Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Deliverable 2

Polish Literature Language Game

Aleksandra Struzik, Wiktoria Włodarczyk

thesis supervisor

DSc (Eng.) Marcin Luckner

version 1.3

01.11.2025

Table of Contents

1 Abstract

The aim of the project is to develop a web application that serves as an interactive language game based on Polish literary texts, as well as to design and implement an algorithm capable of automatically generating language puzzles from a given text. The system allows users to work with selected literary fragments by solving various word-based tasks that enhance linguistic competencies such as contextual understanding, orthographic and stylistic accuracy, and vocabulary knowledge.

The project encompasses both the design and implementation of a browser-based web application and the development of text processing mechanisms responsible for preparing and transforming literary material into interactive exercises. The literary texts used in the game are sourced from publicly available materials and are appropriately processed to fit the game's mechanics.

This document presents the overall concept of the system, the project objectives, the development process, and the supporting algorithmic components that enable the dynamic creation of educational content.

1.1 History of changes

Date	Author	Description	Version
17.10.2025	Wiktoria Włodarczyk	Abstract, Specification, Use case Diagram, Project Schedule	1.0
18.10.2025	Aleksandra Struzik	SWOT analysis, Vocabulary, Project Schedule	1.0
19.10.2025	Wiktoria Włodarczyk	correction of linguistic and stylistic errors	1.1
24.10.2035	Aleksandra Struzik	System Architecture, Architecture Diagram, Module Design, Communication	1.2
25.10.2025	Wiktoria Włodarczyk	Class Diagram, Technology Selection	1.2
31.10.2025	Aleksandra Struzik	User Activity Diagram with description	1.2
31.10.2025	Wiktoria Włodarczyk	User Interface Vision	1.2
02.11.2025	Wiktoria Włodarczyk	Text and reference formatting	1.2
04.11.2025	Aleksandra Struzik	Correction based on feedback from the supervisor	1.3

2 Vocabulary

Book - the entire literary work available in the application. Each literary work is treated as a single book, even if only part of it is used in the game. A standalone poem can also be considered a book.

Extract - a selected fragment of a book that serves as a playable unit in the game. The player progresses through the book by completing extracts in sequence. It is not possible to move to the next extract without finishing the previous one.

Page - extract is divided into pages. A page is a single screen of text; it is completely visible at once. Player must complete pages in order as they appear to finish the extract. Extract is automatically divided into pages.

Line - a row of text on the page. Pages are divided into line for organization and structure.

Minigame - A specific type of language-based puzzle. Player can choose what type of minigame they want to play or set the game to pick one randomly.

Level - A playable instance of an extract. Each time a player starts an extract generates a different level. Once completed or exited a level cannot be revisited.

Riddle - A single interactable challenge on a single page. It presents player with the text of the page and the problem to solve based on the active minigame.

Word - a sequence of characters separated by spaces or punctuation, treated as a distinct token by the system. The same word with different capitalization (e.g., Pokrzywa and pokrzywa) is recognized as identical. Punctuation marks are not considered part of the word.

Letter - a single character from the Polish alphabet. Uppercase and lowercase letters are treated as the same letter.

Blank - a visual indication that an empty space is a part of the riddle.

Highlight - a visual indication that a word in the text is a part of the riddle.

Option - a blank, a letter or a word that is highlighted or visually separated from the text of the page. It is a part of the solution if the riddle.

Decoy - an incorrect option shown to the player in the riddle. Its purpose is to distract the player from correct answer. Blanks, words or letters can be decoys.

Score - the points awarded for completing the extract. The score is calculated based on number of errors made in the level. If an extract is played again a higher score will overwrite the existing one.

3 Specification

3.1 Executive summary

The main goal of the project is to create a web application that serves as an interactive language game based on Polish literary texts, inspired by the concept of *Dear Reader* [1] and similar literary-based learning tools. In addition, the project aims to develop an algorithm capable of automatically generating language puzzles from these texts. The system seeks to increase interest in Polish literature by presenting its content in an engaging and entertaining way, while also providing a technological foundation for adaptive and automated exercise creation.

The application is designed for two main user groups:

- Polish native speakers, who will be encouraged to rediscover classical and modern Polish literature through interactive, language-based challenges, and
- foreign learners of Polish, who will be able to expand their vocabulary, improve comprehension skills, and explore Polish culture through authentic literary materials.

The system consists of a collection of language games and word puzzles based on diverse fragments of Polish literary works obtained from open-source repositories. Each activity will dynamically adapt selected texts into interactive exercises such as word completion, word order correction, or identification of incorrect words, providing users with a varied and educational experience. The generation of these exercises is supported by a dedicated text analysis and transformation algorithm, designed to automatically process literary texts and extract suitable linguistic patterns for gameplay.

The platform is accessible through a standard web browser, ensuring intuitive navigation, responsiveness, and compatibility with multiple devices. The project emphasizes both educational value and technical innovation, combining linguistic learning with the cultural richness of Polish literature and the automation capabilities of modern natural language processing techniques.

3.2 Functional requirements

The use case diagram presented below illustrates the functional behavior of the Polish Literature Language Game application.

The system allows a single actor - User - to interact with the platform by browsing available literary texts, participating in language-based games, and viewing their results.

The main functionalities of the system are divided into two logical modules:

- Library browsing,
- Gameplay.

