**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 тетсування 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., 10 рис., 5 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., 10 fig., 5 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](#_Toc137495312)

[Abstract 9](#_Toc137495313)

[List of abbreviations 12](#_Toc137495314)

[INTRODUCTION 13](#_Toc137495315)

[Chapter 1. Review of Related Literature and Existing Language Learning Applications 15](#_Toc137495316)

[1.1. Current language learning techniques 15](#_Toc137495317)

[1.2. The role of technology in language learning 18](#_Toc137495318)

[1.3. Online learning platforms and tools 19](#_Toc137495319)

[1.4. Real-time communication in online learning 21](#_Toc137495320)

[1.5. Advantages and disadvantages of current language learning applications 23](#_Toc137495321)

[1.5.1. Advantages 23](#_Toc137495322)

[1.5.2. Disadvantages 23](#_Toc137495323)

[Chapter 2. Research Design and Application Conceptualization 24](#_Toc137495324)

[2.1. Study design 24](#_Toc137495325)

[2.1.1. Research approach 24](#_Toc137495326)

[2.1.2. Rationale for chosen approach 25](#_Toc137495327)

[2.1.3. Research procedures 26](#_Toc137495328)

[2.2. Data collection and analysis methods 28](#_Toc137495329)

[2.2.1. Data Collection 28](#_Toc137495330)

[2.2.2. Data Analysis 28](#_Toc137495331)

[2.3. Description of the application 28](#_Toc137495332)

[2.3.1. Overview of the application 28](#_Toc137495333)

[2.3.2. Application key features and functionality high-level overview 29](#_Toc137495334)

[2.3.3. Technical specifications 30](#_Toc137495335)

[2.3.4. Development process 32](#_Toc137495336)

[2.3.5. Expected user experience 33](#_Toc137495337)

[Chapter 3. Architectural Design and Implementation of the Language Exchange Application 35](#_Toc137495338)

[3.1. Technologies used 35](#_Toc137495339)

[3.1.1. ASP.NET Core 35](#_Toc137495340)

[3.1.2. SQL Server 36](#_Toc137495341)

[3.1.3. SignalR 36](#_Toc137495342)

[3.1.4. Angular 37](#_Toc137495343)

[3.1.5. NgRx 38](#_Toc137495344)

[3.1.6. Database management 39](#_Toc137495345)

[3.2. System design and architecture 40](#_Toc137495346)

[3.3. Frontend development 42](#_Toc137495347)

[3.3.1. User interface design 42](#_Toc137495348)

[3.3.2. Responsiveness and accessibility 43](#_Toc137495349)

[3.4. Backend development 44](#_Toc137495350)

[3.4.1. Projects structure 44](#_Toc137495351)

[SyncLink.Server (API). 45](#_Toc137495352)

[SyncLink.Infrastructure. 45](#_Toc137495353)

[SyncLink.Application 45](#_Toc137495354)

[SyncLink.Common 45](#_Toc137495355)

[3.4.1. Database management 46](#_Toc137495356)

[3.4.2. API and functionality 49](#_Toc137495357)

[3.5. Backend Real-Time interactive features implementation 51](#_Toc137495358)

[3.6. Frontend Real-Time interactive features implementation 52](#_Toc137495359)

[3.7. Implementing Gamified learning features 54](#_Toc137495360)

[3.7.1. Minigames involving language skills 55](#_Toc137495361)

[3.7.2. Word/Phrase of the day discussions 55](#_Toc137495362)

[Chapter 4. Approaches for Testing and evaluation of the web application 57](#_Toc137495363)

[4.1. User testing 57](#_Toc137495364)

[4.1.1. Test Preparation 58](#_Toc137495365)

[4.1.2. Test Execution 58](#_Toc137495366)

[4.1.3. Observations and Data Collection 58](#_Toc137495367)

[4.1.4. Feedback Collection 58](#_Toc137495368)

[4.1.5. Analysis 59](#_Toc137495369)

[4.2. Performance testing 59](#_Toc137495370)

[4.2.1. Load Testing 59](#_Toc137495371)

[4.2.2. Stress Testing 59](#_Toc137495372)

[4.2.3. Scalability Testing 60](#_Toc137495373)

[4.2.4. Endurance Testing 60](#_Toc137495374)

[4.2.5. Analysis 60](#_Toc137495375)

[4.3. Results and feedback 60](#_Toc137495376)

[4.3.1. User Testing Feedback 60](#_Toc137495377)

[4.3.2. Performance Testing Results 61](#_Toc137495378)

[4.3.3. Evaluation and Action Plan 61](#_Toc137495379)

[Conclusions 62](#_Toc137495380)

[REFERENCES 63](#_Toc137495381)

# List of 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

Learning a language, especially one as globally prevalent as English, is a multifaceted process. Over the years, numerous techniques and tools have been developed to facilitate this process. In this section, we review the current methods utilized in English language learning.

