Learn@Home: Use case diagrams

Prepared by: Kirill Derevenski
Project: Définissez les besoins pour une app de soutien scolaire (Project 10)
Path: Développeur d'application - JavaScript React
Date: August 18, 2022

TABLE OF CONTENTS

LIST OF DIAGRAMS

1. DOMAIN MODEL

1.1 About the domain model

The proposed domain model summaries suggested top-down architecture for the Learn@Home application.

It identifies key actors and relationships between them and the system, as well as identifies key design domains of the app.

Importantly, in the diagram below, **Core domain modules** reflect requirements captured in the original client specifications (e.g. dashboard, chat, calendar etc), while **Additional domain modules** (e.g. settings, contacts etc) propose additional design domains which are needed to implement requested functionality.

In addition, the **Administrator** role has not been mentioned in the client specification but is needed in a software system. This actor and associated functionality has been reflected in the domain model below.

Note that this document does not contain use cases either for the additional domain models or for the administrator role.

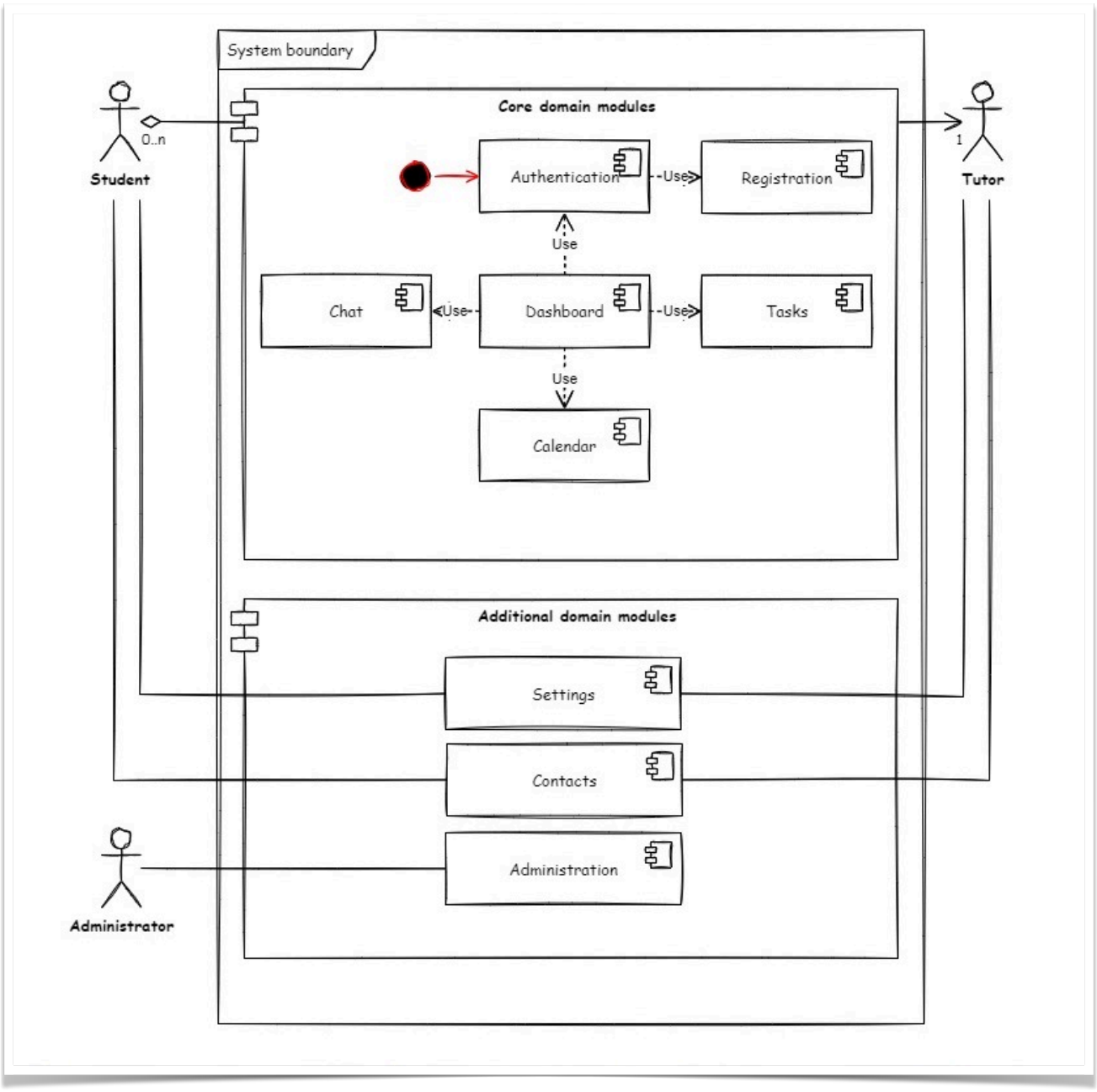

Diagram 1. Domain model

2. USE CASES

2.1 Authentication

This use case represents the point of actors' entry into the application.

In the diagram below, two additional use cases are specified (i.e. Verify user and Send email) to better illustrate proposed functionality.

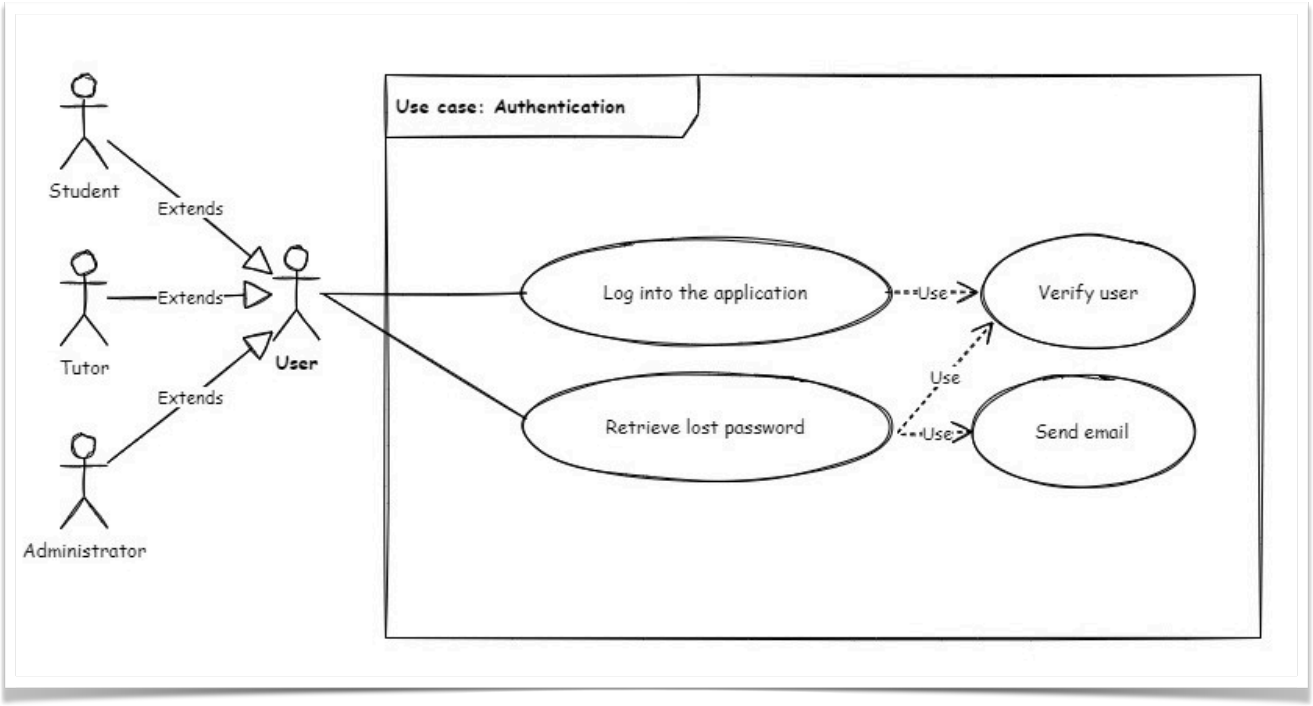

Diagram 2. Use case: Authentication

2.2 Registration

This use case supplements authentication.

In the diagram below, two additional use cases are specified (I.e. Verify user and Send email) to better illustrate proposed functionality.

