**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет кібербезпеки та програмної інженерії

Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ

В.о завідувача кафедри

Олексій ГОРСЬКИЙ

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_2023 р.

**КВАЛІФІКАЦІЙНА РОБОТА**

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ**

**“БАКАЛАВР”**

**Тема:** “Вебзастосунок для мовно-

культурного обміну студентів

англомовної групи”

**Виконавець:**  Барило Олександр Григорович

**Керівник:** к.ф-м.н доц. Ткаченко Олександр Андрійович

**Нормоконтролер:** к.ф-м.н доц. Оленін Михайло Вікторович

Київ 2023

**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE NATIONAL AVIATION UNIVERSITY**

**Faculty of cybersecurity and software engineering**

**Software engineering department**

ALLOW FOR DEFENSE

Acting head of the department

Oleksiy GORSKY

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_2023 y.

**QUALIFICATION WORK**

**(EXPLANATORY NOTE)**

**GRADUATE OF THE EDUCATIONAL DEGREE**

**“BACHELOR”**

**Theme:** “Web application for language and

cultural exchange for English group

students”

**Performer:**  Barylo Oleksandr Hryhorovych

**Supervisor:** Ph.D. in Physics and Mathematics, Associate Professor Tkachenko Oleksandr Andriyovych

**Standard Controller:** Ph.D. in Physics and Mathematics, Associate Professor Olenin Mykhailo Viktorovych

Kyiv 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

**Факультет** кібербезпеки та програмної інженерії

**Кафедра** інженерії програмного забезпечення

**Освітній** **ступінь**: бакалавр

**Спеціальність**: 121 Інженерія програмного забезпечення

**Освітня**-**професійна програма**: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Олексій ГОРСЬКИЙ

"\_\_\_" \_\_\_\_\_\_\_ 2023 р

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Барила Олександра Григоровича

1. Тема кваліфікаційної роботи: “Вебзастосунок для мовно-культурного обміну студентів англомовної групи”

затверджена наказом ректора від 28.04.2023 р. № 597/ст

2. Термін виконання проекту: з 15.05.2023 р. до 25.06.2023 р.

3. Вихідні данні до проекту: програмний продукт розроблений за допомогою програмних середовищ Visual Studio 2022 та WebStorm на мовах програмування C# та TypeScript.

4. Зміст пояснювальної записки:

1. Визначення мотивації, мети та потсановка проблем дослідження.

2. Огляд існуючих програмних продуктів, їхніх підходів, методів вивчення англійської мови.

3. Вибір методології розробки та високорівневий опис результуючого прогрманого продукту.

4. Проектування та розробка прототипу вебзастосунку.

5. Підхід до тестування та оцінки результатів розробки програмного продукту.

6. Висновки.

5. Перелік обов'язкових слайдів презентації:

1. Високорівнева архітектура вебзастосунку.

2. Діаграма класів одного з варіантів використання.

3. Діаграма зв’язків бази даних вебзастосунку.

4. Інтерфейс частини функціоналу.

6. Календарний план-графік

|  |  |  |  |
| --- | --- | --- | --- |
| № пор | Завдання | Термін виконання | Відмітка про виконання |
| 1. | Ознайомлення з постановкою задачі та вивчення літератури  Написання 1 розділу, представлення керівнику | 15.05.2023 –  17.05.2023 |  |
| 2. | Попередній друк 1 розділу та допоміжних сторінок (черновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел. 1-ий нормо-контроль. | 18.05.2023 –  21.05.2023 |  |
| 3. | Написання 2 розділу, представлення керівнику | 22.05.2023  –26.05.2023 |  |
| 4. | Написання 3 розділу, представлення керівнику | 27.05.2023 –  31.05.2023 |  |
| 5. | Написання 4 розділу, представлення керівнику | 01.06.2023 –  04.06.2023 |  |
| 6. | Проходження нормо-контролю,  перевірка на антиплагіат, перепліт  пояснювальної записки. | 05.06.2023 –  09.06.2023 |  |
| 7. | Розробка тексту доповіді. Оформлення  графічного матеріалу для презентації. | 10.06.2023 –  14.06.2023 |  |
| 8. | Отримання відгуку керівника, рецензії. | 15. 06.2023 |  |
| 9. | Підготовка матеріалів для передачі  секретарю ДЕК (ПЗ, ГМ, CD-R з  електронними копіями ПЗ, ГМ,  презентації, відгук керівника, рецензія,  довідка про успішність, 1 папка, 1  конверт) | 16.06.2023 –  18.06.2023 |  |

7. Дата видачі завдання 15.05.2023 р.

Керівник: к.ф-м.н доцент Ткаченко Олександр Андрійович Завдання прийняв до виконання: Олександр БАРИЛО

Дата

NATIONAL AVIATION UNIVERSITY

Faculty of cybersecurity and software engineering

Software engineering department

**Education degree:** bachelor

**Specialty:** 121 Software engineering

**Educational and professional program:** Software Engineering

APPROVED BY

Head of the department

Oleksiy GORSKY

"\_\_\_" \_\_\_\_\_\_\_ 2023

TASK

to the qualification work execution for

Barylo Oleksandr Hryhorovych

1. Topic of the qualification work: “Web application for language and cultural exchange for English group students”

approved by the order of the rector from 28.04.2023 р. № 597/st

2. Project implementation period: from 15.05.2023 y. to 25.06.2023 y.

3. Project output data: software product prototype developed in IDEs Visual Studio 2022 and WebStorm in C# and TypeScript progrramming languages.

4. Explanatory paper contents:

1. Definition of motivation, objectives and formulation of research problems.

2. Review of existing software products, their approaches, methods of studying the English language.

3. Selection of development methodology and high-level description of the resulting software product.

4. Design and development of a web application prototype.

5. Approach to testing and evaluation of software product development results.

6. Conclusions.

5. List of presentation mandatory slides:

1. Web application high level architecture diagram.

2. Class diagram of one of use case.

3. Diagram of the application database relations.

4. User interface of a piece of functionality.

6. Calendar schedule:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Task | Duration of performing | Evaluation of the performance |
| 1 | Introduction to the problem statement and study of literature | 15.05.2023 –  17.05.2023 |  |
| 2 | Writing of 1 chapter, presentation of section to the head | 18.05.2023 –  21.05.2023 |  |
| 3 | Writing of 2 chapter, presentation of section to the head | 22.05.2023  –26.05.2023 |  |
| 4 | Writing of 3 chapter, presentation of section to the head | 27.05.2023 –  31.05.2023 |  |
| 5 | Writing of 4 chapter, presentation of section to the head | 01.06.2023 –  04.06.2023 |  |
| 7 | Control of norms, binding of explanatory paper | 05.06.2023 –  09.06.2023 |  |
| 8 | Development of text of presentation. Design of graphic materials for presentation | 10.06.2023 –  14.06.2023 |  |
| 9 | Getting of head feedback and work review | 15. 06.2023 |  |
| 10 | Preparation of materials for transmission to secretary of SEC | 16.06.2023 –  18.06.2023 |  |

7. A task given on 15.05.2023.

Supervisor: \_\_\_\_\_\_\_\_\_\_ Ph.D. in Physics and Mathematics, Associate Professor Tkachenko Oleksandr Andriyovych

A task taken to perform by: \_\_\_\_\_\_\_\_\_\_ Oleksandr BARYLO

Date

# Реферат

Пояснювальна записка до кваліфікаційної роботи на тему «Вебзастосунок для мовно-культурного обміну студентів англомовної групи» 50 с., 23 рис., 4 табл., 17 інформаційнх джерел.

**Об'єкт дослідження:** Веб-додаток для вивчення англійської мови та культурного обміну між студентами англомовного проекту.

**Предмет дослідження:** Функціональність та ефективність веб-додатка, розробленого для вдосконалення знань з англійської мови та поліпшення комунікації серед студентів англомовного проекту.

**Мета кваліфікаційної роботи:** Розробити веб-додаток, який покращує вивчення англійської мови, зміцнює взаєморозуміння між студентами і підвищує їхнє задоволення від навчання через використання реального спілкування та гаміфікації.

Робота зосереджується на вивченні ролі технологій у вивченні мови, вивченні існуючих веб-додатків для вивчення мови, а також розумінні важливості реального спілкування та гаміфікації у навчанні. Робота також включає розробку та оцінювання веб-додатка, з детальним описом технологій, що використовуються (ASP.NET Core, SQL Server, Angular, SignalR), розробкою системи та архітектури, а також тестуванням і оцінкою веб-додатка.

ВЕБ-ДОДАТОК, ВИВЧЕННЯ МОВИ, СПІЛКУВАННЯ, ГАМІФІКАЦІЯ, ASP.NET CORE, ANGULAR, SQL SERVER, SIGNALR, СПІВРОБІТНИЦТВО, РЕАЛЬНИЙ ЧАС, ДИСТАНЦІЙНЕ НАВЧАННЯ.

# Abstract

Explanatory note to the qualification work “Web application for language and cultural exchange for English group students” 50 p., 23 fig., 4 table., 17 information sources.

**Object of work:** A web application for English language learning and cultural exchange among students.

**Subject of work:** Functionality and efficacy of a web application developed to enhance English language proficiency and improve communication among English-speaking students.

**Aim of the qualification work:** To develop a web application that enhances English language learning, fosters understanding among students, and increases their learning satisfaction through real-time communication and gamification.

The study focuses on exploring the role of technology in language learning, examining existing web applications for language learning, and understanding the importance of real-time communication and gamification in learning. The work also involves developing and evaluating the web application, with a detailed description of the technologies used (ASP.NET Core, SQL Server, Angular, SignalR), the development of the system and architecture, as well as testing and evaluation of the web application.

WEB APPLICATION, LANGUAGE LEARNING, CULTURAL EXCHANGE, ASP.NET CORE, SQL SERVER, ANGULAR, SIGNALR, GAMIFICATION, REAL-TIME COMMUNICATION

**Table of contents**

[Реферат 8](#_Toc137508341)

[Abstract 9](#_Toc137508342)

[List of abbreviations 11](#_Toc137508343)

[INTRODUCTION 12](#_Toc137508344)

[Chapter 1. Review of Related Literature and Existing Language Learning Applications 14](#_Toc137508345)

[1.1. Current language learning techniques 14](#_Toc137508346)

[1.2. The role of technology in language learning 16](#_Toc137508347)

[1.3. Online learning platforms and tools 17](#_Toc137508348)

[1.4. Real-time communication in online learning 19](#_Toc137508349)

[1.5. Advantages and disadvantages of current language learning applications 20](#_Toc137508350)

[1.5.1. Advantages 20](#_Toc137508351)

[1.5.2. Disadvantages 20](#_Toc137508352)

[Chapter 2. Research Design and Application Conceptualization 21](#_Toc137508353)

[2.1. Study design 21](#_Toc137508354)

[2.1.1. Research approach 21](#_Toc137508355)

[2.1.2. Rationale for chosen approach 21](#_Toc137508356)

[2.1.3. Research procedures 22](#_Toc137508357)

[2.2. Data collection and analysis methods 23](#_Toc137508358)

[2.2.1. Data Collection 23](#_Toc137508359)

[2.2.2. Data Analysis 23](#_Toc137508360)

[2.3. Description of the application 24](#_Toc137508361)

[2.3.1. Overview of the application 24](#_Toc137508362)

[2.3.2. Application key features and functionality high-level overview 24](#_Toc137508363)

[2.3.3. Technical specifications 25](#_Toc137508364)

[2.3.4. Development process 26](#_Toc137508365)

[2.3.5. Expected user experience 27](#_Toc137508366)

[Chapter 3. Architectural Design and Implementation of the Language Exchange Application 28](#_Toc137508367)

[3.1. Technologies used 28](#_Toc137508368)

[3.1.1. ASP.NET Core 28](#_Toc137508369)

[3.1.2. SQL Server 28](#_Toc137508370)

[3.1.3. SignalR 29](#_Toc137508371)

[3.1.4. Angular 30](#_Toc137508372)

[3.1.5. NgRx 30](#_Toc137508373)

[3.1.6. Database management 31](#_Toc137508374)

[3.2. System design and architecture 31](#_Toc137508375)

[3.3. Frontend development 32](#_Toc137508376)

[3.3.1. User interface design 32](#_Toc137508377)

[3.3.2. Responsiveness and accessibility 33](#_Toc137508378)

[3.4. Backend development 34](#_Toc137508379)

[3.4.1. Projects structure 34](#_Toc137508380)

[SyncLink.Server (API). 35](#_Toc137508381)

[SyncLink.Infrastructure. 35](#_Toc137508382)

[SyncLink.Application 35](#_Toc137508383)

[SyncLink.Common 35](#_Toc137508384)

[3.4.1. Database management 36](#_Toc137508385)

[3.4.2. API and functionality 39](#_Toc137508386)

[3.5. Backend Real-Time interactive features implementation 41](#_Toc137508387)

[3.6. Frontend Real-Time interactive features implementation 42](#_Toc137508388)

[3.7. Implementing Gamified learning features 43](#_Toc137508389)

[3.7.1. Minigames involving language skills 44](#_Toc137508390)

[3.7.2. Text Plot Game – Collaborative Story Game 44](#_Toc137508391)

[3.7.3. Whiteboard – Collaborative Editing 51](#_Toc137508392)

[Chapter 4. Approaches for Testing and evaluation of the web application 53](#_Toc137508393)

[4.1. User testing 53](#_Toc137508394)

[4.1.1. Test Preparation 53](#_Toc137508395)

[4.1.2. Test Execution 53](#_Toc137508396)

[4.1.3. Observations and Data Collection 54](#_Toc137508397)

[4.1.4. Feedback Collection 54](#_Toc137508398)

[4.1.5. Analysis 54](#_Toc137508399)

[CONCLUSIONS 55](#_Toc137508400)

[REFERENCES 56](#_Toc137508401)

# LIST OF ACRONYMS AND ABBREVIATIONS

ANOVA – Analysis of Variance

API – Application Programming Interface

ASP.NET – Active Server Pages .NET

ASP.NET Core – A free, open-source, cross-platform framework for building modern, cloud-based, internet-connected applications.