Traditionally, language learning has relied heavily on classroom-based teaching. This approach employs a mixture of techniques such as grammar translation, the direct method, and the audio-lingual method, which emphasize the rules of the language, direct association without translation, and pattern drills, respectively (Richards & Rodgers, 2014) [5]. Despite the benefits, these traditional methods are often criticized for their lack of context and real-world application.

In contrast, communicative language teaching (CLT) focuses on meaningful communication and real-world tasks to teach language. CLT stresses the importance of interaction as both the means and the ultimate goal of learning a language (Nunan, 2014) [4]. This method facilitates the development of linguistic fluency rather than just accuracy, creating more well-rounded learners.

In recent years, the advent of technology has given rise to numerous digital language learning tools. Computer-Assisted Language Learning (CALL) uses computers and multimedia resources to enhance language teaching and learning. The benefits of CALL include interactivity, instant feedback, and a wealth of resources catering to diverse learning styles (Beatty, 2013) [1].

Moreover, language learning apps such as Duolingo, Babbel, and Rosetta Stone have become increasingly popular. These apps use a mixture of methods such as spaced repetition, gamification, and immersion to make language learning more engaging and effective (Godwin-Jones, 2014) [3].

Despite their popularity, these digital tools often focus on individual learning, neglecting the communicative and social aspects of language learning. Additionally, while they can help learners acquire vocabulary and understand grammar rules, they often fall short in teaching the cultural nuances of language.

In summary, 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** |
| --- | --- | --- | --- |
| 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 |

Continuation of table 1.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **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.

|  |  |  |  |
| --- | --- | --- | --- |
| **Technique** | **Description** | **Pros** | **Cons** |
| 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 |

## 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

With the evolution of technology, various online platforms and tools have been developed to facilitate language learning. These platforms leverage digital technology to offer accessible, flexible, and diverse language learning experiences.

Popular language learning platforms like Duolingo, Babbel, and Rosetta Stone have revolutionized self-paced language learning. Duolingo, for instance, employs gamified lessons to engage learners and make the learning process fun. It offers immediate feedback, allowing learners to identify and rectify their mistakes promptly. Babbel focuses on conversational skills and provides lessons developed by language experts. Rosetta Stone uses an immersive approach, teaching languages through images, intuition, interactivity, and instruction in the target language (Chapelle, 2019) [2].

Another category of online platforms are language exchange platforms such as Tandem and HelloTalk. These platforms connect learners from different parts of the world to facilitate language exchange. They allow learners to practice the target language with native speakers through text, voice, or video chat.

In the realm of formal education, Learning Management Systems (LMS) such as Canvas, Blackboard, and Moodle are used widely. They offer a range of features such as course management, assessment tools, forums for discussion, and resources for self-study.

While these online tools have brought significant advancements in language learning, they also have their limitations. Language learning apps often lack sufficient opportunities for real-world, spontaneous conversation. Language exchange platforms, while offering conversational practice, often lack structured learning. LMS, although comprehensive, may lack the interactive and engaging 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 |

Continuation of table 1.2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform** | **Description** | **Pros** | **Cons** |
| 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 in online learning refers to live interaction between learners, instructors, or between learners themselves. These interactions can take place via video conferencing tools, instant messaging platforms, or collaborative online spaces, enhancing the educational experience by allowing immediate feedback and promoting active learning.

For language learning in particular, real-time communication is critical. Learning a language involves more than just acquiring vocabulary or understanding grammar. It also includes the ability to use the language spontaneously and appropriately in real-time conversations. Such skills are best developed through direct interaction with other speakers of the language.

Many studies have emphasized the value of real-time communication in language learning. Wang (2015) [6] found that real-time communication can facilitate the development of speaking and listening skills, as it promotes natural language use and offers immediate feedback. Similarly, Sun (2016) [7] highlighted that real-time communication can enhance learners’ confidence and motivation, making the learning process more engaging.

