Dokumentacja Gry "Krzyżówka"

1. Wprowadzenie

Niniejsza dokumentacja opisuje grę słowną (krzyżówkę) stworzoną w Pythonie z wykorzystaniem biblioteki Tkinter. Gra jest przeznaczona dla od 1 do 4 graczy i oferuje różnorodne tryby oraz poziomy trudności. Projekt został zrealizowany zgodnie z dobrymi praktykami programowania, z podziałem na moduły odpowiedzialne za logikę gry, interfejs użytkownika, zarządzanie bazą danych i generowanie dokumentacji.

2. Opis Gry

Gra "Krzyżówka" umożliwia użytkownikom rywalizację w rozwiązywaniu zagadek słownych.

2.1. Tryby Gry:

- **Tryb "Na czas" (For Time):** Gracze mają ograniczony czas na rozwiązanie wszystkich pytań. Im szybciej i dokładniej odpowiadają, tym lepiej. Limit czasu zależy od wybranego poziomu trudności.
- **Tryb "Do błędu" (To mistake):** Gra kończy się dla gracza po pierwszej błędnej odpowiedzi lub gdy skończą się pytania.

2.2. Poziomy Trudności:

- **Łatwy (Easy):** Pytania o niskim stopniu trudności, dłuższy limit czasu w trybie "Na czas".
- **Średni (Medium):** Umiarkowany poziom trudności, standardowy limit czasu.
- **Trudny (Hard):** Trudne pytania, krótszy limit czasu.

2.3. Liczba Graczy:

Gra obsługuje od 1 do 4 graczy.

2.4. Proces Gry:

1. **Główne menu:** Umożliwia dostęp do opcji "Rozpocznij grę", "Ustawienia", "Gracze" i "Wyjście".
2. **Ustawienia gry:** Przed rozpoczęciem gry użytkownik wybiera liczbę graczy, poziom trudności i tryb gry. Można także wybrać język pytań (polski lub angielski) w osobnym oknie ustawień.

3. **Rejestracja/Logowanie graczy:** Każdy gracz musi się zalogować lub zarejestrować. Istnieje opcja generowania nazw "Gość" dla szybkiej gry. Hasła są haszowane przed zapisaniem w bazie danych.
4. **Ekran gry:** Wyświetla bieżące pytanie, pole do wprowadzania odpowiedzi, stoper (jeśli tryb "Na czas"), aktualnego gracza i wyniki. Użytkownik widzi również podpowiedź w postaci długości słowa.
5. **Sprawdzanie odpowiedzi:** Po wprowadzeniu odpowiedzi system weryfikuje jej poprawność. W przypadku poprawnej odpowiedzi gracz zdobywa punkty i przechodzi do następnego pytania. W przypadku błędnej odpowiedzi, w trybie "Do błędu", kolejka przechodzi na następnego gracza.
6. **Pauza:** W trybie "Na czas" dostępna jest opcja pauzy, która zatrzymuje stoper i ukrywa podpowiedź do odpowiedzi.
7. **Zakończenie gry:** Gra kończy się po rozwiązaniu wszystkich pytań, upłynięciu czasu (w trybie "Na czas") lub gdy gracze się poddają. Wyświetlane są ostateczne wyniki i zwycięzcy.
8. **Zapisywanie wyników:** Wyniki każdej gry są zapisywane w lokalnej bazie danych, aktualizując statystyki graczy (liczbę rozegranych gier, zwycięstwa, porażki).

3. Struktura Projektu

Projekt składa się z następujących głównych modułów:

- `mainScript.py` : Główny skrypt, który uruchamia aplikację, zarządza menu i nawigacją między ekranami.
- `auth_module.py` : Odpowiada za proces rejestracji i logowania graczy. Obsługuje haszowanie haseł.
- `gameProcess.py` : Zawiera logikę samej gry w krzyżówkę, zarządzanie pytaniami, timerem, punktacją i interfejsem gry.
- `questionsLogic.py` : Obsługuje ładowanie pytań z pliku JSON do bazy danych i pobieranie losowych pytań na podstawie języka i trudności.
- `databaseLogic.py` : Definiuje modele bazy danych (Gracz, Gra) i konfiguruje połączenie z bazą danych SQLite.
- `pdfLogic.py` : Zawiera funkcje do generowania raportów PDF, np. listy graczy i wyników gry.
- `word_questions.json` : Plik JSON zawierający pytania do krzyżówki.
- `players.db` : Baza danych SQLite przechowująca informacje o graczach i historiach gier.
- `words.db` : Baza danych SQLite przechowująca pytania do krzyżówki.

4. Eksport Wyników

W ramach projektu zaimplementowano możliwość eksportu wyników w formacie PDF.

- **Lista graczy:** W menu "Gracze" dostępny jest przycisk "Download Players PDF", który generuje raport PDF z listą zarejestrowanych graczy i ich statystykami.