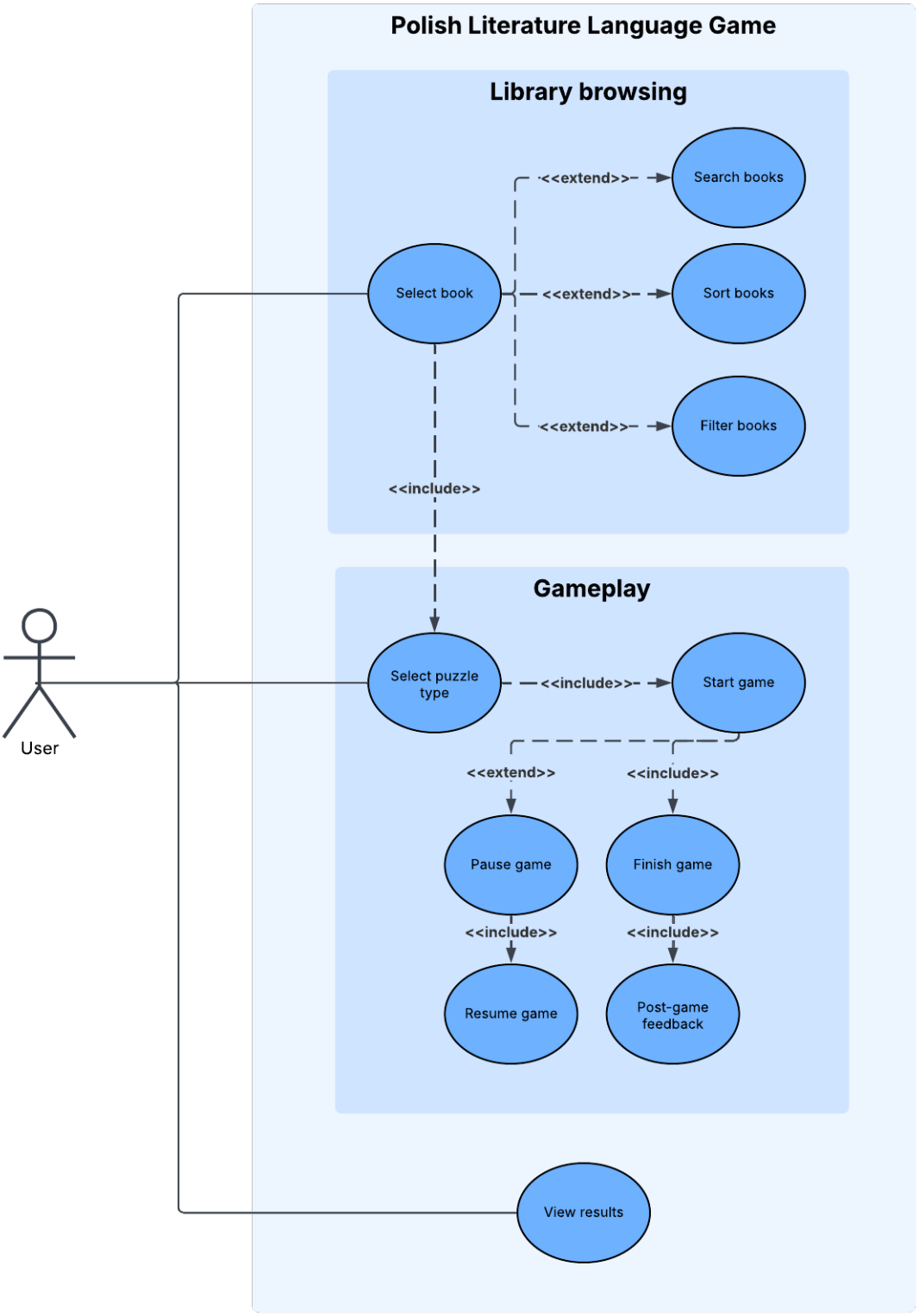

Figure 1 Use case diagram showing interaction of the user with an app

Table 1 Description of use cases for the user with an app

ID	Actor	Use Case Name	Description	Preconditions	Postconditions
UC1	User	Browse Books	User can browse the list of available literary texts.	User has opened the application.	List of available texts displayed.
UC2		Search Books	User searches for a specific book by title, author, or keyword.	User is in the Library view.	Search results are displayed.
UC3		Filter Books	User filters texts by genre, period, or difficulty level.	User is in the Library view.	Filtered list of books displayed.
UC4		Sort Books	User sorts books alphabetically or by date added.	User is in the Library view.	Sorted list displayed.
UC5		Select Book	User selects a specific literary text to use in gameplay.	User browsed or searched for books.	Text fragment selected for game.
UC6		Choose Puzzle Type	User selects the desired type of language task or puzzle.	Book has been selected.	Puzzle type stored for next stage.
UC7		Start Game	Starts an interactive game session.	Book and puzzle type selected.	Game interface displayed.
UC8		Pause Game	Temporarily suspends the game.	Game in progress.	Game state saved.
UC9		Resume Game	Resumes a previously paused game.	Game is paused.	Game resumes from saved state.
UC10		Finish Game	Ends the current game session.	Game in progress.	Results screen displayed.
UC11		Receive Post-Game Feedback	Provides the user with feedback about their performance and linguistic accuracy.	Game session completed.	Feedback displayed to user.
UC12		View My Results	Displays the user's past results and statistics.	User has completed at least one game session.	Results list shown.

This section presents a set of user stories describing the main functionalities of the Polish Literature Language Game application from the user's perspective. Each story expresses a specific need or goal of the end user and defines how the system should respond to it. The user stories serve as a foundation for system design, development, and testing, ensuring that the implemented features align with user expectations and learning objectives.

US1 - Browse Books

As a user,
I want to browse a collection of available Polish literary texts,
so that I can select one for language exercises.

Acceptance Criteria:

- Books are displayed in a list or grid format.
- Each book shows title, author, and a short description.
- Option to open book details.

US2 - Search, Filter, and Sort Books

As a user,
I want to search, filter, and sort available texts,
so that I can quickly find literature that matches my interests or language level.

Acceptance Criteria:

- Search field allows searching by title, author, or keyword.
- Filter options include genre, time period, and difficulty.
- Sort by alphabetical order or publication date.

US3 - Select Book

As a user,
I want to select a book from the library,
so that I can use its text in the language game.

Acceptance Criteria:

- Selected book becomes the input for generating puzzles.
- System confirms selection before starting gameplay.

US4 - Choose Puzzle Type

As a user,
I want to choose the type of language challenge,
so that I can practice specific linguistic skills such as vocabulary or grammar.

Acceptance Criteria:

- Multiple puzzle types are available (e.g., fill-in-the-blank, reorder sentences).

US5 - Start Game

As a user,
I want to start the game after selecting a book and puzzle type,
so that I can begin the learning activity.

Acceptance Criteria:

- Start button initiates the gameplay.
- The game screen loads dynamically.
- Timer or progress indicator visible.

US6 - Pause and Resume Game

As a user,
I want to pause and later resume the game,
so that I can take a break without losing progress.

Acceptance Criteria:

- "Pause" button saves game state.
- "Resume" continues from the saved point.

US7 - Finish Game and Receive Feedback

As a user,
I want to finish the game and receive feedback,
so that I can see my score and understand my mistakes.

Acceptance Criteria:

- Game ends when all tasks are completed.
- System displays performance summary (accuracy, time, and score).
- Optional suggestions for improvement appear.

US8 - View My Results

As a user,
I want to access my past results and progress history,
so that I can monitor my language learning improvement over time.

