ELO Learning

REQUIREMENTS DOCUMENT

ZERO DAY

Last updated: 18 August 2025

University of Pretoria

Name	Student number
RM (Rene) Brancon	u22556771
NF (Nigel) Mofati	u22528084
TM (Tukelo) Mokwena	u22536800
S (Saskia) Steyn	u17267162
NG (Ntokozo) Tonga	u22506773

Team contact:

ZeroDay0D4y@gmail.com

Contents

1. Introduction

1.1 Purpose of the Document

The purpose of this document is to define the user stories, use cases, functional and non-functional requirements for the ELO Learning platform. It outlines the objectives of the system, describes the features and interactions expected from users, and sets the foundation for system design and development. This document serves as a point of reference for our development team, project stakeholders, and any future maintenance efforts.

1.2 Scope of the System

ELO Learning is a gamified math app designed to help students from Grade 8 to first-year university level improve their mathematical proficiency. Using an ELO-based rating system inspired by competitive games, the platform matches students with questions that reflect their skill level.

Students begin by filling in their profile and completing a baseline test, which dynamically adjusts in difficulty using a decision tree structure. Based on their performance, the system assigns them an initial ELO score. The platform includes features such as secure user authentication, gamification elements (badges, leaderboards), and personalised analytics to promote engagement and continuous learning.

1.3 Intended Audience

This document is intended for:

- **Developers and Designers:**
  To understand the system functionality and implement the design accordingly.
- **Project Stakeholders (Proking Solutions):**
  To ensure the platform meets business and user goals.
- **COS 301 Supervisors and Evaluators:**
  To assess the completeness and feasibility of the project.
- **Testers:**
  To develop test plans and verify the system against requirements.
- **Future Maintenance Teams:**
  To provide a clear reference for enhancements or troubleshooting.

2. User Stories

The user stories are discussed from 2 different perspectives, that of students, and general users.

1. Students (Primary Users)

Grade 8 to first-year university students.

1. As a student, I want to sign up and create my profile by providing my personal (name, surname, username, email address) and academic details (age, grade, and confidence level) so that the platform can tailor the experience to my level.

2. As a student, I want the ELO algorithm to balance my rating based on matches I play..

3. As a student, I want to answer questions using a math keyboard so that I can input notation like fractions and exponents correctly.

4. As a student, I want to view my current ELO rating and performance stats (such as accuracy and progress over time) so that I can track my improvement.

5. As a student, I want to choose between different game modes– Ranked Matches, Practice Rounds, and Single player Rounds– so that I can learn in a way that suits my goal (be it to progress or speed).

6. As a student, I want to solve math problems in Ranked Matches based on my ELO so that I can improve my rating and feel challenged.

7. As a student, I want to enter Practice mode with questions at my current level so that I can improve at my own pace.

8. As a student, I want to attempt Single player matches where I solve as many questions as possible under a time limit so that I can sharpen my speed and accuracy.

9. As a student, I want to view a full memorandum (correction and explanation) for each practice question at the end so that I can learn from my mistakes.

10. As a student, I want to retake problems I previously got wrong so that I can reinforce learning and improve on my weaknesses.

11. As a student, I want to earn badges and rewards as I complete goals and milestones so that I stay motivated.

12. As a student, I want to earn XP for correctly solving questions so that I feel rewarded for my effort.

13. As a student, I want to appear on a leaderboard ranked by XP, so that I can compare myself to others and engage in healthy competition.

14. As a student, I want to access the platform on both my phone and desktop so that I can practice anytime, anywhere.

2. System

Adaptive Testing & ELO Calculation

1. As the system, I want to assign difficulty levels to questions from 1 to 10 so that I can measure user performance accurately.

2. As the system, I want to update a student's ELO rating after each problem submission (especially in ranked mode) so that future challenges reflect their current skill level.

3. As the system, I want to select problems based on a student's ELO and topic history so that they receive appropriate and varied challenges.

4. As the system, I want to adapt question difficulty dynamically based on recent performance trends so that the challenge remains balanced.

5. As the system, I want to store and track each student's ELO history over time so that trends and improvement can be visualized.

Submission Evaluation & Feedback

6. As the system, I want to parse and evaluate math expressions submitted in LaTeX format so that I can accurately assess correctness.

7. As the system, I want to provide immediate visual feedback (correct/incorrect, color indicators) after a submission so that students can learn effectively.

8. As the system, I want to suggest short tutorials or hints when a student fails a problem multiple times so that they don't feel stuck.

9. As the system, I want to recommend a review topic when a student repeatedly struggles in a specific area so that they can focus their improvement.

Tracking & Analytics

10. As the system, I want to log every problem attempt with metadata (time taken, number of attempts, type of answer) so that analytics remain comprehensive.

11. As the system, I want to track the time of day and session duration so that I can optionally suggest breaks when students seem fatigued (wellness feature).

Leaderboard and Abuse Prevention

12. As the system, I want to update the leaderboard in real-time when a user's XP/ELO changes so that rankings are always current.

13. As the system, I want to detect abnormal behavior (e.g, spamming submissions, solving too fast) so that abuse or misuse of the platform can be flagged.

Notifications & Engagement

14. As the system, I want to send push notifications reminding students to complete their daily lesson so that they stay consistent in their learning routine.

15. As the system, I want to notify students when new badges or achievements are unlocked so that they feel recognised and motivated.

16. As the system, I want to remind students if they haven't logged in for several days so that they do not fall behind or forget to practice.

17. As the system, I want to notify students when their leaderboard position changes so that they stay engaged and motivated to keep progressing.

3. Admin (Developers and Stakeholders)

Content & Question Management

1. As an admin, I want to upload, manage, and categorize math problems by topic and difficulty so that the problem pool stays educational, relevant, and diverse.

2. As an admin, I want to update or delete outdated questions so that users always receive accurate content.

3. System Requirements

FR1: User Registration and Profile Creation

- **FR1.1** The system must provide a registration form for students to input name, surname, age, email address, grade and math confidence level.
    - **FR1.1.1:** If a username that is already selected is chosen, prevent the student from selecting that username.
- **FR1.2** The system shall validate and securely store user information.
- **FR1.3** The system shall allow students to edit their profile data after registration.

FR2: Secure Login and Authentication

- **FR2.1** The system shall allow students to log in using their username and password.
- **FR2.2** The system shall implement password hashing and secure authentication mechanisms.
- **FR2.3** The system shall provide error messages for incorrect login attempts and allow password resets.

FR3.1 The system shall show one math question at a time to reduce cognitive load.

- **FR3.2** The system shall determine whether to branch left (easier) or right (harder) in the decision tree based on correctness of the student's answer.
- **FR3.3** The system shall use question difficulty levels ranging from 1 to 10.
- **FR3.4** The system shall compute an initial ELO rating based on consistent performance at a given level.

