Technical Report M2

Küng Pascal

Mettler Micha

Zehnder Jonas

Gonçalves Rafael

Thayananthan Ragavan

ZHAW-School of Engineering

IT22tb_WIN

Software Project 3

Lectured by Nina Schnatz and Kurt Bleisch

Table of Contents

List of Figures

1. Intro

At the heart of our Mathify[1]-app development for schoolchildren is a steadfast commitment to usability and user experience. From the outset, we pondered how to ensure and enhance effectiveness, efficiency and user satisfaction. In terms of usability, for our prototype, we focused particularly (but not exclusively) on three of the seven requirements:

1. Suitability for the task (German: Aufgabenangemessenheit): It is crucial for children to encounter a simple interface that displays on the necessary content. Additionally, the app should require a minimal number of steps and user input for logging in and solving exercises[2], as children often lack technical proficiency and should not be overwhelmed by unnecessary hurdles.
2. Conformity with user expectations (German: Erwartungskonformität): It is essential that the app's terminology, behavior and task presentation are consistent. This means once a child is oriented in terms of design and interaction, they will know what to do next. This is especially important since children often read slowly or not at all and may struggle to comprehend complex terms and interactions.
3. Error tolerance (German: Fehlertoleranz): It's also vital that the app clearly communicates what input is expected, checks for incorrect entries, and assists in correcting them.

As the app development progresses and leaves the prototype status, customizability (German Individualisierbarkeit) will eventually become a focus too. We start with the school grade[3] as anchor point for exercise type, allowing for progression from basic to advanced level within the app. In a later phase, extensive customization will be available (e.g. individual types of exercises, tailoring by teachers, exam mode, school challenges, etc.).

Furthermore, learning facilitation (German: Lernförderlichkeit) becomes another priority in more advanced states too. Our goal is to provide, upon user request, information about underlying mathematical concepts and easy access to tooltips as well as further tutorials, including videos, enabling them to learn complex concepts while using the app.

[1] See Glossary for a definition of this term.
[2] See Glossary for a definition of this term.
[3] See Glossary for a definition of this term.

2. Use-Case-Model

In defining our use case model, we followed these steps:

2.1. Step 1: Defining System Boundaries

Defining the system boundaries is an essential initial step in the development of Mathify, as it is instrumental in establishing the scope and objectives of the project. By defining Mathify itself as the system boundary, we can concentrate on crafting pertinent use cases, delineating clearly the functions and actors that fall within the remit of Mathify and those that do not. This clarity not only promotes the efficient utilization of resources but also facilitates transparent communication with stakeholders.

2.2. Step 2: Identifying the Actors, with a focus on Primary Actors

Primary Actor

A primary actor initiates a use case to achieve a specific goal and, as a result, gains the principal benefit. In our scenario, these include:

1. Child[4]
2. Teacher

Supporting Actor

A supporting actor aids the system in handling a use case. In our scenario, these include:

1. App-Team[5]
2. Payment Service

Offstage Actor

An offstage actor is an additional stakeholder who does not directly interact with the system. In our scenario, these include:

1. School
2. Department of Education
3. Tax Department
4. Parents

[4] See Glossary for a definition of this term.
[5] See Glossary for a definition of this term.