Acceptance Criteria:

- Results page shows completed games and scores.

3.3 Non-functional requirements

This section defines the non-functional requirements for the developed web application. These requirements describe the qualitative attributes of the system, such as usability, reliability, performance, and maintainability. They complement the functional specification by setting measurable standards that ensure the system's stability, efficiency, and accessibility for all user

Table 2 List of non-functional requirements for Polish Literature Language Game

Requirements area	No.	Description
Usability	1	The interface should fit on screens with a minimum resolution of 1280×720, without horizontal scrolling. Text should be easily readable, with a font size of at least 14 px.
	2	The application should allow a new user to start a game from the library in no more than three clicks from the main page.
	3	During usability testing, at least 80% of users should be able start a game and complete a puzzle without help.
Reliability	4	The game should automatically save progress every 15 seconds and whenever a puzzle is completed. After a crash, the user should lose no more than one action.
Performance	5	The application should load the main screen in less than 3 seconds on a broadband connection (minimum 25 Mbps download speed) when accessed from a modern browser with an empty cache.
	6	Searching and filtering books in the library should take no longer than 1 second for up to 5,000 records.
	7	Starting a game (loading the selected text and first puzzle) should take less than 2 seconds.
Supportability	8	The system should be easy to maintain. The code should be documented and tested before deployment.
	9	The application should keep logs of system errors and important actions to help identify and fix problems.
Portability / Compatibility (+)	10	The web app should work on the latest versions of major browsers (Chrome, Firefox, Edge, Safari) and adapt to mobile screens.
Internationalization (+)	11	The application should be available in Polish and English. Language can be changed from the interface.
Data Quality (+)	12	All literary texts must include information about the author and source. Generated puzzles should be grammatically and logically correct.

4 Project schedule

Table 3 Project schedule.

Phase / Lab	Task ID	Task / Subtask Description	Start Date	End Date	Duration
1. Organization, Requirements and Risk Analysis (Lab 1)	1.1	Define the main goal and scope of the application	01.10.2025	03.10.2025	3 days
	1.2	Write system vision and describe target users	02.10.2025	05.10.2025	4 days
	1.3	Specify functional and non-functional requirements	05.10.2025	10.10.2025	6 days
	1.4	Conduct risk analysis and identify project constraints	08.10.2025	12.10.2025	5 days
	1.5	Create project schedule and assign team roles	10.10.2025	15.10.2025	6 days
	1.6	Prepare documentation for submission (business goal & scope)	14.10.2025	**22.10.2025**	9 days
2. Analysis and System Design (Lab 2)	2.1	Analyze the problem and define system modules	20.10.2025	25.10.2025	6 days
	2.2	Analyze the preprocesing of the data	23.10.2025	29.10.2025	7 days
	2.3	Design the puzzles structers and data entities (Book, Extract, Page, etc.)	27.10.2025	02.11.2025	7 days
	2.4	Define API endpoints and inter-module relations	31.10.2025	04.11.2025	5 days
	2.5	Prepare UI mockups and navigation flow	02.11.2025	**05.11.2025**	4 days
3. Implementation (Lab 3)	3.1	Implement backend structure	04.11.2025	11.11.2025	8 days
	3.2	Develop NLP preprocessing and riddle generation modules	08.11.2025	15.11.2025	8 days
	3.3	Implement main game mechanics (extracts, levels, scoring)	13.11.2025	20.11.2025	8 days
	3.4	Create frontend components	15.11.2025	23.11.2025	9 days
	3.5	Test modules with sample data	21.11.2025	**26.11.2025**	6 days
4. Testing and Optimization (Lab 4)	4.1	Develop unit tests for backend and NLP	24.11.2025	01.12.2025	8 days
	4.2	Perform functional tests and fix bugs	30.11.2025	06.12.2025	7 days
	4.3	Optimize response times	04.12.2025	08.12.2025	5 days
	4.4	Conduct internal system validation	07.12.2025	09.12.2025	3 days
	4.5	Document test results and improvements	08.12.2025	**10.12.2025**	3 days
5. Integration, Finalization and Documentation (Lab 5)	5.1	Integrate frontend and backend modules	09.12.2025	15.12.2025	7 days
	5.2	Conduct acceptance and usability tests	14.12.2025	20.12.2025	7 days
	5.3	Add multilingual (PL/EN) interface and final UI polish	18.12.2025	23.12.2025	6 days
	5.4	Prepare deployment documentation and setup guide	22.12.2025	29.12.2025	8 days
	5.5	Write user's guide and finalize documentation	28.12.2025	04.01.2026	8 days
	5.6	Conduct final review and project presentation	03.01.2026	**07.01.2026**	5 days

5 Risk Analysis

The analysis identifies key internal and external factors that may influence the success of the system. It evaluates the project's strengths and weaknesses (internal factors) as well as potential opportunities and threats (external factors).

The purpose of this analysis is to provide a comprehensive overview of the project's current position and development environment, highlighting areas that can support or hinder its completion. It also helps in risk identification and strategic planning, allowing the team to anticipate potential challenges and prepare appropriate preventive measures.

Each identified threat is later described in detail with an assessment of its likelihood, potential sources, and proposed actions to minimize its occurrence.

Table 4 SWOT analysis.

SWOT	Threats	Opportunities
Internal	1. Not enough time due to other projects 2. One of team members not being able to finish the project due to personal reasons 3. Problems with communication between members of the team 4. One member of the team falls behind in their work, blocking the progress of the other 5. Small development team and limited technical experience	1. Motivated team because this project must be finished before the team members can graduate 2. Possibility of asking a tutor for advice 3. Risk free experimentation because budget is virtual so the project will not lose any money 4. Growing demand for mobile and browser-based learning tools 5. Fluency in polish language
External	6. The main source of open-domain texts closing 7. Warsaw University of Technology is forced to close	6. No similar web-applications that focus on polish literature 7. Easy access to open-domain texts 8. Access to a group of people who speak polish as second language, who can help in testing the application

Threat 1: Not enough time due to other projects

The team members have other assignments, courses, projects, and personal business to attend. This can result in delays and missed deadlines.

Probability: Very high

Preventive actions:

- Create a detailed schedule and ensure each team member is following.
- Include time buffers in the schedule to handle any unexpected delays.
- Hold team meetings to monitor progress.

Value of threat: Medium, lack of time directly affects quality of work.