Despite its benefits, real-time communication is not always effectively incorporated in online language learning platforms. For instance, while apps like Duolingo and Rosetta Stone provide structured lessons, they often lack sufficient real-time interaction opportunities. On the other hand, language exchange platforms like Tandem, though offering real-time conversation practice, often lack the structure and support of formal 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

This chapter forms the foundation of this qualification work, laying out the strategies and techniques used to investigate the outlined questions, design the application, and subsequently assess its effectiveness. The rigorous methodological framework utilized in this study ensures that the resulting findings are both reliable and valid.

In crafting the study design, meticulous consideration was given to choosing an approach that would facilitate comprehensive exploration of the complex interplay between language learning, technology use, and collaborative communication in English-speaking student groups.

Data collection and analysis methods were carefully selected to gather insightful, meaningful data, which was critically evaluated to generate nuanced understanding of the research problem. This empirical data then informed the design of the application, ensuring that it was tailored to effectively address the identified issues.

With a deep understanding of the problem at hand and armed with substantial empirical data, the application was conceptualized, developed, and polished. This process involved detailed planning and robust design strategies to ensure the application effectively harnessed real-time communication, facilitating language and cultural exchange among users.

## 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.

Reporting: The final stage involved comprehensive and systematic presentation of findings, including future recommendations.

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 of this work is designed to serve as an online platform that facilitates language learning and cultural exchange among students in English-speaking groups. Leveraging technology and real-time communication features, the application aims to enhance students' English language skills while promoting cross-cultural understanding and collaboration.

The application is intended to offer a rich, interactive, and engaging environment that makes language learning more accessible and enjoyable. Through its various features, the application seeks to address the challenges identified in the context of English language learning, particularly those accentuated by the remote education setting necessitated by the recent pandemic.

While the primary focus is on enhancing language skills, the application also seeks to cultivate a sense of community among students. Through real-time interactions, group activities, and cultural exchange, the application aims to foster a sense of belonging, making learning more collaborative and less isolating.

Designed as a web application, it can be accessed from various devices such as desktop computers, laptops, tablets, and smartphones, providing flexibility and convenience to users. The backend of the application is built using ASP.NET Core, with SQL Server serving as the database. The frontend is developed using Angular, and SignalR is employed for real-time interactive features.

In summary, the application is a comprehensive, interactive, and user-friendly platform that combines language learning and cultural exchange, addressing the challenges faced by non-native English-speaking students.

### 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.

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.

Forum for Q&A and Discussions: SyncLink provides a forum for students to ask questions, share insights, or engage in discussions related to language learning or course content. This is a moderated space that fosters respectful and constructive interaction among students.

Cultural Exchange Space: This feature allows students to share and learn about different cultures. Students can post about their own culture or inquire about others, fostering mutual respect and understanding among a diverse group of 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.

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.

Progress Tracker: Each student has access to a personal progress tracker that monitors their activity on the platform, their performance in the language learning games, and their overall progress in language learning.

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's architecture comprises of both a front-end and a back-end that communicate with each other to deliver the app's functionalities. In addition, the app makes use of real-time communication features to support instant interaction among users.

Backend Development: The backbone of the application is developed using ASP.NET Core, a robust, high-performance, open-source framework for building modern, cloud-based applications. The backend is responsible for the business logic of the application, such as managing user accounts, storing and retrieving data, and handling real-time communication among users. Data is stored in a SQL Server database, providing secure and efficient data management capabilities.

Frontend Development: The user interface of the application is built using Angular, a popular web application framework developed by Google. Angular provides a powerful platform for developing complex, single-page applications with a rich, responsive user interface. It facilitates clean and maintainable code, ensuring that the application is scalable and easy to update.

Real-Time Communication Features: Real-time communication among users is facilitated using SignalR, a library that simplifies the process of adding real-time web functionality to applications. SignalR handles connection management automatically and allows for broadcasting messages to all connected clients simultaneously, which is essential for the real-time chat and collaborative whiteboard features of the application.

Together, these technologies provide a secure, efficient, and responsive platform for “SyncLink”, ensuring that users have a smooth and engaging experience while using the application. The choice of these technologies was informed by their proven capabilities, extensive community support, and suitability for the specific requirements of the application.

