Dokumentacja Projektu

Blackjack

Kamil Siwek, Monika Smędzik, Maksymilian Stach, Kamil Tracz, Artur Turak

04-06.2024

Spis treści

Cel Projektu

Celem projektu było stworzenie w pełni funkcjonalnej gry Blackjack z możliwością obstawiania i jak największą liczbą dodatkowych funkcji podchodzących pod kategorię „quality of life”.

Zakres Projektu

Projekt obejmował trzy główne segmenty. Są to następująco prace wstępne, analiza rynkowa i testy oraz dopracowanie projektu. Chcieliśmy stworzyć poza samą grą funkcjonalny interface okienkowy, samouczek oraz możliwość zmian kosmetycznych, te opcje jednak nie są w pełni gotowe więc nie znajdują się w wydaniu gry na dzień premiery. Będą one jednak dodane w późniejszych aktualizacjach.

Technologie i Narzędzia

W projekcie wykorzystaliśmy język programowania Python oraz środowisko programistyczne PyCharm. Całość projektu dodawana była do repozytorium w serwisie Github, gdzie również znajduje się backlog produktu. Do komunikacji ze sobą zdecydowaliśmy się na wykorzystanie komunikatorów: Messenger – do komunikacji tekstowej i przekazywania ogłoszeń, Discord – do pracy zdalnej. Najważniejsze kwestie projektu były za to omawiane na spotkaniach, które odbywały się na początku i na końcu każdej części projektu, plus jedno dodatkowe omawiające przeprowadzone testy (w sumie siedem).

Funkcjonalności Systemu

- **main.py**
  - Importowane Moduły
    - time: Umożliwia pracę z czasem (np. opóźnienia).
    - class_Deck: Moduł zawierający klasę Deck do obsługi talii kart.
    - F_bot: Moduł zawierający funkcję obsługującą bota
    - random: Umożliwia generowanie liczb losowych.
    - players: Moduł zawierający klasę players do obsługi graczy.
    - os: Umożliwia np. czyszczenie ekranu konsoli
  - Zmienne Globalne
    - sum_total: Pomaga w przechowywaniu aktualnej wartości pieniędzy, którą postawił gracz.
  - Funkcje
    - logo():
      - Tworzy logo gry w konsoli
    - get_deck(n=0):
      - Tworzy listę talii kart.
      - Dla każdej talii w liczbie n:

- Tworzy obiekt Deck z modułu class_Deck.
- Tasuje talię (p.Shuffle()).
- Dodaje talię do listy tab_deck.
- Zwraca listę talii tab_deck.

▪ get_card(tab, n):
- Losowo dobiera kartę z jednej z talii w liście tab.
- Jeśli talia jest pusta, usuwa ją z listy i zmniejsza licznik talii n.
- Zwraca dobraną kartę.

▪ get_players(i, p, p_total):
- Tworzy listę graczy tab_players.
- Dla każdego gracza w liczbie i:
- Tworzy obiekt players z początkową sumą pieniędzy p_total i wkładem p.
- Dodaje gracza do listy tab_players.
- Funkcja zwraca listę graczy.

▪ split(tab_players, number_player, total):
- Wykonuje akcję "split" na wskazanym graczu number_player.

▪ player_pass(tab_players, number_player):
- Gracz pasuje, traci pieniądze w wysokości swojego zakładu, i jest usuwany z listy graczy.

▪ win(tab_win):
- Dodaje wartość wygranej sum_total dla graczy w liście tab_win.

➢ Główna Funkcja if __name__ == "__main__":

▪ Obsługa głównej pętli gry:
- Na samym początku są sprawdzane wszystkie możliwości wygrania i przegrania gracza.
- Następnie losowane są karty dla Krupiera, bota i gracza
- Na 3 turze: bot i gracz mają możliwość: podbicia stawki lub wyciągnięcia 3 karty
- Po zakończonej grze, gracz może pozostać w grze i zacząć nową rozgrywkę z aktualną wartością pieniędzy.

❖ **Class_Card.py**

➢ Klasa Card

▪ Stałe
- Konstruktor __init__

▪ def __init__(self, window, rank, suit, value):
- window: Okno Pygame, w którym karta będzie wyświetlana.
- rank: Ranga karty (np. "As", "Król", "2").
- suit: Kolor karty (np. "Pik", "Karo").
- value: Wartość karty (liczba całkowita).

        - cardName: Nazwa karty, złożona z jej rangi i koloru.
    - Metody
        - getName(self): Zwraca nazwę karty (rank + suit).
        - getValue(self): Zwraca wartość karty.
        - getSuit(self): Zwraca kolor karty.
        - getRank(self): Zwraca rangę karty.

- **Class_Deck.py**
    - Importowane Moduły
        - random: Umożliwia generowanie liczb losowych, w tym przypadku używane do tasowania talii.
        - class_Card: Moduł zawierający klasę Card, która reprezentuje pojedynczą kartę w talii.
    - Stałe
        - color: Krotka zawierająca kolory kart ("karo", "pik", "karo", "trefl").
        - BLACKJACKDictionary: Słownik zawierający rangi kart i ich wartości punktowe zgodnie z zasadami Blackjacka.
    - Konstruktor __init__
        - def __init__(self, window, ValueDictionary=BLACKJACKDictionary):
            - window: Okno Pygame, w którym karty będą wyświetlane (Przygotowane pod rozbudowę gry).
            - ValueDictionary: Słownik zawierający rangi kart i ich wartości punktowe. Domyślnie jest to BLACKJACKDictionary.
            - self.startDeck: Lista zawierająca wszystkie karty w talii.
            - self.playingDeck: Lista zawierająca karty aktualnie używane w grze.
                - Tworzy pełną talię kart, iterując przez kolory (Deck.color) i rangi kart z ValueDictionary. Każda karta jest tworzona jako obiekt klasy Card z modułu class_Card.
    - Metody
        - Shuffle(self): Tasuje talię kart.
            - Kopiuje self.startDeck do self.playingDeck.
            - Ustawia wszystkie karty na stan "zakryte" (card.conceal())(Przygotowane pod rozbudowę gry).
            - Tasuje karty w self.playingDeck używając random.shuffle().
        - getCard(self): Dobiera kartę z talii.
            - Jeśli self.playingDeck jest pusta, zwraca False.
            - Usuwa i zwraca ostatnią kartę z self.playingDeck.
        - returnCard(self, card): Zwraca kartę do talii.
            - Wstawia kartę na początek self.playingDeck.