Threat 2: One of team members not being able to finish the project due to personal reasons

One of the team members can potentially drop out, take a leave, or have an emergency. This can cause their part of the project to remain unfinished.

Probability: Medium

Preventive actions:

- Conduct regular status updates so other members understand each part of the system.

Value of threat: High, this can be critical especially if this happens near the deadline.

Threat 3: Problems with communication between members of the team

Misunderstandings or irregular communication can cause delays and lead to inefficiency.

Probability: Medium

Preventive actions:

- Work on communication in the team.
- Solve any conflicts as soon as they arise.
- Decide early on how to share feedback.

Value of threat: Medium, lack of teamwork affects the quality of work.

Threat 4: One member of the team falls behind in their work, blocking the progress of the other

If one team member misses a deadline, the dependent parts may be delayed.

Probability: High

Preventive actions:

- Identify dependent task early in the project.
- Track task competition.

Value of threat: High, delays can accumulate causing the whole project to be delayed.

Threat 5: Small development team and limited technical experience

The project is developed by a small student team with limited prior experience in full-stack development, natural language processing, and system deployment. This can lead to slower progress, technical errors, or difficulties integrating different components of the system.

Probability: High

Preventive actions:

- Divide the project into small, manageable modules with clear ownership and deadlines.
- Use well-documented, open-source frameworks and existing libraries to reduce development complexity.
- Schedule short weekly review meetings to track progress and identify problems early.
- Consult online documentation, academic materials, or mentors when facing technical challenges.

Value of threat: High, lack of experience can lead to mistakes.

Threat 6: The main source of open-domain texts closing

The project relies on open-source repositories such as *Wolne Lektury* [2]. If access to these databases is restricted or discontinued, it would limit the availability of materials used for text-based gameplay.

Probability: Low

Preventive actions:

- Download and locally archive all public-domain texts used in the project.
- Prepare fallback texts (e.g., manually digitized or university-provided materials).

Value of threat: Medium, another source would need to be found.

Threat 7: Warsaw University of Technology is forced to close

In an extreme scenario (e.g., public emergency, long-term lockdown, or institutional shutdown), the university may suspend student access to infrastructure, supervision, or laboratory sessions.

Probability: Very low

Preventive actions:

- Maintain off-campus copies of all project files using cloud repositories (GitHub, Google Drive).
- Ensure all development tools are open-source and available for remote use.
- Establish online communication channels between team members and the supervisor.

Value of threat: High, this is a Bachelor Thesis project so closing of the University would cause a critical disruption.

6 System Architecture

The Language Game based on Polish Literature is planned to be implemented as a web application consisting of four components shown below. Two of those components: the text management module and the scoring module, are responsible for accessing and modifying datasets: of books and of user's scores respectively.

The frontend module handles user interface. It displays the riddles and allows users to solve them with the help of interactive elements that users can click or drag. While a riddle is being solved the frontend module is constantly communicating with the riddle module to check if the user is solving the riddle correctly.

The riddle module has two main goals: to first generate a riddle and then to manage it by checking if the user is solving it correctly. This process is repeated until the whole level is solved.

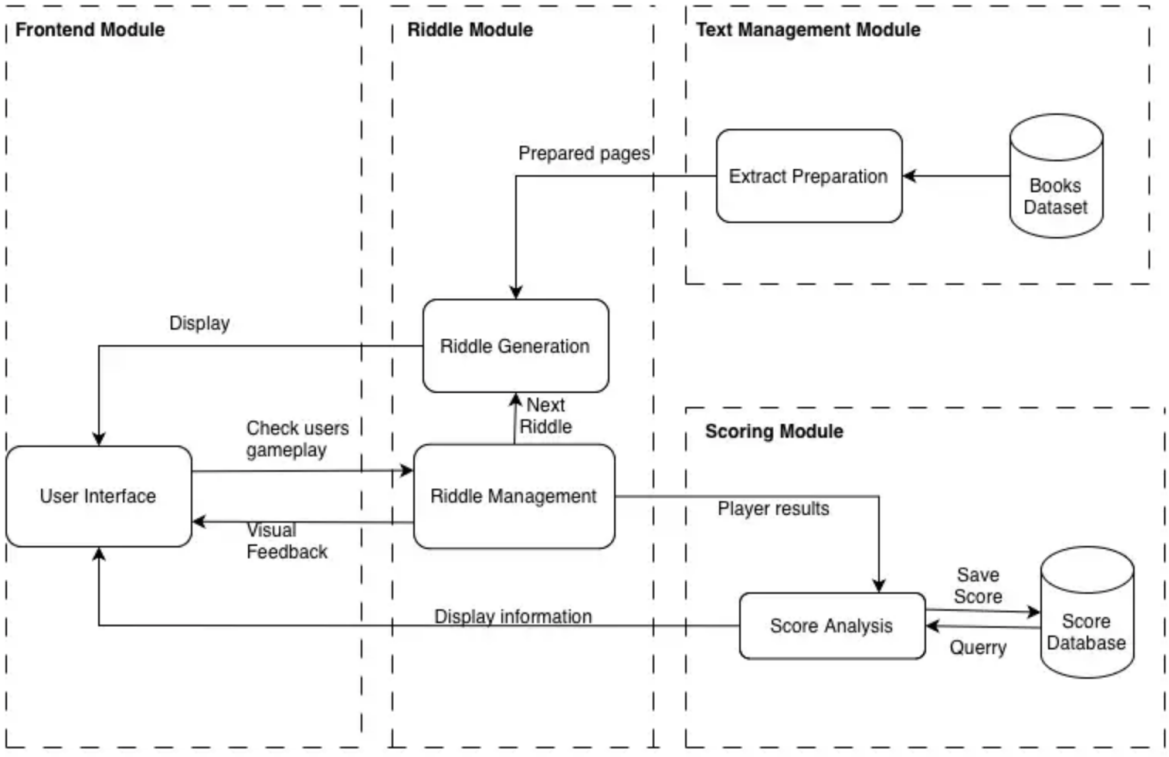

Figure 2 Application Architecture Diagram

7 Modules design

7.1 Text Management Module

The text management module handles extracts. The system stores all extracts as files sorted into folders corresponding to the books they belong to. The text management module is responsible for navigating the files, fetching the correct extract, and dividing it into pages.

7.2 Riddle Module

The riddle module has two main functions. Firstly, it is responsible for choosing a minigame based on user preferences and generating the riddle. After that the riddle module communicates with the frontend module to check if the user is solving the riddle correctly. When the riddle is fully solved the riddle module repeats the process for the next page until either the user loses, or the level is completed.