FR4: Game Modes and Math Practice

- **FR4.1** The system shall allow students to choose from three modes: Ranked Matches, Practice Rounds, and Single player Rounds.
- **FR4.2** The system shall deliver Ranked Match questions based on the user’s current ELO.
- **FR4.3** The system shall limit Single Rounds to a predefined time and track the number of correct answers within that period.
- **FR4.4** The system shall update the student’s ELO rating after every ranked match.
- **FR4.5** The system shall allow students to retake previously incorrect problems from their practice history.

FR5: Math Keyboard Input

- **FR5.1** The system shall include a math keyboard supporting symbols like fractions, exponents, square roots, and Greek letters.

FR6: Feedback and Memorandum

- **FR6.1** The system shall display immediate visual feedback (correct/incorrect) after every answer.
- **FR6.2** The system shall display a full memorandum or worked solution after each practice or test session.
- **FR6.3** The system shall recommend review topics or hints when a student struggles repeatedly.

FR7: XP, Badges, and Leaderboards

- **FR7.1** The system shall award XP for correct answers in Practice and Single player Rounds.
- **FR7.2** The system shall assign badges for milestone achievements.
- **FR7.3** The system shall maintain a real-time leaderboard ranked by XP.
- **FR7.4** The system shall allow filtering of the leaderboard by ELO ranking.

FR8: ELO & Performance Analytics

- **FR8.1** The system shall display the student's current ELO rating and progress chart over time.
- **FR8.2** The system shall store a history of ELO changes and match data.
- **FR8.3** The system shall allow students to view accuracy, number of questions attempted, and topic mastery.

FR9: Push Notifications

- **FR9.1** The system shall notify students to complete their daily lesson or activity.
- **FR9.2** The system shall notify students when badges or achievements are unlocked.
- **FR9.3** The system shall notify inactive users after a predefined period.
- **FR9.4** The system shall notify users when their leaderboard rank changes.
- **FR9.5** The system shall notify students when their account is deleted or deactivated by admin.

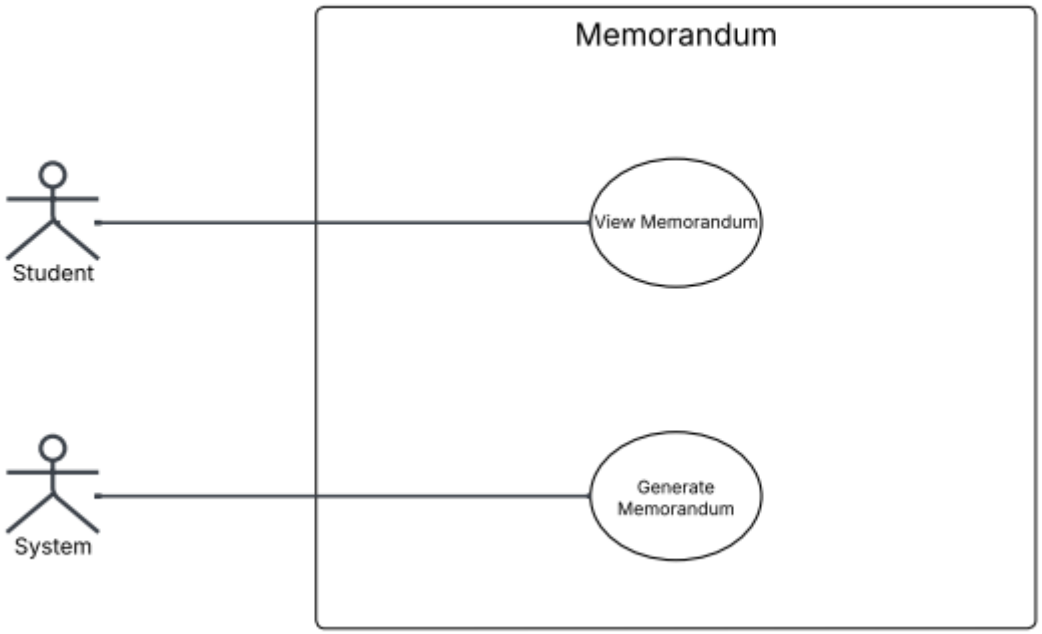
FR10: User Accessibility

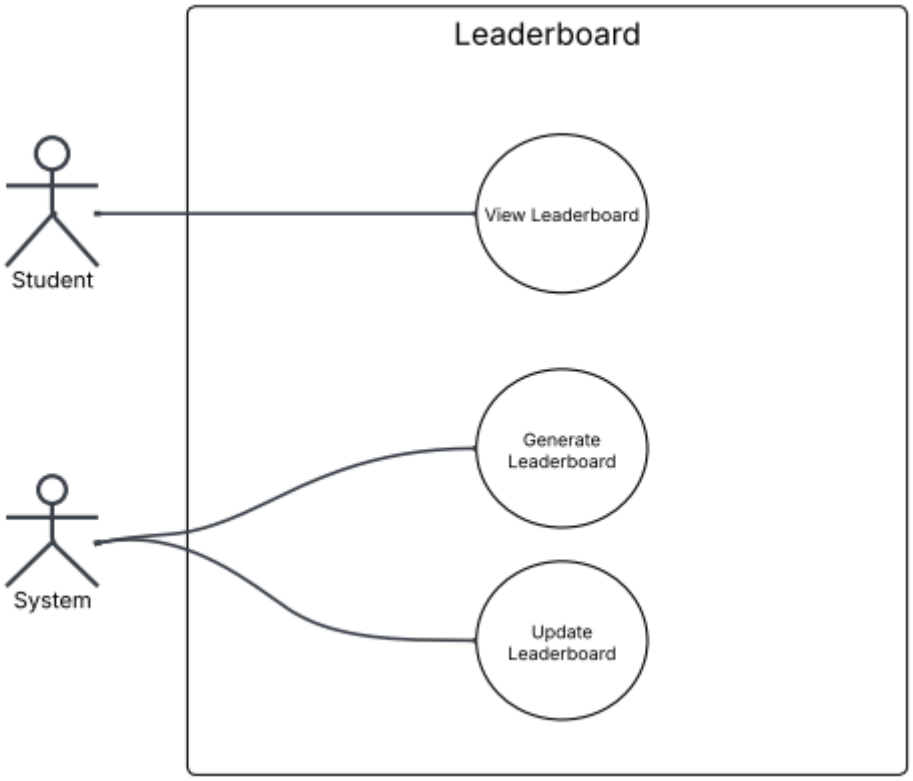
- **FR10.1** The system shall be accessible on desktop and mobile browsers.
- **FR10.2** The interface shall be responsive and support input from touchscreens and keyboards.