The figure below depicts application high-level architecture overview:



Fig. 2.1. Application high-level architecture

### 2.3.4. Development process

The development of SyncLink followed a structured process to ensure that the resulting application meets its intended goals and provides a smooth and engaging user experience. This process consisted of three main stages: Planning and Design, Implementation, and Testing and Debugging.

Planning and Design: The first phase involved clearly defining the application's goals and the specific functionalities required to achieve these goals. This was based on the needs and challenges identified through the literature review and data analysis. Once the functionalities were defined, the overall structure and architecture of the application were designed. This included designing the database schema, determining the main components and services of the application, and designing the user interface. Wireframes and prototypes were created to visualize the user interface and user flow through the application.

Implementation: The implementation phase involved coding the application based on the designs and specifications established in the planning phase. The backend was developed using ASP.NET Core, with the business logic and database interactions being implemented first. The frontend was developed using Angular, with each component and service being implemented and tested individually before being integrated into the application. The real-time communication features were implemented using SignalR, and integrated with both the backend and frontend.

Testing and Debugging: After the implementation, the application underwent extensive testing to ensure it functions as expected and provides a smooth user experience. This included unit testing, integration testing, and system testing. Any bugs or issues identified during testing were fixed, and the application was re-tested to ensure the fixes were successful. The application was also tested on different devices and browsers to ensure compatibility.

The development process was iterative, with regular feedback and testing at each stage to ensure that the application meets its objectives and provides a high-quality user experience. This approach allowed for any issues or changes to be addressed promptly, reducing the risk of major problems or delays in the later stages of development.

### 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.

Overall, 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 framework developed by Microsoft [8], was selected as the primary technology for the backend of the application. As a modern web development framework, ASP.NET Core offers several advantages that made it a compelling choice for the project.

ASP.NET Core was selected for its capabilities that extend beyond just serving web requests. It is equipped with a rich set of features that allow for implementing various architectural patterns like Command Query Responsibility Segregation (CQRS) [9] and elements of Clean Architecture. The use of CQRS enables clear separation of read and write operations, enhancing the application's performance, scalability, and security. Clean Architecture elements were integrated to promote maintainability, flexibility, and independence from frameworks, thereby allowing the application's business rules and policies to drive the design.

Moreover, this framework is highly scalable and can handle large numbers of users simultaneously, an important factor considering the application's nature that encourages real-time interaction and collaboration. Its cross-platform compatibility, allowing the application to run on Windows, macOS, and Linux, also adds flexibility and broadens potential user demographics.

ASP.NET Core's built-in dependency injection and ability to integrate seamlessly with popular databases and other tools made it a fitting choice for developing a comprehensive and efficient backend for the application.

### 3.1.2. SQL Server

SQL Server, a relational database management system developed by Microsoft [10], is used for managing and storing data for the web application. The selection of SQL Server as the database for this project was driven by its numerous advantageous features, ease of use with ASP.NET Core, and its comprehensive support for data management tasks.

SQL Server is renowned for its high performance, security, and scalability. It provides advanced features such as data compression, encryption, and a high level of compliance with standards, ensuring the safety and integrity of user data. Its transactional system allows the execution of complex queries, further facilitating robust data management.

The database structure in SQL Server is designed to efficiently store user details, game scores, discussion records, and more, all of which are crucial to the web application's functionality. Furthermore, SQL Server integrates seamlessly with ASP.NET Core, simplifying the process of data operations in the backend, such as CRUD (Create, Read, Update, Delete) operations, which are fundamental in the interactive nature of the web application.

Moreover, SQL Server's compatibility with Entity Framework Core [11], an Object-Relational Mapping (ORM) framework, facilitates the translation of object-oriented domain models into a database, further simplifying database management and enhancing developer productivity. The support of LINQ (Language Integrated Query) queries to directly interact with the database using C# also bolsters its efficiency and usability in the context of this project.

In conclusion, SQL Server provides a powerful, reliable, and secure platform for the application's data storage and management needs.

### 3.1.3. SignalR

SignalR, a library developed by Microsoft, is incorporated into the application for real-time web functionality [12]. This technology enables server-side code to push content updates to connected clients instantly as they become available, rather than having the server wait for a client to request new data. This bidirectional communication is pivotal for creating highly interactive applications.