System Boundary Diagram

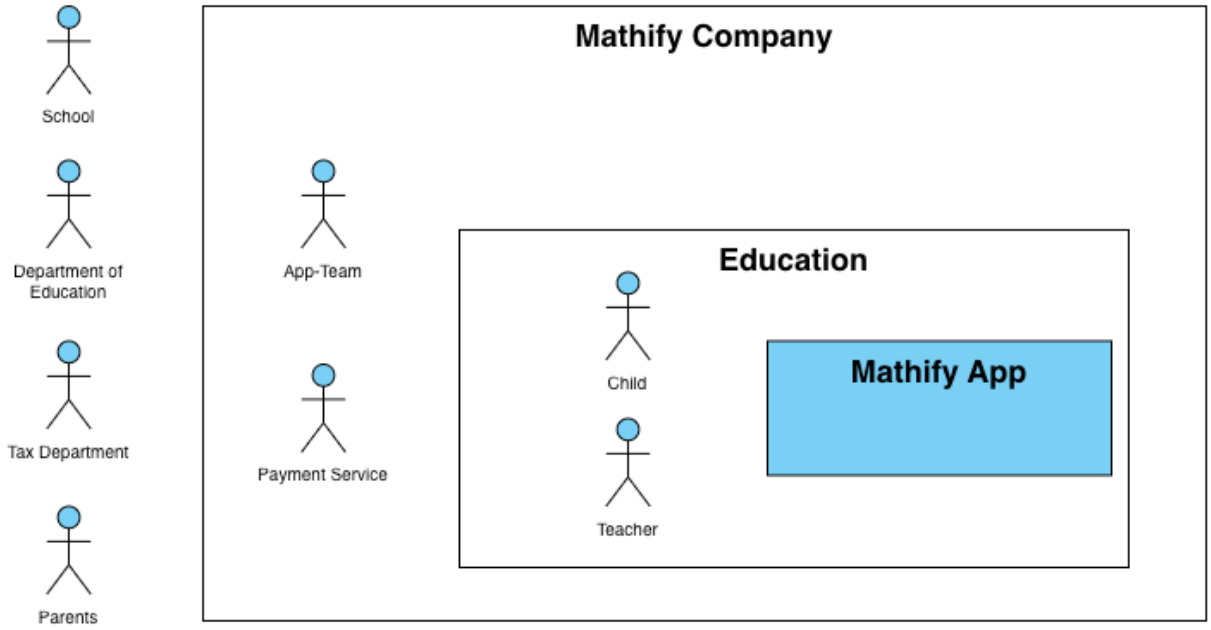


Figure 1: System Boundary Diagram

2.3. Step 3: Identifying Goals and Tasks of Primary Actors

Tasks

- Solving math problems (child)
- Competing with others to improve and feel good (child)
- Creating exercises and mock exams (teacher)

Goals

- Enhancing math skills (child)
- Building self-esteem (child)
- Transferring knowledge (teacher)

2.4. Step 4: Definition of Use Cases

2.4.1. Register / Login of Child (Casual UC)

- Upon their first login, a child is prompted to enter their username, password, mail address, class and payment details. If the child already has an account, they need to input their username and password. The system verifies all the provided

information. In case of three failed logins, the child can reset the password via mail address.
- After successful registration or login, the child selects a game mode[6], and the system determines possible game statuses.
- The system advances to the exercise screen, where it generates exercises based on the inputs.

2.4.2. Solve Math Exercise (Fully-Dressed UC)

Use Case Name:

- Solve Math Exercise

Scope:

- Mathematics Quiz

Primary Actor:

- Child

Stakeholders and Interest:

- Parents
  - Aim to support their children.
  - Desire to track their children's learning progress.

- Schools and Education Department
  - Aim to elevate educational standards nationwide.

- Tax Authority
  - Intends to collect tax revenues.

[6] See Glossary for a definition of this term.

Preconditions:

- The child must be logged in correctly.

Success Guarantee / Postconditions:

- Exercises are generated accurately.
- Submissions are validated effectively.
- Scores are calculated correctly and updated accordingly.

Standard Process:

1. The system generates an exercise.
2. The child completes the exercise.
3. The system validates the child's submission.
4. The system verifies the accuracy of the result.
5. The system calculates and records the score.
6. The child repeats steps 1-5 to solve additional exercises.
7. The child views the scoreboard[7].

Extensions / Alternate Flows:

<u>If the child solves an exercise incorrectly:</u>

1. The system alerts the child to the incorrect result.
2. The correct answer is displayed.
3. A link to learning material is provided.
4. The score is adjusted downward.

<u>If the child's answer is in the wrong format (e.g. a word instead of a number):</u>

1. The system generates an error message.
2. The system instructs the child on the correct format.
3. The child is prompted to re-enter their answer.

[7] See Glossary for a definition of this term.

Special Requirements:

1. Text must be legible from 0.5 meters away.
2. The system must process validations within 2 seconds.
3. The level of difficulty[8] must increase if the child consistently solves exercises correctly.

List of Technical and Data Variations:

- 2a) Input using a standard keyboard.
- 2b) Input using an on-screen keyboard in the browser.

Frequency of Occurrence:

- Whenever the child wishes to practice, which is the application's primary use case.

Open Questions:

- Which payment system is to be implemented?
- How many exercise types from the syllabus will the program incorporate?