CALL – Computer-Assisted Language Learning

CLT – Communicative Language Teaching

CPU – Central Processing Unit

CRUD – Create, Read, Update, Delete

CQRS – Command Query Responsibility Segregation

DTOs – Data Transfer Objects

EF Core – Entity Framework Core

JSON – JavaScript Object Notation

LINQ – Language Integrated Query

LMS – Learning Management System

NgRx – Angular Reactive Extensions

ORM – Object-Relational Mapping

SPAs – Single-Page Applications

SQL – Server: Microsoft SQL Server

SQL – Structured Query Language

UI – User Interface

VR – Virtual Reality

# INTRODUCTION

Learning English, spoken by over 1.5 billion people globally, can be challenging for non-native speakers due to its grammatical complexities, vocabulary, and cultural subtleties. These difficulties have been exacerbated by the shift to virtual learning during the global pandemic, which limits face-to-face communication and collaborative learning opportunities. Traditional resources often focus on individual learning, neglecting the benefits of real-time communication, collaborative learning, and practical application, which leads to difficulties in effective communication within English-speaking groups.

Remote learning, while flexible and accessible, has introduced new challenges including lower student engagement and insufficient face-to-face interaction, hindering the practice of spoken English and understanding of cultural nuances. Conventional tools lack collaboration and community-building elements, especially crucial in a remote learning setup.

To address these issues, a user-friendly web application is to be developed to facilitate language learning and cultural exchange among English learners. The application merges communication with gamified learning to create an immersive and collaborative environment, catering to the diverse needs and proficiency levels of learners. The aim is to bridge the communication gap, foster effective collaboration, and enhance English language skills and cultural understanding amongst students, irrespective of their geographic location.

This work aims to answer the following research questions:

• How does the language barrier impact collaboration and communication among students in English language groups, particularly in a remote learning environment?

• How can technology be leveraged to help non-native English speakers improve their language skills while also fostering better communication and collaboration?

• What are the benefits and challenges of incorporating real-time communication and gamification into a web application aimed at language and cultural exchange among English language group students?

• How does ASP.NET Core, SQL Server, Angular, and SignalR contribute to the development of an effective and interactive language learning application?

• What are the user feedback and performance results of the developed application and how do they inform possible future improvements?

These questions will guide the research and development process, providing a clear direction for the design and implementation of the application.

# Chapter 1. Review of Related Literature and Existing Language Learning Applications

## 1.1. Current language learning techniques

Language learning, particularly of globally prevalent languages like English, involves a range of techniques. Traditional methods lean on classroom-based teaching employing techniques like grammar translation, the direct method, and the audio-lingual method (Richards & Rodgers, 2014) [5], often criticized for their lack of context and real-world application. Conversely, communicative language teaching (CLT) focuses on meaningful communication and real-world tasks, promoting interaction and linguistic fluency (Nunan, 2014) [4]. The rise of technology has spurred digital tools like Computer-Assisted Language Learning (CALL), enhancing language learning with interactivity, immediate feedback, and diverse resources (Beatty, 2013) [1]. Language apps like Duolingo, Babbel, and Rosetta Stone employ methods like spaced repetition, gamification, and immersion (Godwin-Jones, 2014) [3], but often neglect the communicative, social, and cultural aspects of language learning.

Therefore while traditional and digital language learning techniques have their merits, there seems to be a gap in integrating these methods into a comprehensive, socially interactive, and culturally inclusive language learning tool. The next section will further explore the role of online platforms and tools in language learning and their limitations.

Table 1.1.

Language learning techniques pros and cons

| **Technique** | **Description** | **Pros** | **Cons** |
| --- | --- | --- | --- |
| Direct Method | Teaches the language directly without using the learner's native language; emphasis on everyday vocabulary and grammar | Provides a more immersive learning experience | Might be challenging for beginners as there's no translation involved |
| Audio-Lingual Method | Focuses on pattern drills and repeated practice of dialogues and phrases | Useful for improving pronunciation and fluency | Can become monotonous and boring; lacks focus on meaningful communication |
| Communicative Language Teaching (CLT) | Emphasizes interaction and real-world communication skills | Promotes fluency and understanding of language in context | Might neglect the importance of accurate grammar |
| Computer-Assisted Language Learning (CALL) | Uses computers and multimedia resources to assist language learning | Offers interactivity and instant feedback; caters to diverse learning styles | Often lacks the social aspect of language learning |

Continuation of table 1.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Language Learning Apps (e.g., Duolingo, Babbel, Rosetta Stone) | Use gamification, spaced repetition, and other methods to teach language | Makes learning engaging and fun; allows learning at one's own pace | Often fails to cover the cultural aspects of language; lacks real-world communication practice |
| Grammar Translation | Focuses on the grammatical rules of the language, usually involving translation between the target language and the native language | Helps students understand the structural complexities of the language | Lacks context and real-world application; Not conducive for improving speaking and listening skills |

## 1.2. The role of technology in language learning

In the past few decades, technology has transformed the field of language learning. The advent of the Internet, mobile devices, and software applications has enabled learners to access language learning resources at any time, from anywhere in the world. This section of the study will explore the various roles that technology plays in language learning, emphasizing its advantages and potential drawbacks.

Access to Resources: Technology provides learners with a wealth of resources for language learning, including online dictionaries, grammar guides, and language learning websites and apps. There are also online language communities where learners can interact with native speakers and other learners.

* Interactivity and Engagement: Interactive software and applications can engage learners in a way that traditional textbooks cannot. This can include multimedia content, quizzes, games, and virtual reality experiences that make learning more interesting and fun.
* Personalization: Technology allows for personalized learning experiences. Learners can work at their own pace, focus on their areas of interest, and choose the learning methods that suit them best.
* Real-Time Feedback: Many language learning apps offer immediate feedback, allowing learners to correct their mistakes in real time. This can accelerate the learning process and increase learner confidence.
* Language Practice: Technology offers learners the opportunity to practice their language skills in realistic contexts. For example, they can participate in online discussions, listen to podcasts in the target language, or use language exchange apps to communicate with native speakers.

## 1.3. Online learning platforms and tools

The advent of technology has yielded diverse online platforms and tools for language learning. Popular platforms like Duolingo, Babbel, and Rosetta Stone have transformed self-paced learning with their unique approaches—gamification (Duolingo), conversation-focused lessons (Babbel), and immersive learning (Rosetta Stone) (Chapelle, 2019) [2]. Language exchange platforms such as Tandem and HelloTalk connect learners worldwide for language practice through text, voice, or video. Learning Management Systems (LMS) like Canvas, Blackboard, and Moodle offer course management, assessment tools, discussion forums, and self-study resources. Despite advancements, these tools have limitations: language apps often lack real-world conversation opportunities, language exchange platforms may lack structure, and LMS might lack the interactive features of language-specific platforms.

Below is a table providing an overview of the platforms that were previously mentioned:

Table 1.2.

Existing language learning platforms overview

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform** | **Description** | **Pros** | **Cons** |
| Duolingo | Gamified language learning app | Engaging and fun; Immediate feedback | May lack depth in grammar and culture aspects |
| Babbel | Language learning app focusing on conversation | Lessons developed by language experts; Focus on conversation | Subscription-based; Might be less engaging compared to gamified apps |
| Rosetta Stone | Immersive language learning platform | Provides comprehensive lessons; Focus on pronunciation | Relatively expensive; Some find the immersive method challenging |
| Tandem | Language exchange platform | Offers real-world conversation practice with native speakers | Lacks structured learning; Quality of learning depends on the partner |
| Canvas/Blackboard/Moodle | Learning Management Systems | Comprehensive tools for course management and self-study | Might lack the engaging and interactive features of language-specific platforms |

## 1.4. Real-time communication in online learning

Real-time communication (RTC) in online learning involves live engagement between learners and instructors or among learners. It employs video conferencing, instant messaging, and collaborative online spaces, offering immediate feedback and active learning. Particularly crucial for language learning, RTC provides the opportunity for spontaneous and appropriate language use in conversations. Research indicates RTC's importance in developing speaking and listening skills (Wang, 2015) [6] and boosting learners' confidence and motivation (Sun, 2016) [7]. However, online language platforms often inadequately implement RTC, with apps like Duolingo and Rosetta Stone lacking real-time interaction, and language exchange platforms like Tandem missing structured learning environments.

Below is a table illustrating the presence of real-time communication features in various popular platforms:

Table 1.3.

Real-time features in popular language learning platforms

|  |  |
| --- | --- |
| **Platform** | **Real-time Communication Feature** |
| Duolingo | Limited to forums, no direct real-time interaction |
| Babbel | No real-time interaction |
| Rosetta Stone | No real-time interaction |
| Tandem | Text, voice, and video chat |
| Canvas/Blackboard/Moodle | Depending on the course setup, may include video conferencing, instant messaging |

This review thus suggests a gap in the current landscape of online language learning tools: the need for a platform that combines structured learning, real-time communication, and cultural exchange. The work aims to address this gap by designing and implementing a web-based application with these features.

## 1.5. Advantages and disadvantages of current language learning applications

### 1.5.1. Advantages

Flexibility: Learning apps offer self-paced learning, suitable for individuals with busy schedules.

Interactive Learning: Most applications employ interactive exercises, enhancing engagement and retention.

Instant Feedback: Immediate error correction, a feature of these applications, ensures effective learning.

Accessibility: Language learning is more accessible with smartphones, as apps allow learning anywhere, anytime.

Resources: Apps often offer diverse resources, aiding a well-rounded language understanding.

### 1.5.2. Disadvantages

Lack of Interaction: A significant downside is the absence of personal interaction, vital in traditional classroom settings.

Over-reliance on Technology: There's a risk of learners becoming dependent on apps, neglecting other valuable resources.

Variable Quality: The quality of content varies across apps, with some neglecting crucial language aspects.

Limited Customization: Many apps lack in-depth personalization, limiting effectiveness for certain learners.

Cost: Advanced features in many apps are behind a paywall, potentially inaccessible for learners on a budget.

# Chapter 2. Research Design and Application Conceptualization

## 2.1. Study design

### 2.1.1. Research approach

This study used a mixed-method approach, combining quantitative and qualitative research methods. The quantitative approach provided a wide, statistical view of the challenges students face in remote English language learning and the effectiveness of current tools. On the other hand, qualitative methods offered detailed insights into students' perceptions and experiences of language learning, cultural exchange, and technology use. This approach capitalized on both methods' strengths for a balanced, comprehensive understanding of the research problem.

### 2.1.2. Rationale for chosen approach

The complexity of the research problem led to the choice of a mixed-method approach. Quantitative data collection through surveys allowed for objective data gathering about language learning issues. It provided broad numerical data to gauge the scale and prevalence of these challenges. Conversely, qualitative methods like interviews explored students' experiences and perceptions, bringing out the reasons and context behind the numbers. Together, they offered a broader and deeper understanding, each enhancing the reliability of the other. The data collected informed the application design, with quantitative data guiding feature selection and qualitative data influencing their implementation.

### 2.1.3. Research procedures

Problem Identification: Initial stage where challenges for non-native English speakers were recognized, primarily through literature reviews, surveys, and personal observations.

Objectives and Questions: Based on identified issues, the study outlined specific objectives and research questions.

Methodology Design: A mixed-method approach was adopted, tailored to fit the research questions, study context, and available resources.

Data Gathering: Both primary (through surveys, interviews, focus groups) and secondary (literature and online resources) data were collected.

Data Interpretation: Analysis of collected data was performed using appropriate techniques for both quantitative and qualitative information.

Application Building: Utilizing insights from data analysis, the application was designed, coded, and implemented with relevant features.

Evaluation and Testing: Functionality, usability, and effectiveness of the developed application were tested, and user feedback was gathered for evaluation.

Below is a tabular representation of the research procedures and associated methods:

Table 2.1.

Research procedures and associated methods

|  |  |
| --- | --- |
| **Stage** | **Associated Methods** |
| Identification of the Problem | Literature review, preliminary surveys, observations |
| Definition of the Research Objectives and Questions | Review of literature and preliminary data |
| Design of the Research Methodology | Methodological literature review, expert consultation |
| Data Collection | Surveys, interviews, focus groups, literature review |
| Data Analysis | Statistical analysis, thematic analysis |
| Application Development | Software development, user-centered design |
| Testing and Evaluation | Usability testing, user feedback collection, performance evaluation |
| Reporting of Findings | Academic writing, data visualization |

## 2.2. Data collection and analysis methods

The robustness of the research findings and the subsequent development of the application hinge on the efficacy of the data collection and analysis methods. Given the mixed-method approach of this study, both quantitative and qualitative data were collected and analyzed. This section elucidates the methods employed for data collection and analysis and their relevance to the research objectives.

### 2.2.1. Data Collection

This research employed both primary and secondary data collection methods. Primary data was gathered directly from students via online surveys, personal interviews, and focus groups. These tools captured student experiences, challenges, and opinions about language learning. Secondary data, sourced from academic literature, online reports, and databases, helped understand the context, identify existing language learning tools, and recognize gaps in current solutions.

### 2.2.2. Data Analysis

Data analysis comprised both quantitative and qualitative approaches. Quantitative data from surveys was checked for accuracy, summarized using statistical measures, and analyzed using inferential statistics and correlational analysis. Visualizations further simplified data understanding. Qualitative data from interviews and focus groups underwent transcription, coding, thematic development, review, and interpretation. The data analysis findings informed the design and development of the application.

## 2.3. Description of the application

### 2.3.1. Overview of the application

The application is an online platform designed to facilitate language learning, cultural exchange, and foster community among English-speaking student groups. By leveraging technology and real-time communication, it aims to enhance English proficiency, cross-cultural understanding, and collaboration, while making learning accessible and engaging.

The platform addresses challenges specific to English language learning, especially those amplified by pandemic-induced remote education. Alongside language skills enhancement, it nurtures a sense of community through real-time interactions, group activities, and cultural exchanges, making learning collaborative and inclusive.