- **Game_screen.py** (Przygotowane pod rozbudowę gry)
    - Importowane Moduły
        - pygame: Biblioteka Pygame do tworzenia gier.
        - sys: Moduł systemowy Pythona, używany do zakończenia programu.
    - Funkcja main
        - Argumenty

- screen: Obiekt pygame.Surface, który reprezentuje główny ekran gry.

- Opis działania:
  - def main(screen):
    - Przechwytywanie Zdarzeń:
      - for event in pygame.event.get():
        - Pętla iterująca przez wszystkie zdarzenia w kolejce zdarzeń Pygame.
    - Obsługa Zamykania Okna:
      - if event.type == pygame.locals.QUIT: pygame.quit() sys.exit()
        - Jeśli typ zdarzenia to QUIT (kliknięcie przycisku zamknięcia okna), Pygame jest zamykany (pygame.quit()), a program kończy działanie (sys.exit()).
    - Kod Gry:
      - print("gra")
        - W obecnej formie funkcja po prostu drukuje "gra" przy każdym zdarzeniu. W rzeczywistej grze tutaj znajdowałby się kod obsługujący logikę gry.
    - Zwracanie Stanu:
      - return 1
        - Funkcja zwraca wartość 1, co może być używane do określenia stanu gry lub następnego ekranu, który ma być wyświetlany.

❖ **menu_screen.py** (Przygotowane pod rozbudowę gry)

- Importowane Moduły
  - pygame: Biblioteka Pygame do tworzenia gier.
  - sys: Moduł systemowy Pythona, używany do zakończenia programu.

- Funkcja main

  - Argumenty
    - screen: Obiekt pygame.Surface, który reprezentuje główny ekran gry.
  - Opis Działania
    - def main(screen):
      - Przechwytywanie Zdarzeń:
        - for event in pygame.event.get():
          - Pętla iterująca przez wszystkie zdarzenia w kolejce zdarzeń Pygame.
      - Obsługa Zamykania Okna:
        - if event.type == pygame.locals.QUIT: pygame.quit() sys.exit()
          - Jeśli typ zdarzenia to QUIT (kliknięcie przycisku zamknięcia okna), Pygame jest zamykany (pygame.quit()), a program kończy działanie (sys.exit()).
      - Obsługa Kliknięć Myszy:

- ♦ if event.type == pygame.locals.MOUSEBUTTONDOWN and event.button == 1:
  - ➢ Sprawdza, czy zdarzenie to naciśnięcie lewego przycisku myszy (event.button == 1).
- Sprawdzanie Obszarów Kliknięcia:
  - ♦ Obszar 1:
    - ➢ if 705 < event.pos[0] < 1214 and 341 < event.pos[1] < 1033: return 1
      - Sprawdza, czy kursor myszy znajduje się w prostokącie z współrzędnymi od (705, 341) do (1214, 1033). Jeśli tak, funkcja zwraca 1.
  - ♦ Obszar 2:
    - ➢ elif 1394 < event.pos[0] < 1814 and 500 < event.pos[1] < 1033: return 2
      - Sprawdza, czy kursor myszy znajduje się w prostokącie z współrzędnymi od (1394, 500) do (1814, 1033). Jeśli tak, funkcja zwraca 2.
  - ♦ Obszar 3:
    - ➢ elif 103 < event.pos[0] < 519 and 500 < event.pos[1] < 1033: return 3
      - Sprawdza, czy kursor myszy znajduje się w prostokącie z współrzędnymi od (103, 500) do (519, 1033). Jeśli tak, funkcja zwraca 3.
- Domyślne Zwracanie:
  - ♦ return 0
    - ➢ Jeśli żaden z powyższych warunków nie jest spełniony, funkcja zwraca 0.

❖ **player.py**

- ➢ Definicja Klasy:
  - Klasa Player jest definiowana za pomocą słowa kluczowego class.
- ➢ Metoda __init__:
  - Konstruktor __init__ inicjuje nowego gracza.
    - Przyjmuje argumenty total_player i prices, z których total_player oznacza ilość pieniędzy, a prices reprezentuje aktualny zakład gracza.
    - Inicjalizuje atrybuty total_player, prices, cards (lista kart gracza) i sum_value (suma wartości kart gracza).
- ➢ Metoda split:
  - Metoda split umożliwia graczowi podzielenie się zakładem.
  - Przyjmuje argument total, który określa kwotę, jaką gracz chce podzielić się ze stołem.
  - Jeśli gracz ma wystarczającą ilość pieniędzy (total_player >= total), dodaje kwotę do zakładu (prices) i odejmuje ją od całkowitej ilości pieniędzy gracza (total_player), zwracając True.
  - W przeciwnym razie zwraca False.
- ➢ Metoda get_card:
  - Metoda get_card dodaje kartę do ręki gracza.

- Przyjmuje argumenty card (obiekt karty) i value (wartość karty).
- Dodaje wartość karty do sumy wartości kart gracza (sum_value) oraz kartę do listy kart gracza (cards).