2.4.3. Gather additional Math Knowledge (Casual UC)

- After scoring poorly on an exercise, the system recommends additional learning material. A child accesses the tutorial feature to gain additional knowledge. A child selects appropriate learning materials from a teaching library[9].
- Alternatively, the teacher provides material to the children via app.
- Once confident with the topic, the child returns to the exercise screen to attempt the exercise types they previously struggled with.

[8] See Glossary for a definition of this term.
[9] See Glossary for a definition of this term.

2.4.4. Calculate / Display Score (Casual UC)

- The system updates the child's score, awarding double points for each correct answer compared to a wrong one. Following each exercise, the scoreboard entry is updated.
- Upon reaching a specified threshold, the scoreboard is displayed, and a new level is unlocked, introducing more challenging exercises.
- The child can then choose to progress to the new level or opt to continue at the current level. After making their selection, the system reverts to exercise mode.

2.4.5. Create individual Exercises / Tutorials (Casual UC)

- The teacher creates exercises or learning materials tailored to the class and uploads them to the app. The application checks the format and file size for compatibility.
- The teacher assigns the material to a specific topic and difficulty level or creates new topics altogether.
- The new content is then made available in the child's version of the app.

2.5. Step 5: Draw Use-Case-Diagram

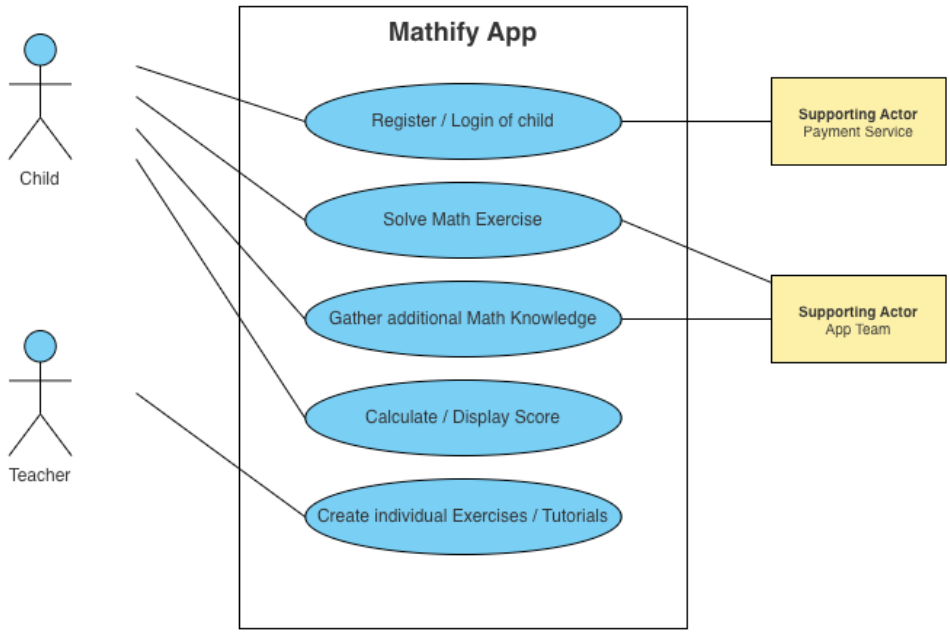
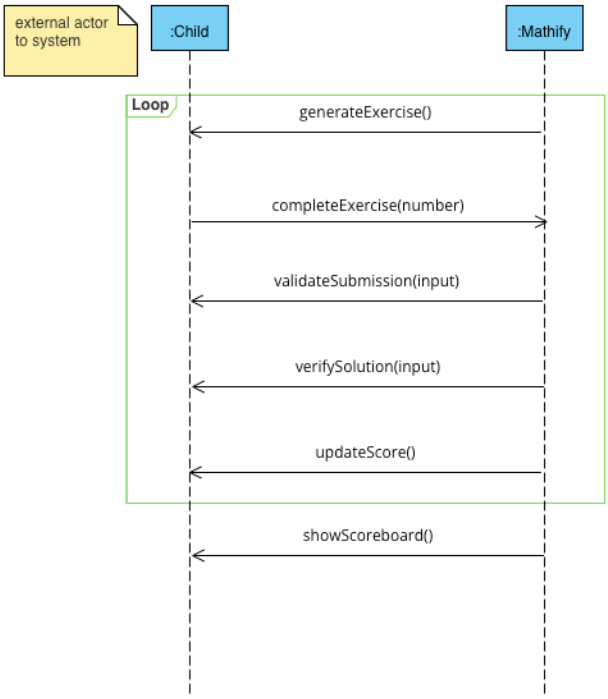