Accessible on various devices, the web application is built with ASP.NET Core for backend, SQL Server as the database, Angular for frontend, and SignalR for real-time interactions.

In essence, it is a comprehensive, interactive, and user-friendly platform that marries language learning with cultural exchange, addressing non-native English-speaking students' challenges.

### 2.3.2. Application key features and functionality high-level overview

The application "SyncLink" incorporates a variety of features designed to stimulate effective and enjoyable language learning, cultural exchange, and real-time communication among students. These features are deliberately designed to address the unique needs and obstacles faced by non-native English-speaking students in English-speaking groups, particularly within the context of a remote learning environment.

Language Learning Mini-Games: These games provide an interactive and enjoyable method for students to challenge and improve their English language skills. Games could involve vocabulary quizzes, grammar exercises, listening tasks, or speaking challenges, among others. Points are awarded based on performance, promoting a friendly competition among students.

Real-Time Chat: A real-time chat feature facilitates instantaneous communication among students. This can be utilized for informal conversations, rapid clarifications, or real-time discussions about the course material or the language learning games.

Text Plot Game: Users initiate a game around a topic, setting a storyline where others contribute entries.

Collaborative Whiteboard: This feature facilitates group study sessions, brainstorming sessions, or problem-solving discussions. Users can write, draw, or annotate on the whiteboard in real-time, encouraging active participation and collaboration.

Word/Phrase of the Day Discussions: Each day, a new word or phrase is featured on the platform. Students can discuss its meaning, usage, synonyms, and contribute sentences using the word/phrase. This encourages daily active engagement with the language.

Collectively, these features of “SyncLink” create a comprehensive language learning environment that promotes active learning, real-time communication, cultural exchange, and community building among students.

### 2.3.3. Technical specifications

“SyncLink” utilizes a front-end and back-end architecture along with real-time communication features to deliver its functionalities.

The backend, built on ASP.NET Core, handles business logic like managing user accounts and data storage/retrieval using a SQL Server database.

The front-end, created with Angular, offers a responsive user interface and ensures scalability with clean, maintainable code.

Real-time communication is facilitated by SignalR, crucial for real-time chat and collaborative features.

These technologies, selected for their proven capabilities and suitability for the application, provide a secure, efficient platform for a smooth user experience in SyncLink.

The figure below depicts application high-level architecture overview:



Fig. 2.1. Application high-level architecture

### 2.3.4. Development process

The creation of “SyncLink” followed a structured three-step procedure: Planning and Design, Implementation, and Testing and Debugging. Initially, the application's goals and needed features were determined through a literature review and data analysis. This led to the design of the application structure, including the database schema, main components, services, and user interface.

Implementation followed, with the backend developed using ASP.NET Core, and frontend via Angular. SignalR was used for real-time communication features, integrating with both backend and frontend.

Post-implementation, extensive testing was performed for functionality and smooth user experience. Unit testing, integration testing, and system testing were conducted. Bugs were addressed and compatibility with various devices and browsers was ensured.

This iterative process, with consistent feedback and testing, allowed for prompt issue resolution, minimizing risks and potential delays in later stages.

### 2.3.5. Expected user experience

“SyncLink” is designed for an optimal user experience, ensuring a user-friendly, engaging, and efficient platform for language learning and cultural exchange. The application promises:

An Intuitive Interface: SyncLink has a straightforward, logically organized user interface that simplifies navigation and usage.

Engaging Interactivity: Features like language mini-games, real-time chat, and a collaborative whiteboard encourage active learning and participation.

Collaborative Learning: SyncLink creates a supportive learning community through discussion forums and real-time chats, fostering motivation and reducing learner isolation.

To sum up, the user experience of SyncLink is designed to be engaging, supportive, and personalized, facilitating effective language learning and cultural exchange in a remote learning environment.

# Chapter 3. Architectural Design and Implementation of the Language Exchange Application

## 3.1. Technologies used

### 3.1.1. ASP.NET Core

ASP.NET Core, an open-source, cross-platform Microsoft framework [8], was chosen as the backend technology for the application due to its multitude of benefits. It provides robust features enabling the implementation of architectural patterns such as Command Query Responsibility Segregation (CQRS) [9] and aspects of Clean Architecture, thereby improving performance, scalability, security, and maintainability. This framework also supports high scalability and cross-platform compatibility, critical for handling large user interactions and promoting flexibility. Its built-in dependency injection and seamless integration capabilities with various databases further enhance its suitability for backend development.

### 3.1.2. SQL Server

SQL Server, Microsoft's relational database management system [10], was selected for this project due to its robust features, compatibility with ASP.NET Core, and extensive data management support. Known for high performance, security, and scalability, it offers data compression, encryption, and compliance with standards, protecting user data integrity.

SQL Server handles complex queries via its transactional system and efficiently stores essential data, including user details, game scores, and discussion records. Its seamless integration with ASP.NET Core simplifies data operations like CRUD (Create, Read, Update, Delete) in the backend, vital for the application's interactivity.

Compatibility with Entity Framework Core [11], an Object-Relational Mapping (ORM) framework, enables translation of object-oriented domain models into a database, thereby simplifying database management and boosting developer productivity. Its support for LINQ (Language Integrated Query) queries allows direct interaction with the database using C#, enhancing its project utility.

In sum, SQL Server serves as a powerful, reliable, and secure solution for the application's data storage and management needs.

### 3.1.3. SignalR

SignalR, a Microsoft-developed library, is integrated into the application to facilitate real-time web functionality [12]. It allows server-side code to instantly push updates to connected clients, enabling bidirectional communication crucial for high interactivity.

SignalR's relevance is notable given the application's collaborative nature, where instantaneous updates and real-time communication are essential. It supports various real-time scenarios including live chatting, real-time gaming, and instant notifications, crucial for user engagement.

Key benefits of SignalR include automatic connection management, creating persistent connections between the client and server, and handling connection interruptions, ensuring a seamless user experience.

SignalR supports various transport protocols (WebSockets, Server-Sent Events, Long Polling) and automatically selects the best transport based on client-server capabilities, ensuring optimal performance.

In conjunction with ASP.NET Core, SignalR offers an effective real-time communication approach, enhancing the application's interactive and collaborative elements, and enriching the overall user experience.

### 3.1.4. Angular

The web application's frontend is built using Angular [13], an open-source framework from Google, chosen for its feature-rich capabilities and scalability, particularly for creating single-page applications (SPAs). Angular uses TypeScript [14], a JavaScript superset, which ensures better code quality and maintainability thanks to compile-time error checking. The framework's component-based architecture promotes code reusability, modularity, and separation of concerns, improving development speed and reducing code duplication. Two-way data binding automates data synchronization between model and view components, enhancing performance. Tools like Angular Material [15] for UI components and RxJS for asynchronous operations, along with compatibility with third-party libraries, allow for extensive customization. Angular Material, providing reusable and accessible UI components, ensures a consistent user experience while decreasing development time.

To sum up, Angular and Angular Material help in crafting an engaging UI for the gamified learning web application.

### 3.1.5. NgRx

NgRx [16], a set of Angular libraries for reactive extensions, is used in the application for state management, especially in complex scenarios with shared and mutable state. Based on Redux, NgRx employs a unidirectional data flow [17] with immutable state, enhancing predictability and traceability of state changes. The application's state is managed through Actions, dispatched in response to events; Reducers, pure functions that return new states based on actions and previous state; and Store, an observable and observer of the state and actions respectively. This approach ensures a single, predictable, and consistent state source, vital for real-time, interactive applications. NgRx's seamless integration with Angular's architecture, RxJS, and Redux DevTools extension makes it an optimal choice for state management in Angular applications.

### 3.1.6. Database management

Database management is crucial in app development, driving functionality through data storage, organization, and retrieval. In our application, we use SQL Server as the primary database technology for its excellent performance, scalability, and security. The database schema, a relational design, includes tables for users, minigames, and daily discussions. SQL enables data manipulation operations, performed using ASP.NET Core and Entity Framework Core in the application's backend for efficient communication with the database. To maintain data consistency and integrity, we implement constraints and transactions. Data security is tackled with measures such as encryption and user authentication. Hence, through SQL Server, ASP.NET Core, and Entity Framework Core, we ensure efficient, secure, and flexible database management.

## 3.2. System design and architecture

The web application uses a three-tier architecture—Frontend (Presentation Layer), Backend (Application Layer), and Database (Data Layer)—for modularity, scalability, and separation of tasks.

The Presentation Layer utilizes Angular, Angular Material, and NgRx. Angular Material designs user interfaces according to Google's Material Design, while NgRx manages application states to enhance performance.

The Application Layer, built on ASP.NET Core, controls application functions and business logic. It uses the CQRS pattern for data handling and employs elements of Clean Architecture for a maintainable design.

The Data Layer uses SQL Server for data storage, retrieval, and manipulation, preserving data consistency through transactions.

SignalR provides real-time communication between server and client, enabling live updates.

In a minigame scenario, a user's answer submission is processed from the Presentation Layer to the Application Layer. The Application Layer updates the game state and the Data Layer stores it. SignalR sends updates to other users instantly.

The system design ensures a scalable, efficient, and engaging user experience.

## 3.3. Frontend development

The frontend of the web application plays a critical role in providing users with an engaging, interactive, and intuitive platform for language and cultural exchange. It is developed using Angular, an advanced JavaScript framework for building single-page applications, supplemented with NgRx for efficient state management and Angular Material for a consistent, appealing UI design.

### 3.3.1. User interface design

The user interface (UI) design blends aesthetics with functionality for a visually pleasing, intuitive, and user-friendly experience. Using Angular Material, a library following Google's Material Design principles, the UI consists of various components, including dialogues, forms, and game interfaces, ensuring visual consistency across the application. Additionally, the UI encourages user engagement with gamified learning features like minigames and interactive elements, making learning enjoyable.

The figure below demonstrates UI related dependencies of one of the application frontend modules:



Fig. 3.1. Example of a UI module dependencies

### 3.3.2. Responsiveness and accessibility

Responsiveness and accessibility are crucial in frontend development due to the diverse range of devices users may employ. The UI is responsive, adjusting to different screen sizes and orientations for optimal interaction across devices. Accessibility practices are observed, including alternative text for images, proper color contrast, and keyboard navigation, ensuring usability for individuals with various abilities. The intuitive UI, responsive layout, and accessibility features offer a user-centric experience that caters to diverse user needs.

The figure below demonstrates a piece of application main UI:



Fig. 3.2. Example of the application prototype UI

## 3.4. Backend development

The backend development of the application forms the core of the system, providing all the necessary logic and operations to support and drive the application's functionalities. In order to facilitate the communication and learning processes, the system relies on an ASP.NET Core web API, a SQL Server database and the use of Entity Framework Core for data access. This chapter outlines the key features and functions that were developed in the backend.

### 3.4.1. Projects structure

The application bakend consist of the five projects.

### SyncLink.Server (API).

This is the entry point of our application and is responsible for handling incoming HTTP requests from clients and returning appropriate responses. All API endpoints will be defined in this project. The controllers in this project will depend on services provided by the SyncLink.Application project. The API project also contains the startup class for the ASP.NET Core application, configuring the necessary services, middleware, and dependency injection.

### SyncLink.Infrastructure.

This project encapsulates all the infrastructural concerns of the application. It is responsible for accessing the database and any other external systems or resources. It contains the Entity Framework Core DbContext and entity configurations, migrations, as well as implementation of the interfaces defined in the Application project. Furthermore, this project handles the implementation details for any third-party services or libraries, like SignalR for real-time interaction.

### SyncLink.Application

The Application project contains the application's business logic. It is where the Command and Query Handlers are implemented as part of the CQRS pattern. These handlers use interfaces (whose implementations will be provided by the Infrastructure project) to interact with the database. This project also contains the definitions for the DTOs (Data Transfer Objects), view models, and any other structures used to exchange data between the API and Application layer.

The purpose of keeping business logic separate from infrastructure and API concerns allows for greater flexibility, easier testing, and better separation of concerns.

### SyncLink.Common

The Common project contains functionalities, utilities, and classes that are shared across the other projects. It might contain constants, extension methods, helper classes, base classes, or any domain-specific abstractions. Placing shared elements in a common project prevents code duplication and promotes code reuse, which in turn can make the solution more maintainable and scalable.

Below is the diagram of backend projects dependencies:



Fig. 3.3. Backend projects hierarchy

### 3.4.1. Database management

The heart of our web application lies in its database, which is responsible for the storage, retrieval, and manipulation of data. We have chosen Microsoft's SQL Server for this project due to its efficient handling of complex data relationships, robustness, scalability, and broad support for complex queries.

The database schema is a comprehensive representation of the various entities in our system, including User, Group, Room, Message, Whiteboard, WhiteboardElement, TextPlotEntry, TextPlotGame, TextPlotVote, UserGroup, and UserRoom. These entities encapsulate the various functionalities and relationships in the system.

* User: Represents a user in the system. Users can belong to multiple groups and rooms, and they can send multiple messages.
* Group: Represents a group in the system. Groups have multiple users and rooms associated with them, and they can host games.
* Room: Represents a room in the system. Rooms can be associated with multiple users and have multiple messages.
* Message: Represents a message sent by a user in a room.
* Whiteboard: Represents a collaborative space where users can create and manipulate WhiteboardElements. Each Whiteboard is associated with a user (Owner) and a Group.
* WhiteboardElement: Represents an element on the Whiteboard. Each element is associated with a user (Author).
* TextPlotEntry: Represents an entry in a TextPlotGame. Each entry is associated with a user and can receive multiple votes.
* TextPlotGame: Represents a game in the system. Each game is associated with a group and has multiple entries.
* TextPlotVote: Represents a vote cast on a TextPlotEntry. Each vote is associated with a user.
* UserGroup: Represents the membership of a user in a group.
* UserRoom: Represents the membership of a user in a room.

Entity Framework Core is used to interact with the SQL Server database. This ORM allows us to work with the database using .NET objects, which reduces the amount of data-access code that needs to be written and maintained. It also provides an abstraction layer over the underlying database system, improving the resilience of the application to potential changes in the database technology.

