



**WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA**
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Filip Walat
Nr albumu studenta w67204

System do zarządzania parkingiem

Prowadzący: mgr inż. Ewa Żesławska

Praca projektowa programowanie obiektowe C#

Rzeszów 2023

Spis treści

Wstęp	3
1 Opis założeń projektu	4
1.1 Cele projektu	4
1.2 Wymagania funkcjonalne i нефункционалне	4
1.3 Wymagania Funkcjonalne	5
1.4 Wymagania Niefunkcjonalne	5
1.5 Oczekiwania jakościowe aplikacji dedykowanej	6
2 Opis struktury projektu	7
2.1 Komponenty i Organizacja Systemu	7
2.2 Opis Techniczny Projektu	8
2.3 Repozytorium i System Kontroli Wersji	8
3 Harmonogram Realizacji Projektu	9
4 Prezentacja warstwy użytkowej projektu	11
4.1 Warstwa Użytkowa Projektu	11
4.2 Opis interakcji z użytkownikiem	14
5 Podsumowanie	15
Spis rysunków	17

Wstęp

Codzienne wyzwania związane z zarządzaniem miejscami parkingowymi w gęsto zaludnionych obszarach miejskich wymagają innowacyjnych i efektywnych rozwiązań. Nasz projekt, wykorzystujący zaawansowane technologie platformy C#, ma na celu wprowadzenie kompleksowego systemu obsługi parkingów, który zarówno usprawni zarządzanie ruchem pojazdów, jak i znacząco podniesie komfort użytkowników. Dążymy do rozwiązania problemów logistycznych przez optymalizację wykorzystania dostępnej przestrzeni, stosując zaawansowane algorytmy i interaktywne rozwiązania. Nasze podejście ma na celu nie tylko ułatwienie gospodarowania przestrzenią parkingową, ale także zmniejszenie przeciążeń w centrach miast, co przyczyni się do poprawy ogólnej jakości życia mieszkańców oraz efektywności miejskiej infrastruktury transportowej, łącząc w sobie praktyczne umiejętności programistyczne z realnymi wyzwaniami urbanistycznymi.

Rozdział 1

Opis założeń projektu

1.1 Cele projektu

Celem naszego projektu jest stworzenie zaawansowanego systemu obsługi parkingu wykorzystującego platformę C#. Dążymy do zaprojektowania i implementacji systemu, który nie tylko efektywnie zarządza ruchem pojazdów na parkingu, ale również zapewnia wygodę użytkownikom. Projekt ten ma kluczowe znaczenie dla naszych studiów informatycznych, koncentrując się na wykorzystaniu praktycznych umiejętności programistycznych w środowisku C# do rozwiązania realnych problemów. ‘’

- **Jaki jest cel projektu?** Stworzenie systemu zarządzania parkingiem, który usprawnia parkowanie i zarządzanie przestrzenią parkingową.
- **Jaki jest problem, który będzie rozwiązywany oraz proszę wskazać podstawowe źródło problemu?** Problemem jest niewystarczająca automatyzacja zarządzania parkingami, co prowadzi do trudności z lokalizacją wolnych miejsc oraz zarządzaniem ruchem pojazdów.
- **Dlaczego ten problem jest ważny oraz jakie są dowody potwierdzające jego istnienie?** Zwiększająca się liczba pojazdów w miastach wymaga bardziej efektywnych rozwiązań parkingowych, aby zmniejszyć zatory i poprawić doświadczenia użytkowników.
- **Co jest niezbędne, aby problem został rozwiązany przez Zespół i dlaczego?** Niezbędne jest zastosowanie nowoczesnych technologii i metod programowania, aby stworzyć elastyczny i skalowalny system.
- **W jaki sposób problem zostanie rozwiązany?** Poprzez zaprojektowanie i implementację systemu na platformie C#, wykorzystującego zasady programowania obiektowego do zarządzania danymi i procesami parkingowymi.

1.2 Wymagania funkcjonalne i нефункционалне

Definicja:

Wymagania funkcjonalne określają konkretną funkcjonalność lub zachowanie systemu, które musi zostać zaimplementowane. Obejmują one specyficzne zadania lub funkcje, które system powinien być w stanie wykonać, takie jak przetwarzanie danych, wykonanie obliczeń, reakcja na określone wejścia użytkownika, i inne wymagane operacje.