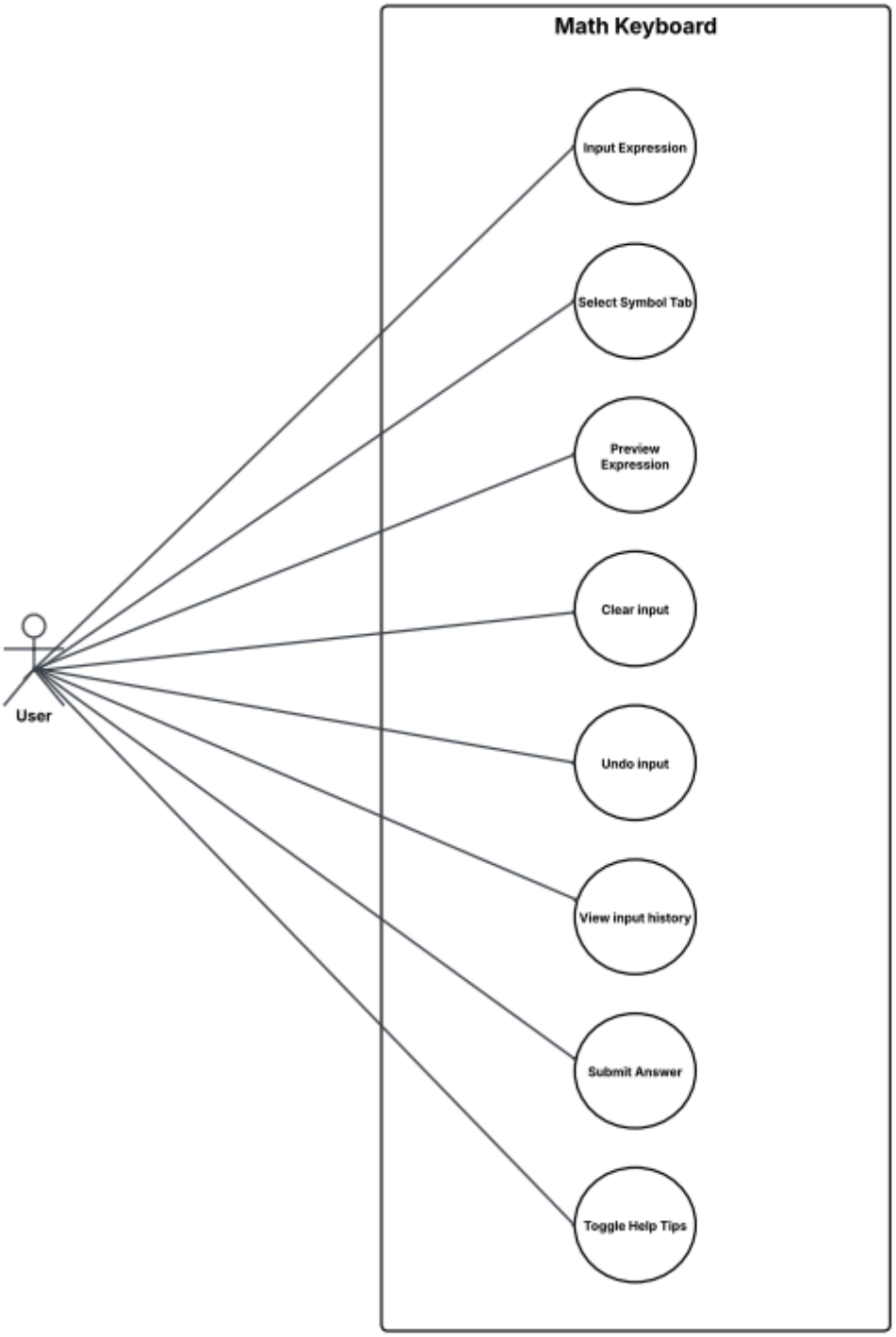
FR11: Admin – Content & Question Management

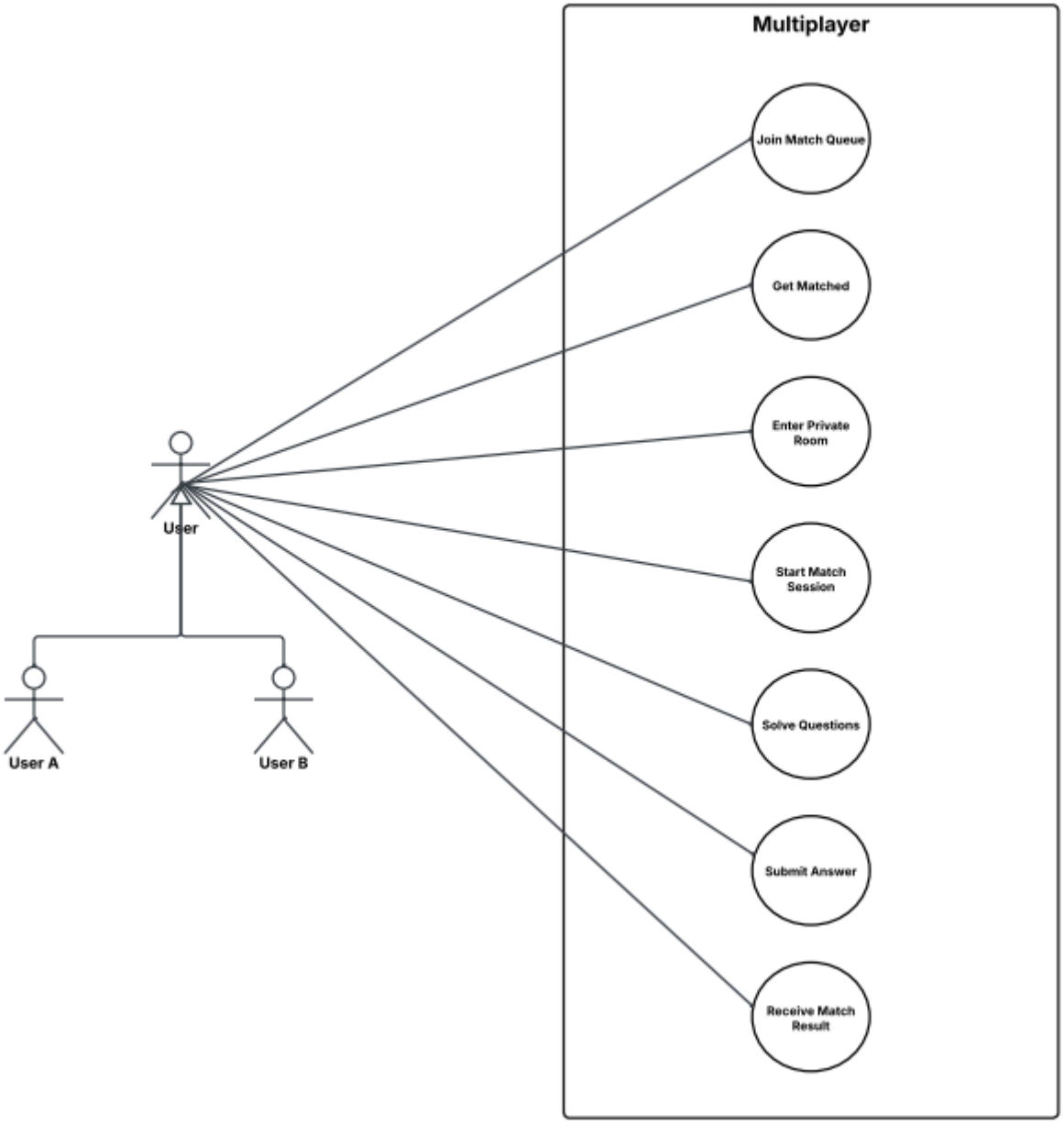
- **FR11.1** The system shall allow admins to upload, edit, delete, and categorize math problems by topic and difficulty.