Figure 2: Use-Case Diagram

2.6. Step 6: Draw a System Sequence Diagram (SSD)



Noch zu erledigen kommende Woche:

UI Sketches für die wichtigsten Use-Cases

Evtl. Navigationsmöglichkeiten (Dialogablauf) in Diagramm

Figure 3: System Sequence Diagram of Main UC

3. Additional Requirements

The following chapter describes additional functional and non-functional requirements that were not covered in the use cases. The requirements are formulated as user stories and are all measurable / verifiable. Additionally, the team has limited itself to only stating requirements that are truly necessary without providing specific solutions how to achieve them.

Our subject expert demands that...:

1. the child requires a maximum of three clicks after logging in to reach the first exercise, as exceeding this number of clicks may overwhelm cognitive capacity.
2. a screen should never contain more than two full sentences, and a sentence should never exceed six words, as experience shows that excessive reading quickly diminishes the playful component.
3. results are savable to track learning progress together.
4. the app should have no downtime between 08.00 and 18.00 on weekdays to enable proper integration into the curriculum.

Parents demand that...:

1. the payment method is universally accepted, as no new payment medium is opened solely for using an app.

Schools demand that...:

1. changes in the national syllabus[10] or new learning insights are continuously reflected in the app, as this is the only way a usage in the educational context makes sense.

The government demands that...:

1. data security must be always ensured because sensitive data[11] is stored in the system, which makes security necessary from a regulatory point of view.

The company mathify demands that...

1. the application guarantees a traffic of 10,000 parallel users without breaking down to ensure credibility in the market and hence achieve profitability of the company.
2. The application must be compatible with common operating systems to quickly gain widespread adoption among schoolchildren and push sales figures up.

[10] See Glossary for a definition of this term.
[11] See Glossary for a definition of this term.

4. Domain Model

The domain model[12] illustrated below represents Mathify through a simplified UML class diagram. It delineates the principal concepts with interconnection to each other to visualize their relationships. Attributes have been included where they enhance understanding.