➢ Metoda win:
- Metoda win aktualizuje stan gracza po wygranej rundzie.
- Przyjmuje argument total, który określa wygraną sumę pieniędzy.
- Dodaje wygraną sumę do całkowitej ilości pieniędzy gracza (total_player), zeruje zakład (prices), czyści listę kart gracza (cards) oraz resetuje sumę wartości kart gracza (sum_value).

➢ Metoda get_prices i get_sum_value:
- Metoda get_prices zwraca aktualny zakład gracza.

➢ Metoda get_sum_value zwraca sumę wartości kart gracza.

❖ **settings_screen.py** (Przygotowane pod rozbudowę gry)

➢ Przechwytywanie Zdarzeń:
- Pętla for iteruje przez wszystkie zdarzenia w kolejce zdarzeń Pygame, które zostały zgromadzone od ostatniego wywołania funkcji pygame.event.get().
- Każde zdarzenie jest analizowane w pętli.

➢ Obsługa Zamknięcia Okna:
- Warunek if event.type == pygame.locals.QUIT: sprawdza, czy zdarzenie to zamknięcie okna (czy użytkownik kliknął przycisk zamknięcia okna).
- Jeśli tak, wywoływane są funkcje pygame.quit() i sys.exit(), które odpowiednio zamykają bibliotekę Pygame i kończą działanie programu.

➢ Kod Ustawień:
- Wewnątrz pętli for znajduje się kod związany z obsługą ustawień gry.
- W tym przypadku kod został zastąpiony przez prostą instrukcję print("ustawienia"), która wypisuje "ustawienia" w konsoli.

➢ Zwracanie Stanu:
- Na koniec funkcja zwraca wartość 2.

❖ **f_bot.py**

➢ Funkcja f_bot:
- Przyjmuje dwa argumenty: sum (całkowita wartość kart bota) i card (aktualna karta Krupiera).
- Po przeanalizowaniu tych wartości funkcja podejmuje decyzje o działaniu
- Funkcja zwraca wartości 1, 2 lub 3, oznaczające kolejno: Hit, Stand i Double down

❖ **tutorial_screen.py** (Przygotowane pod rozbudowę gry)

➢ Przechwytywanie Zdarzeń:

    - Pętla for iteruje przez wszystkie zdarzenia w kolejce zdarzeń Pygame, które zostały zgromadzone od ostatniego wywołania funkcji pygame.event.get().
    - Każde zdarzenie jest analizowane w pętli.
- Obsługa Zamknięcia Okna:
    - Warunek if event.type == pygame.locals.QUIT: sprawdza, czy zdarzenie to zamknięcie okna (czy użytkownik kliknął przycisk zamknięcia okna).
    - Jeśli tak, wywoływane są funkcje pygame.quit() i sys.exit(), które odpowiednio zamykają bibliotekę Pygame i kończą działanie programu.
- Kod Samouczka:
    - Wewnątrz pętli for znajduje się kod związany z obsługą samouczka gry.
    - W tym przypadku kod został zastąpiony przez prostą instrukcję print("samouczek"), która wypisuje "samouczek" w konsoli.
- Zwracanie Stanu:
    - Na koniec funkcja zwraca wartość 3.
    - Może to oznaczać, że po wywołaniu tej funkcji gracz powinien być przekierowany do ekranu samouczka.

Harmonogram Projektu

- 16.04.2024 – pierwsze spotkanie projektowe, na którym został omówiony wstępny podział projektu, specjalizacje uczestników projektu, podział ról, wstępne terminy oraz ogólny zamysł wyglądu gry (szczegółowe informacje dostępne w pliku „Notatka ze spotkania1.pdf”)
- 16.04 – 03.05.2024 – pierwsza cześć projektu zajmująca się stworzeniem podstawy programu oraz stworzenie podstawki pod interface. Zadania zostały przypisane według specjalizacji uczestników. Zadania, które miały przypisaną większą liczbę osób były wykonywane na dwa sposoby zależnie od ich natury: jeśli zadania posiadało kilka osobnych funkcji to osoby nad nimi pracujące dzieliły się tym kto wykonuje którą oraz w swoim zakresie dodawali je do repozytorium, jeśli jednak zadanie było w swojej naturze mocno połączone to aby nie tworzyć problemu z późniejszą niespójnością w trakcie łączenia osoby przypisane do zadania pracowały nad nim razem oraz tylko jedna z nich wysyłała skończony plik do repozytorium. Zadania wykonywane w trakcie tej części projektu dostępne są w backlogu oraz w zakładce issues repozytorium po przejściu na zakładkę milestones, później closed, a następnie wybierając milestone: „Cel 1: Podstawowe zasady gry”. Przykładowymi zadaniami, które znajdowały się w tej części projektu są:
    - Implementacja kart – Monia Smędzik, Kamil Siwek
    - Wstępny projekt interface – Kamil Tracz, Maksymilian Stach
    - System obstawiania – Artur Turak, Monika Smędzik
- 03.05.2024 – drugie spotkanie projektowe, na którym omówiona została pierwsza część projektu. Została tam przeprowadzona analiza wykonania zadań, ich efektywności oraz tępa. Podniesiono tam także temat braku wykonania analizy rynkowej potrzebnej do wydania gry. Zarządzono więc kilkudniową pół przerwę, w trakcie której każdy miał za zadanie przygotować tematy do poruszenia na spotkaniu rozpoczynającym drugą część projektu tak aby skorygować brakujące elementy naszego projektu (szczegółowe informacje dostępne w pliku „Notatka ze spotkania2.pdf”).