4. Use Cases

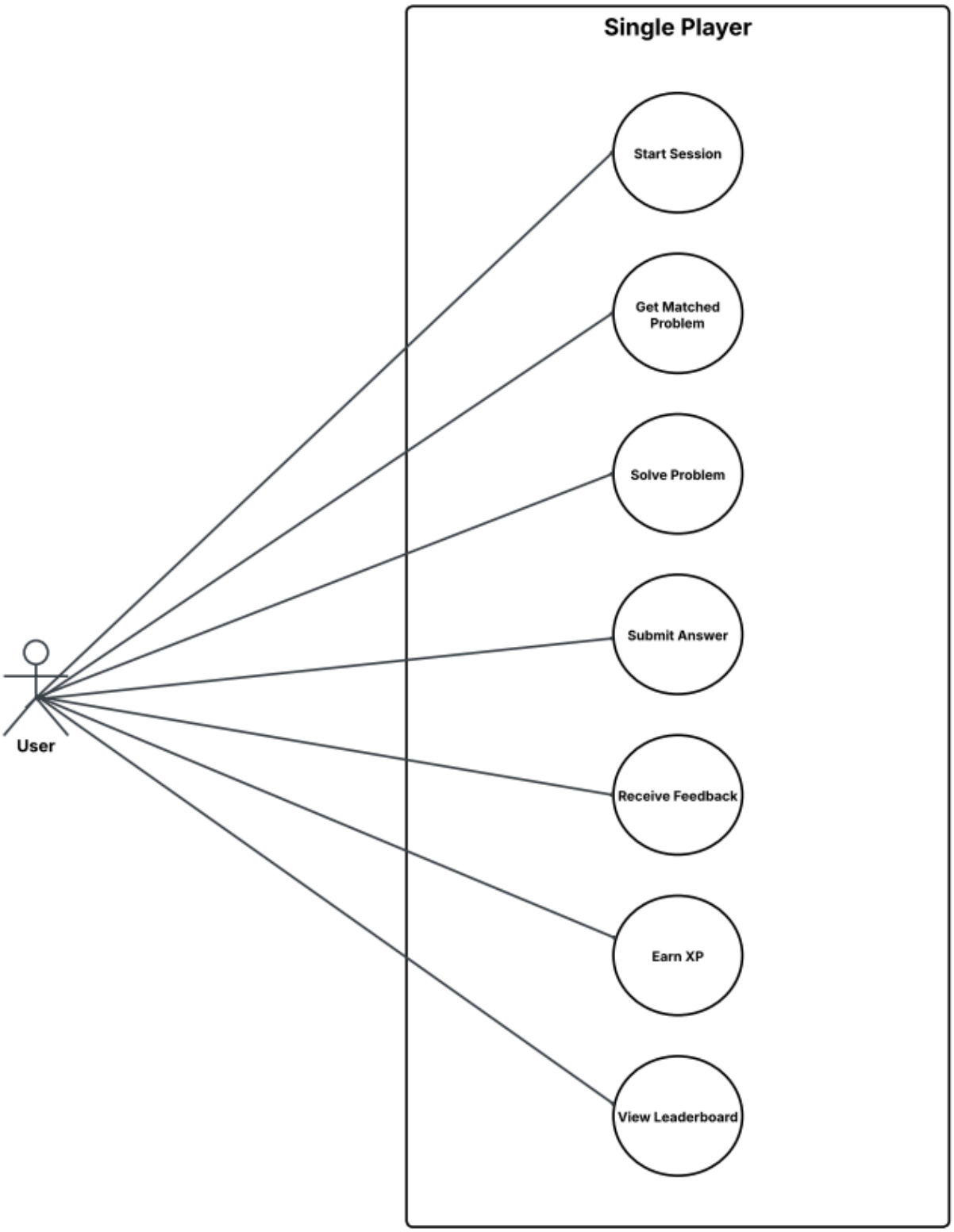

Memorandum
Student
View Memorandum
System
Generate Memorandum

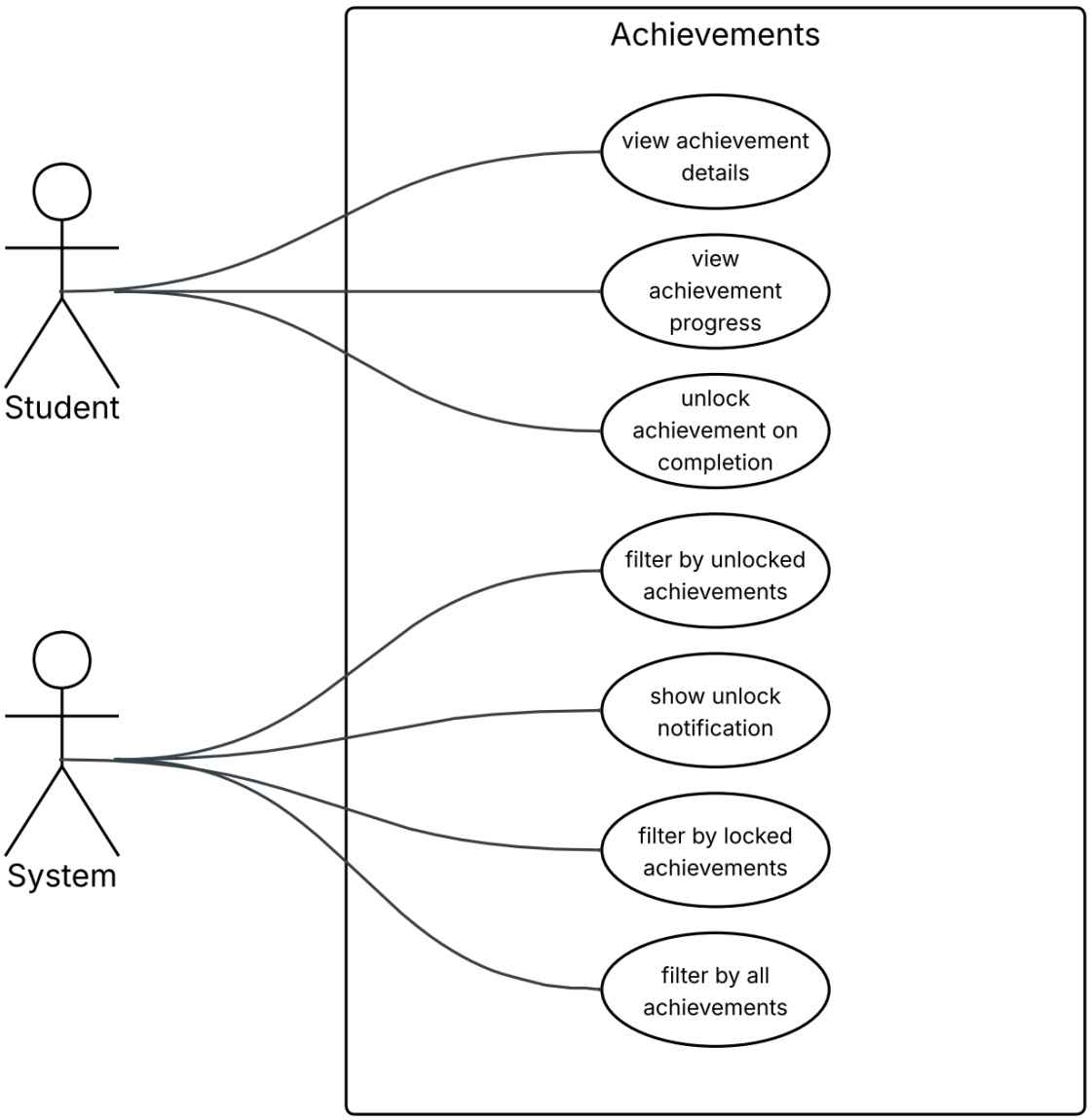

Leaderboard
Student
View Leaderboard
Generate Leaderboard