The inclusion of SignalR is particularly relevant for the collaborative and interactive nature of the web application, where instantaneous updates and real-time communication are fundamental. SignalR caters to various real-time scenarios, such as live chatting, real-time gaming, and instant notifications, all of which are crucial in enhancing user engagement within the application.

One of the key advantages of SignalR is its automatic handling of connection management. It can create persistent connections between the client and the server, enabling them to call methods on each other directly. Also, SignalR elegantly handles connection interruptions and can automatically reestablish a disrupted connection, providing a seamless user experience.

SignalR supports different types of transport protocols (WebSockets, Server-Sent Events, Long Polling) and automatically chooses the best available transport based on the client's and server's capabilities. This ensures optimal performance regardless of the server and client environments.

In conjunction with ASP.NET Core, SignalR provides a straightforward and effective approach to real-time communication, enhancing the interactive and collaborative aspects of the web application and contributing to its overall user experience.

### 3.1.4. Angular

The frontend of the web application is built using Angular [13], a popular open-source web application framework developed by Google. Angular was chosen for its robust set of features, scalability, and the capability to build interactive single-page applications (SPAs) that provide a superior user experience.

Angular employs TypeScript [14], a statically-typed superset of JavaScript, which adds a layer of security due to its compile-time error checking capabilities. This ensures better code quality, easier debugging, and overall improved maintainability, which are vital for the long-term sustainability of the application.

The framework's component-based architecture is another significant advantage. This architecture promotes reusability, modularity, and separation of concerns, thereby leading to a well-structured and easier-to-maintain codebase. Components can be reused across the application, improving the development speed and reducing the likelihood of code duplication.

Angular also offers two-way data binding, which automates the synchronization of data between the model and view components. This feature simplifies the programming model, reduces the amount of boilerplate code, and further improves performance.

Angular's comprehensive set of tools and libraries, such as Angular Material for UI components and RxJS for handling asynchronous operations, further enhance the application's development and functionality. Additionally, Angular's compatibility with third-party libraries allows for flexibility and customization, ensuring that the application can adapt to future needs and changes.

Angular Material, a UI component library, was utilized [15] to craft visually appealing, consistent, and functional interfaces. The library provides a set of reusable, well-tested, and accessible UI components based on Google's Material Design specification. The use of Angular Material helps in ensuring a unified user experience across the application, decreasing development time and promoting user interaction.

With its robust capabilities, Angular together with Angular Material UI components library facilitates the creation of an interactive and engaging user interface, which is crucial for the gamified learning environment that the web application aims to provide.

### 3.1.5. NgRx

NgRx is a group of Angular libraries for reactive extensions [16]. It's used in the application for efficient state management. The need for a state management tool like NgRx arises when an application grows complex, with many components sharing and affecting the state.

Based on Redux, a popular state management library in the JavaScript ecosystem, NgRx employs a unidirectional data flow [17] where the state is immutable. This means that when a change in the state is required, a new state is created instead of modifying the existing one. This practice greatly enhances the predictability and traceability of state changes.

In NgRx, the application's state is managed through three core concepts: Actions, Reducers, and Store.

Actions are dispatched in response to user events or operations like clicking a button or API responses. Each action represents a unique event with a type and optional data payload.

Reducers, pure functions, handle these actions and return a new state based on the action type and the previous state.

The Store, an observable of the state and an observer of dispatched actions, brings actions and reducers together. Components can select slices of state from the store, and the state can be updated by dispatching actions to reducers through the store.

By employing NgRx, the application maintains a single source of truth, making the state predictable and consistent. This is especially crucial in a real-time, interactive application where the state can be affected by multiple users simultaneously.

NgRx also integrates seamlessly with Angular's component architecture, RxJS, and the Redux DevTools extension for debugging and performance tracking. This makes it an excellent choice for state management in Angular applications.

### 3.1.6. Database management

Database management is a key aspect of the application development process. As the heart of most applications, the database stores, organizes, and retrieves the data that drives functionality. In the context of this application, we use SQL Server as the main database technology due to its performance, scalability, and security features.

The design of the database schema is crucial for the application's operation and performance. It involves the careful planning of how data is structured and related to ensure efficient storage and quick retrieval. For this application, a relational database design is adopted, employing tables to store data related to users, minigames, and word/phrase of the day discussions. Relationships are defined among these tables to enable efficient data access and maintain data integrity.