- 07.05.2024 – trzecie spotkanie projektowe, na którym omówiono plan przeprowadzenia drugiej części projektu. Podjęliśmy decyzję o postawieniu większego nacisku na sprawy związane z analizą rynkową w tej części projektu. Każda osoba dostała listę zadań do wykonania, które pokryły praktycznie każdy aspekt zarządzania branża gier. Dodatkowo zadecydowaliśmy o przeprowadzeniu testów jednostkowych, które mają za zadanie sprawdzić najważniejsze elementy kodu. Każda osoba dostała swój segment, który miała sprawdzić we własnym zakresie a następnie przedstawić rezultat na specjalnym zaplanowanym na to spotkaniu (szczegółowe informacje dostępne w pliku „Notatka ze spotkania3.pdf").
- 07.05 – 28.05.2024 – prace nad drugą częścią projektu (część testów jednostkowych), która obejmuje szeroko pojętą analizę rynku zaczynając od rozeznania w konkurencji przez monety monetyzacji na optymalizacji czasu premiery kończąc. Dodatkowo rozpoczęliśmy komunikacje z inwestorami oraz wydawcą aby zapewnić budżet oraz wsparcie doświadczonej firmy na premierę naszego projektu. Dodatkowo zostały w tej części przeprowadzone testy jednostkowe oraz po omówieniu ich na specjalnym spotkaniu rozpoczęcie pracy nad beta testami z udziałem graczy tak, aby umożliwić zebranie opinii nie tylko o błędach, ale też o ogólnym stanie naszego produktu od strony użytkownika. Zwiększyliśmy także tępo pracy podnosząc ilość zadań praktycznie trzykrotnie względem poprzedniej części projektu. Odbiło się to jednak w przedłużeniu prac o tydzień dłużej. Poważnym błędem było jednak przekazanie zadań do wykonania poszczególnym osobą w formie papierowej listy ponieważ spowodowało to zamieszanie i brak konsultacji w sprawie skorelowanych ze sobą zadań oraz w niektórych przypadkach pominięcie kluczowych aspektów zleconego zadania. Problem ten został jednak dokładnie przeanalizowany w okolicach spotkania kończącego drugą część projektu i dopiero wtedy skorygowany dodaniem wszystkiego do backlogu. Dokładne wykonane w trakcie tej części zadania dostępne są w repozytorium w zakładce issue -> milestones -> closed - > „Cel 2: Analiza rynku". Przykładowe zadania realizowane w ramach tej części:
    - Zarządzanie budżetem projektu – Monika Smędzik
    - Konsultacja poprawności stworzonych zasad gry – Artur Turak
    - Analiza istniejących na rynku modeli monetyzacyjnych – Kamil Tracz
    - Analiza recenzji innych gier karcianych – Kamil Siwek
    - Konsultacja projektu interface na zasadzie sondy ulicznej – Maksymilian Stach
- 17.05.2024 – czwarte spotkanie projektowe dotyczące analizy przeprowadzonych testów. Wykazano na nim problem z przesyłaniem plików pomiędzy funkcjami oraz zdecydowano się na sposób jego naprawy. Poruszono też temat przeprowadzenia beta testów w czasie pozostałym na drugą część projektu (szczegółowe informacje dostępne w pliku „Notatka ze spotkania4.pdf").
- 28.05.2024 – piąte spotkanie projektowe dotyczące podsumowania prac nad drugą częścią projektu. Zostały na nim omówione rezultaty działań analizy rynkowej, oraz podjęte ważne decyzje marketingowe co do rozwoju i wydania gry. Ponadto omówiono rezultaty rozmów z inwestorami i wydawcą oraz przeanalizowano feedback od testerów oraz osób ankietowanych. Zarządzono także po analizie terminów dodatkowe spotkanie następnego dnia rano w celu podjęcia decyzji dotyczących prac nad trzecią częścią projektu (szczegółowe informacje dostępne w pliku „Notatka ze spotkania5.pdf").
- 29.05.2024 – szóste spotkanie projektowe dotyczące sposobu realizacji trzeciej części projektu. Zostały na nim przeanalizowane dokładne deadliny i opóźnienie prac oraz podjęte decyzje dotyczące hierarchii prac w kolejności od najważniejszych:

dopracowanie gry do wersji zdatnej do wydania bez dodatkowych funkcji, działania potrzebne do premiery gry niezwiązane z kodowaniem, dodanie dodatkowych funkcji. Narzuciliśmy jednak bardzo wysokie tempo prac z powodu utrzymania ilości zadań podobnej do części drugiej projektu, ale zmiany terminu na kilka dni (szczegółowe informacje dostępne w pliku „Notatka ze spotkania6.pdf").

- 29.05 – 02.06.2024 – prace nad trzecią częścią projektu obejmowały złożenie kodu w całość, przetestowanie go, ewentualne poprawki oraz dopięcie spraw administracyjnych związanych z wydaniem gry. Aby zmieścić się w deadlinie musieliśmy wybierać czym się zająć. Stosowaliśmy się więc do hierarchii ustalonej na spotkaniu rozpoczynającym te część projektu. Nie udało nam się wyrobić z dodatkowymi funkcjami jednak wszystkie najważniejsze aspektu potrzebne do wydania gry zostały zrealizowane. Najwżniejszym aspektem dodanym w tej części projektu jest bot, który stosuje strategie podstawową. Dokładne wykonane w trakcie tej części zadania dostępne są w repozytorium w zakładce issue -> milestones -> open - > „Cel 3: Dopracowanie projektu i dodatkowe funkcje". Przykładowe zadania realizowane w ramach tej części:
    - Projekt modelu monetyzacyjnego – Kamil Tracz
    - Zebranie feedbacku od beta testerów – Maksymilian Stach
    - Analiza prawa hazardowego krajów z rynku docelowego – Artur Turak
    - Analiza testów przystosowania bota do gry z graczami – Monika Smędzik
    - Nawiązanie współpracy z influencerami – Kamil Siwek, Monika Smędzik
- 02.06.2024 – siódme spotkanie projektowe podsumowujące prace nad projektem, określono na nim co udało się zrobić, a czego nie. Dodatkowo zaplanowano strategie dodawania obiecanych funkcji w późniejszym terminie (szczegółowe informacje dostępne w pliku „Notatka ze spotkania7.pdf").