7.3 Frontend Module

The frontend is responsible for the user interface. It has two tabs, one allowing user to choose the types of minigames they want to play and the other to choose the book. After the user chooses the book the frontend module allows them to choose one of unlocked extracts and starts the game. The frontend module has multiple interactive components that allow the user to complete riddles.

7.4 Scoring Module

The scorings module purpose is to allow player to receive feedback by processing their results into numerical score. Then the scoring module checks if the new score is better than existing one for the extract – which for new extracts is 0 – and save it. The scoring module then informs the frontend module if a new extract should become unlocked.

8 Communication

8.1 Frontend Communication

The frontend of the application will be developed using React and the backend – other modules - will be using Python. The communication between the frontend and the backend will be possible thanks to API calls. The backend modules will be communicating between themselves with function calls.

8.2 Scoring Database Communication

The scores user gathered for each extract they played will be stored in the database implemented in SQLite3. The Scoring Module is responsible for modifying and accessing those data. An unlocked, unplayed extract will be stored in the database with points set to 0, a locked extract's score will be –1. The Scoring Module will send the retrieved scores to frontend module so they can be formatted and displayed.

8.3 Riddle Communication

The Riddle Module serves and communication coordinator between backend modules. When it receives the instruction to begin a level from the Frontend module it uses function call to use the Text Management Module to fetch the extract needed. After the game is completed it sends the feedback to the Scoring Module. During the game the Riddle Module constantly communicates with Frontend in real-time through API calls.

9 Main Components Description

9.1 Activity Diagram

The activity diagram shown in Figure 3 describes the actions user can do in the application. After starting the application user can choose to either switch to a tab that will allow them to select the types of riddles they would like to play or stay on the default tab which allows them to pick one of the books available.
After selecting a book user can then select one of the extracts of that book, if it is unlocked the user can then play the level based on that extract. When the user is done playing the level they are informed of the results.

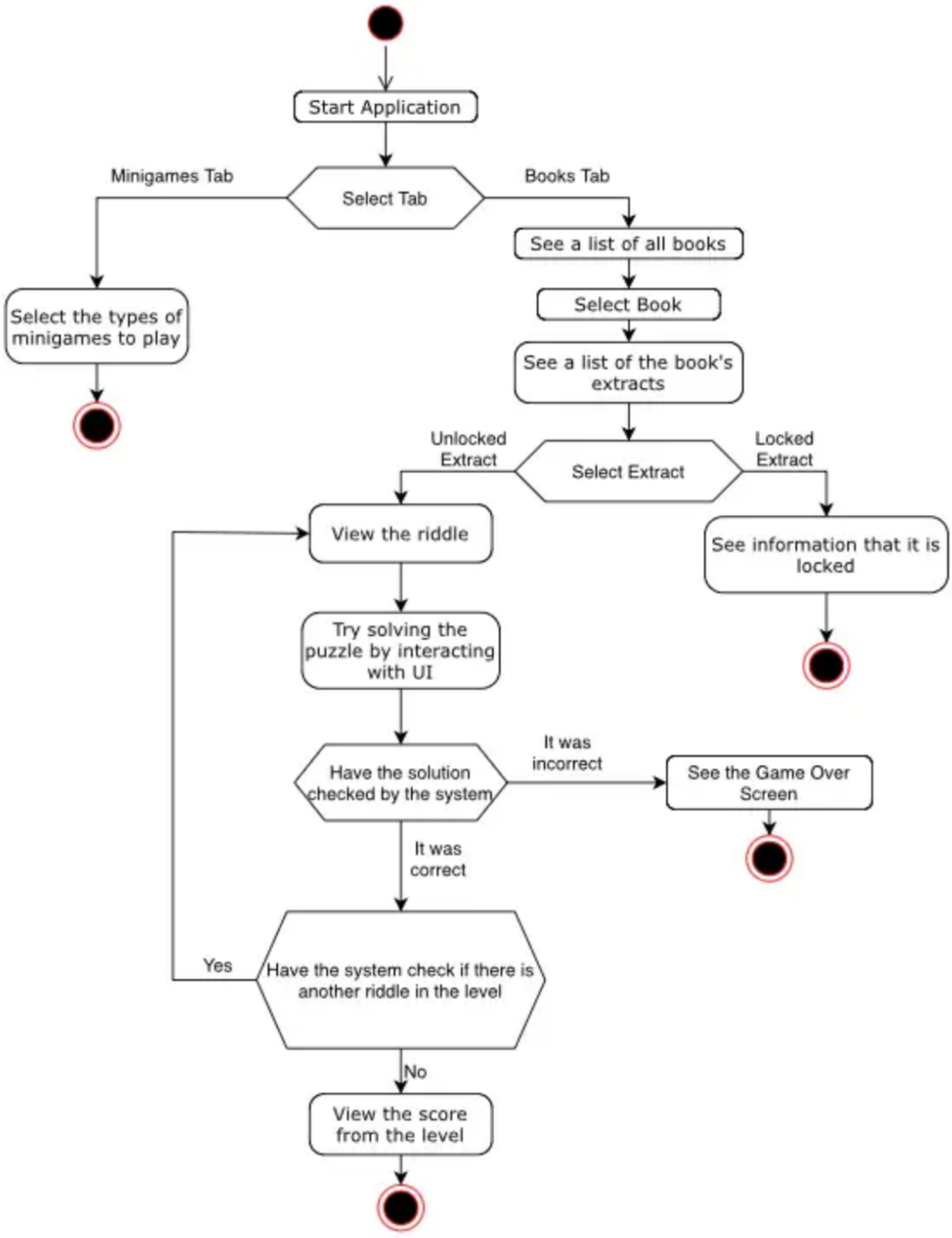

Figure 3 Activity Diagram

9.2 Class Diagram

Figure 4 presents the class diagram, which describes how the frontend, backend, and text-processing modules collaborate to make the application function.
The frontend is responsible for handling user interaction, displaying book content, puzzles, progress results, and feedback. It is implemented in React and structured around components such as LibraryView, PuzzleView, and ResultsView. These components communicate with the backend through REST API endpoints and manage the user session stored in the browser. The frontend does not store gameplay logic locally - all puzzle generation and evaluation are delegated to the backend.