System
Update Leaderboard

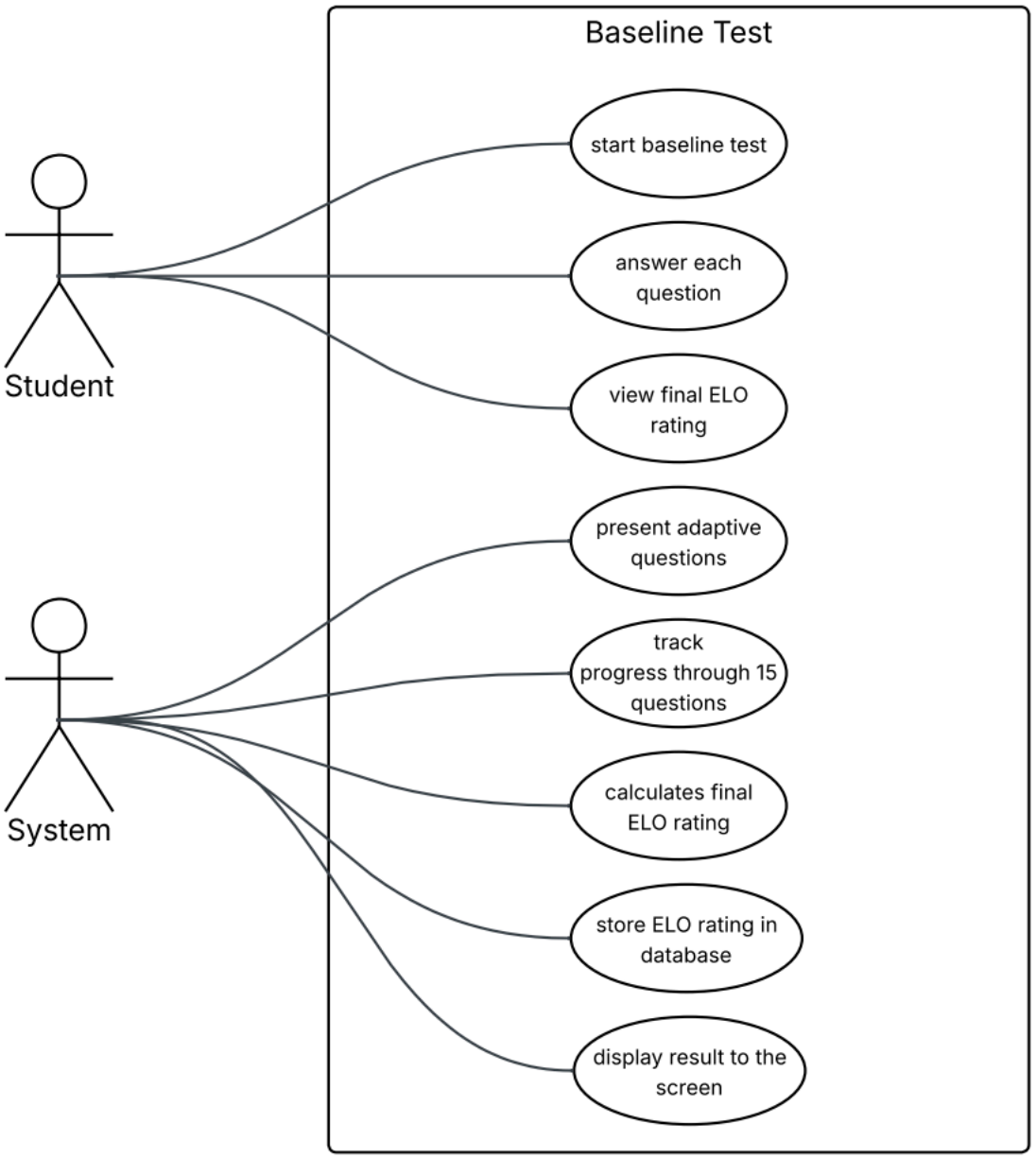

Math Keyboard
Input Expression
Select Symbol Tab
Preview Expression
Clear input
Undo input
View input history
Submit Answer
Toggle Help Tips
User