The database management involves creating, reading, updating, and deleting (CRUD) records in the database. These operations are performed by various API endpoints, and the data returned by these endpoints is formatted as JSON for easy use in the Angular frontend.



Fig. 3.4. Application database relations schema diagram

In the next section the various APIs and functionalities provided by the backend will be outlined.

### 3.4.2. API and functionality

In order to create a platform for English language and cultural exchange for students, several functionalities were developed on the backend of the application. The features provided by the application's backend, and exposed through APIs, are integral to ensure a collaborative and gamified learning environment.

In the backend, we've employed a layered architecture utilizing principles of CQRS and Clean Architecture. It facilitates separation of concerns and makes the codebase more maintainable and adaptable to changes. MediatR is used for dispatching commands and queries, which are processed by separate handlers, and AutoMapper is utilized for mapping between various DTOs and domain entities.

The following are some of the key controllers that have been developed for the application:

* AuthController

This controller is responsible for authentication. Students can log in and register using the methods provided by this controller.

* GroupsController

This controller is essential for group management. Students can form and join different groups for collaboration and interaction. It also provides features to search for groups, get members of a group, create and manage rooms within groups and handle private and room messages.

* MessagesController

This controller handles the sending of messages, which is a crucial part of the communication between students.

* RoomsController

This controller is responsible for creating rooms, which are sub-groupswhere students can collaborate on specific topics or activities.

* TextPlotGamesController

This controller manages a text plot game, where students can practice their language skills in a fun and interactive manner.

* WhiteboardsController

This controller manages the whiteboard feature of the application, which can be used for interactive learning and collaboration within a group.

Every API endpoint defined in the controllers provides the necessary functionality for the collaborative web app, facilitating both communication and learning. For instance, using the "register" and "login" endpoints in the AuthController, students can create an account and log into the application. They can use the various endpoints provided by the GroupsController to create groups, join groups, send messages within the group, etc.

Similarly, the TextPlotGamesController offers endpoints for creating and participating in text-plot games, which adds the element of gamification to the learning process. The WhiteboardsController enables students to visually express their ideas and enhance their learning experience through collaboration.

The following table provides an overview of the primary controllers used in the application, their purpose, and some of their key endpoints:

Table 3.1.

Listing of the main application API endpoints

|  |  |  |
| --- | --- | --- |
| **Controller** | **Purpose** | **Key Endpoints** |
| AuthController | Handles user authentication | POST /api/auth/login  POST /api/auth/register |
| GroupsController | Manages groups and group interactions | POST /api/groups  GET /api/groups/{id}  GET /api/groups/{id}/members  GET /api/groups/search  GET /api/groups/{id}/rooms |
| MessagesController | Facilitates sending messages | POST /api/messages |
| RoomsController | Creates and manages rooms | POST /api/rooms |
| TextPlotGamesController | Manages text plot games | POST /api/features/textplotgames/start  POST /api/features/textplotgames/submitEntry  POST /api/features/textplotgames/endGame |

Continuation of table 3.1.

|  |  |  |
| --- | --- | --- |
| **Controller** | **Purpose** | **Key Endpoints** |
| WhiteboardsController | Manages the whiteboard feature for group collaboration | POST /api/groups/{groupId}/features/whiteboards  GET /api/groups/{groupId}/features/whiteboards/{whiteboardId} |

The following picture shows part of OpenApi Swagger documentation of the application API:



Fig. 3.5. Part of application’s API Swagger specification

## 3.5. Backend Real-Time interactive features implementation

Our application's real-time interactivity is achieved through SignalR, enabling instant content updates and fostering dynamic communication among students. Central to this interactivity is the SyncLinkHub class, inheriting from SignalR's Hub, which facilitates real-time features like messaging, whiteboard updates, and game events. Methods within SyncLinkHub align with user actions, and communication with various notification services allows changes to be propagated to all connected users.

Key real-time features include:

Messaging: The MessageReceived method in ISyncLinkHub interface triggers real-time message notifications within groups.

Whiteboard Updates: The BoardUpdated method in ISyncLinkHub reflects shared whiteboard changes to all group users instantly.

Text Plot Games: Real-time game events correspond to GameStarted, NewEntry, VoteReceived, and GameEnded methods in ISyncLinkHub interface.

In summary, real-time interactive features are crucial to the application, creating an interactive learning environment. SignalR ensures seamless real-time communication and data synchronization among users.

## 3.6. Frontend Real-Time interactive features implementation

The application uses a SignalRService to manage real-time updates from the server to the Angular frontend. The service operates a hubConnection and uses Subjects from the RxJS library to propagate real-time changes, such as shared whiteboard alterations. SignalRService methods align with real-time events, e.g., opening/closing a group or updating the whiteboard. Event listeners are set in the SignalRService constructor for actions like receiving new messages or updates to the shared whiteboard. These mechanisms ensure updated, synchronized views for all connected clients. Additionally, SignalR's integration with Angular and NgRx offers a neat and scalable approach for handling real-time updates, enabling predictable management of application state.

Below is a diagram of dependencies of real-time services on Frontend. The central point is dedicated to the SignalR serivice, which is responsible for establishing persisted connection to the server and then dispatching/listening to actions providing the API which is then used by NgRx effects, which dispatch appropriate actions to change the application state:



Fig. 3.6. Frontend real-time service dependencies

To conclude, the real-time interactive features are a vital aspect of the application, creating a live, collaborative environment that enhances learning and interaction among students. Leveraging SignalR in conjunction with Angular and NgRx helps achieve a robust, scalable, and responsive real-time system.

## 3.7. Implementing Gamified learning features

In this project, gamified learning features are employed as a strategy to facilitate both language learning and cultural exchange among the students. This approach recognizes the power of gaming in boosting student engagement, fostering motivation, and creating a more enjoyable and interactive learning environment. Two of the major gamified learning features embedded within the web application include minigames involving language skills and word/phrase of the day discussions.

### 3.7.1. Minigames involving language skills

The application includes multiplayer minigames that focus on language skills, such as word matching or sentence building. These games, facilitated by SignalR, encourage real-time interaction, cooperation, and competition among students.

The example below demonstrates a simple type dependency diagram of a WordsChain game entities:



Fig. 3.7. Type dependency diagram of an example game domain entities



Fig. 3.8. Type dependency diagram of communication use cases entities

### 3.7.2. Text Plot Game – Collaborative Story Game

The TextPlotGame is an interactive feature that promotes language proficiency and user engagement. Users initiate a game around a topic, setting a storyline where others contribute entries.

Submitted entries are voted on by users on a scale of 1 to 10. Comments can also be added to entries, encouraging deeper interaction. A one-minute countdown begins with the first vote, adding a sense of urgency and excitement.

The highest scoring entry is added to the story at the end of the voting period, and the cycle repeats. Users can view game statistics at any time, fostering competition and tracking progress.

Thus, the TextPlotGame provides an engaging platform for practicing English, stimulating creative thinking, writing, and critical analysis.

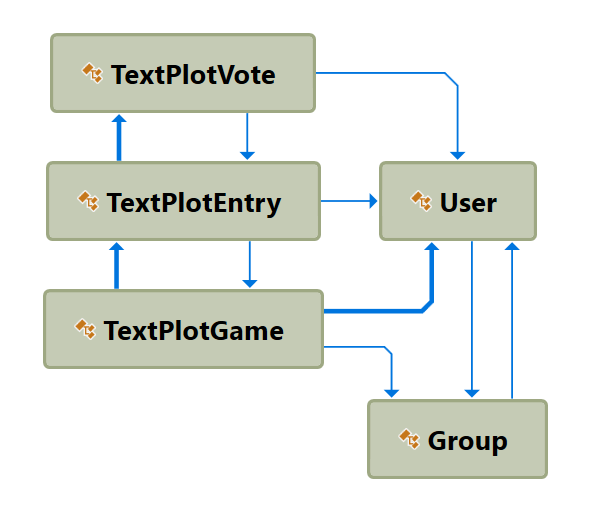


Fig. 3.9. Type dependency diagram of TextPlotGame entities

To better illustrate how the TextPlotGame feature works within the application, let's walk through the user interface (UI) and the process from the user's perspective.

**Accessing the TextPlotGame Feature.**

The first step involves navigating to the TextPlotGame tab, located in the application's main features section. Here, users can view all the active games and participate in them.

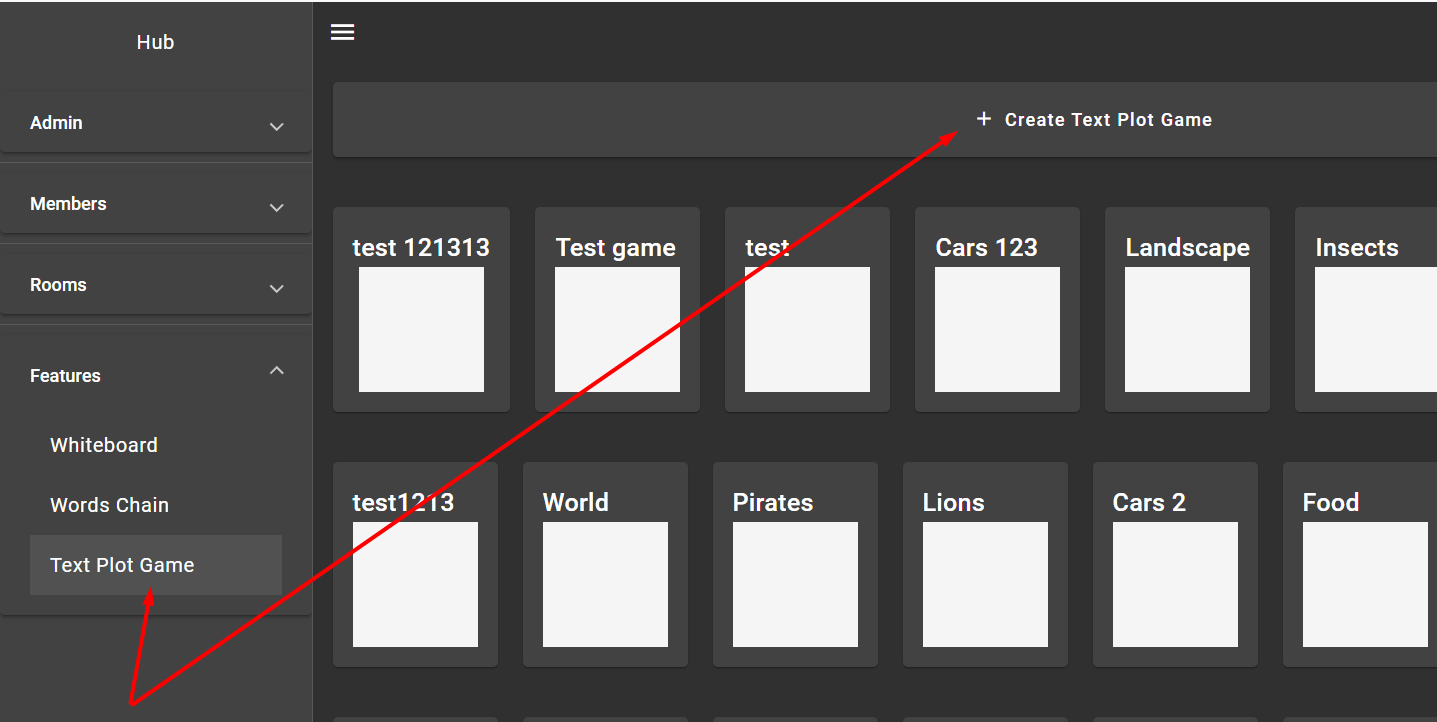


Fig. 3.10. Screenshot showing the TextPlotGame tab in the application

**Selecting or Starting a Game.**

Upon entering the TextPlotGame section, the user can either choose an existing game from the displayed list or start a new game by clicking on the 'Start Game' button. If a user decides to start a new game, they will be prompted to enter the game's topic.

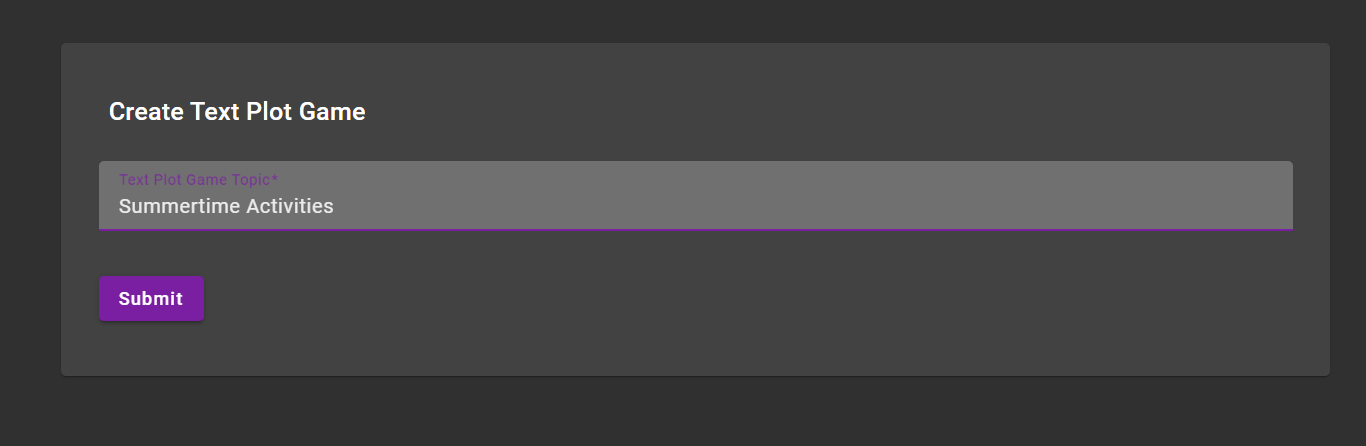


Fig. 3.11. Screenshot showing the form for starting a new game

**Notification of a New Game.**

After the game host submits the topic, a notification is sent to all users, informing them about the new game. The game is also displayed on the main page of the TextPlotGame section.