- **Wyniki gry:** Po zakończeniu każdej gry, na ekranie wyników dostępny jest przycisk "Download Results PDF", który generuje plik PDF z podsumowaniem wyników danej gry.

Opis kodu źródłowego

Poniżej znajdują się docstringi funkcji i klas z poszczególnych modułów projektu.

Opis pliku `mainScript.py`

Opis funkcji `on_enter(e)`

Event handler for button hover (mouse enter).

Opis funkcji `on_leave(e)`

Event handler for button hover (mouse leave).

Opis funkcji `create_styled_button(parent, text, x, y, command=None)`

Creates a standardized styled button with hover effects and specific font.

Args:
parent (tk.Widget): The parent widget.
text (str): The text to display on the button.
x (int): The x-coordinate for placing the button.
y (int): The y-coordinate for placing the button.
command (callable, optional): The function to call when the button is clicked.
Defaults to None.
Returns:
tk.Button: The created button widget.

Opis funkcji `show_frame(frame_to_show)`

Hides all frames and shows the specified frame.

Args:
frame_to_show (tk.Frame): The frame to display.

Opis funkcji `clear_frame(frame)`

Destroys all widgets within a given frame.

Args:
frame (tk.Frame): The frame to clear.

Opis funkcji `show_main_menu()`

Displays the main menu frame.

Opis funkcji `display_error(message)`

Displays an error message box.

Args:
message (str): The error message to display.

Opis funkcji `select_players_func(num)`

Updates the selected player count and its display label.

Args:
num (int): The number of players selected.

Opis funkcji `select_difficulty_func(diff)`

Updates the selected difficulty and its display label.

Args:
diff (str): The selected difficulty level.

Opis funkcji `select_mode_func(mode)`

Updates the selected game mode and its display label.

Args:
mode (str): The selected game mode.

Opis funkcji `create_settings_option_section(parent_frame, title_text, y_offset, buttons_data)`

Creates a section for game settings (players, difficulty, mode).

Args:
parent_frame (tk.Frame): The frame to place widgets in.
title_text (str): Title for the section (e.g., "Choose the amount of players").
y_offset (int): Vertical position offset for the section.
buttons_data (list): List of tuples (button_text, command_func, arg).

Opis funkcji `create_game_settings_ui()`

Initializes/re-creates the game settings UI on the game_frame.

Opis funkcji `start_game()`

Handles showing the game settings frame and preparing it for a new game setup.

Opis funkcji `check_ready(error_message)`

Checks if all game settings are selected. If so, initiates player registration.

Args:
error_message (str): The message to display if settings are incomplete.

Opis funkcji `settings_menu()`

Creates and displays the settings window for language selection.

Opis funkcji `set_language(lang)`

Sets the selected language and updates the display label.

Args:
lang (str): The language code ('pl' or 'en').

Opis funkcji `show_players()`

Displays the list of registered players with their statistics.

Opis `auth_module.py`

Opis funkcji `start_registration(parent_frame, num_players, difficulty, game_mode, language, on_game_finish_callback)`

Starts the player registration process.

Args:
parent_frame (tk.Frame): The parent frame for registration UI.
num_players (int): The total number of players to register.
difficulty (str): The selected game difficulty.
game_mode (str): The selected game mode.

language (str): The selected language for questions.
on_game_finish_callback (callable): Callback function to run after game finishes.

Opis funkcji `display_registration_ui(parent_frame, current_player_index, num_players, difficulty, game_mode, language, on_game_finish_callback, registered_players_data)`

Displays the registration/login UI for the current player.

Args:
parent_frame (tk.Frame): The parent frame for the UI.
current_player_index (int): The index of the current player (1-based).
num_players (int): Total number of players.
difficulty (str): Game difficulty.
game_mode (str): Game mode.
language (str): Language for questions.
on_game_finish_callback (callable): Callback for game finish.
registered_players_data (list): List to store registered player data (name, id).

Opis funkcji `hash_password(password)`

Hashes a password using SHA256.

Args:
password (str): The password to hash.

Returns:
str: The hashed password.

Opis funkcji `verify_password(stored_password, provided_password)`

Verifies a provided password against a stored hashed password.

Args:
stored_password (str): The stored hashed password.
provided_password (str): The password provided by the user.

Returns:
bool: True if passwords match, False otherwise.

Opis funkcji `handle_login_register(username_entry, password_entry, mode, current_player_index, num_players, difficulty, game_mode, language, on_game_finish_callback, registered_players_data, reg_window)`

Handles user login or registration.

Args:
username_entry (tk.Entry): Entry widget for username.
password_entry (tk.Entry): Entry widget for password.
mode (str): 'register' or 'login'.
current_player_index (int): Current player index.
num_players (int): Total number of players.
difficulty (str): Game difficulty.
game_mode (str): Game mode.
language (str): Language for questions.
on_game_finish_callback (callable): Callback for game finish.
registered_players_data (list): List of registered player data.
reg_window (tk.Toplevel): The registration window.