Data manipulation operations, including data insertion, deletion, updating, and retrieval, are performed using SQL (Structured Query Language). These CRUD operations are implemented in the application's backend using ASP.NET Core, which communicates with SQL Server through Entity Framework Core, a powerful ORM tool that simplifies data access by enabling us to interact with the database using C# objects.

Additionally, data consistency and integrity are ensured through the use of constraints and transactions. Constraints restrict the data that can be stored in tables, while transactions ensure that database operations are carried out completely or not at all, preventing data inconsistency in case of errors or system failures.

Data security is another critical concern in database management. It is addressed by implementing measures such as data encryption, user authentication, and authorization to protect sensitive data and prevent unauthorized access.

In summary, efficient database management is vital for the application's smooth operation, performance, and security. The use of SQL Server, in conjunction with ASP.NET Core and Entity Framework Core, ensures that these requirements are met while providing a scalable and flexible data management solution.

## 3.2. System design and architecture

The web application adheres to a three-tier architecture, consisting of the Presentation Layer (Frontend), Application Layer (Backend), and Data Layer (Database). This architecture ensures the separation of concerns, modularity, and scalability, which align perfectly with the collaborative and interactive nature of the application.

The Presentation Layer is crafted using Angular and Angular Material, facilitating the creation of dynamic, interactive, and visually appealing user interfaces. Angular Material, a UI component library, is employed to design consistent and functional user interfaces based on Google's Material Design guidelines. Furthermore, NgRx, a robust state management library, is integrated into this layer. By managing both local and global states, NgRx promotes performance, predictability, and consistency across the application.

The Application Layer, created with ASP.NET Core, serves as the application's core, handling business logic and controlling the functionality of the application. This layer employs the Command Query Responsibility Segregation (CQRS) pattern for efficient handling of data manipulation operations, enhancing performance and scalability. Elements of Clean Architecture are incorporated to achieve a flexible, maintainable design that focuses on the application's business rules and policies.

The Data Layer, driven by SQL Server, is entrusted with the task of data storage, retrieval, and manipulation. This layer ensures the organized and ready availability of data, preserving data consistency and integrity through transactions and other measures.

SignalR, a real-time library, bridges the gap between the server and client, enabling them to maintain a persistent connection and exchange information in real time. This adds a dimension of live updates and instant communication to the application, which is particularly beneficial in a collaborative environment.

Considering the interaction among these components, suppose a user participates in a minigame. When the user submits an answer, this event is captured by the Presentation Layer, which then communicates it to the Application Layer. The Application Layer processes the request, updates the game state, and stores the relevant data using the Data Layer. Meanwhile, SignalR propagates these updates to all other participants in real time.

In conclusion, the system's design and architecture are instrumental in ensuring a scalable, maintainable, and efficient application, providing a seamless, engaging experience for the users.

## 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 is primarily concerned with ensuring a visually appealing, intuitive, and user-friendly experience. It combines aesthetics with functionality, enabling users to interact with the application effectively.

The UI is crafted using Angular Material, a component library implementing Google's Material Design principles. The application UI consists of numerous components like dialogues, forms, navigation bars, buttons, and game interfaces. These components follow a coherent visual language, providing consistency across the application.

Furthermore, the UI is designed to promote user engagement through gamified learning features. It includes interfaces for minigames, word/phrase of the day discussions, and other interactive elements that make learning and communication fun and 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

Given the diverse range of devices that users might use to access the application, responsiveness is a key aspect of the frontend development. The application's UI is designed to be responsive, ensuring that it adapts to different screen sizes and orientations, providing an optimal viewing and interaction experience across a variety of devices.

Accessibility is another critical concern, ensuring that the application is usable by people with different abilities. Best practices for web accessibility are followed, including providing alternative text for images, ensuring proper color contrast, and allowing keyboard navigation, among others.

Through an intuitive UI design, responsive layout, and accessibility features, the frontend development ensures that the application provides a user-centric experience that caters to diverse user needs and preferences.

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 |
| 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

One of the key features of our application is its real-time interactivity, achieved using SignalR. SignalR is a library that simplifies the process of adding real-time web functionality to applications, allowing for instant content updates without requiring a page refresh. This functionality is critical for creating a dynamic and responsive environment that fosters communication and collaboration among students.