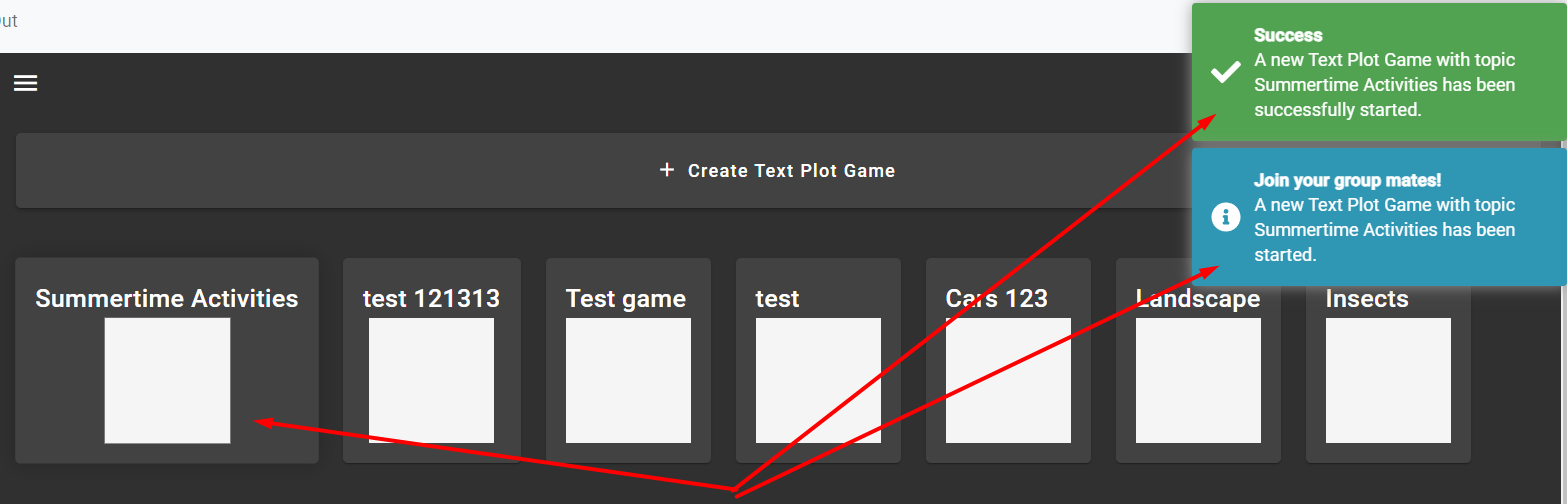


Fig. 3.12. Screenshot showing the notification of the new game

**Navigating the Game Interface.**

Once inside a game, the game host can begin by submitting an initial entry. Users then add their own entries, which are displayed for everyone to see and vote on. The interface is designed to be user-friendly, promoting easy interaction and navigation.

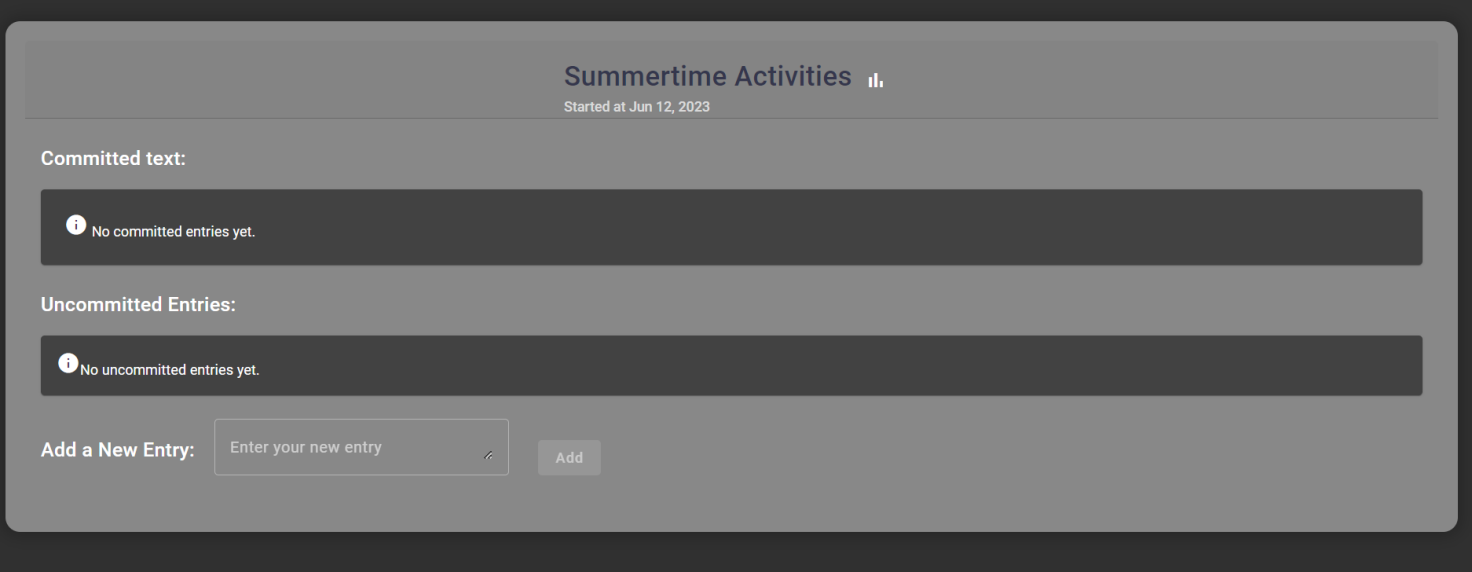


Fig. 3.13. Screenshot showing the game interface with the initial entries

**Submitting game entries.**

When the user submits an entry, it propagates to other users partcipating in the game allowing them to vote.

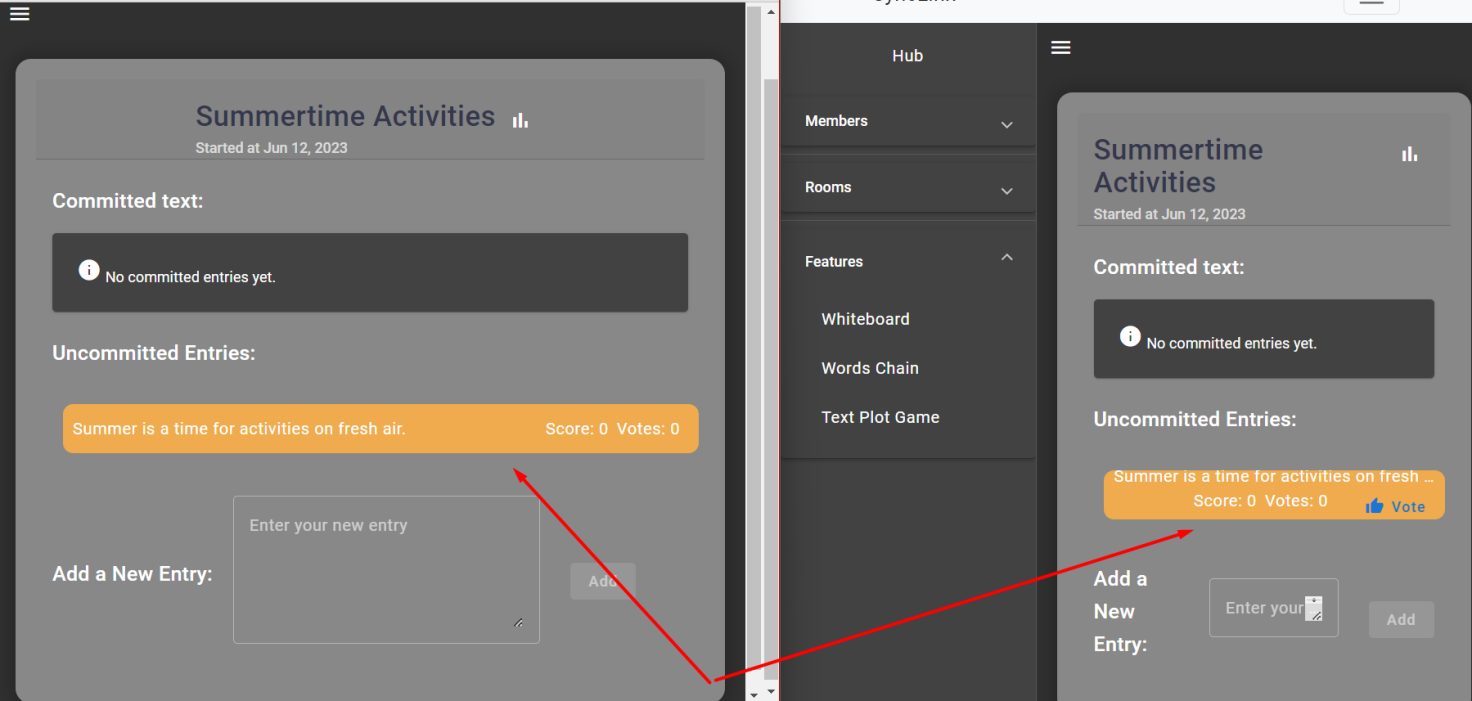


Fig. 3.14. Submitting a game entry.

**Voting and Viewing Statistics.**

After a game entry is submitted, other users can vote it assigning a score from 1 to 10. Optionally, users can add comment which will later be displayed in statistics:

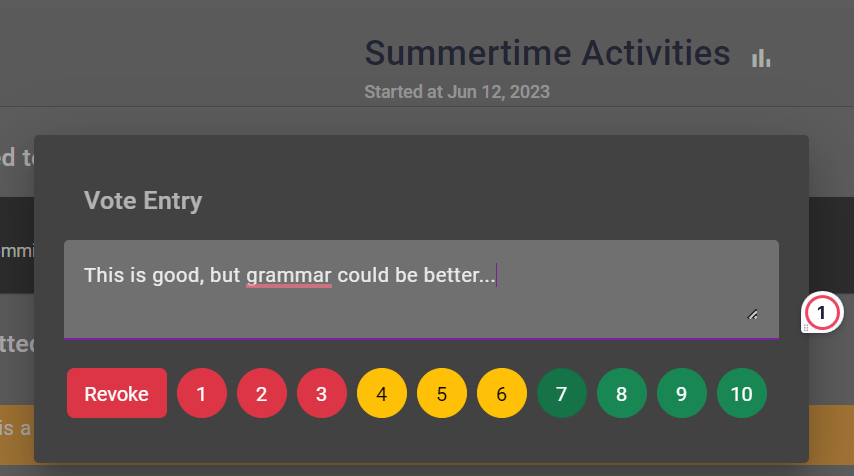


Fig. 3.15. Screenshot showing the voting process.

When the voting is initiated, a timer is triggered showing a 1-minute proggress bar.

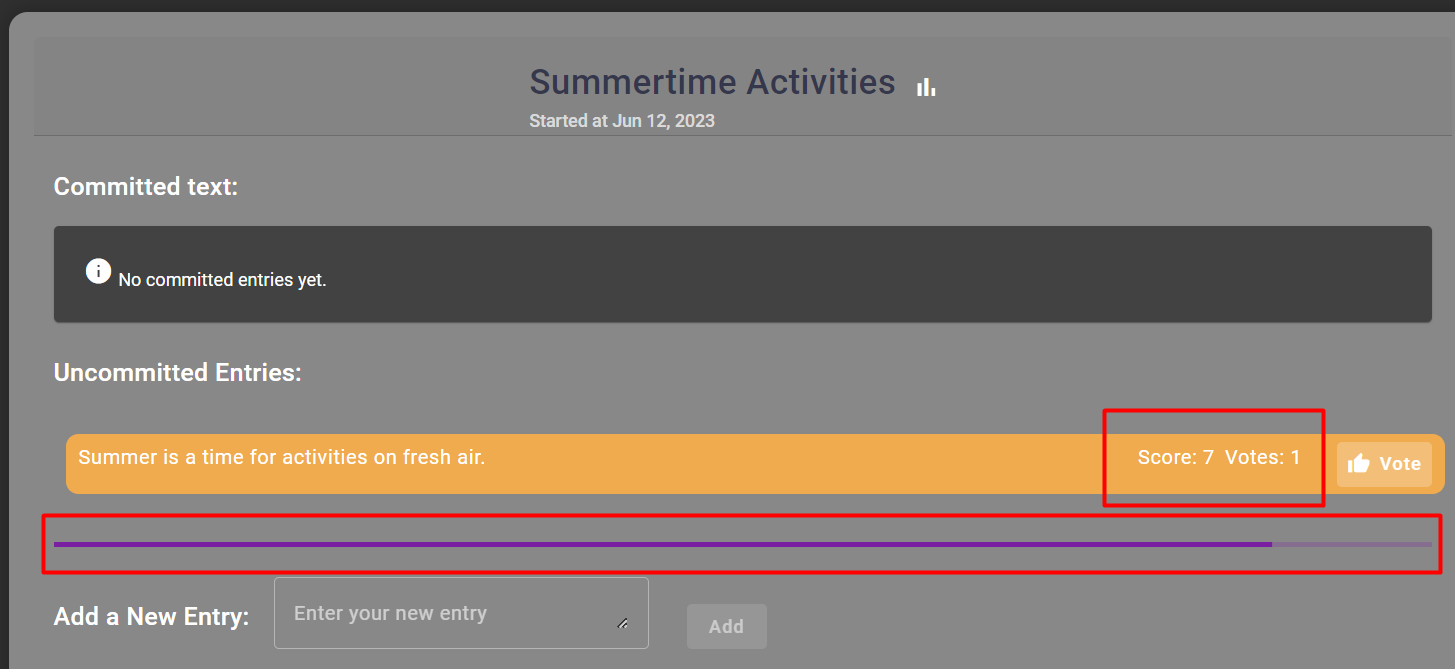


Fig. 3.16. Screenshot of voting triggering a proggress bar.