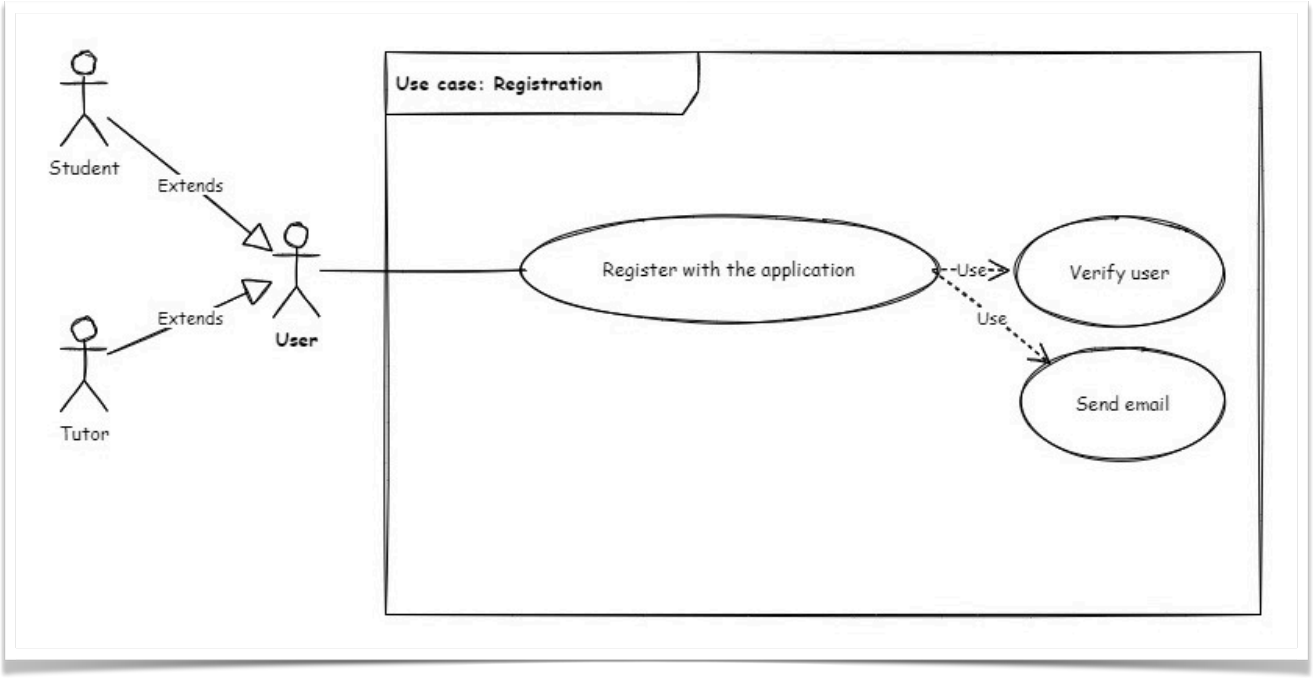

Diagram 3. Use case: Registration

2.3 Dashboard

This use case represents user dashboard and a number of other components on which it relies for its functionality.

Note that in the proposed system, students will only be able to see own items and those created for them by their tutors, while tutors can see their own items as well as those of their students.