Wymagania нефункционалне dotyczą ogólnych jakości systemu, takich jak wydajność, bezpieczeństwo, skalowalność, niezawodność, łatwość użytkowania, i zgodność ze standardami. Te wymagania nie opisują bezpośrednio działań systemu, ale określają atrybuty, które muszą być spełnione, aby system był użyteczny i efektywny w swoim środowisku pracy.

1.3 Wymagania Funkcjonalne

Wymagania funkcjonalne naszego systemu obejmują:

- Sprawdzenie dostępności miejsc parkingowych: Użytkownik za pomocą prostego interfejsu może szybko zweryfikować, czy na parkingu znajduje się wolne miejsce.
- Rejestracja pojazdu na parkingu: Użytkownik podając dane pojazdu (numer rejestracyjny, kolor, typ), inicjuje proces rejestracji wjazdu.
- Rejestracja wyjazdu pojazdu: Przy wyjeździe z parkingu użytkownik informuje system o zwolnieniu miejsca. Dzięki temu procesowi, system na bieżąco aktualizuje dostępne miejsca parkingowe, co pozwala na optymalizację zarządzania parkingiem.
- Dostęp do ogólnych informacji o projekcie: System oferuje użytkownikom interfejs do przeglądania informacji o projekcie, w tym celach, funkcjonalnościach oraz sposobie korzystania z systemu.
- Bezpieczne zakończenie pracy z aplikacją przez użytkownika.

Metody takie jak `CheckAvailability`, `EnterParking`, i `ExitParking` są kluczowe dla funkcjonowania systemu, co zapewnia sprawne zarządzanie parkingiem i dostępnością miejsc.

1.4 Wymagania Niefunkcjonalne

Wymagania niefunkcjonalne projektu są równie istotne, zapewniając:

- Skalowalność i wydajność: System został zaprojektowany z myślą o obsłudze dużej liczby pojazdów i użytkowników, zapewniając płynną pracę nawet przy wysokim obciążeniu, co jest kluczowe dla zapewnienia ciągłości działania w obszarach miejskich o dużej intensywności ruchu.
- Szybki czas odpowiedzi i efektywność działania aplikacji. Utrzymywalność i łatwość modyfikacji: Kod źródłowy systemu jest zgodny z najlepszymi praktykami programistycznymi, co ułatwia wprowadzanie zmian, aktualizacji oraz szybką diagnozę i naprawę ewentualnych błędów.
- Możliwość przeprowadzania testów jednostkowych i integracyjnych.

Zastosowanie zorientowanej obiektowo hierarchii klas oraz wykorzystanie zaawansowanych struktur danych pozwala na skuteczne zarządzanie stanem parkingowym i obsługę różnych typów pojazdów.

1.5 Oczekiwania jakościowe aplikacji dedykowanej

Teraz nadchodzi część, w której definiujemy oczekiwania jakościowe aplikacji dedykowanej zarządzania parkingiem. Te atrybuty opisują sposoby, w jakie oczekujemy, że aplikacja będzie się zachowywała:

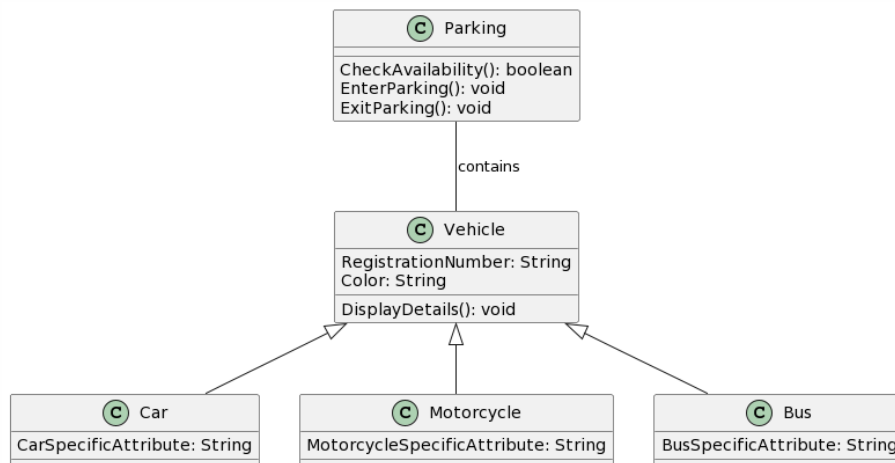
- **Użyteczność produktu:** Aplikacja powinna charakteryzować się intuicyjnym i łatwym w użyciu interfejsem, minimalizującym potrzebę szkoleń i umożliwiającym szybki dostęp do wszystkich kluczowych funkcji.
- **Prawa i regulacje:** Aplikacja musi być zgodna z ogólnym rozporządzeniem o ochronie danych (RODO) oraz lokalnymi przepisami dotyczącymi parkowania i płatności elektronicznych.
- **Dostępność aplikacji:** System powinien być dostępny 24/7/365, z zapewnieniem ciągłości działania nawet w przypadku awarii czy nieprzewidzianych sytuacji.
- **Wydajność systemu IT:** Oczekuje się, że czas odpowiedzi systemu na kluczowe operacje (np. ładowanie listy dostępnych miejsc) nie będzie przekraczał 3 sekund, a funkcje offline będą dostępne przez co najmniej 24h.

Rozdział 2

Opis struktury projektu

2.1 Komponenty i Organizacja Systemu

Projekt Systemu Zarządzania Parkingiem skupia się na zorientowanej obiektowo architekturze, co ułatwia zarządzanie różnymi typami pojazdów i interakcje z parkingiem. Centralnym elementem jest klasa `Parking`, która zarządza miejscami parkingowymi, a także wprowadzaniem i wyjazdem pojazdów. Klasy pojazdów, takie jak `Car`, `Motorcycle` i `Bus`, dziedziczą z abstrakcyjnej klasy `Vehicle`, co pozwala na polimorficzne traktowanie różnych typów pojazdów. Poniżej przedstawiono diagram klas, który ilustruje relacje między głównymi komponentami systemu.



Rysunek 2.1: Diagram klas systemu zarządzania parkingiem.

Struktura projektu jest zaprojektowana w taki sposób, aby maksymalizować ponowne wykorzystanie kodu i ułatwić rozszerzanie systemu o nowe funkcjonalności.

2.2 Opis Techniczny Projektu

Projekt został zrealizowany w języku C#, co zapewnia szerokie możliwości w zakresie programowania obiektowego i zarządzania danymi. Do zarządzania projektem i kodem źródłowym wykorzystano środowisko Visual Studio Code z dodatkowymi wtyczkami, takimi jak PlantUML dla generowania diagramów klas UML, oraz Git jako system kontroli wersji.

System jest zaprojektowany z myślą o niskich wymaganiach sprzętowych, co czyni go dostępnym na większości współczesnych komputerów i serwerów. Minimalne wymagania to:

- Procesor: 1 GHz lub szybszy.
- Pamięć RAM: 512 MB dla klienta, 2 GB dla serwera.
- Przestrzeń na dysku: 100 MB.
- System operacyjny: Windows 7 lub nowszy, Linux, MacOS.

Projekt wykorzystuje mechanizm zarządzania danymi oparty na bazie danych w pliku tekstowym (txt), co pozwala na prostą i efektywną manipulację danymi bez potrzeby korzystania z zewnętrznych systemów DBMS. Taki wybór umożliwia łatwą portowalność i minimalizuje wymagania sprzętowe oraz konfiguracyjne. Struktura plików tekstowych jest zaprojektowana w taki sposób, aby umożliwić szybkie odczytywanie i zapisywanie stanu miejsc parkingowych oraz informacji o pojazdach, co zapewnia wysoką wydajność działania systemu.

