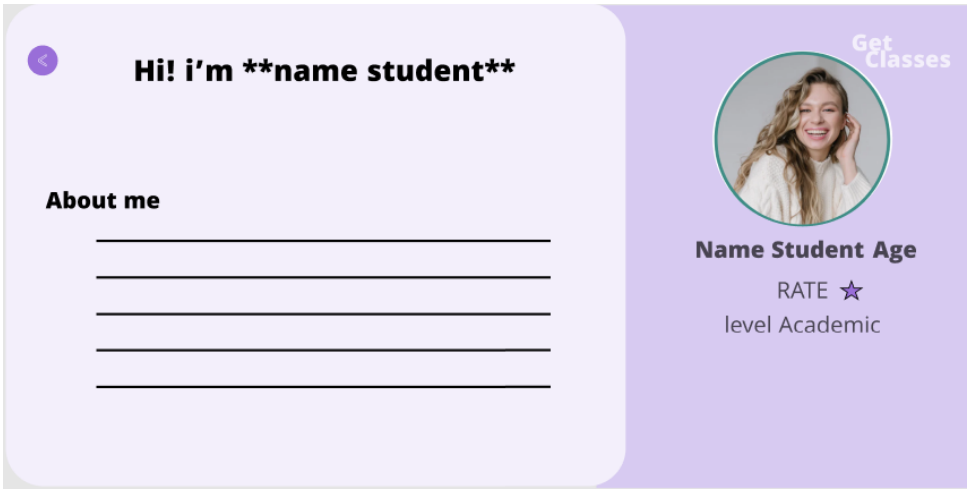


Figure A.5: Student Profile Page

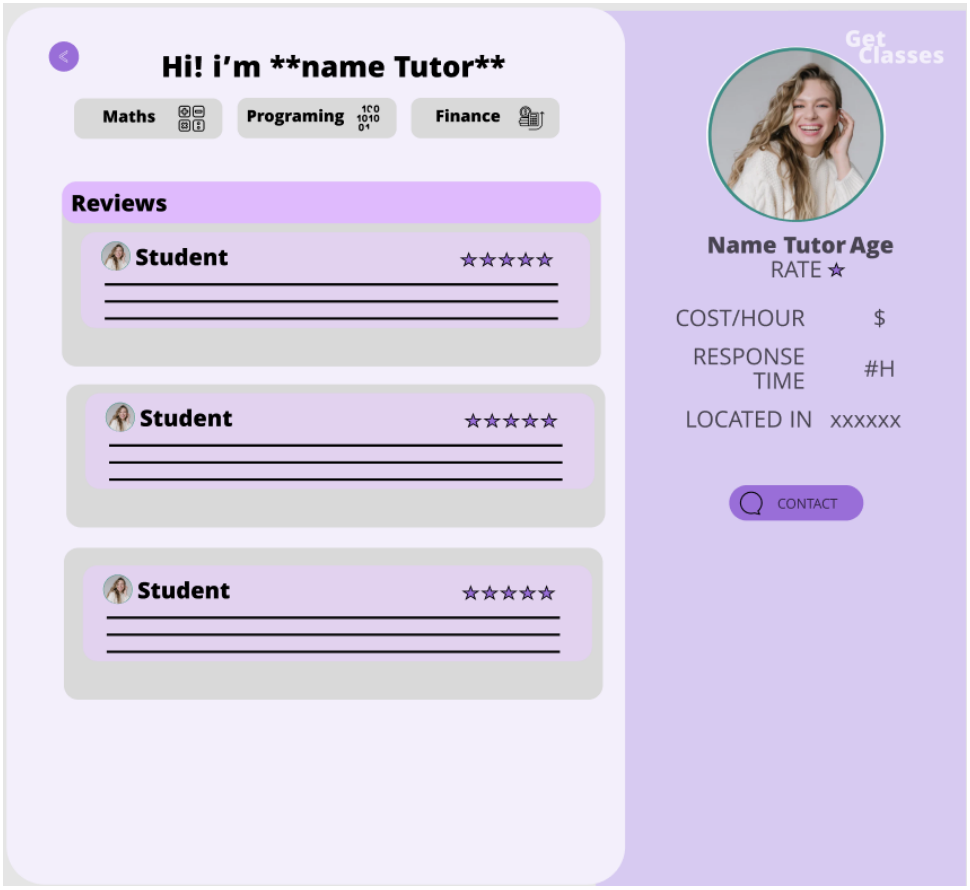


Figure A.6: Review List Page

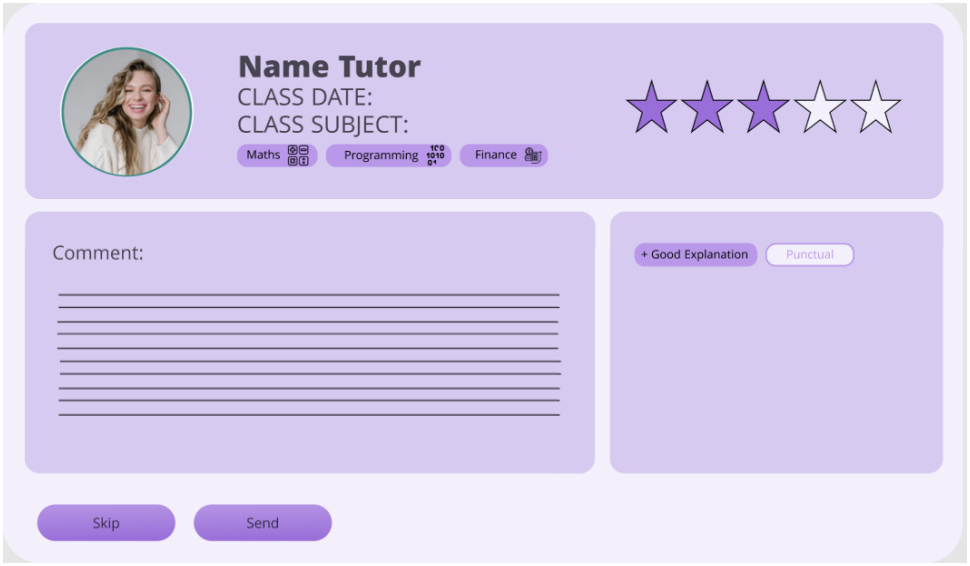


Figure A.7: Make Review page

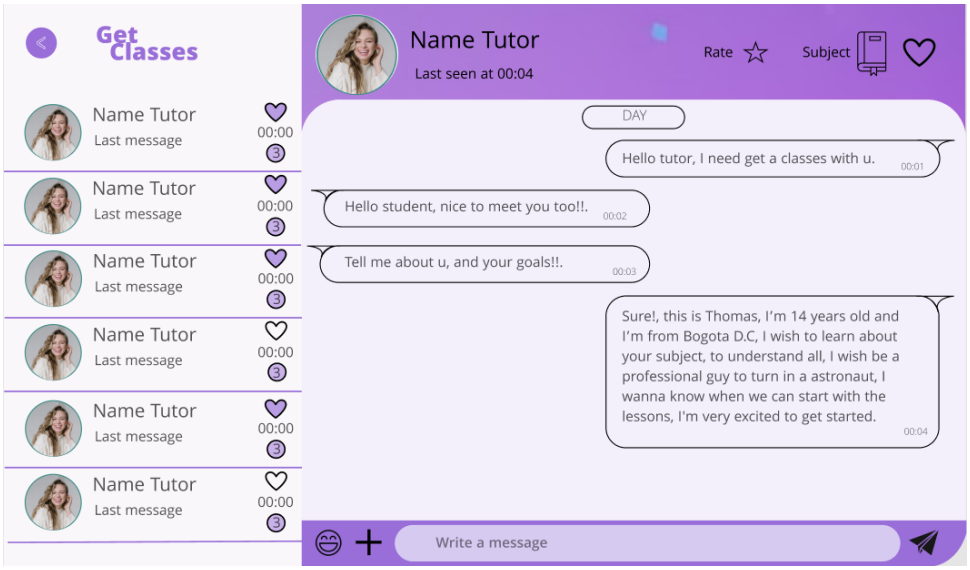


Figure A.8: Chat Page

# Appendix B

# Glossary

**OOP** Object-Oriented Programming.

**SOLID** A set of design principles for clean, maintainable code.

**UML** Unified Modeling Language for software system design.

**JavaFX** A graphical framework for building Java-based GUI applications.