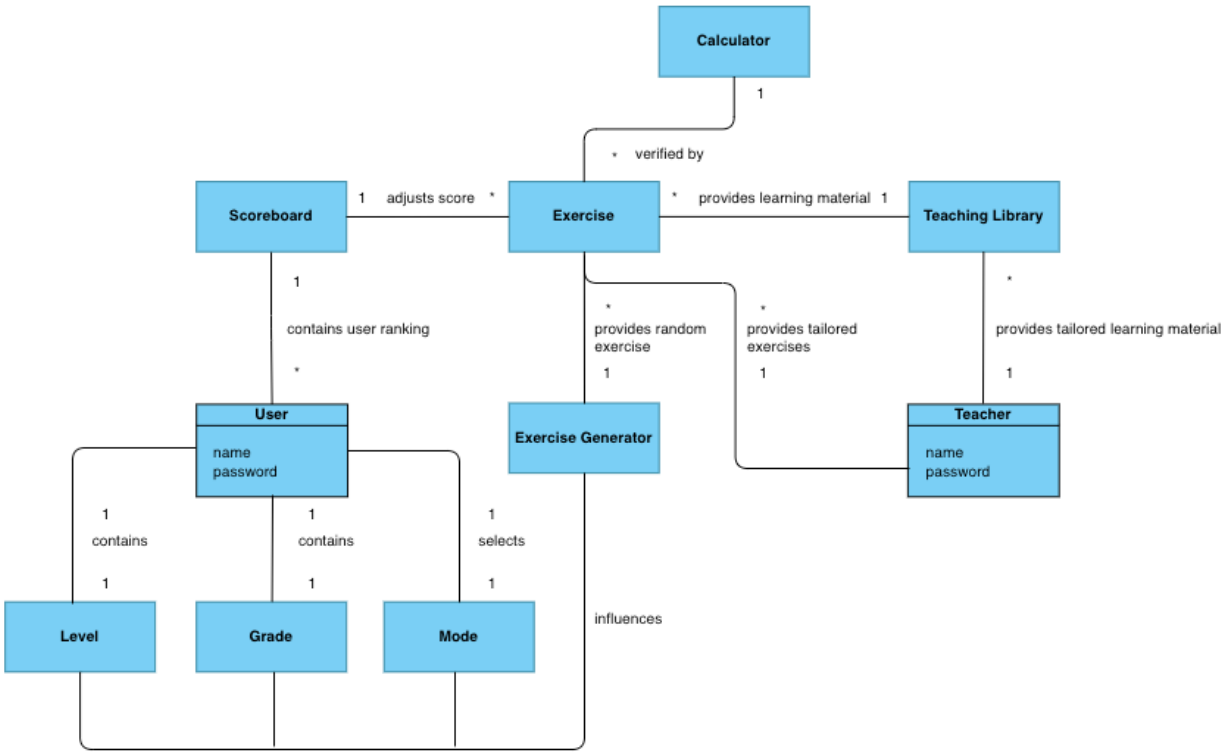


Figure 4: Domain Model

[12] See Glossary for a definition of the technical terms in the domain model.

5. Software Architecture

The decision on a specific software architecture and technology stack is a critical step in the development of a project, which has far-reaching implications for its success, maintainability, and scalability. In our case, the decision was made based on various factors such as team experience, specific project requirements, and past experiences with technologies.

5.1. Programming Languages

Java was chosen as the primary programming language, mainly because of the existing experience within the team. This decision allows the team to be immediately productive, as no additional learning time for a new language is required. Java is also known for its robustness, performance, and wide support through a variety of libraries and frameworks, making it a safe choice for our backend development.

Python will be introduced later for specialized mathematical calculations. Python offers an extensive selection of scientific libraries like NumPy and SciPy, which are ideal for mathematical tasks. This decision enables us to leverage Python's capabilities in this area without having to build our entire architecture around it.

5.2. Frameworks

Javalin a lightweight framework for the backend, was chosen to simplify development and maximize performance. Renowned for its simplicity and efficiency, Javalin enables backend development without the complexity and overhead associated with more extensive frameworks.

For the frontend, a combination of Angular and Bootstrap was selected. Angular allows for a modular and component-based architecture, promoting code reusability and a clear structuring of the application. Bootstrap complements Angular with simple and responsive design templates, accelerating the development of a professionally looking frontend.

5.3. Patterns and Architecture

The use of the Factory Pattern supports the creation of the Exercise Generator by providing a clear separation between the creation of objects and their use. This enhances the flexibility and extensibility of the code.

The Model-View-Controller (MVC) pattern is employed to ensure a clear separation between the user interface, business logic, and data access. This facilitates both maintenance and expansion of the application.

The Client-Server architecture forms the basis of our system, enabling light and efficient communication between the frontend and backend.

5.4. Reusable Logic and Modular Structures

Outsourcing the calculation logic into a separate maths package allows for its reuse in different parts of the application, improving consistency and reducing the likelihood of errors.

The Exercise Generator is an example of a modular structure that is exclusively accessible through the Factory Pattern. This ensures that changes to the creation logic are centralized, simplifying the maintenance and testing of the application.