After the voting round an entry with most votes is committed to the story:

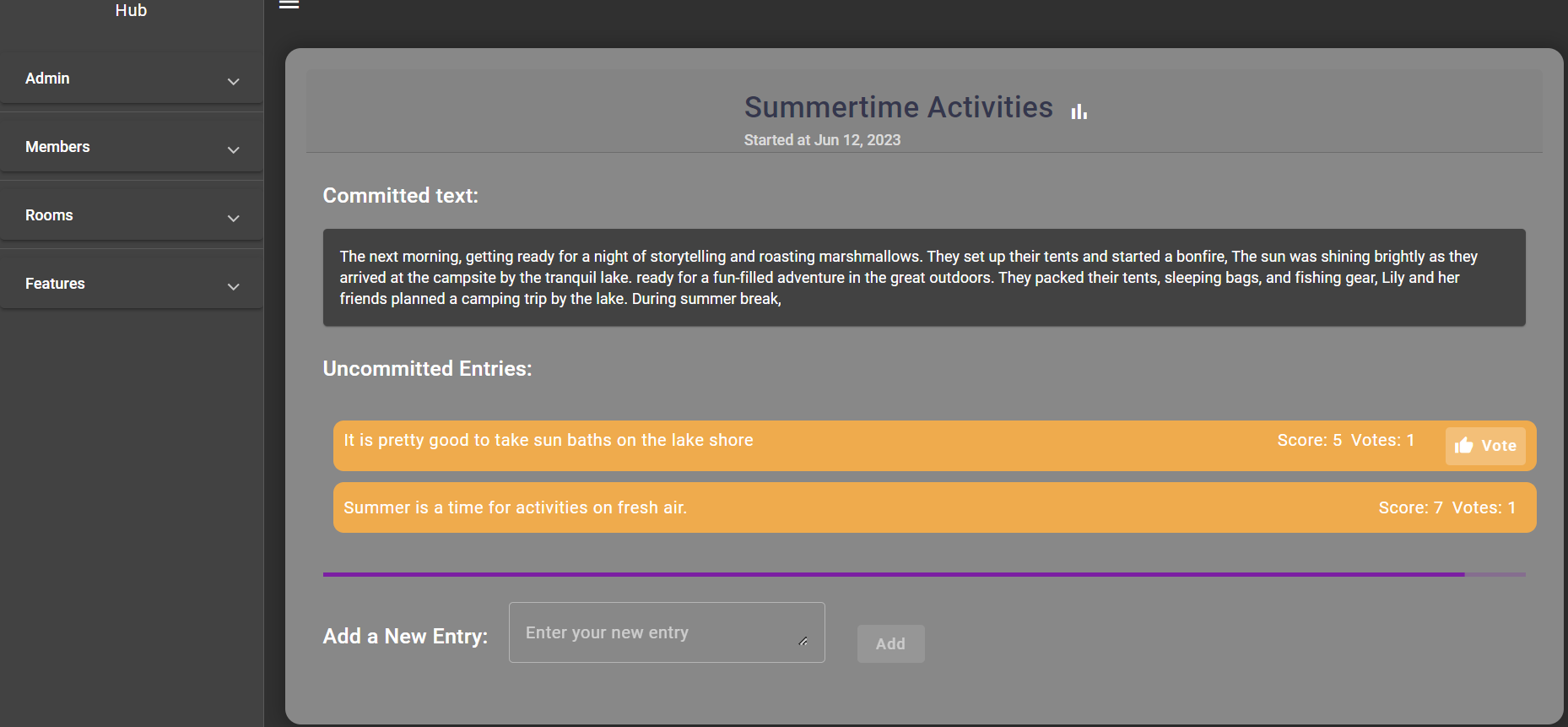


Fig. 3.17. Game after commiting a couple of entries.

At any moment of time users can view game statistics with leaderboard and comments received by other participants:

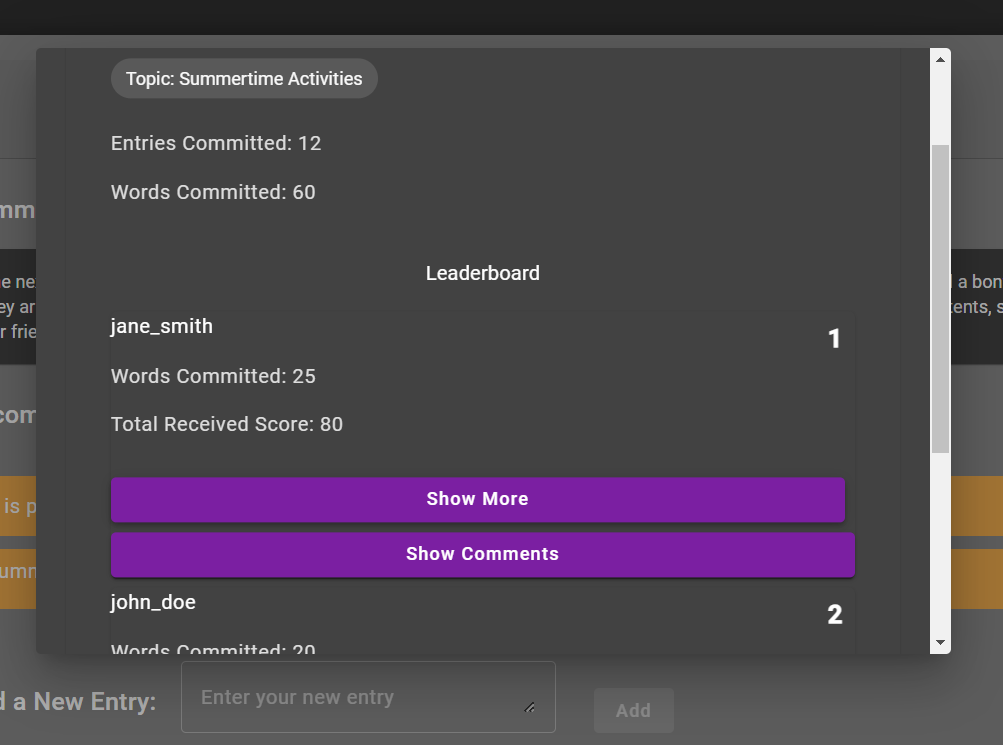


Fig. 3.18. Text Plot Game statistics tab.

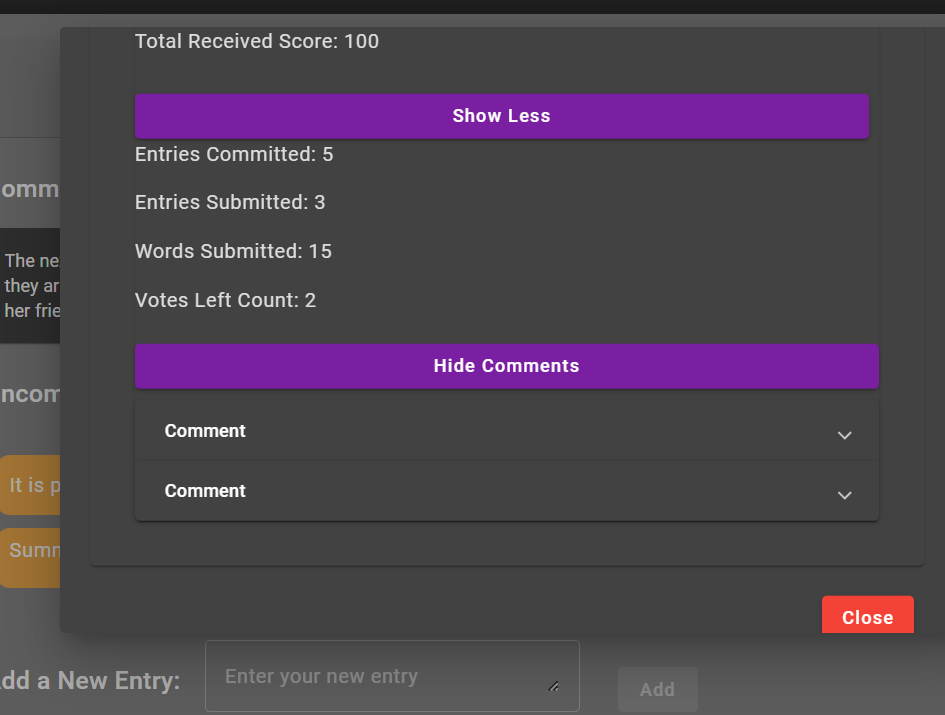


Fig. 3.19. Text Plot Game detailed statistics.

### 3.7.3. Whiteboard – Collaborative Editing

An additional feature of the application is the Collaborative Whiteboard, designed to foster real-time interaction and collaboration among users.



Fig. 3.20. Whiteboard feature selection.

This whiteboard acts as a virtual canvas where users can draw, write, or edit images together in real-time. Users can use a variety of tools to express their thoughts, solve problems, or just for fun, such as pens of different colors and thicknesses, shapes, text, and erasers.

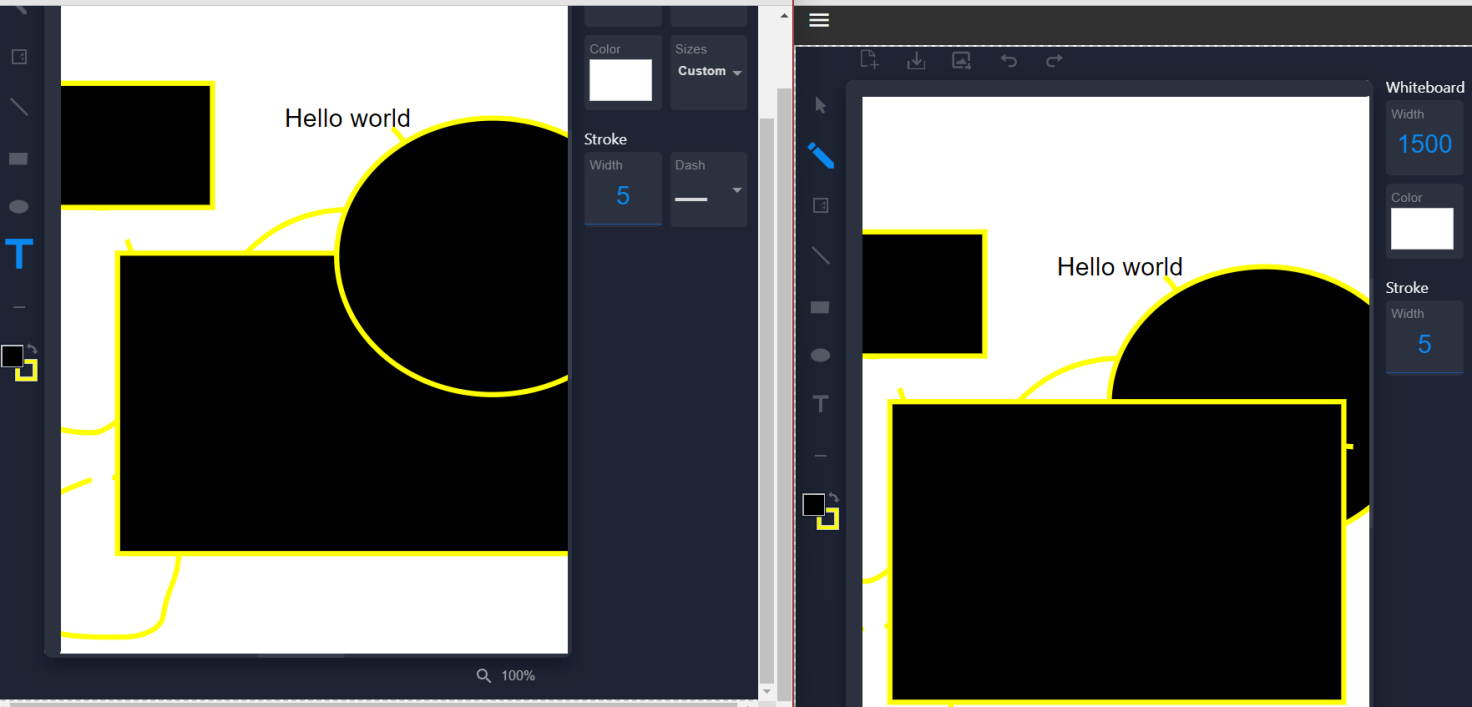


Fig. 3.21. Whiteboard updates in real-time.

The whiteboard can import images, allowing users to collaboratively edit or annotate them. This can be useful for explaining concepts, brainstorming ideas, or discussing the cultural contexts of images.