2.3 Repozytorium i System Kontroli Wersji

Projekt wykorzystuje system kontroli wersji Git, co umożliwia skuteczne zarządzanie historią zmian kodu źródłowego. Repozytorium kodu znajduje się na platformie GitHub pod adresem:

<https://github.com/filwalu/ProjektOOP> i będzie dostępne publicznie do dnia 30.09.2024. Bezpieczne

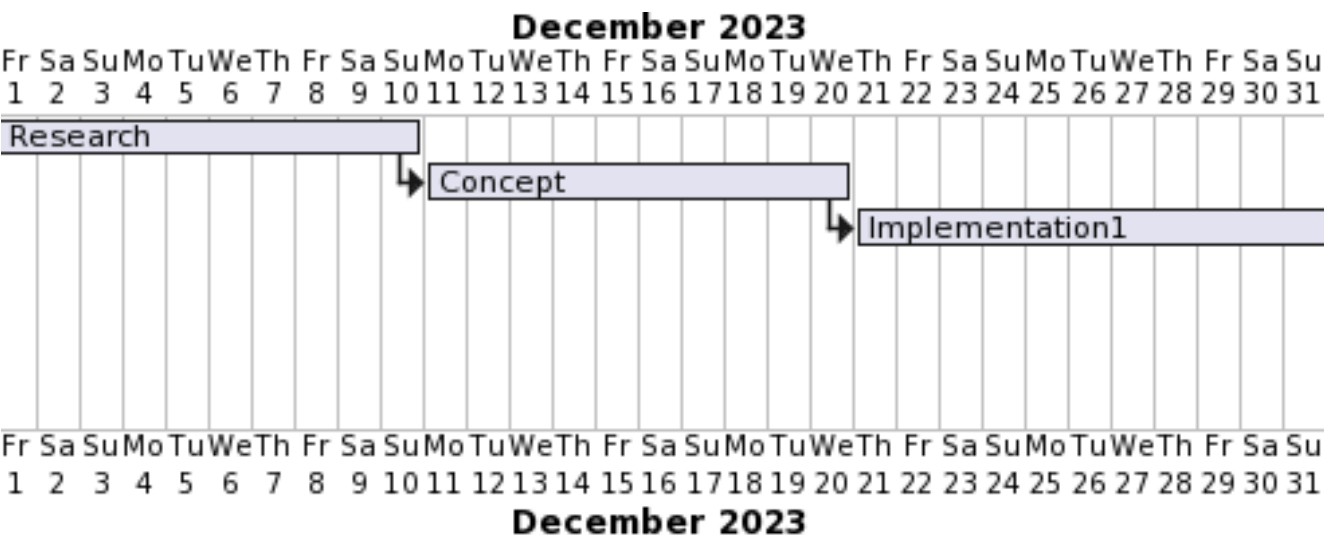
połączenie z repozytorium zabezpieczono za pomocą pary kluczy SSH. Poniżej przedstawiono opis użytych poleceń Git:

1. `git init` - inicjalizacja nowego repozytorium Git.
2. `git clone [URL]` - klonowanie repozytorium przy użyciu SSH.
3. `git add -Av` - dodawanie zmian do kolejki commitów.
4. `git status` - sprawdzanie statusu zmian.
5. `git commit -m "[wiadomość]"` - commitowanie zmian z opisem.
6. `git push` - wysyłanie zmian do zdalnego repozytorium przez SSH.