Zespół Projektowy

- Kamil Tracz
    - Grafika
    - Podstawa Programowania
    - Interface- projektowanie
    - Animacje
- Maksymilian Stach
    - Programowanie frontend (C++, C, Python)
    - Interface- projektowanie
    - Pisanie botów
- Artur Turak
    - Programowanie backend (Python, PHP, CSS)
    - Pisanie botów
    - Robienie podstawowych grafik
- Kamil Siwek
    - Grafika
    - Animacja
    - Podstawy Python
- Monika Smędzik
    - Programowanie (Python, C++)
    - Frontend
    - Pisanie botów

Plan Testów

Testy zostały przeprowadzone w trzech segmentach:

- Pierwszy segment – testy jednostkowe sprawdzające najważniejsze funkcje programu pod kątem wszelkich nieprawidłowości przez zespół projektowy
- Drugi segment – oddanie prototypu gry beta testerom, aby uzyskać ich opinie o produkcie i zwiększyć szanse na wykrycie błędów, które zespół mógł potencjalnie pominąć
- Trzeci segment – ponowne testy jednostkowe wersji projektu po poprawkach przeznaczonej do wydania

Zarządzanie Projektem

Projekt zarządzany był w formie przypisania każdej osobie jej segmentu, którym zarządza i postawienia jednej osoby jako głównego zarządcy projektu. Podział ról wygląda następująco:

- Kamil Tracz – główny zarządca projektu
- Artur Turak – główny programista
- Maksymilian Stach – specjalista do spraw społeczności
- Monik Smędzik – specjalistka do spraw administracyjnych
- Kamil Siwek – główny grafik

Raporty zespołu projektowego

Raporty zostały przygotowane przez każdą z osób uczestniczących w projekcie indywidualnie:

Kamil Tracz

Kamil Tracz, KamilT121, ktracz@student.agh.edu.pl

Zrealizowane zadania:

- Dodanie finalnej wersji projektu do repozytorium
- Napisanie notki ze spotkania rozpoczynającego trzecią cześć projektu
- Spotkanie rozpoczynające trzecią część projektu
- Spotkanie dotyczące podsumowania drugiej części projektu
- Napisanie notki ze sprawozdania rozpoczynającego drugą część projektu
- Porównanie opłacalności różnych sposobów wydania gry
- Analiza kalendarza premier w branży gier
- Koordynacja działań związanych z premierą gry
- Spotkanie dotyczące stanu produktu przed premierą
- Ponowne testy jednostkowe
- Sporządzenie notki ze spotkania omawiającego produkt przed premierą
- Prezentacja finalnego produktu przed inwestorami
- Wprowadzanie ograniczeń na bota pozwalających na bardziej uczciwą rozgrywkę
- Projekt modelu monetyzacyjnego
- Rozmowy ze wydawcą
- Zlecenie stworzenia reklamy produktu
- Opracowanie strategii na rozmowę z wydawcą
- Gra z botem

- Napisanie notki ze spotkania rozpoczynającego projekt
- Napisanie notki ze spotkania omawiającego testy
- Badanie rynków docelowych
- Analiza istniejących na rynku modeli monetyzacyjnych
- Organizacja spotkania omawiającego rezultaty testów projektu
- Przydzielenia fragmentów kodu do testowania do osób
- Analiza programu w poszukiwaniu poważnych błędów
- Wysłanie reprezentantów na konferencje w branży gier
- Napisanie notki ze sprawozdania podsumowującego pierwszą część projektu
- Organizacja spotkania podsumowującego pierwszy etap projektu
- Rozplanowanie zadań na pierwszy sprint
- Napisanie notki ze sprawozdania podsumowującego drugą część projektu
- Rozplanowanie zadań na drugi sprint
- Organizacja sprawozdania dotyczącego drugiej części projektu
- Przygotowanie testów sprawdzających funkcjonalność programu
- Poprawki do konceptu interface/Wykonanie interface
- Implementacja zasad gry
- Wstępny projekt interface

Nazwy commitów z datą:

- first commit – 24.04
- Merge pull request #4 from KamilT121/3-zrobienie-wstępnej-tali-kart-graficznie-rewers-i-awers – 24.04
- Merge pull request #4 from KamilT121/3-zrobienie-wstępnej-tali-kart-graficznie-rewers-i-awers – 25.04
- Delete trefl directory - 25.04
- Delete rewers_podstawowy.png – 25.04
- Koncept menu w formie png - 27.04
- Koncept menu jako projekt w programie Photoshop – 27.04
- Merge pull request #7 from KamilT121/wstepny-koncept-menu – 27.04
- Merge pull request #8 from KamilT121/2-system-obstawiania – 27.04
- Merge pull request #9 from KamilT121/6-implementacja-kart – 28.04
- Przywrócenie przypadkowo usuniętego pliku – 30.04
- Przywrócenie przypadkowo usuniętego pliku – 30.04
- Merge pull request #12 from KamilT121/10-implementacja-zasad-gry – 02.05
- Dodanie kody interface – 07.05
- Add files via upload – 31.05
- Dodanie notki ze spotkania rozpoczynającego prace – 31.05
- Dodanie testu jednostkowego – 02.06
- Dodanie notki ze spotkania 2 – 02.06
- Dodatnie notki ze spotkania numer 4 – 02.06
- Dodanie notki ze spotkania numer 3 – 02.06
- Dodanie notki ze spotkania numer 5 – 02.06
- Dodanie notki ze spotkania numer 6 – 02.06
- Merge pull request #93 from KamilT121/14-przygotowanie-testów-sprawdzających-funkcjonalność-programu – 02.06