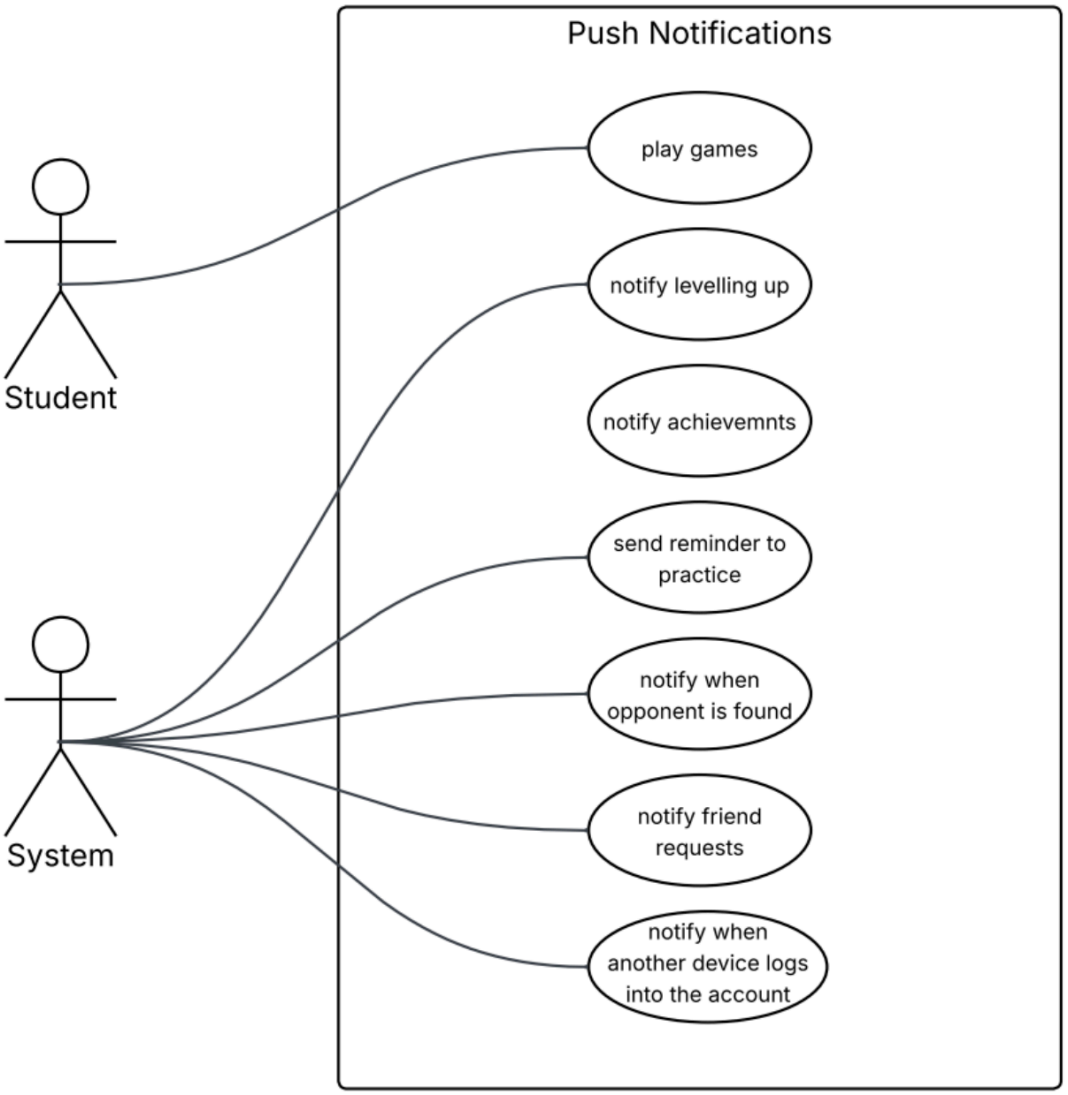

Multiplayer
Join Match Queue
Get Matched
Enter Private Room
Start Match Session
Solve Questions
Submit Answer
Receive Match Result
User
User A
User B


Practice
Choose Section
Answer Question
Submit Answer
Choose Table
Check Answer
Track Time
Assign XP
Generate Memorandum
Student
System

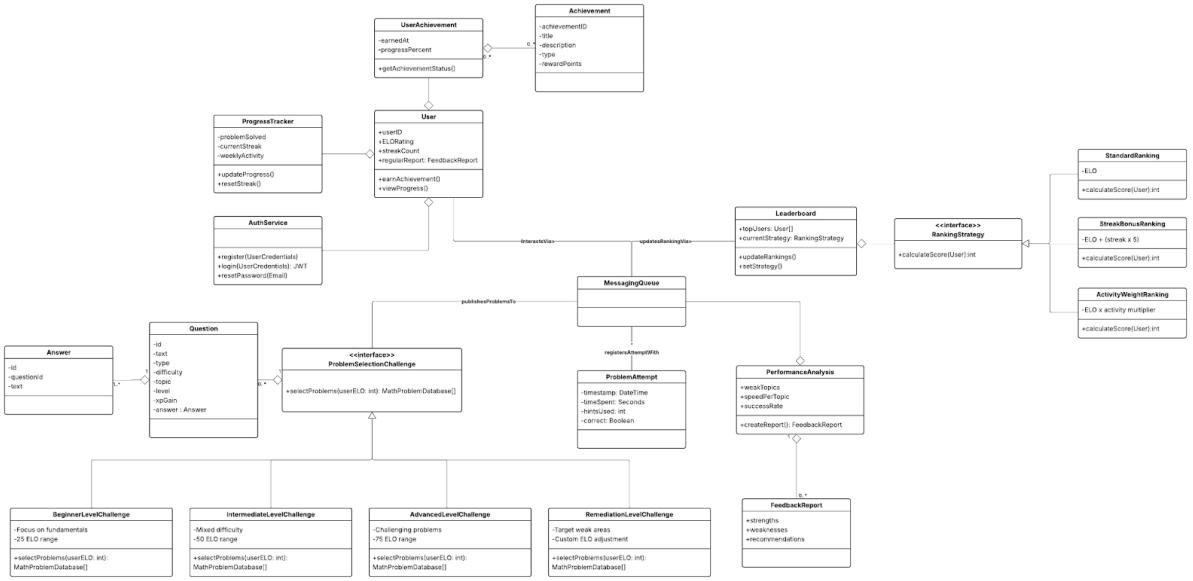

Single Player
Start Session
Get Matched Problem
Solve Problem
Submit Answer
Receive Feedback
Earn XP
View Leaderboard
User


Achievements
view achievement details
view achievement progress
unlock achievement on completion
filter by unlocked achievements
show unlock notification
filter by locked achievements
filter by all achievements
Student
System


Baseline Test
start baseline test
answer each question
view final ELO rating
present adaptive questions
track progress through 15 questions
calculates final ELO rating
store ELO rating in database
display result to the screen
Student
System


Push Notifications
play games
notify levelling up
notify achievemnts
send reminder to practice
notify when opponent is found
notify friend requests
notify when another device logs into the account
Student
System

5. Domain Model



6. Math Sections

Grade 8:

Paper 1	Paper 2
Fractions	Construction of Geometric Figures
Integers	Geometry of 2D Shapes
Exponents	Geometry of Straight Lines
Functions and Relationships	
Algebraic Expressions	
Algebraic Equations	

Grade 9:

Paper 1	Paper 2
Fractions	Construction of Geometric Figures
Integers	Geometry of 2D Shapes
Exponents	Geometry of Straight Lines
Numeric and Geometric Patterns	Pythagoras' Theorem
Functions and Relationships	Area and Perimeter of 2D Shapes
Algebraic Expressions	
Equations	