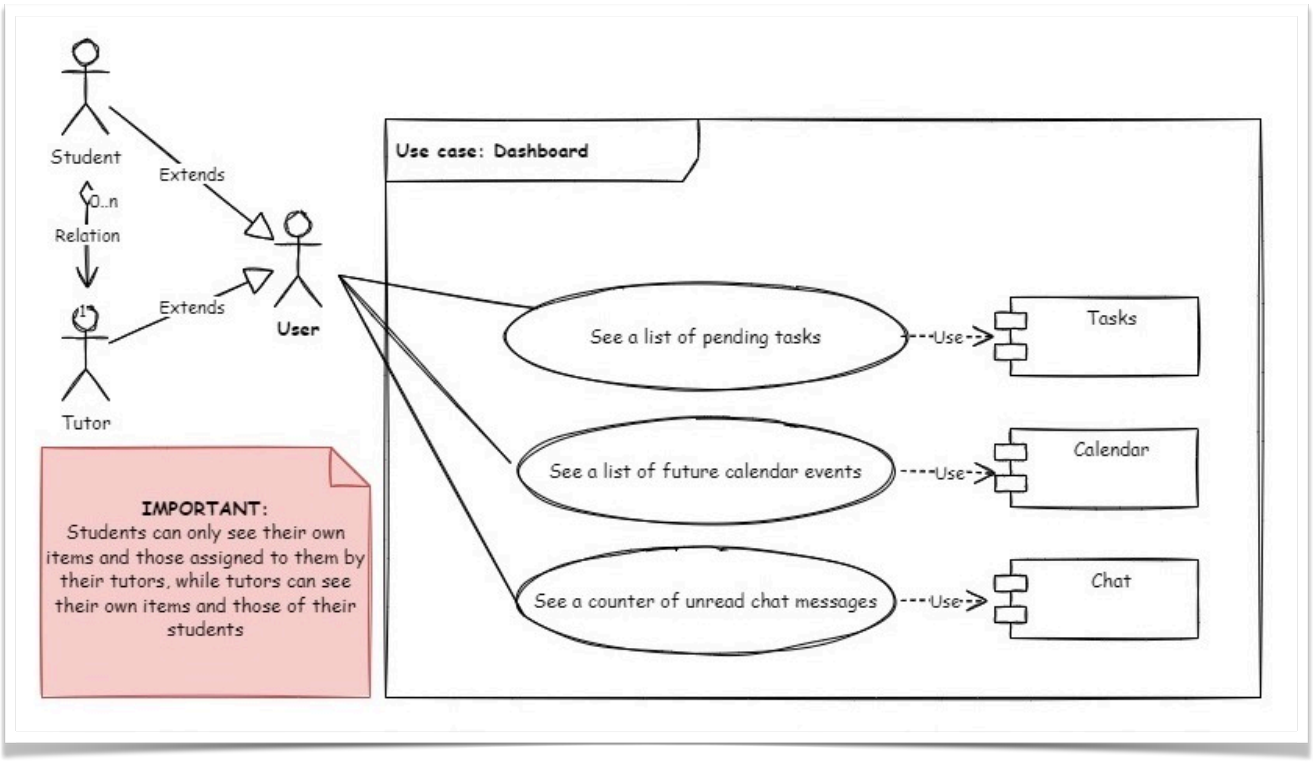

Diagram 4. Use case: Dashboard

2.4 Tasks

This use case represents management system for the tasks (todo list) functionality of the proposed application.

Note that in the proposed system, students will only be able to see own tasks and those created for them by their tutors, while tutors can see their own tasks as well as those of their students.