Opis funkcji `handle_guest_login(current_player_index, num_players, difficulty, game_mode, language, on_game_finish_callback, registered_players_data, reg_window)`

Handles guest login, creating a unique guest name.

Args:
current_player_index (int): Current player index.
num_players (int): Total number of players.
difficulty (str): Game difficulty.
game_mode (str): Game mode.
language (str): Language for questions.
on_game_finish_callback (callable): Callback for game finish.
registered_players_data (list): List of registered player data.
reg_window (tk.Toplevel): The registration window.

Opis `gameProcess.py`

Opis funkcji `start_game_process(parent_frame, current_players_data, difficulty, game_mode, language, on_game_finish_callback)`

Initializes and starts the main game process.

Args:
parent_frame (tk.Frame): The frame where the game UI will be displayed.
current_players_data (list): List of Player objects for the current game.
difficulty (str): The selected difficulty level.

game_mode (str): The selected game mode ("For Time" or "To mistake").
language (str): The language for questions.
on_game_finish_callback (callable): Callback function to run after game finishes.

Opis funkcji `display_game_ui(parent_frame, current_player_id, current_player_name, question, player_scores, difficulty, game_mode, language, on_game_finish_callback, players_turn_order)`

Displays the main game UI for the current player's turn.

Args:
parent_frame (tk.Frame): The frame to display the UI.
current_player_id (int): ID of the current player.
current_player_name (str): Name of the current player.
question (dict): Dictionary containing 'question' and 'answer'.
player_scores (dict): Dictionary of player_id -> score.
difficulty (str): Game difficulty.
game_mode (str): Game mode.
language (str): Language for questions.
on_game_finish_callback (callable): Callback for game finish.
players_turn_order (list): List of (player_id, player_name) in turn order.

Opis funkcji `update_timer()`

Updates the countdown timer in "For Time" mode.

Opis funkcji `check_answer(entry_widget, current_player_id, correct_answer, player_scores, difficulty, game_mode, language, on_game_finish_callback, players_turn_order, timer_id, game_window)`

Checks the player's submitted answer.

Args:
entry_widget (tk.Entry): The entry widget containing the player's answer.
current_player_id (int): ID of the current player.
correct_answer (str): The correct answer to the question.
player_scores (dict): Dictionary of player_id -> score.
difficulty (str): Game difficulty.
game_mode (str): Game mode.
language (str): Language for questions.
on_game_finish_callback (callable): Callback for game finish.
players_turn_order (list): List of (player_id, player_name) in turn order.

timer_id (str): ID of the timer after() call.
game_window (tk.Toplevel): The game window.

Opis funkcji `display_results(parent_frame, final_scores, game_mode, difficulty, language, on_game_finish_callback)`

Displays the final results of the game.

Args:
parent_frame (tk.Frame): The frame to display the results.
final_scores (dict): Dictionary of player_id -> score for all players.
game_mode (str): Game mode.
difficulty (str): Game difficulty.
language (str): Language for questions.
on_game_finish_callback (callable): Callback function to run after game finishes.

Opis `questionsLogic.py`

Opis funkcji `load_questions_to_db(filename="word_questions.json")`

Loads questions from a JSON file into the database.

Args:
filename (str): The name of the JSON file containing questions.\

Returns:
bool: True if questions were loaded successfully, False otherwise.

Opis funkcji `get_random_question(language, difficulty)`

Retrieves a random question based on language and difficulty.

Args:
language (str): The language of the question ('pl' or 'en').
difficulty (str): The difficulty level ('Easy', 'Medium', 'Hard').\

Returns:
dict or None: A dictionary with 'question' and 'answer' if found, None otherwise.

Opis `databaseLogic.py`

Opis klasy `Base`

Base class for declarative models.

Opis klasy `Player(Base)`

SQLAlchemy model for storing player information.

Opis klasy `Game(Base)`

SQLAlchemy model for storing game history.

Opis funkcji `init_db()`

Initializes the database, creates tables if they don't exist, and loads questions.

Opis `pdfLogic.py`

Opis klasy `PDF(FPDF)`

Custom PDF class extending FPDF for common header/footer.

Opis funkcji `generate_players_list_pdf(filename_prefix="players_list")`

Generates a PDF report of all registered players and their statistics.

Args:
filename_prefix (str): Prefix for the output PDF filename.

Opis funkcji `generate_game_results_pdf(players_data, final_scores, game_mode, difficulty, language, filename_prefix="game_results")`

Generates a PDF report for a single game's results.

Args:
players_data (list): List of Player objects for players in the game.
final_scores (dict): Dictionary mapping player_id to final score.
game_mode (str): The game mode (e.g., "For Time").
difficulty (str): The game difficulty (e.g., "Medium").
language (str): The language of questions.
filename_prefix (str): Prefix for the output PDF filename.