Grade 10:

Paper 1	Paper 2
Fractions	Construction of Geometric Figures
Integers	Geometry of 2D Shapes
Exponents	Geometry of Straight Lines

Numeric and Geometric Patterns	Pythagoras' Theorem
Functions and Relationships	Area and Perimeter of 2D Shapes
Algebraic Expressions	
Equations	

Grade 11+12:

Paper 1	**Paper 2**
Algebra & equations	Statistics
Patterns & Sequences	Analytical geometry
Financial Mathematics	Trigonometry
Functions and Graphs	Euclidean geometry (Circle Geometry)
Probability	Measurement
Introduction to Calculus	

7. Technology Choices

Frontend Development: React.js and Next.js (PWA)

Alternatives Considered:

- **Vue.js & Nuxt.js:**
  Strong SSR capabilities and gentle learning curve
- **SvelteKit:**
  Excellent performance with minimal bundle size
- **React.js & Next.js:**
  (Selected) Mature ecosystem with comprehensive PWA support

Selection Justification:
React + Next.js provides the best balance of development velocity, PWA capabilities, and ecosystem support for complex mathematical interfaces. The mature component ecosystem includes specialized math rendering libraries that directly support our usability quality requirements.

Backend Development: Express.js

Alternatives Considered:

- **Express.js:**
  (Selected) Lightweight, flexible Node.js framework with extensive middleware ecosystem
- **NestJS:**
  Structured TypeScript framework with built-in dependency injection and testing
- **Spring Boot (Java):**
  Enterprise-grade framework with comprehensive features but steeper learning curve

Selection Justification:
Express.js was chosen for its simplicity and rapid development capabilities, which align perfectly with our project timeline and team expertise. While NestJS offers more structure through its opinionated architecture, Express.js provides the flexibility needed to implement our SOA pattern without the overhead of learning a complex framework.

The extensive middleware ecosystem allows us to add exactly the features we need for authentication, WebSocket support, and API routing without unnecessary complexity. This choice directly supports our performance quality requirements through minimal overhead and our maintainability requirements through the team's existing familiarity with Express.js patterns. The framework's lightweight nature also supports our scalability goals by reducing resource consumption per service instance.

Database Strategy: PostgreSQL + InfluxDB

Alternatives Considered:

- **MongoDB + PostgreSQL:**
  NoSQL flexibility with relational consistency
- **MySQL + Prometheus:**
  Standard relational with monitoring-focused time-series
- **PostgreSQL + InfluxDB:**
  (Selected) Robust relational with specialized time-series capabilities

Selection Justification:
PostgreSQL provides the ACID compliance needed for user data and ELO calculations, while InfluxDB offers optimized time-series storage for learning analytics. This combination directly supports our performance and scalability quality requirements.

Real-Time Communication: NestJS WebSocket Gateway

Alternatives Considered:

- **Socket.IO (standalone):**
  Feature-rich but requires additional integration overhead
- **Firebase Realtime Database:**
  Easy setup but vendor lock-in concerns
- **NestJS WebSocket Gateway:**
  (Selected) Integrated with existing backend architecture

Selection Justification:
Native integration with our SOA services eliminates additional complexity while providing the <300ms latency required by our performance quality requirements.

Authentication: OAuth 2.0 + JWT

Alternatives Considered:

- **Firebase Auth:**
  Simplified implementation but vendor dependency
- **Session-based Auth:**
  Traditional approach but limited scalability

- **OAuth 2.0 + JWT:**
  (Selected) Industry standard with scalable token-based architecture

Selection Justification:
Provides the security requirements while supporting our SOA pattern's stateless service communication. The standard approach ensures long-term maintainability and compliance with security best practices.

8. Services Contracts

The ELO Learning platform provides a comprehensive set of REST APIs and WebSocket services for managing users, questions, answers, gameplay, and mathematical validation. All endpoints return JSON responses and follow standard HTTP status codes.

Base URL:

```
http://localhost:3000 (development) / https://your-domain.com
(production)
```

1.Authentication

Most endpoints require Bearer token authentication. Include the token in the Authorization header:

```
Authorization: Bearer <jwt_token>
```

Tokens are obtained through the login endpoint and have a 1-hour expiration time.

2.User Management APIs

2.1 GET /users

Retrieve all users in the system

Authentication: Not required

2.2 GET /user/:id

Retrieve a specific user by ID

Authentication: Required (Bearer token)

- id (path): User UUID

2.3 GET /users/:id/achievements

Retrieve all achievements for a specific user

Authentication: Required (Bearer token)

- id (path): User UUID

2.4 POST /user/:id/xp

Update a user's XP points

Authentication: Required (Bearer token)

- id (path): User UUID

2.5 POST /user/:id/avatar

Update a user's avatar

Authentication: Not required

- id (path): User UUID

- Request Body: { "pfpURL": "string" }

- Response: Updated user object

2.6 GET /users/rank/:rank

Retrieve all users with a specific rank

Authentication: Not required

- rank (path): Rank name

2.7 POST /register

Register a new user account

Authentication: Not required

2.8 POST /login

Authenticate user and get access token

Authentication: Not required

2.9 POST /forgot-password

Request password reset email

Authentication: Not required

2.10 POST /reset-password

Reset password using reset token

Authentication: Not required

2.11 GET /verify-reset-token/:token

Verify if a password reset token is valid

Authentication: Not required

- token (path): Reset token

3.Question Management APIs

3.1 GET /questions

Retrieve all questions in the system

Authentication: Not required

3.2 GET /questionsById/:id

Retrieve a specific question by ID

Authentication: Not required

- id (path): Question UUID

3.3 GET /question/:level

Retrieve all questions for a specific level

Authentication: Required (Bearer token)

- level (path): Level number

3.4 GET /question/:id/answer

Retrieve the correct answer for a specific question

Authentication: Required (Bearer token)

- id (path): Question UUID

3.5 GET /questions/topic

Retrieve questions by topic

Authentication: Not required

- topic (query): Topic name

3.6 GET /questions/level/topic

Retrieve questions filtered by both level and topic

Authentication: Not required

- level (query): Level number

- topic (query): Topic ID

3.7 GET /topics

Retrieve all available topics

Authentication: Not required

3.8 GET /questions/random

Retrieve random questions for a specific level

Authentication: Not required

- level (query): Level number (required)

3.9 POST /question/:id/submit

Submit and validate an answer for a specific question

Authentication: Not required

- id (path): Question UUID

3.10 GET /questions/type/:type

Retrieve questions by type

Authentication: Not required

- type (path): Question type

3.11 GET /questions/mixed

Retrieve a mixed set of questions

Authentication: Not required

4.Answer Management APIs

4.1 GET /answers/:id

Retrieve all answers for a specific question

Authentication: Not required

- id (path): Question UUID

4.2 POST /submit-answer

Submit an answer (alternative to /question/:id/submit)