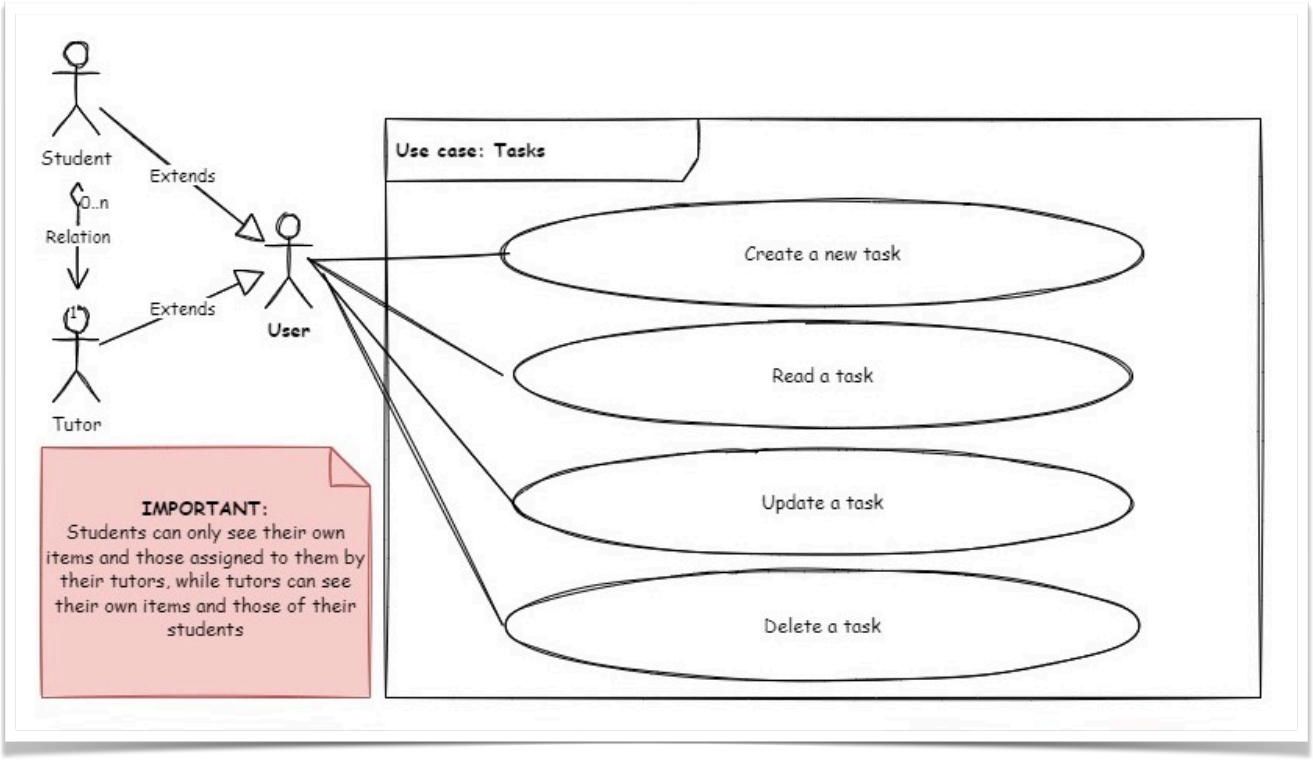

Diagram 5. Use case: Tasks

2.5 Calendar

This use case represents management system for the calendar event functionality of the proposed application.

Note that in the proposed system, students will only be able to see own events and those created for them by their tutors, while tutors can see their own events as well as those of their students.

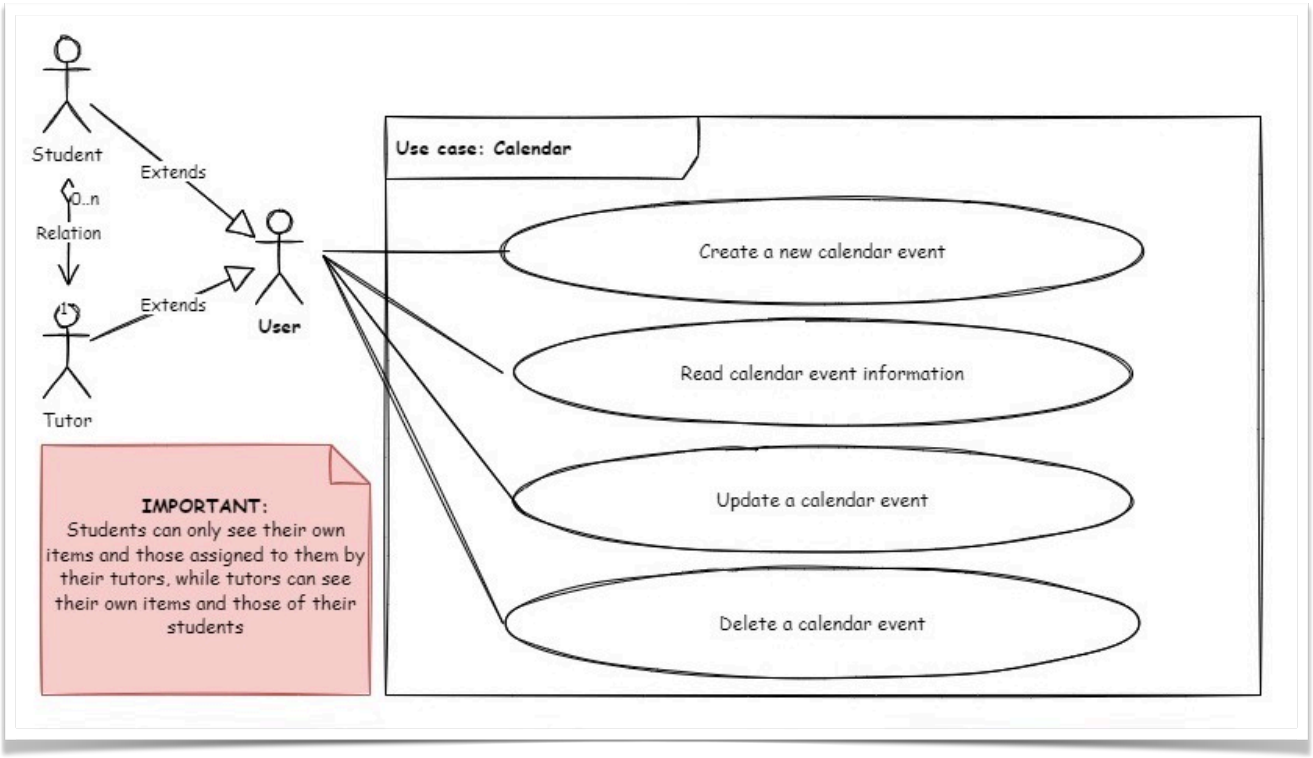

Diagram 6. Use case: Calendar

2.6 Chat

This use case represents management system for the chat functionality of the proposed application.