- Merge pull request #94 from KamilT121/Notki-ze-spotkań – 02.06
- Merge pull request #95 from KamilT121/80-ponowne-testy-jednostkkowe – 02.06
- Merge pull request #96 from KamilT121/39-przygotowanie-bazy-dokumentacji-projektowej – 02.06
- Merge pull request #97 from KamilT121/69-testy-przystosowania-bota-do-gry-z-graczami – 02.06
- Merge pull request #98 from KamilT121/51-gra-z-botem – 02.06
- Dodanie wersji projektu gotowej do wydania – 02.06
- Merge pull request #99 from KamilT121/92-dodanie-finalnej-wersji-projektu-do-repozytorium – 02.06
- Dodanie notki ze spotkania numer 7 – 03.06
- Dodanie notki ze spotkania numer 6 – 03.06
- Merge pull request #100 from KamilT121/Notki-ze-spotkań – 03.06

Aspekt	Parametry	Wkład
Role	Wymienić	Zarządzanie zespołem Planowanie spotkań Tworzenie notek za spotkań Sprawdzanie poprawności kodu Praca przy interface Testowanie kodu Przygotowanie finalnej wersji projektu
Kodowanie	Liczba linii kodu	385
	Funkcje (wymienić)	Połączenie kodu w całość Praca nad kodem interface Testy jednostkowe Programowanie bota
Repozytorium	Liczba commit-ów	18
	Liczba utworzonych gałęzi	16
	Gałąź (używana – nazwa)	Notki-ze-spotkań 92-dodanie-finalnej-wersji-projektu-do-repozytorium 80-ponowne-testy-jednostkkowe 14-przygotowanie-testów-sprawdzających-funkcjonalność-programu 33-analiza-programu-w-poszukiwaniu-poważnych-błędów 10-implementacja-zasad-gry 11-poprawki-do-konceptu-interfacewykonanie-interface wstepny-koncept-menu

		69-testy-przystosowania-bota-do-gry-z-graczami Interface wstepny-koncept-menu 11-poprawki-do-konceptu-interfacewykonanie-interface
	Liczba połączonych gałęzi	0
	Liczba dni aktywności GIT	13
Dokumentowanie	Liczba standup-ów	16
	Opisy na Wiki	
Aktywność	Liczba zrealizowanych zadań	36
	Szacowana liczba godzin	49
	Ocena procentowego wkładu	22%

Artur Turak
Artur Turak, Arturllllkkkk, aturak@student.agh.edu.pl

1. Zrealizowane zadania:

1) Zadania organizacyjne:
   a) Pomoc w zebraniu odpowiedniego zespołu pod kątem potrzebnych umiejętności
   b) Pomoc w rozdzieleniu poszczególnych zadań na odpowiednie osoby
   c) Konsultacje w sprawie poprawności zasada w grze
   d) Planowanie wydania gry
   e) Planowanie rozwoju gry po premierze
   f) Analiza potrzebnej ilości miejsca na serwerze i jego zakup
2) Zadania projektowe:
   a) Konsultacje nad wyborem odpowiedniego języka programowania do projektu
   b) Wybranie odpowiednich technologii (Bibliotek)
3) Zadania programistyczne:
   a) System obstawiania: Stworzenie funkcji, która będzie umożliwiała graczowi podejmowanie decyzji: czy chce podbić stawkę, itp.
   b) Zaprojektowanie i stworzenie bota do gry, który będzie symulował grę prawdziwego gracza.
   c) Tworzenie testów do napisanych funkcji, takich jak sprawdzenie działalności bota, funkcji gracza i funkcji kart
   d) Wykrywanie na bieżąco wynikających z testów błędy i ich poprawianie
   e) Spięcie całego program w całość, czyli tak aby program główny odpowiednio komunikował się z wcześniej napisanymi funkcjami i klasami. Ponadto przygotowanie wstępnego szablonu do programu głównego (Tworzenie talii kart i graczy)
4) Zadania związane z inwestorami:
   a) Rozmowy z inwestorami
   b) Przedstawianie aktualnych posunięć w pracy nad grą i statystyk dotyczących zbieranych danych od przyszłych użytkowników
   c) Przedstawienie i omówienie gotowej gry i zapowiedz późniejszych działań nad projektem
5) Zadania marketingowe:
   a) Przygotowanie kampanii marketingowej
6) Zadania związane z dokumentacją:
   a) Pomoc w opracowaniu dokumentacji kodu
   b) Końcowe zdanie raportu z przebiegu mojej pracy

2. Raporty:

1) Zadania organizacyjne (Nazwy commit'ów):
   - Konsultacja poprawności stworzonych zasad gry #38
   - Analiza przewidywanej potrzebnej pojemności serwerowej #76
   - Rozmowy dotyczące kupna pojemności serwerowej #77
   - Analiza prawa hazardowego krajów z rynku docelowego #86
   - Identyfikacja głównych konkurentów #82

- Koordynacja działań związanych z premierą gry #83
- Opracowanie planu wydania gry #62
- Projektowanie planu rozwoju gry po premierze #60
- Organizacja beta testów #26

2) Zadania programistyczne (Nazwy commit'ów):