Authentication: Not required

- Request Body:

```
{ "question_id": "uuid", "studentAnswer": "string", "userId": "uuid (optional)" }
```

- Response:

```
{ "isCorrect": "boolean", "studentAnswer": "string", "correctAnswer": "string", "message": "string", "xpAwarded": "number", "updatedUser": { "id": "uuid", "xp": "number" } }
```

5.Practice Mode APIs

5.1 GET /practice

Retrieve 10 practice questions with answers

Authentication: Not required

5.2 GET /practice/type/:questionType

Retrieve practice questions by type

Authentication: Not required

- questionType (path): Type of question

6.Single Player Game APIs

6.1 POST /singleplayer

Process single player game attempt and update user statistics

Authentication: Not required

7.Multiplayer Game APIs

7.1 POST /multiplayer

Process multiplayer game results and update both players' statistics

Authentication: Not required

8.OAuth APIs

8.1 POST /oauth/user

Create or retrieve OAuth authenticated user

Authentication: Not required

9.Math Validation APIs

9.1 POST /validate-answer

Validate a mathematical answer

Authentication: Not required

9.2 POST /quick-validate

Quick validation for real-time feedback

Authentication: Not required

9.3 POST /validate-expression

Validate mathematical expression format

Authentication: Not required

10.WebSocket Events

The system uses Socket.IO for real-time multiplayer functionality. Connect to the same base URL with Socket.IO client.

10.1 Client Events (Sent to Server)

10.1.1 queue-for-game

Join the multiplayer game queue

```
{ "id": "uuid", "name": "string", "username": "string", "xp": "number" }
```

10.1.2 leave-queue

Leave the multiplayer game queue

None

10.1.3 disconnect

Handle client disconnection

None

10.2 Server Events (Sent to Client)

10.2.1 startGame

Game has started with matched opponent

```
{ "gameId": "uuid", "opponent": { "name": "string", "username":
"string", "xp": "number" } }
```

10.2.2 queueUpdate

Update on queue status

```
{ "position": "number", "totalInQueue": "number" }
```

10.2.3 gameEnded

Game has ended

```
{ "gameId": "uuid", "result": "string", "stats": "object" }
```

10.2.4 achievementsUnlocked

Achievements unlocked while in queue

```
{ "achievements": [ { "achievement_id": "uuid", "achievement_name":
"string", "description": "string", "earned_date": "ISO date string" } ]
}
```

11.Error Handling

11.1 Standard HTTP Status Codes

- **200 OK**: Request successful
- **201 Created**: Resource created successfully
- **400 Bad Request**: Invalid request data
- **401 Unauthorized**: Authentication required or invalid
- **404 Not Found**: Resource not found
- **409 Conflict**: Resource already exists (e.g., email during registration)
- **500 Internal Server Error**: Server error

11.2 Error Response Format

```
{ "error": "string (error message)" }
```

11.3 Common Error Scenarios

11.3.1. Missing Authorization Header:

```
{ "error": "You are unauthorized to make this request." }
```

11.3.2. Invalid Token:

```
{ "error": "Invalid or expired token" }
```

11.3.3. Missing Required Fields:

```
{ "error": "All fields are required" }
```

11.3.4. User Not Found:

```
{ "error": "User doesn't exist" }
```

11.3.5. Database Error:

```
{ "error": "Failed to fetch/update/create resource" }
```

12.Response Formats

12.1 Success Response

All successful responses return data in JSON format with appropriate HTTP status codes.

12.2 Date Formats

All dates are returned in ISO 8601 format: YYYY-MM-DDTHH:mm:ss.sssZ

12.3 UUID Format

All IDs use UUID v4 format: xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx

12.4 Pagination

Currently, pagination is not implemented. The /practice endpoint limits results to 10 items, and /questions/random limits to 5 items.

12.5 Data Validation

- Email addresses must be valid format

- Passwords must be at least 8 characters for registration/reset

- Level parameters must be positive integers

- XP values must be numbers

- Boolean fields only accept true or false

9. WOW Factors

WF1: Classroom Wars (Gamified Competitions)

- **WF1.1** The system shall allow teachers to create a "Classroom War" match between groups of students or entire classes.
- **WF1.2** The system shall automatically match students within a Classroom War based on similar ELO ratings.
- **WF1.3** The system shall provide real-time progress tracking of each class or group during the competition.
- **WF1.4** The system shall award bonus XP, badges, and achievements to winners of Classroom Wars.
- **WF1.5** The system shall provide a leaderboard for Classroom Wars, visible to all participants.

WF2: Customer Support & Help Desk

- **WF2.1** The system shall provide an in-app customer support portal accessible from the user dashboard.
- **WF2.2** The system shall allow students, parents, or teachers to submit support tickets describing technical or account issues.
- **WF2.3** The system shall allow support staff to view, respond, and resolve tickets within an admin panel.
- **WF2.4** The system shall notify users via push notifications or email when their ticket status changes.
- **WF2.5** The system shall provide a searchable FAQ and knowledge base for common issues.

WF3: AI Assistance & Adaptive Learning

- **WF3.1** The system shall use AI to recommend personalized practice sets based on student weaknesses and learning history.
- **WF3.2** The system shall allow students to request AI-generated hints or step-by-step explanations for difficult problems.
- **WF3.3** The system shall allow AI to detect patterns of repeated mistakes and suggest targeted review topics.
- **WF3.4** The system shall provide teachers with AI-generated analytics on student progress and areas of struggle.
- **WF3.5** The system shall adapt question difficulty dynamically using AI predictions of user performance.

WF4: Location-Based Rankings

- **WF4.1** The system shall allow leaderboards to be filtered by country, region, school, or classroom.

- **WF4.2** The system shall determine a user's location using profile data or device geolocation.
- **WF4.3** The system shall allow students to compare their performance against peers in the same city, school, or region.
- **WF4.4** The system shall award location-based badges (e.g., "Top 10 in Johannesburg").
- **WF4.5** The system shall ensure location data privacy and allow students to opt-out of public rankings.

WF5: Community & Social Learning

- **WF5.1** The system shall allow students to join or create communities (e.g., "Grade 10 Geometry Club").
- **WF5.2** The system shall allow community members to post questions, share solutions, and discuss math problems.
- **WF5.3** The system shall allow admins to moderate communities and remove inappropriate content.
- **WF5.4** The system shall provide community achievements and badges for active participation.
- **WF5.5** The system shall allow students to invite friends or classmates to join their community.

WF6: Avatar Creation & Unlockables

- **WF6.1:** Users shall have the ability to create personalised avatars.
- **WF6.2:** The system shall store the avatar and display it on the frontend.
- **WF6.3:** Users shall have the ability to unlock wearables for their avatar through playing matches on the application.