The real-time interaction in our application is centered around the SyncLinkHub class, which inherits from SignalR's Hub class. This hub acts as a conduit for real-time communication, facilitating various interactive features of the application such as messaging, whiteboard updates, and text plot game events.

The SyncLinkHub class contains several methods tied to user actions, and it is through these methods that real-time communication is implemented. For example, the GroupOpened and GroupClosed methods manage users joining and leaving a group, while the BoardUpdated method handles whiteboard updates.

Furthermore, the SyncLinkHub class communicates with various notification services (GeneralGeneralNotificationsService, TextPlotGameNotificationService) to propagate changes to all connected users. For instance, when a new message is received, the GeneralGeneralNotificationsService notifies all connected users in the respective group through the MessageReceived method.

Let's take a closer look at the main real-time interactive features:

Messaging: This feature allows users to send and receive messages in real time. When a message is sent, the MessageReceived method in ISyncLinkHub interface is triggered, notifying all users within the group of the new message.

Whiteboard Updates: The application supports a shared whiteboard where users can collaboratively work. Any update made on the whiteboard triggers the BoardUpdated method in ISyncLinkHub interface, instantly reflecting the changes to all users in the group.

Text Plot Games: In a text plot game, the start of the game, new entries, received votes, and the end of the game are all events that are communicated to users in real time. These events correspond to the GameStarted, NewEntry, VoteReceived, and GameEnded methods respectively in ISyncLinkHub interface.

In summary, the real-time interactive features are a significant part of the application's overall functionality, contributing to the interactive and dynamic learning environment. SignalR has been instrumental in implementing these features, ensuring seamless real-time communication and synchronization of data among users.

## 3.6. Frontend Real-Time interactive features implementation

The application uses a SignalRService, which handles the setup and management of the SignalR connection from the client-side (Angular frontend). The service creates and manages the hubConnection and also ensures that the client can handle real-time updates from the server-side SyncLinkHub.

The SignalRService makes use of Subjects from the RxJS library, a type of Observable, for propagating whiteboard changes to the relevant components in real time. The boardChange$ subject specifically listens for changes in the state of the shared whiteboard, emitting updates as they occur.

Several methods in SignalRService correspond to real-time events. For example, when a group is opened or closed, the groupOpened and groupClosed methods are invoked respectively. Similarly, when the shared whiteboard is updated, the whiteboardUpdated method is triggered.

In the SignalRService constructor, listeners are established for various events:

messageReceived: When a new message is received from the server, the method dispatches the sendMessageSuccess action with the received message details. This allows the state of the application to be updated and reflects the new message in the user interface.

boardUpdated: Listens for updates related to the shared whiteboard. When an update is received, it dispatches the whiteboardUpdatedExternal action which updates the shared whiteboard state in the application.

All these methods and event listeners contribute to the application's real-time interactivity. They ensure that all connected clients maintain an updated and synchronized view of the shared state, promoting collaboration and engagement among the students.

Moreover, the integration of SignalR with Angular and NgRx (Redux-like state management for Angular applications) provides a clean and scalable way of handling real-time updates. The use of NgRx allows the application state to be predictably managed and updated in response to actions dispatched when real-time updates are received.

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 first gamified learning feature involves a series of interactive minigames specifically designed to enhance language skills. These minigames vary in scope and style but they all share the common goal of promoting English language proficiency in a playful, non-threatening manner.

For example, one such minigame could involve word matching where students are tasked with associating English words with their correct meanings, images, or translations. Other minigames might involve sentence building or grammar puzzles, encouraging students to practice their syntax and grammatical knowledge.

Crucially, these minigames are designed to be multiplayer, meaning students are able to interact with their peers in real time. This not only enhances the fun factor but also encourages cooperation and friendly competition among the students. The real-time updates and scores are facilitated by SignalR, allowing all the participants to see their progress and rank immediately.

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

### 3.7.2. Word/Phrase of the day discussions

The second major gamified feature is the "Word/Phrase of the Day" discussion. Each day, a new English word or phrase is presented to the students within the application. This word or phrase can be explored in multiple ways – its definition, its synonyms, its usage in sentences, and its cultural or historical context.

However, rather than simply providing this information, the application encourages students to participate in discussions around the word/phrase of the day. Students can share their own sentences using the word/phrase, suggest similar words/phrases, or discuss any cultural connotations associated with it. These discussions are then facilitated by the application's real-time interactive features, ensuring lively, dynamic conversation.