5.5. Package Diagram

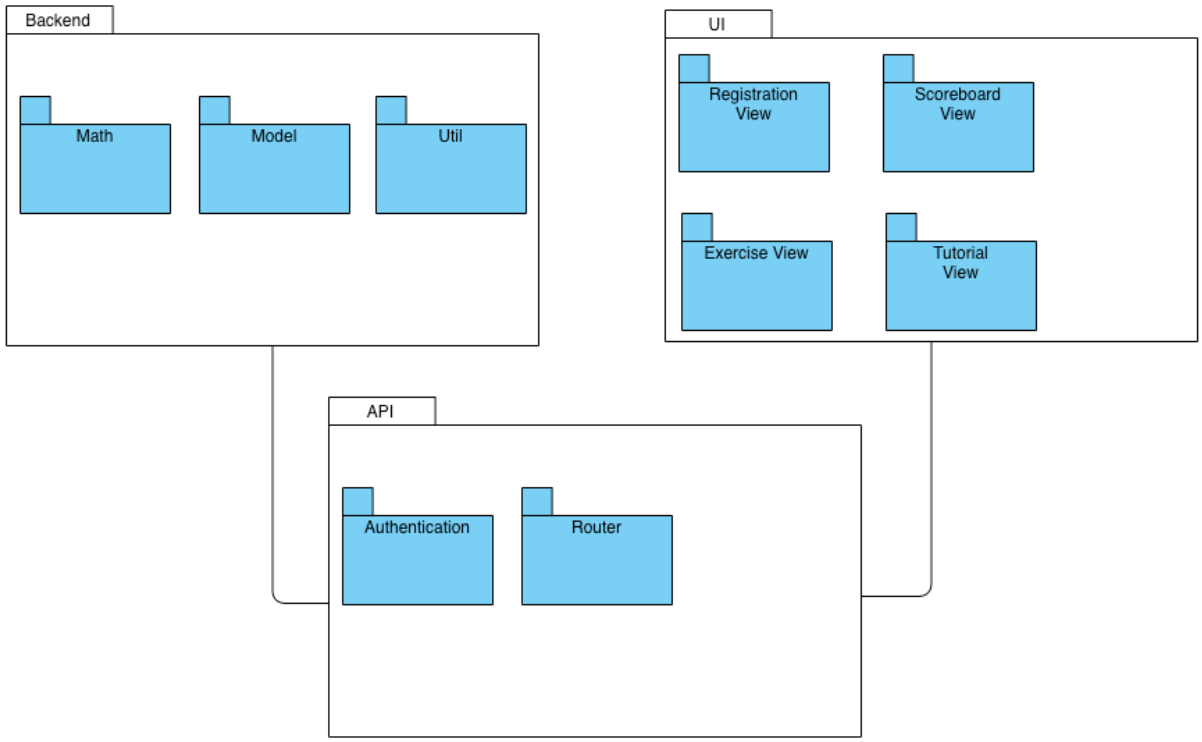


Figure 5: Package Diagram

5.6. Conclusion

The chosen architecture and technology stack reflect a careful consideration of the project requirements, existing team skills, and best practices in software development. These decisions enable us to develop a robust, maintainable, and scalable application that meets both current and future needs.

6. Design-Artefact

Design-Klassendiagramm (DCD) und geeignete Interaktionsdiagramme zeichnen
→ für mind. 4 Systemoperationen ein Interaktionsdiagramm zeigen
→ gefällte Design-Entscheide dokumentieren
→ siehe unten

6.1. Design-Class Diagram (DCD)

Input from the other.

6.2. Interaction Diagrams

Sequence diagrams illustrate the chronological sequence of interactions between objects during a quiz run. They illustrate how Student objects interact with Quiz and Question objects to answer questions and receive scores. Collaboration diagrams also show the relationships and message exchange between the objects, which is crucial for implementing the logic behind the quiz interactions.

7. Implementation

Beschreiben wie die gewählte Softwarearchitektur verifiziert wurde
→ partielle und unvollständige Implementation (Prototyp) der wichtigsten Use Cases verifizieren

Erklärung der Teststrategie: Welche Tests werden auf welchen Teststufen gemacht inkl. Begründung
→ Unit-Tests
→ Integration-Tests
→System-Tests

8. Project management

1. Planung der bisherigen Iterationen sowie entsprechender Aufwand erfassen und darstellen
2. Pro Iteration tatsächlicher Aufwand und erreichten Resultate mit Planung und gesteckten Zielen vergleichen
3. Allfällig getroffene Massnahmen festhalten
4. Detaillierte Planung der nächsten Iteration (inkl. Aufwandschätzung & aufdatierte Risikoliste)

9. Glossary

Term	Definition / Explanation
App-Team	The collective responsible for the development and maintenance of the Mathify quiz platform.
Calculator	A component within the app designed to assess and validate the accuracy of answers submitted by the user.
Child	The primary user group for the Mathify quiz, encompassing students from grades 1 through 9.
Exercise	A mathematical challenge tailored to the student's grade, selected mode and reached level of difficulty.
Exercise Generator	A system component that creates a variety of random mathematical exercises for users.
Game Mode	An initial selection feature that allows users to choose a play style such as mixed or custom.
Grade	A categorization that ranges from 1 to 9, dictating the types of exercises generated based on educational level.
Teaching Library	A repository of educational resources, including videos, texts, and graphs to support learning.
Level of Difficulty	A progression parameter within the game that progresses from basic to intermediate and finally to advanced.
Mathify	The official name of the quiz application.
National Syllabus	The "Lehrplan 21" dictates the variety and complexity of exercises tailored to each grade and level of difficulty.
Scoreboard	A feature that enables students to compare their performance with peers and their learning journey.
Sensitive Data	All personal information that can identify clients, requiring secure storage to protect privacy.