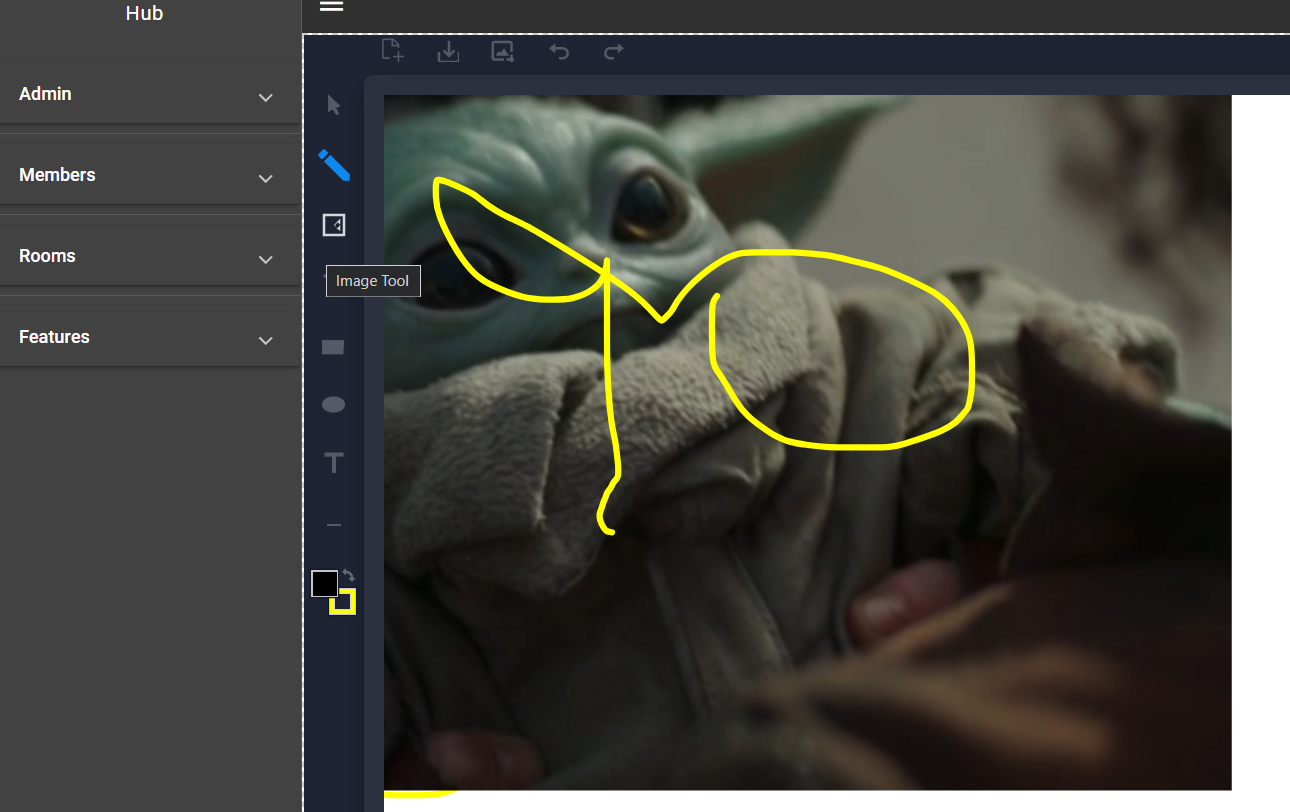


Fig. 3.2. Whiteboard image importing feature.

SignalR technology ensures that every user's action on the whiteboard is instantly visible to all other participants. This real-time interaction encourages cooperation and makes the learning process more engaging and interactive.

Thus, the Collaborative Whiteboard feature enriches the learning experience by enabling users to visually express ideas, engage in group problem-solving, and interact in an immersive, real-time environment.

# Chapter 4. Approaches for Testing and evaluation of the web application

The importance of testing in any software development process cannot be overemphasized. Testing ensures that all software components are functioning as intended, providing confidence to both the developers and the end users. For our web application aimed at facilitating language and cultural exchange for English group students, a systematic and rigorous approach was undertaken in its testing and evaluation.

## 4.1. User testing

User Testing forms a crucial aspect of any application's development process as it aims to evaluate the product's functionality, usability, and consistency from the user's perspective. The primary focus here is to ensure the user's needs are met and any possible confusion or difficulties are addressed and eliminated.

In the case of our language and cultural exchange web application, a group of English group students participated in the testing process. To conduct this User Testing, the following steps were taken:

### 4.1.1. Test Preparation

The first step was preparing the test environment. All features and functionalities of the application were ensured to be operational. A set of tasks encompassing various application functions was created for the users to perform. These tasks were designed to mimic real-life scenarios that users might encounter while using the application.

### 4.1.2. Test Execution

Students were given a brief tutorial on how to use the application, after which they were presented with the tasks. They were asked to:

* Create a new user profile
* Join a group discussion
* Start a private chat with another user
* Contribute to the 'Word/Phrase of the Day' discussion
* Participate in the minigames focusing on language skills

While performing these tasks, students were encouraged to think aloud about their actions and experiences, providing real-time feedback.

### 4.1.3. Observations and Data Collection

During the execution of the tasks, the facilitator observed the users' interactions with the application, noting any difficulties faced, errors encountered, and the overall ease of use. Users were also asked to rate the ease or difficulty of each task on a predetermined scale.

### 4.1.4. Feedback Collection

After completion of the tasks, students were asked to provide their feedback on various aspects of the application. This included their overall experience, the application's ease of use, the usefulness of the features, any encountered difficulties, and suggested improvements.

### 4.1.5. Analysis

All the feedback and observational data were compiled and analyzed to identify any trends, common difficulties, and usability issues. These findings were instrumental in understanding the application from a user’s perspective and served as valuable input for future enhancements.

The ultimate goal of User Testing was to ensure that the application is intuitive and user-friendly, and that it effectively aids English group students in language learning and cultural exchange. The feedback and findings from this phase were used to refine the application to better suit the needs and preferences of the users.

# CONCLUSIONS

The development of a web-based language and cultural exchange application for English group students has been a rewarding journey, providing valuable lessons in design, implementation, and user engagement. This application, a blend of communication, gamification, and real-time interaction features, stands as an innovative response to the remote learning challenges highlighted during the pandemic.

This project has demonstrated the effectiveness of ASP.NET Core, SQL Server, Angular, SignalR, and other related technologies in creating a dynamic and interactive language learning platform. It also showcased how elements of the CQRS and clean architecture principles could contribute to a maintainable and scalable backend design.

The successful deployment of the application and the positive results obtained from the user and performance testing indicate that such a platform can contribute to overcoming language barriers and improving collaboration among students. It demonstrated the potential of technology and innovative web design in transforming the landscape of language learning and intercultural communication.

However, it is important to remember that the journey does not stop here. The ever-evolving nature of technology and the diverse needs of users necessitate continuous iteration and improvement. The suggestions put forth for future research underline the unending possibilities for enhancements and further exploration.

In conclusion, the realization of this project underlines the possibility of building more inclusive, collaborative, and engaging learning environments through the right mix of technology, pedagogical strategy, and user-centric design. It provides hope and direction for those striving to create tools and resources that not only educate but also connect and empower individuals across cultural and linguistic boundaries.

## REFERENCES

1. Beatty, K., Teaching & Researching: Computer-Assisted Language Learning – Routledge, 2013;
2. Chapelle, C. A., Computer-Assisted Language Learning: Learners, Teachers, and Tools – Oxford University Press, 2019;
3. Godwin-Jones, R., Emerging Technologies: Mobile Apps for Language Learning – Language Learning & Technology, 2014, 18(2), pp. 3-19;
4. Nunan, D., The Learner-Centred Curriculum: A Study in Second Language Teaching – Cambridge University Press, 2014;
5. Richards, J. C., Rodgers, T. S., Approaches and Methods in Language Teaching – Cambridge University Press, 2014;
6. Wang, Y., Promoting Active Learning in Online Language Courses through the Integration of Synchronous Communication Tools – International Journal of Distance Education, 2015, 14(3), pp. 105-120;
7. Sun, Y. C., Examining the Effectiveness of Real-Time Interaction in Online Learning: A Comparative Analysis – Journal of Educational Technology & Society, 2016, 19(3), pp. 183-197;
8. Microsoft, ASP.NET Core, [Electronic resource] – Mode of access: https://docs.microsoft.com/en-us/aspnet/core/, 2023;
9. Microsoft, CQRS pattern, [Electronic resource] – Mode of access: https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs, 2023;
10. Microsoft, SQL Server, [Electronic resource] – Mode of access: https://www.microsoft.com/en-us/sql-server/, 2023;
11. Microsoft, Entity Framework Core, [Electronic resource] – Mode of access: https://docs.microsoft.com/en-us/ef/core/, 2023;
12. Microsoft, SignalR, [Electronic resource] – Mode of access: https://docs.microsoft.com/en-us/aspnet/signalr/, 2023;
13. Google, Angular, [Electronic resource] – Mode of access: https://angular.io/, 2023;
14. TypeScript, TypeScript, [Electronic resource] – Mode of access: https://www.typescriptlang.org/, 2023;
15. Angular Material, Angular Material, [Electronic resource] – Mode of access: https://material.angular.io/, 2023;
16. NgRx, NgRx, [Electronic resource] – Mode of access: https://ngrx.io/, 2023;
17. Redux, Redux, [Electronic resource] – Mode of access: https://redux.js.org/, 2023.

APPENDIX A

Backend code

API startup file code

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddConfiguredDbContext(builder.Configuration);

builder.Services.AddConfiguredIdentity(builder.Configuration);

builder.Services.AddApiWithSwagger(builder.Configuration);

builder.Services.AddInfrastructure();

builder.Services.AddApplicationServices();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

await DbSeeder.SeedAsync(app.Services);

}

app.UseRouting();

app.UseMiddleware<ErrorHandler>();

app.UseCors(Constants.AllowAllCorsPolicy);

app.UseAuthentication();

app.UseAuthorization();

app.UseHttpsRedirection();

app.MapControllers();

app.MapHub<SyncLinkHub>("hubs/general");

app.Run();

Application SignalR Hub class code

public interface ISyncLinkHub

{

Task MessageReceived(int? roomId, int? otherUserId, bool isPrivate, MessageDto message);

Task BoardUpdated(int groupId, int whiteboardId, WhiteboardElementDto[] change);

#region TextPlotGame

Task GameStarted(TextPlotGame game);

Task NewEntry(TextPlotEntry entry);

Task VoteReceived(TextPlotVote vote);

Task GameEnded(TextPlotGame game);

#endregion

#region WordsChainGame

Task NewWordsChainGame(WordsChainGameOverviewDto game);

Task NewEntry(WordsChainGameEntryDto entry);

#endregion

}

[Authorize]

public class SyncLinkHub : Hub<ISyncLinkHub>

{

private readonly IUserRepository \_userRepository;

private readonly IMediator \_mediator;

public SyncLinkHub(IUserRepository userRepository, IMediator mediator)

{

\_userRepository = userRepository;

\_mediator = mediator;

}

#region General

public async Task GroupOpened(int groupId)

{

var isUserInGroup = await \_userRepository.IsUserInGroupAsync(UserId, groupId, CancellationToken.None);

if (!isUserInGroup)

{

throw new AuthException(new[] { $"User {UserId} has no access to group {groupId}" });

}

var groupName = HubHelper.GetGroupNameForGroupId(groupId);

await Groups.AddToGroupAsync(ConnectionId, groupName);

}

public Task GroupClosed(int groupId)

{

var groupName = HubHelper.GetGroupNameForGroupId(groupId);

return Groups.RemoveFromGroupAsync(ConnectionId, groupName);

}

public override Task OnConnectedAsync()

{

return base.OnConnectedAsync();

}

public override Task OnDisconnectedAsync(Exception? exception)

{

return base.OnDisconnectedAsync(exception);

}

#endregion

#region Whiteboard

public async Task BoardUpdated(int groupId, int whiteboardId, string updateJson)

{

try

{

var update = JsonConvert.DeserializeObject<WhiteboardElementDto[]>(updateJson);

var command = new UpdateWhiteboard.Command

{

GroupId = groupId,

Update = update?.ToArray()!,

UserId = UserId,

WhiteboardId = whiteboardId

};

var result = await \_mediator.Send(command);

await Clients.GroupExcept(GetGroupNameByGroupId(groupId), new[] { ConnectionId }).BoardUpdated(groupId, whiteboardId, result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

#endregion

#region Utils

protected string GetGroupNameByGroupId(int groupId) => HubHelper.GetGroupNameForGroupId(groupId);

protected string ConnectionId => Context.ConnectionId;

protected int UserId => AppUserIdClaimHelper.RetrieveUserId(Context.User!) ?? throw new AuthException(new[] { "User id should be present in a hub." });

#endregion

}

Application mapping profile

public class ApplicationProfile : Profile

{

public ApplicationProfile()

{

CreateMap<Login.Command, LoginData>();

CreateMap<Register.Command, RegistrationData>();

CreateMap<UserGroup, GroupMemberDto>()

.ForMember(dest => dest.Id, opt => opt.MapFrom(src => src.UserId))

.ForMember(dest => dest.IsCreator, opt => opt.MapFrom(src => src.IsCreator))

.ForMember(dest => dest.IsAdmin, opt => opt.MapFrom(src => src.IsAdmin))

.ForMember(dest => dest.Username, opt => opt.MapFrom(src => src.User.UserName));

CreateMap<UserGroup, GroupDto>()

.ForAllMembers(opt => opt.MapFrom(src => src.Group));

CreateMap<UserRoom, RoomMemberDto>()

.ForMember(dest => dest.Id, opt => opt.MapFrom(src => src.UserId))

.ForMember(dest => dest.IsAdmin, opt => opt.MapFrom(src => src.IsAdmin))

.ForMember(dest => dest.Username, opt => opt.MapFrom(src => src.User.UserName));

CreateMap<UserWordsChainGame, WordsChainGameParticipantDto>()

.ForMember(dest => dest.Id, opt => opt.MapFrom(src => src.UserId))

.ForMember(dest => dest.IsCreator, opt => opt.MapFrom(src => src.IsCreator))

.ForMember(dest => dest.Score, opt => opt.MapFrom(src => src.Score))

.ForMember(dest => dest.Username, opt => opt.MapFrom(src => src.User.UserName));

CreateMap<Group, GroupDto>();

CreateMap<Room, RoomDto>();

CreateMap<User, GroupMemberDto>();

CreateMap<Message, MessageDto>();

CreateMap<Whiteboard, WhiteboardDto>();

CreateMap<WhiteboardElementDto, WhiteboardElement>().ReverseMap();

CreateMap<WhiteboardElementOptionsDto, WhiteboardElementOptions>().ReverseMap();

CreateMap<WordsChainGame, WordsChainGameDto>()

.ForMember(dest => dest.Participants, opt => opt.MapFrom(src => src.Participants))

.ReverseMap();

CreateMap<WordsChainGame, WordsChainGameOverviewDto>().ReverseMap();

CreateMap<WordsChainEntry, WordsChainGameEntryDto>().ReverseMap();

CreateMap<TextPlotGame, TextPlotGameDto>().ReverseMap();

CreateMap<TextPlotEntry, TextPlotEntryDto>().ReverseMap();

CreateMap<TextPlotVote, TextPlotVoteDto>().ReverseMap();

CreateMap(typeof(PaginatedResult<>), typeof(PaginatedResult<>));