The backend handles text processing, puzzle generation, scoring, and database interaction. The central class is GameEngine, which coordinates three main services:

- TextProcessor, responsible for linguistic analysis using the Morfeusz2 library [3] and additional NLP tools. It extracts tokens, parts of speech, and syntactic patterns from book fragments.
- PuzzleGenerator, which transforms processed text into playable riddles based on the selected puzzle type (e.g., fill-in-the-blank, word replacement, error detection).
- ScoreManager, which evaluates player answers, calculates final scores, and stores the best result for each extract and session.

The DatabaseManager class ensures interaction with the SQLite database, handling queries related to books, extracts, generated levels, and stored user results. Since the system no longer offers user accounts, results are linked to an anonymous Session identifier stored in the browser. Finally, the SessionTracker module manages session lifecycle, allowing users to return to unfinished games without authentication.

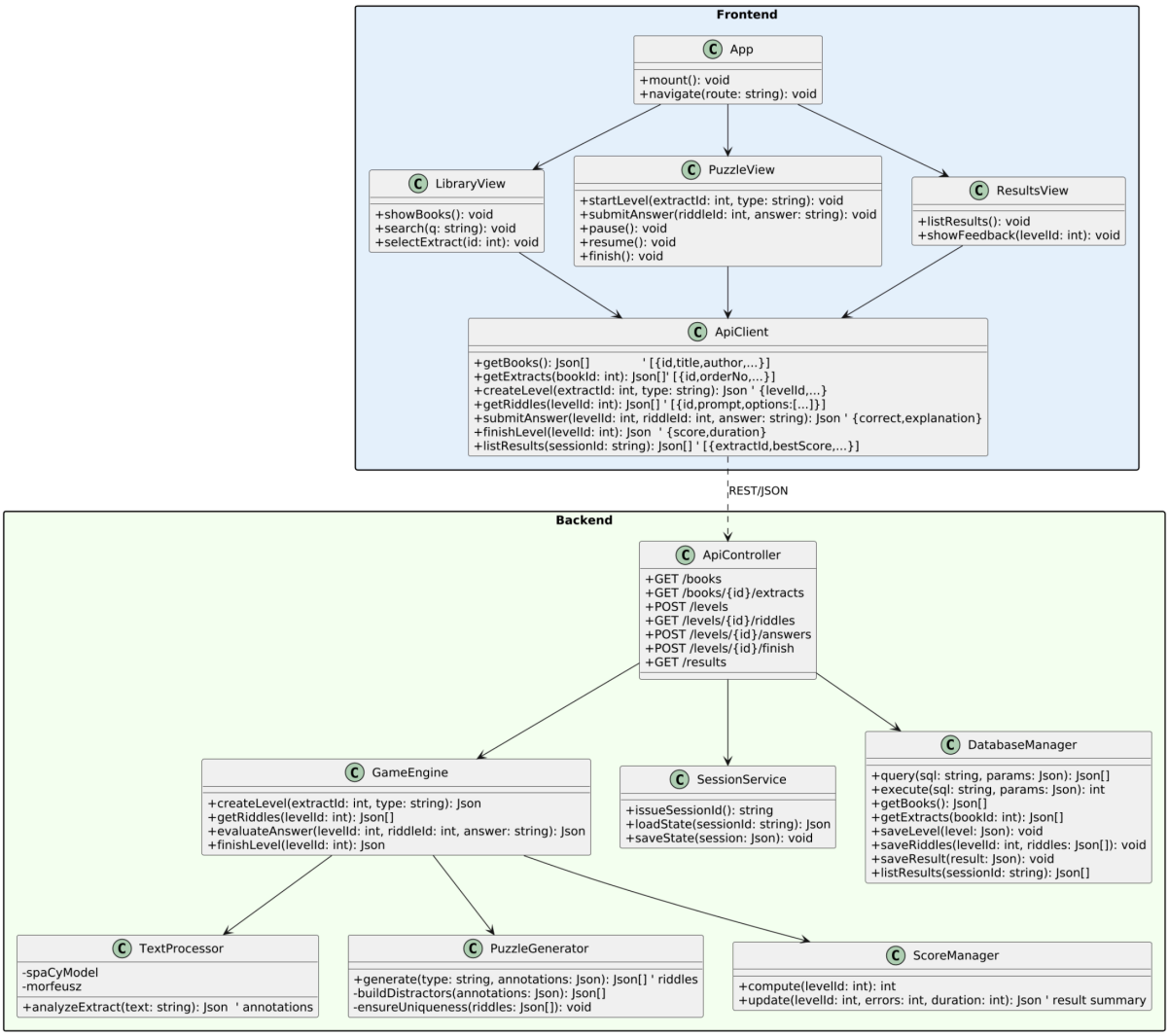

Figure 4 Class Diagram

10 Technology Selection

This section presents the technologies selected for the development of the application. The stack was chosen to support natural language processing in Polish, fast web development, and ease of deployment in an academic environment. All selected tools are open-source, well documented, and actively maintained.

10.1 Python

Python is used as the main programming language for the backend and text-processing pipeline. It is well suited for this project due to its rich ecosystem of natural language processing, data analysis, and web development libraries. Python enables rapid prototyping and readable code, which is essential for a small development team working under time constraints.

10.2 FastAPI

FastAPI was selected as the backend web framework responsible for serving the REST API. It offers high performance, automatic OpenAPI documentation, async support, and a simple learning curve [4]. Compared to heavier frameworks (e.g. Django), FastAPI provides the necessary functionality without enforcing unnecessary architectural overhead. This makes it a suitable choice for a lightweight, data-driven project.

10.3 SQLite

SQLite is used as the embedded database engine. The backend communicates with the database directly through Python's built-in sqlite3 module, without using an ORM layer. This keeps the architecture minimal and avoids overhead associated with migrations and model abstraction, which would be unnecessary for a single-user, file-based system.

10.4 spaCy

SpaCy is the primary NLP library used to process literary texts, including tokenization, part-of-speech tagging, sentence segmentation and lemmatization. It offers a pretrained model for Polish that is sufficiently fast to enable puzzle generation at runtime [5].

10.5 Morfeusz2

Morfeusz2 is used as an auxiliary tool for detailed morphological analysis of Polish words, especially for tasks where inflection, case or ambiguity resolution affects puzzle generation. It complements spaCy by providing more fine-grained morphological tags based on the SGJP dictionary.

10.6 React + TypeScript

The presentation layer of the system is implemented as a React single-page application (SPA). TypeScript provides static type checking, reducing runtime UI errors and ensuring compatibility with typed API responses. React allows efficient rendering of text-based puzzles, dynamic progress updates, and page-to-page transitions without reloading [6].