- Implementacja zasad gry #10
- System obstawiania #2
- Gra z botem #51
- Poprawki błędów wykrytych w fazie testów #24
- Analiza programu w poszukiwaniu poważnych błędów #33
- Przygotowanie testów sprawdzających funkcjonalność programu #14
- Zaplanowanie systemu decyzji którym będzie kierować się bot #63
- Wprowadzanie ograniczeń na bota pozwalających na bardziej uczciwą rozgrywkę #66
- Testy przystosowania bota do gry z graczami #69

3) Zadania związane z inwestorami (Nazwy commit'ów):

- Przygotowanie wizualizacji dla inwestorów #31
- Komunikacja z inwestorami #32
- Przedstawienie planu rozwoju i oczekiwanych zysków inwestorom #59
- Prezentacja finalnego produktu przed inwestorami #68

4) Zadania marketingowe (Nazwy commit'ów):

- Stworzenie kampanii e-maili marketingowych #87

4. Ocena pracy:

a) Moja ocena:
Uważam, że moje praca w projekcie była wystraczająca do oczekiwań reszty zespołu. Każde postawione mi zdanie zostało wykonane w nienaganny sposób i w wyznaczonym czasie. Sądzę, że mój wkład własny w projekt wynosił 80%.

b) Ocena pracy zespołu:
Każdy członek zespołu wykonywał swoją pracę odpowiedni i z dużym zaangażowaniem, co przełożyło się na dobry efekt końcowy, w formie skończonej gry.
Moim zdaniem każdy miał taki sam wkład w projekt.

Aspekt	Parametry	Wkład
Role	Wymienić	Główny programista Komunikacja z inwestorami Napisanie bota do gry Testy kodu
Kodowanie	Liczba linii kodu	239
	Funkcje (wymienić)	Utworzenie klasy Player Napisanie bota Tworzenie głównego szkieletu gry 2 testy programu Łączenie prototypu
Repozytorium	Liczba commit-ów	10
	Liczba utworzonych gałęzi	0
	Gałąź (używana – nazwa)	80-ponowne-testy-jednostkkowe 39-przygotowanie-bazy-dokumentacji-projektowej 69-testy-przystosowania-bota-do-gry-z-graczami 51-gra-z-botem 14-przygotowanie-testów-sprawdzających-funkcjonalność-programu 10-implementacja-zasad-gry 2-system-obstawiania
	Liczba połączonych gałęzi	0
	Liczba dni aktywności GIT	6
Dokumentowanie	Liczba standup-ów	17
	Opisy na Wiki	
Aktywność	Liczba zrealizowanych zadań	23

	Szacowana liczba godzin	47
	Ocena procentowego wkładu	25%

Monika Smędzik

a) Imię i nazwisko: Monika Smędzik

Adres email: mosme@student.agh.edu.pl

Nick na Github: Nika45454

b) Zrealizowane zadania:
- Zaprojektowanie systemu obstawiania i napisanie kodu odpowiedzialnego za obsługę wygranej gracza
- Napisanie kodu odpowiedzialnego za obsługę podziału postawionej przez gracza kwoty.
- Przeprowadzenie implementacji kart- napisano fragment kodu odpowiedzialny za obsługę talii kart- w tym tworzenie samej talii, obsługa tasowania, pobierania karty z talii oraz zwracania karty do talii.
- Przeprowadzenie analizy na rynku serwerów i zakupienie serwerów potrzebnych na premierę gry.
- Przeprowadzenie analizy testów przystosowania bota do gry z uczestnikami.
- Nawiązanie współpracy z influencerami w celu lepszej kampanii marketingowej.
- Rozplanowanie kampanii marketingowej i jej efektywne przeprowadzenie.
- Rozpisanie planu rozwoju i potencjalnych zysków oraz przedstawienie je inwestorom, tak aby pozyskać potrzebne środki.
- Przeprowadzenie analizy trendów w branży gier, tak aby wpasować się w obecne tendencje z formą gry i projektu, dostosować odpowiednie opcje gry do graczy.
- Utworzono materiały promocyjne.
- Przeprowadzono analizę strategii przeprowadzania kampanii marketingowych i określono kierunek promocji, który będzie najwydajniejszy i trafi do jak największej liczby potencjalnych klientów.
- Przeprowadzenie badanie demografii w tym wieku i płci wśród grup graczy gier karcianych, wyciągnięto wnioski, które później posłużyły do określenia odpowiedniej formy kampanii marketingowej i podjęcia dalszych kroków w temacie promocji.
- Przeprowadzenie analizy na rynku serwerów i zakupienie serwerów potrzebnych do beta testów.
- Przyjęto rolę zarządzania budżetem projektu- rozpisanie planowanych kosztów i wydatków, oszacowano przychody i przeznaczono odpowiednią kwotę rezerwową na wypadek niespodziewanych problemów.
- Przeprowadzono komunikację z inwestorami, przedstawiono im wstępny plan działania i prognozy finansowe.
- Skonsultowanie się z prawnikami, aby omówić kwestie prawne dotyczące promowania hazardu poprzez grę.
- Napisanie testów sprawdzających, czy poprawnie działa funkcja wywoływana na obiekcie gracz, dotycząca wyłożonej przez niego kwoty.