}

}

Application Error handler middleware

internal class ErrorHandler : IMiddleware

{

private const string DefaultFallbackErrorMessage = "Something went wrong.";

public async Task InvokeAsync(HttpContext context, RequestDelegate next)

{

try

{

await next(context);

}

catch (Exception e)

{

await HandleExceptionAsync(context, e);

}

}

private static Task HandleExceptionAsync(HttpContext context, Exception exception)

{

return exception switch

{

BusinessException businessException => HandleBusinessExceptionAsync(context, businessException),

RepositoryActionException repositoryException => HandleRepositoryActionException(context, repositoryException),

\_ => HandleUnknownError(context)

};

}

private static Task HandleRepositoryActionException(HttpContext context, RepositoryActionException repositoryException)

{

return repositoryException.Status switch

{

RepositoryActionStatus.NotFound => WriteErrorResponse(context, HttpStatusCode.NotFound, repositoryException.GetClientFacingErrors()),

RepositoryActionStatus.Conflict => WriteErrorResponse(context, HttpStatusCode.Conflict, repositoryException.GetClientFacingErrors()),

RepositoryActionStatus.ValidationFailed => WriteErrorResponse(context, HttpStatusCode.BadRequest, repositoryException.GetClientFacingErrors()),

RepositoryActionStatus.UnknownError => WriteErrorResponse(context, HttpStatusCode.InternalServerError, repositoryException.GetClientFacingErrors()),

\_ => HandleUnknownError(context)

};

}

private static Task HandleBusinessExceptionAsync(HttpContext context, BusinessException businessException)

{

return businessException switch

{

AuthException authException => HandleAuthErrorAsync(context, authException),

\_ => WriteErrorResponse(context, HttpStatusCode.BadRequest, new[] { businessException.Message })

};

}

private static Task HandleAuthErrorAsync(HttpContext context, AuthException authException)

{

return authException switch

{

LoginException loginException => WriteErrorResponse(context, HttpStatusCode.Unauthorized, loginException.Errors),

RegistrationException registrationException => WriteErrorResponse(context, HttpStatusCode.BadRequest, registrationException.Errors),

\_ => HandleUnknownError(context)

};

}

private static Task HandleUnknownError(HttpContext context)

{

return WriteErrorResponse(context, HttpStatusCode.InternalServerError, new[] { DefaultFallbackErrorMessage });

}

private static Task WriteErrorResponse(HttpContext context, HttpStatusCode statusCode, IEnumerable<string>? errors)

{

context.Response.StatusCode = (int)statusCode;

return context.Response.WriteAsJsonAsync(new ErrorDetails

{

StatusCode = statusCode,

Errors = errors?.ToList()

});

}

}

Example of database migrations

namespace SyncLink.Infrastructure.Migrations

{

/// <inheritdoc />

public partial class Whiteboards : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Whiteboards",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: false),

OwnerId = table.Column<int>(type: "int", nullable: true),

GroupId = table.Column<int>(type: "int", nullable: true),

LastUpdatedTime = table.Column<DateTime>(type: "datetime2", nullable: false),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Whiteboards", x => x.Id);

table.ForeignKey(

name: "FK\_Whiteboards\_ApplicationUsers\_OwnerId",

column: x => x.OwnerId,

principalTable: "ApplicationUsers",

principalColumn: "Id",

onDelete: ReferentialAction.SetNull);

table.ForeignKey(

name: "FK\_Whiteboards\_Groups\_GroupId",

column: x => x.GroupId,

principalTable: "Groups",

principalColumn: "Id",

onDelete: ReferentialAction.SetNull);

});

migrationBuilder.CreateTable(

name: "WhiteboardElement",

columns: table => new

{

Id = table.Column<string>(type: "nvarchar(450)", nullable: false),

WhiteboardId = table.Column<int>(type: "int", nullable: false),

WhiteboardElementId = table.Column<int>(type: "int", nullable: false),

Type = table.Column<int>(type: "int", nullable: false),

Value = table.Column<string>(type: "nvarchar(max)", nullable: false),

X = table.Column<int>(type: "int", nullable: false),

Y = table.Column<int>(type: "int", nullable: false),

Rotation = table.Column<int>(type: "int", nullable: false),

Opacity = table.Column<int>(type: "int", nullable: false),

Options\_Width = table.Column<int>(type: "int", nullable: true),

Options\_Height = table.Column<int>(type: "int", nullable: true),

Options\_StrokeWidth = table.Column<int>(type: "int", nullable: true),

Options\_StrokeColor = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_Fill = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_LineJoin = table.Column<int>(type: "int", nullable: false),

Options\_LineCap = table.Column<int>(type: "int", nullable: false),

Options\_Left = table.Column<int>(type: "int", nullable: true),

Options\_Top = table.Column<int>(type: "int", nullable: true),

Options\_FontSize = table.Column<int>(type: "int", nullable: true),

Options\_FontFamily = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_FontStyle = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_FontWeight = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_Color = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_DashArray = table.Column<string>(type: "nvarchar(max)", nullable: false),

Options\_DashOffset = table.Column<int>(type: "int", nullable: true),

Options\_X1 = table.Column<int>(type: "int", nullable: true),

Options\_Y1 = table.Column<int>(type: "int", nullable: true),

Options\_X2 = table.Column<int>(type: "int", nullable: true),

Options\_Y2 = table.Column<int>(type: "int", nullable: true),

Options\_Rx = table.Column<int>(type: "int", nullable: true),

Options\_Ry = table.Column<int>(type: "int", nullable: true),

Options\_Cx = table.Column<int>(type: "int", nullable: true),

Options\_Cy = table.Column<int>(type: "int", nullable: true),

AuthorId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_WhiteboardElement", x => new { x.WhiteboardId, x.Id });

table.ForeignKey(

name: "FK\_WhiteboardElement\_ApplicationUsers\_AuthorId",

column: x => x.AuthorId,

principalTable: "ApplicationUsers",

principalColumn: "Id",

onDelete: ReferentialAction.SetNull);

table.ForeignKey(

name: "FK\_WhiteboardElement\_Whiteboards\_WhiteboardId",

column: x => x.WhiteboardId,

principalTable: "Whiteboards",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_WhiteboardElement\_AuthorId",

table: "WhiteboardElement",

column: "AuthorId");

migrationBuilder.CreateIndex(

name: "IX\_Whiteboards\_GroupId",

table: "Whiteboards",

column: "GroupId");

migrationBuilder.CreateIndex(

name: "IX\_Whiteboards\_OwnerId",

table: "Whiteboards",

column: "OwnerId");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "WhiteboardElement");

migrationBuilder.DropTable(

name: "Whiteboards");

}

}

}

namespace SyncLink.Infrastructure.Migrations

{

/// <inheritdoc />

public partial class AddPlotGame : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "TextPlotGames",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

GroupId = table.Column<int>(type: "int", nullable: false),

CreatorId = table.Column<int>(type: "int", nullable: true),

EndedAt = table.Column<DateTime>(type: "datetime2", nullable: true),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_TextPlotGames", x => x.Id);

table.ForeignKey(

name: "FK\_TextPlotGames\_ApplicationUsers\_CreatorId",

column: x => x.CreatorId,

principalTable: "ApplicationUsers",

principalColumn: "Id");

table.ForeignKey(

name: "FK\_TextPlotGames\_Groups\_GroupId",

column: x => x.GroupId,

principalTable: "Groups",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "TextPlotEntries",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

UserId = table.Column<int>(type: "int", nullable: true),

GameId = table.Column<int>(type: "int", nullable: false),

Text = table.Column<string>(type: "nvarchar(max)", nullable: false),

CreatedAt = table.Column<DateTime>(type: "datetime2", nullable: false),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_TextPlotEntries", x => x.Id);

table.ForeignKey(

name: "FK\_TextPlotEntries\_ApplicationUsers\_UserId",

column: x => x.UserId,

principalTable: "ApplicationUsers",

principalColumn: "Id",

onDelete: ReferentialAction.SetNull);

table.ForeignKey(

name: "FK\_TextPlotEntries\_TextPlotGames\_GameId",

column: x => x.GameId,

principalTable: "TextPlotGames",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "TextPlotVotes",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

UserId = table.Column<int>(type: "int", nullable: true),

EntryId = table.Column<int>(type: "int", nullable: false),

TextPlotEntryId = table.Column<int>(type: "int", nullable: true),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_TextPlotVotes", x => x.Id);

table.ForeignKey(

name: "FK\_TextPlotVotes\_ApplicationUsers\_UserId",

column: x => x.UserId,

principalTable: "ApplicationUsers",

principalColumn: "Id",

onDelete: ReferentialAction.SetNull);

table.ForeignKey(

name: "FK\_TextPlotVotes\_TextPlotEntries\_EntryId",

column: x => x.EntryId,

principalTable: "TextPlotEntries",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_TextPlotVotes\_TextPlotEntries\_TextPlotEntryId",

column: x => x.TextPlotEntryId,

principalTable: "TextPlotEntries",

principalColumn: "Id");

});

migrationBuilder.CreateIndex(

name: "IX\_TextPlotEntries\_GameId",

table: "TextPlotEntries",

column: "GameId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotEntries\_UserId",

table: "TextPlotEntries",

column: "UserId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotGames\_CreatorId",

table: "TextPlotGames",

column: "CreatorId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotGames\_GroupId",

table: "TextPlotGames",

column: "GroupId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotVotes\_EntryId",

table: "TextPlotVotes",

column: "EntryId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotVotes\_TextPlotEntryId",

table: "TextPlotVotes",

column: "TextPlotEntryId");

migrationBuilder.CreateIndex(

name: "IX\_TextPlotVotes\_UserId",

table: "TextPlotVotes",

column: "UserId");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "TextPlotVotes");

migrationBuilder.DropTable(

name: "TextPlotEntries");

migrationBuilder.DropTable(

name: "TextPlotGames");

}

}

}

namespace SyncLink.Infrastructure.Migrations

{

/// <inheritdoc />

public partial class AddFeed : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Discussions",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

WordPhrase = table.Column<string>(type: "nvarchar(max)", nullable: false),

DescriptionOrQuestion = table.Column<string>(type: "nvarchar(max)", nullable: true),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false),

GroupId = table.Column<int>(type: "int", nullable: false),

AuthorId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Discussions", x => x.Id);

table.ForeignKey(

name: "FK\_Discussions\_ApplicationUsers\_AuthorId",

column: x => x.AuthorId,

principalTable: "ApplicationUsers",

principalColumn: "Id");

table.ForeignKey(

name: "FK\_Discussions\_Groups\_GroupId",

column: x => x.GroupId,

principalTable: "Groups",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Votings",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Question = table.Column<string>(type: "nvarchar(max)", nullable: false),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false),

GroupId = table.Column<int>(type: "int", nullable: false),

AuthorId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Votings", x => x.Id);

table.ForeignKey(

name: "FK\_Votings\_ApplicationUsers\_AuthorId",

column: x => x.AuthorId,

principalTable: "ApplicationUsers",

principalColumn: "Id");

table.ForeignKey(

name: "FK\_Votings\_Groups\_GroupId",

column: x => x.GroupId,

principalTable: "Groups",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "WordsQuizzes",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Topic = table.Column<string>(type: "nvarchar(max)", nullable: false),

Question = table.Column<string>(type: "nvarchar(max)", nullable: false),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false),

GroupId = table.Column<int>(type: "int", nullable: false),

AuthorId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_WordsQuizzes", x => x.Id);

table.ForeignKey(

name: "FK\_WordsQuizzes\_ApplicationUsers\_AuthorId",

column: x => x.AuthorId,

principalTable: "ApplicationUsers",

principalColumn: "Id");

table.ForeignKey(

name: "FK\_WordsQuizzes\_Groups\_GroupId",

column: x => x.GroupId,

principalTable: "Groups",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "VotingOption",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Text = table.Column<string>(type: "nvarchar(max)", nullable: false),

VotingId = table.Column<int>(type: "int", nullable: false),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_VotingOption", x => x.Id);

table.ForeignKey(

name: "FK\_VotingOption\_Votings\_VotingId",

column: x => x.VotingId,

principalTable: "Votings",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Vote",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

VoterId = table.Column<int>(type: "int", nullable: false),

VotingOptionId = table.Column<int>(type: "int", nullable: true),

CreationDate = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Vote", x => x.Id);

table.ForeignKey(

name: "FK\_Vote\_ApplicationUsers\_VoterId",

column: x => x.VoterId,

principalTable: "ApplicationUsers",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Vote\_VotingOption\_VotingOptionId",

column: x => x.VotingOptionId,

principalTable: "VotingOption",

principalColumn: "Id");

});

migrationBuilder.CreateIndex(

name: "IX\_Discussions\_AuthorId",

table: "Discussions",

column: "AuthorId");

migrationBuilder.CreateIndex(

name: "IX\_Discussions\_GroupId",

table: "Discussions",

column: "GroupId");

migrationBuilder.CreateIndex(

name: "IX\_Vote\_VoterId",

table: "Vote",

column: "VoterId");

migrationBuilder.CreateIndex(

name: "IX\_Vote\_VotingOptionId",

table: "Vote",

column: "VotingOptionId");

migrationBuilder.CreateIndex(

name: "IX\_VotingOption\_VotingId",

table: "VotingOption",

column: "VotingId");

migrationBuilder.CreateIndex(

name: "IX\_Votings\_AuthorId",

table: "Votings",

column: "AuthorId");

migrationBuilder.CreateIndex(

name: "IX\_Votings\_GroupId",

table: "Votings",

column: "GroupId");

migrationBuilder.CreateIndex(

name: "IX\_WordsQuizzes\_AuthorId",

table: "WordsQuizzes",

column: "AuthorId");

migrationBuilder.CreateIndex(

name: "IX\_WordsQuizzes\_GroupId",

table: "WordsQuizzes",

column: "GroupId");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Discussions");

migrationBuilder.DropTable(

name: "Vote");

migrationBuilder.DropTable(

name: "WordsQuizzes");

migrationBuilder.DropTable(

name: "VotingOption");

migrationBuilder.DropTable(

name: "Votings");

}

}

}