Note that based on the available project information, it is assumed that users will only be able to have chat conversations within their pairings (i.e. tutors with assigned students and students with assigned tutors), and not between themselves (i.e. tutors with other tutors or students with other students).

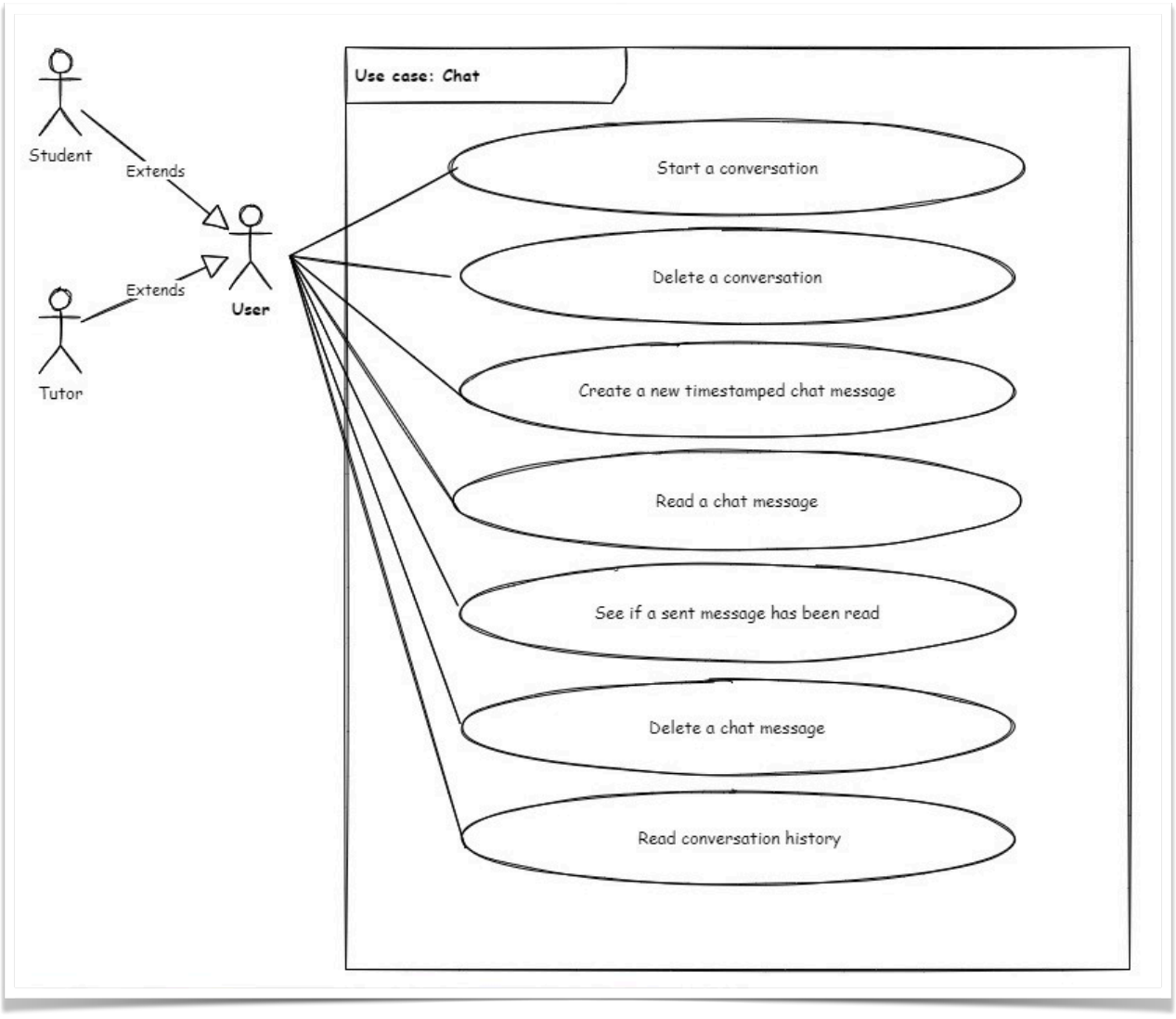

Diagram 7. Use case: Chat

3. DATABASE MODEL

3.1 About the database model

The proposed database model summaries suggested data architecture for the Learn@Home application.