10.7 Tailwind CSS

Tailwind CSS is used to style the frontend interface. Its utility-first approach reduces the need for custom CSS files and accelerates development by relying on predefined responsive layout primitives [7]. This is appropriate for a project that does not require custom graphic design but must remain accessible and readable on multiple screen sizes.

11 User Interface Vision

The following figures present a set of example user interface screens designed for the application. They represent conceptual, simplified mock-ups intended to illustrate the main interaction flow rather than final visual or technical implementation details.
The game interface shown in this section reflects one possible gameplay mode (Fill in the Blanks), while the application is designed to support additional puzzle types and views in later development stages.
The purpose of this section is to demonstrate the planned layout, navigation logic, and user experience, rather than to define the final graphical design.

11.1 Book Selection Screen

The figure 5 presents the Library Screen, which is the starting point of the application. The screen displays a searchable and sortable list of available literary works. Each row contains the book title, author, and year of publication, allowing the user to quickly recognize and select a text of interest. A global search bar at the top enables keyword-based filtering, while the *Sort* and *Filter* buttons allow refinement by metadata such as genre, publication year, or difficulty level.
Once a book is selected, the user is redirected to a preview of available extracts, where the gameplay session begins. This screen acts as a central navigation hub, providing intuitive access to the literary content used throughout the game.

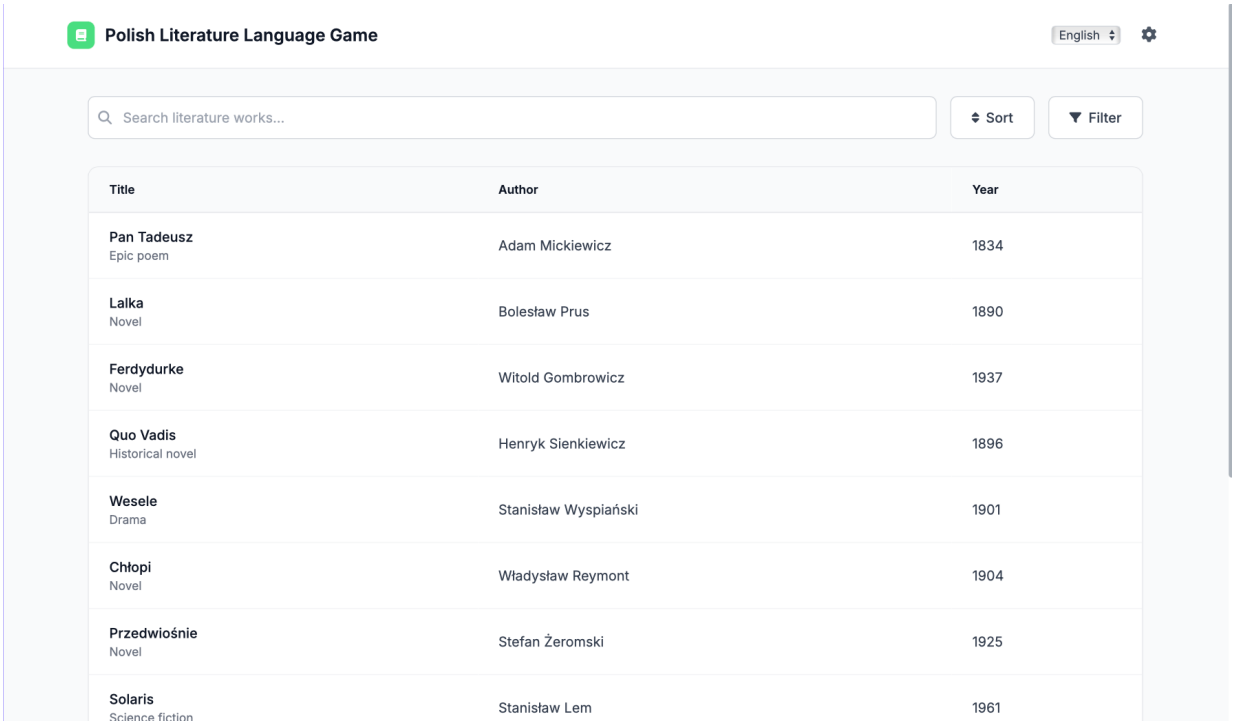

Figure 5 Book Selection Screen

11.2 Gameplay Screen

The figure 6 illustrates the Gameplay Screen for the *Fill in the Blanks* puzzle mode. The user is presented with a literary extract that contains several missing words, each replaced by an empty placeholder indicated with a dashed outline.

Below the text, a set of word options is displayed as selectable buttons. The user must drag or click the correct word to complete each blank. A timer and progress counter are shown in the top bar, tracking the current page number and number of blanks solved.
The interface focuses on clarity and minimal distraction, ensuring that the user's attention is directed toward the text. Incorrect selections are visually marked, while correct answers update the placeholder in real time. Completion of all blanks advances the user to the next page or ends the level, depending on extract length.