The "Word/Phrase of the Day" feature thus serves a dual purpose – it helps to expand the students' English vocabulary and comprehension, and it also encourages active participation and interaction between the students.

In conclusion, the incorporation of these gamified learning features within the web application serves to enhance the educational experience for students. By merging gameplay elements with traditional learning activities, these features aim to foster a more engaging, interactive, and enjoyable learning environment for English group students.

The example below demonstrates a simple type dependency diagram for variaous collaboration-related entities, like Discussion blocks, WordQuizzes, Votings, etc.:



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

# 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.

This chapter provides an overview of the different forms of testing carried out on the web application, including User Testing and Performance Testing. It further discusses the results obtained from these tests, as well as the feedback received from the users. These results and feedback will be used to inform potential improvements and modifications in subsequent iterations of the application.

## 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.

## 4.2. Performance testing

In addition to functionality and usability, the performance of an application plays a significant role in user satisfaction and overall application success. As the application serves multiple users simultaneously, it is essential to ensure it can handle load and still operate smoothly. Performance Testing was conducted to evaluate how the application performs under various conditions, particularly focusing on response time, reliability, resource usage, and scalability.

### 4.2.1. Load Testing

Load Testing is carried out to understand the behavior of the system under a specific load. The application was subjected to the expected number of users that would typically use the application simultaneously. This test helped us assess the system's response time, throughput, CPU utilization, and other critical metrics under typical load conditions. It ensured that the system could handle its expected usage volume.

### 4.2.2. Stress Testing

Stress Testing involves testing an application under extreme workloads to see how it handles high traffic or data processing. The application was tested beyond normal load conditions to observe its performance and stability under these stressful conditions. This test was crucial to identify the application's breaking point or safe usage limit.

### 4.2.3. Scalability Testing

Scalability Testing was performed to measure the application's effectiveness in "scaling up" to support an increase in user load. It determined whether the application could handle a larger number of users and how adding additional resources (like CPU or memory) would impact the performance.

### 4.2.4. Endurance Testing

Also known as Soak Testing, Endurance Testing checks the system's response to a sustained, expected user load over a long duration. It aimed to identify system issues such as memory leaks or performance degradation that might occur over time.

### 4.2.5. Analysis

The performance metrics collected from these tests were analyzed to identify any bottlenecks or weak points. Any performance issues that emerged during these tests were addressed and re-tested to confirm the application's readiness for public usage.

The Performance Testing aimed to ensure that the web application offers a seamless and consistent user experience, even under high usage conditions. The results and findings from these tests are crucial in making informed decisions about system tuning and optimization, and they also provide insights into infrastructure planning to meet future load requirements.

## 4.3. Results and feedback

### 4.3.1. User Testing Feedback

User testing provided a wealth of information about how real users interact with the web application. The data collected from these tests included both qualitative and quantitative feedback, offering a broad understanding of the application's usability and user satisfaction.

Quantitative feedback included data on error rates, completion rates, and task completion times. These metrics provided insight into the overall efficiency and effectiveness of the application.

Qualitative feedback was gathered through user comments, observations, and interviews conducted post-testing. This feedback offered valuable insights into user perceptions, likes, dislikes, and suggestions for improvement.

### 4.3.2. Performance Testing Results

The performance testing provided critical insights into the application's ability to handle load, its reliability, scalability, and resource utilization. These insights were crucial for understanding the application's robustness and readiness for deployment.

Specific results such as load times, server response times, peak load handling, and error rates under stress conditions were scrutinized. The findings guided the optimization process, ensuring that the application can handle a significant amount of load and offer a smooth user experience.

### 4.3.3. Evaluation and Action Plan

The results and feedback from both user and performance testing were used to create an action plan for refining the web application. Each identified issue was prioritized based on its impact on the user experience or application performance. High-priority issues were addressed immediately, while lower-priority items were scheduled for future updates.

The process of evaluation also extended to the effectiveness of the testing procedures themselves. Consideration was given to whether the testing methods provided a comprehensive understanding of the application's strengths and weaknesses, and adjustments were made for future testing cycles.

In conclusion, the feedback and results obtained were instrumental in refining the application and making it more robust, user-friendly, and efficient. They offered an objective basis for decision-making, fostering continuous improvement, and enhancing user satisfaction.

# 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");

}

}

}