– Napisanie testów sprawdzających, czy karty mają prawidłowe kolory oraz figury.

c) Link-commit- system obstawiania- obsługa dwóch przypadków dla środków posiadanych przez uczestnika- pierwszy z nich zachodzi, gdy gracz wygrywa daną kwotę, drugi zachodzi, gdy gracz chce rozdzielić wyłożoną kwotę i grać jako dwóch zawodników tzw. *split*-
https://github.com/KamilT121/Blackjack/pull/8/commits/8f9235deb6d6f7efe7882cbed219f729933698d6
Link-commit-implementacja kart- klasa dla talii kart, obsługująca tworzenie talii, jej tasowanie, pobieranie losowej karty z talii oraz odłożenie pobranej karty do talii-
https://github.com/KamilT121/Blackjack/pull/9/commits/a8f71a256041c5dd9e18164db998b3024ff60654
Link- commit- testy jednostkowe- sprawdzenie czy kwota wyłożona przez gracza w grze jest prawidłowa https://github.com/KamilT121/Blackjack/blob/80-ponowne-testy-jednostkkowe/test_Card_getrank.py
Link-commit- testy jednostkowe sprawdzające czy każda z kart ma odpowiedni kolor i figurę
https://github.com/KamilT121/Blackjack/blob/80-ponowne-testy-jednostkkowe/test_Card_getrank.py

Aspekt	Parametry	Wkład
Role	Wymienić	Ogólnie pojęta działalność administracyjna Pomoc w różnych aspektach kodu Pracowanie w zespołach kilkuosobowych nad dużymi częściami kodu
Kodowanie	Liczba linii kodu	86
	Funkcje (wymienić)	Obsługiwanie sytemu obstawiania 2 testy programu Implementowanie kart
Repozytorium	Liczba commit-ów	6
	Liczba utworzonych gałęzi	0
	Gałąź (używana – nazwa)	80-ponowne-testy-jednostkkowe 6-implementacja-kart 14-przygotowanie-testów-sprawdzających-funkcjonalność-programu
	Liczba połączonych gałęzi	0
	Liczba dni aktywności GIT	6
Dokumentowanie	Liczba standup-ów	11
	Opisy na Wiki	
Aktywność	Liczba zrealizowanych zadań	18
	Szacowana liczba godzin	36
	Ocena procentowego wkładu	19%

Maksymilian Stach
Maksymilian Stach, max2311231 , mstach@student.agh.edu.pl,

Zadania zrealizowane:

- Zebranie feedbacku od beta testerów
- Analuza feedbacku beta testerów
- Analiza przewidywanej potrzebnej pojemności serwerowej
- Analiza długoterminowego zainteresowania produktami konkurencji
- Rozmowy ze wydawcą
- Zlecenie stworzenia reklamy produktu
- Kampania w mediach społecznościowych
- Opracowanie strategii na rozmowę z wydawcą
- Przeprowadzenie ankiet w śród graczy gier karcianych co do ich oczekiwań
- Konsultacja projektu interface na zasadzie sondy ulicznej
- Analiza programu w poszukiwaniu poważnych błędów
- Poprawki błędów wykrytych w fazie testów
- Analiza preferencji użytkowników dotyczących interface
- Przygotowanie testów sprawdzających funkcjonalność programu
- Poprawki do konceptu interface/Wykonanie interface

Raporty:

- Projekt wstępnego interfacu
- Przystosowanie kodu do gry okienkowej
- Napisanie 2 testów sprawdzających działanie programu

Aspekt	Parametry	Wkład
Role	Wymienić	Praca nad interface gry Komunikacja ze społecznością Analiza kodu
Kodowanie	Liczba linii kodu	108
	Funkcje (wymienić)	Przygotowanie interfaceu 2 testy programu
Repozytorium	Liczba commit-ów	4
	Liczba utworzonych gałęzi	0
	Gałąź (używana – nazwa)	80-ponowne-testy-jednostkkowe 14-przygotowanie-testów-sprawdzających-funkcjonalność-programu interface wstepny-koncept-menu
	Liczba połączonych gałęzi	0
	Liczba dni aktywności GIT	4
Dokumentowanie	Liczba standup-ów	11
	Opisy na Wiki	
Aktywność	Liczba zrealizowanych zadań	15
	Szacowana liczba godzin	27
	Ocena procentowego wkładu	17%

Kamil Siwek
Kamil Siwek, kamilsiwek@student.agh.edu.pl

Zadania zrealizowane:

- Rozmowy dotyczące kupna pojemności serwerowej
- Nawiązanie współpracy z influencerami
- Testy przystosowania bota do gracza
- Analiza programu w poszukiwaniu poważnych błędów
- Przygotowanie wizualizacji dla inwestorów
- Tworzenie materiałów promocyjnych
- Wysłanie reprezentantów na konferencje w branży gier
- Poprawki błędów wykrytych w fazie testów
- Analiza konkurencyjnych produktów
- Analiza recenzji innych gier karcianych
- Sprawdzanie funkcji w innych produktach tego typu
- Analiza prototypu gry pod względem przystępności dla użytkownika
- Analiza ryzyka projektu
- Opracowanie planu wydania gry

Raporty:

- Zrobienie wstępnej talii kart do gry
- Stworzenie klasy Card
- Napisanie 2 testów sprawdzających działanie programu
- Przygotowanie bazy dokumentacji projektowej i opisanie w niej kodu gry

Aspekt	Parametry	Wkład
Role	Wymienić	Grafika (stworzenie talii kart) Poboczne zadania
Kodowanie	Liczba linii kodu	69
	Funkcje (wymienić)	Utworzenie klasy Card 2 testy programu
Repozytorium	Liczba commit-ów	10
	Liczba utworzonych gałęzi	0
	Gałąź (używana – nazwa)	3-zrobienie-wstępnej-tali-kart-graficznie-rewers-i-awers 39-przygotowanie-bazy-dokumentacji-projektowej 80-ponowne-testy-jednostkkowe 14-przygotowanie-testów-sprawdzających-funkcjonalność-programu 6-implementacja-kart
	Liczba połączonych gałęzi	0
	Liczba dni aktywności GIT	6
Dokumentowanie	Liczba standup-ów	14
	Opisy na Wiki	
Aktywność	Liczba zrealizowanych zadań	18
	Szacowana liczba godzin	31
	Ocena procentowego wkładu	17%