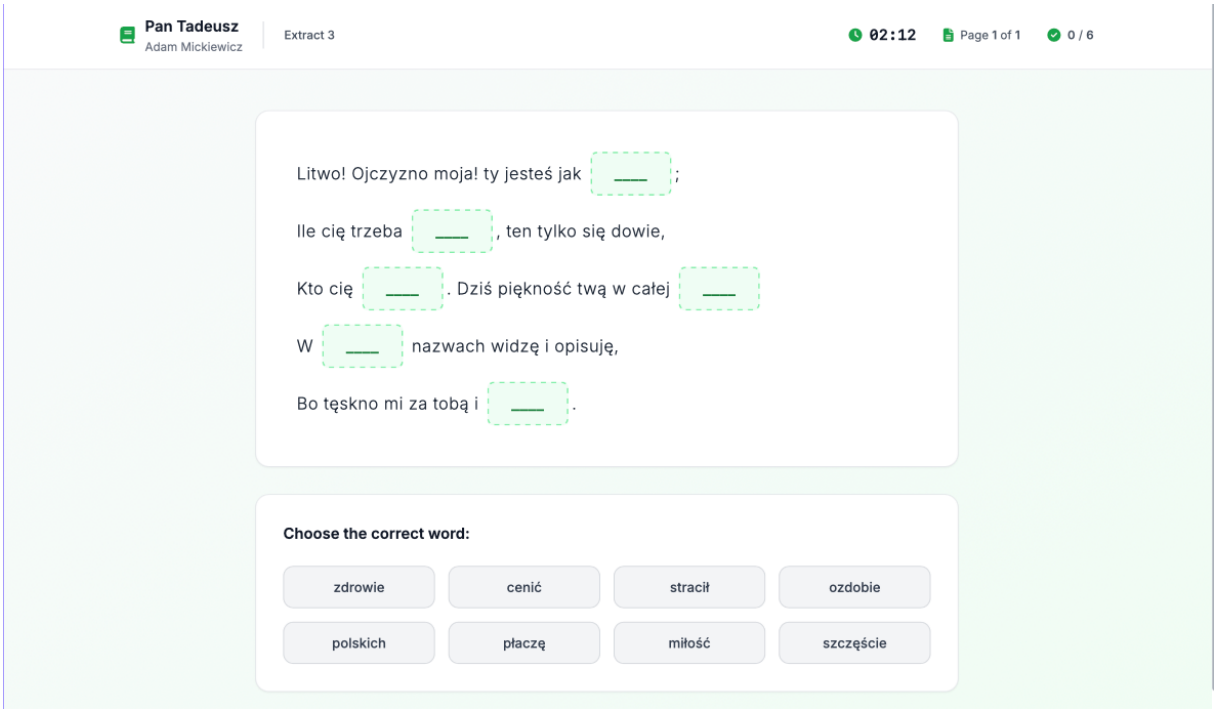

Figure 6 Gameplay Screen

11.3 Post-Game Results Screen

The figure below shows the Post-Game Results Screen, which summarizes the user's performance after completing an extract. The final score is calculated based on accuracy, number of mistakes, and total time spent.
A performance breakdown section provides detailed statistics, including accuracy percentage, completion time, mistake count, and the number of pages solved.
Action buttons at the bottom allow the user to replay the same extract, proceed to the next one in sequence, or return to the Library screen.
The purpose of this screen is to provide immediate feedback and reinforce learning motivation by visualizing progress in a clear and encouraging manner.

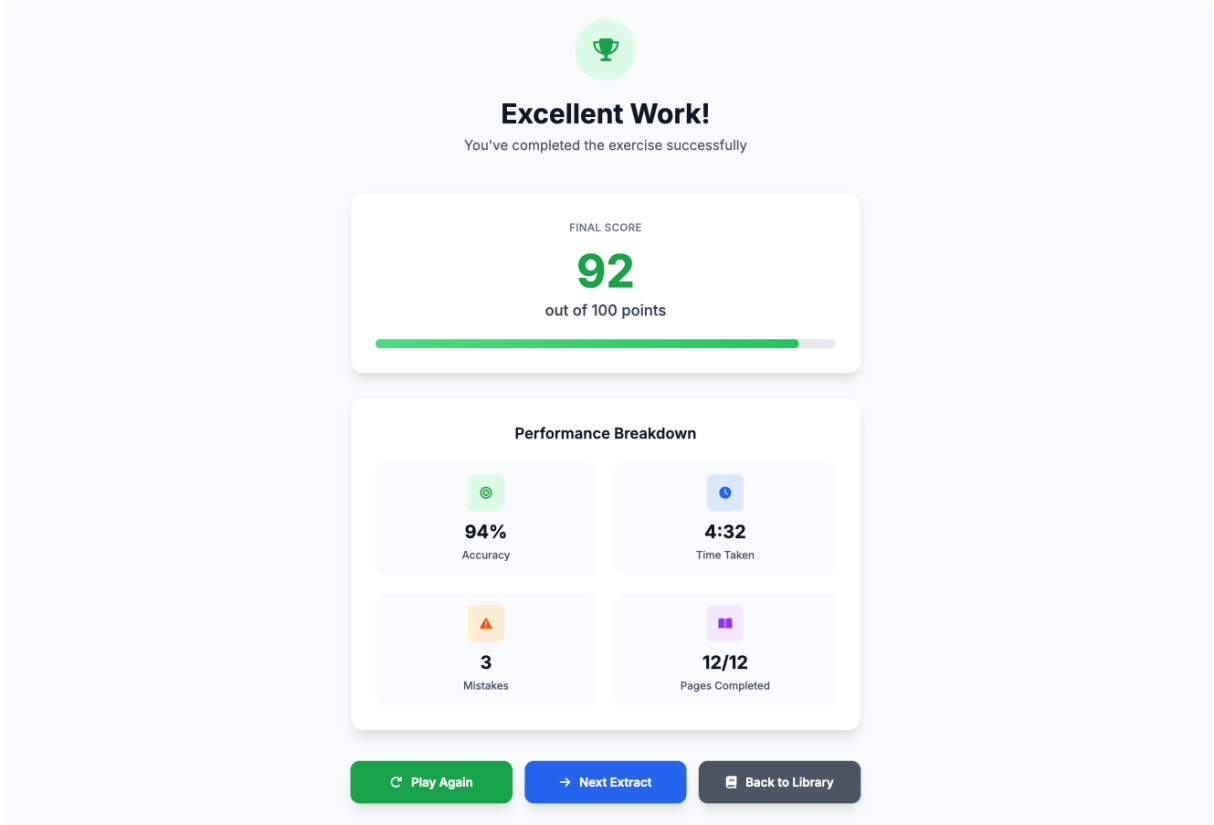

Figure 7 Results Screen

12 Bibliography

[1] Local No. 12, "Dear Reader," [Online]. Available: https://apps.apple.com/us/app/dear-reader/id1470280172. [Accessed 10 9 2025].

[2] F. N. Polska, "Fundacja Nowoczesna Polska," [Online]. Available: https://wolnelektury.pl. [Accessed 19 10 2025].

[3] W. Kieraś and M. Woliński, "Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego," *Język Polski,* vol. XCVII, no. 1, p. 75–83, 2017.

[4] FastAPI Project, "FastAPI – FastAPI documentation," 2024. [Online]. Available: https://fastapi.tiangolo.com/. [Accessed 10 10 2025].

[5] Explosion AI, "spaCy · Industrial-strength Natural Language Processing," [Online]. Available: https://spacy.io/. [Accessed 19 10 2025].

[6] Meta Platforms, Inc., "React – Official Documentation," [Online]. Available: https://react.dev/. [Accessed 22 10 2025].

[7] Tailwind Labs Inc., "Tailwind CSS Documentation," [Online]. Available: https://tailwindcss.com/docs. [Accessed 22 10 2025].