Given that the proposed application contains structured data, a **relational database** is the best option for implementation. This is because we can determine from the start what attributes the database will be capturing for every use case, and there there is a finite number of them.

In the diagrams below, the conceptual data model identifies the domains (or entities) that will exist in the application data and plots the relationships between them. From here, the logical model identifies actual database tables and linkages between attributes within them.

3.2 Conceptual data model

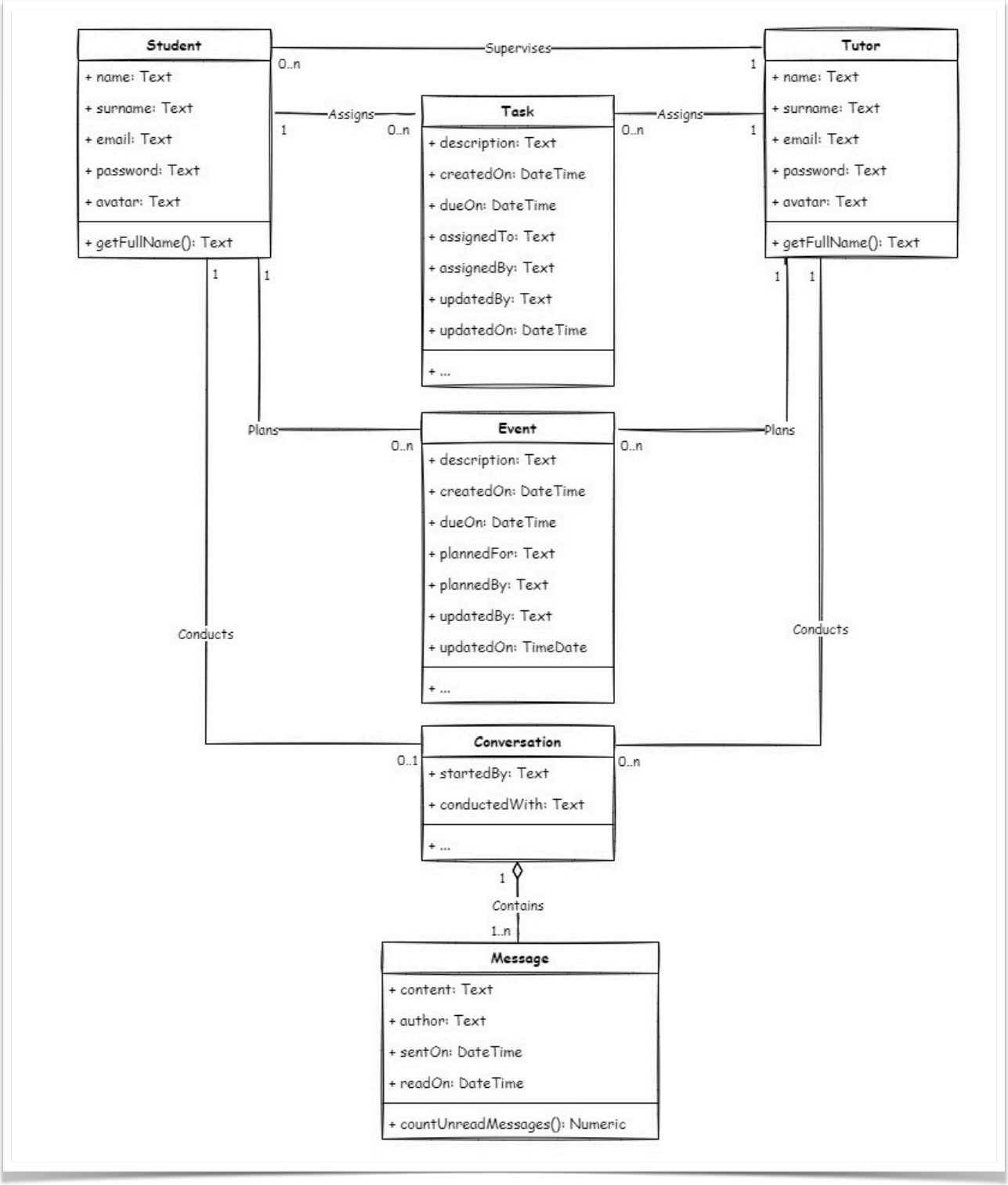

Diagram 8. Conceptual data model

3.3 Logical data model

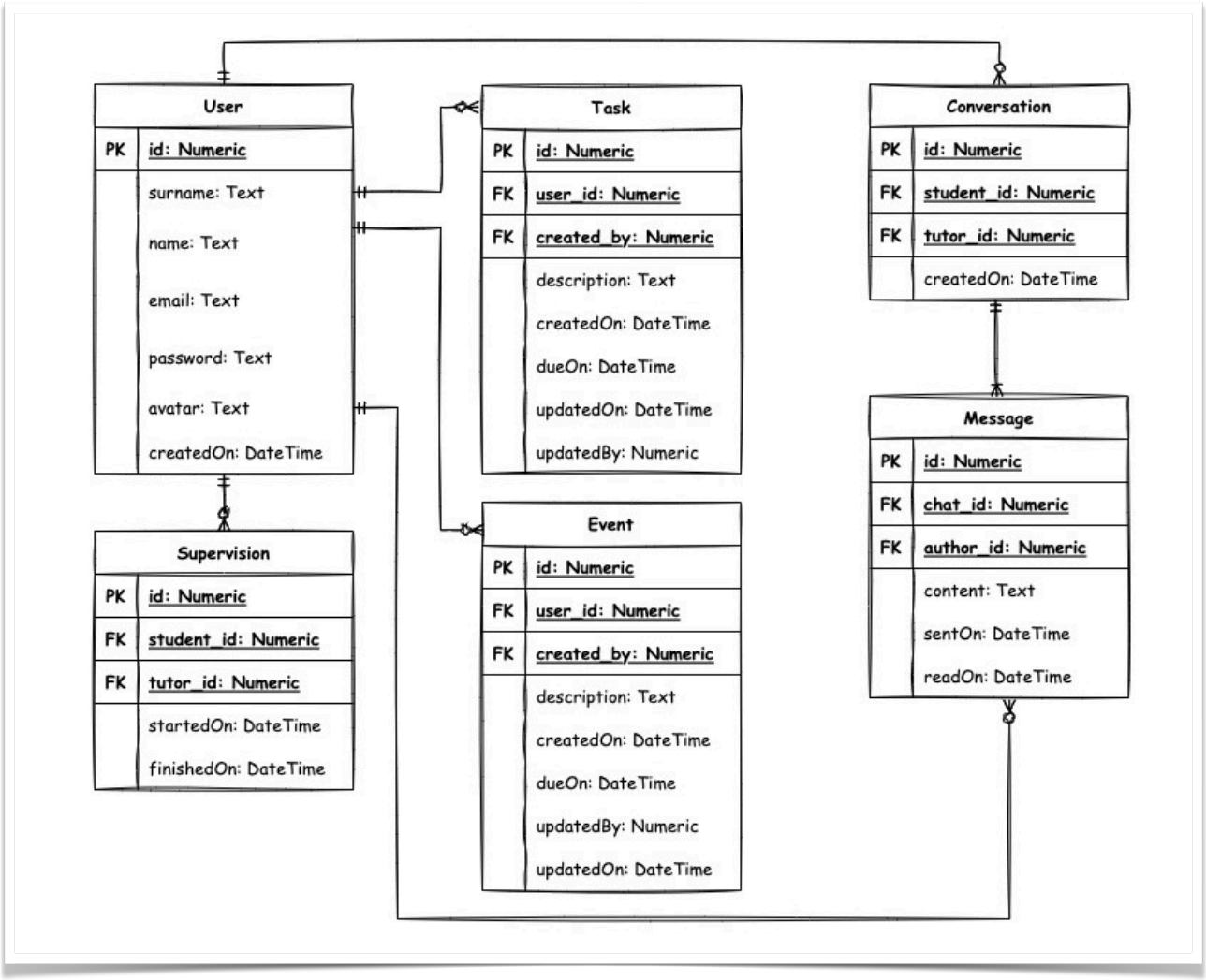

Diagram 9. Logical data model