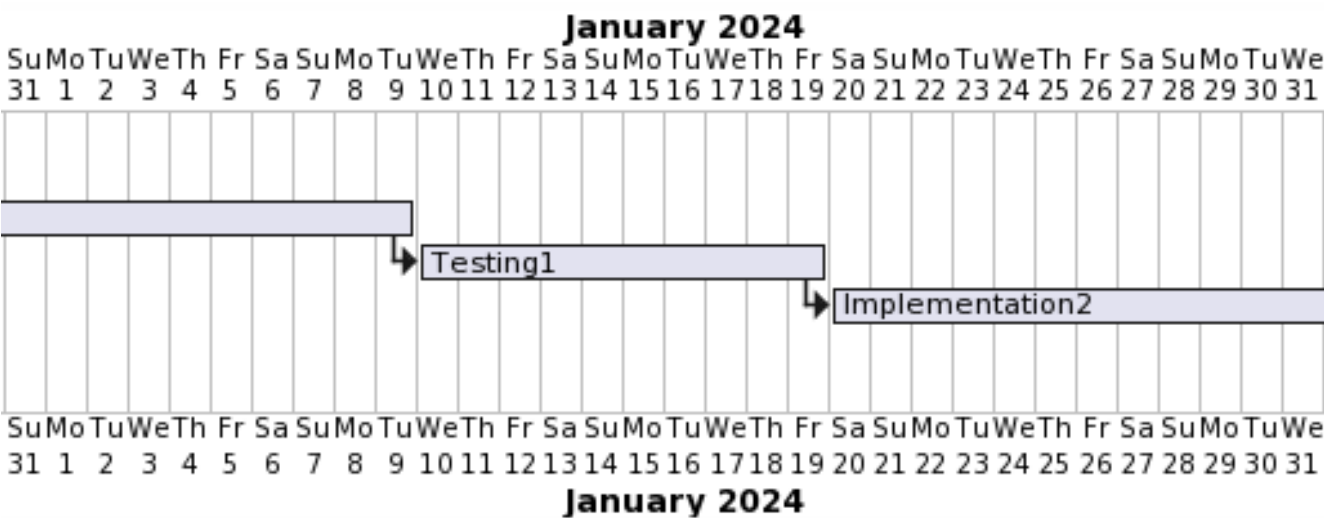
Rozdział 3

Harmonogram Realizacji Projektu

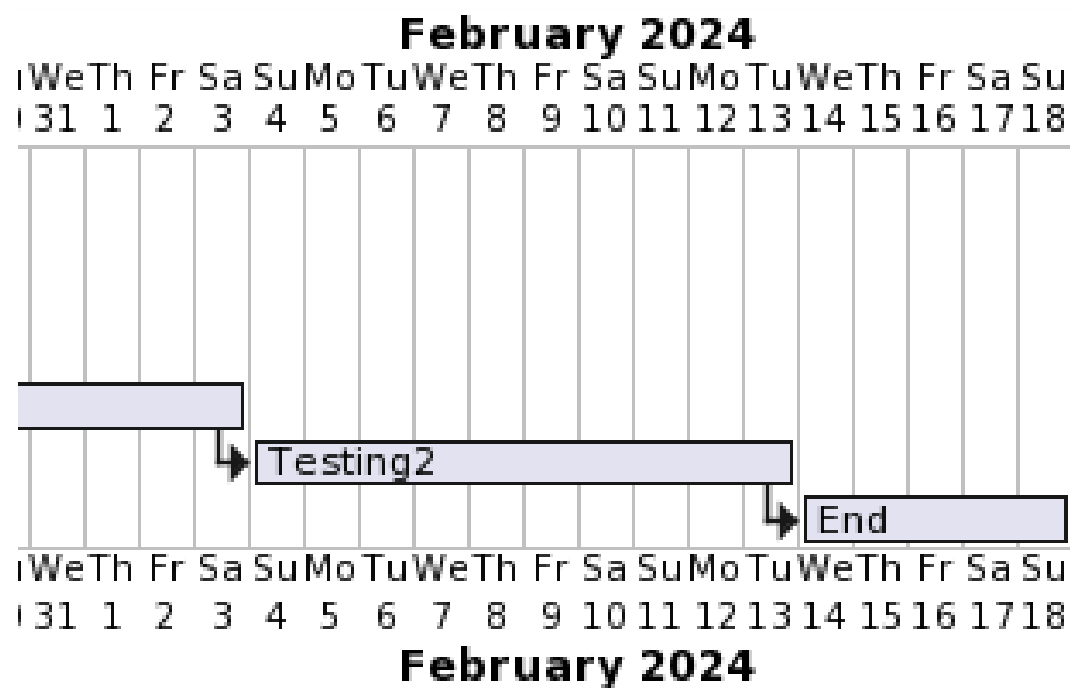
Harmonogram realizacji projektu został zaplanowany z wykorzystaniem diagramu Gantta, który ilustruje kluczowe etapy rozwoju projektu, ich zależności czasowe oraz alokację zasobów. Poniżej przedstawiono diagram Gantta dla projektu System Zarządzania Parkingiem.



Rysunek 3.1: Diagram Gantta projektu System Zarządzania Parkingiem.



Rysunek 3.2: Diagram Gantta projektu System Zarządzania Parkingiem.



Rysunek 3.3: Diagram Gantta projektu System Zarządzania Parkingiem.

Rozdział 4

Prezentacja warstwy użytkowej projektu

4.1 Warstwa Użytkowa Projektu

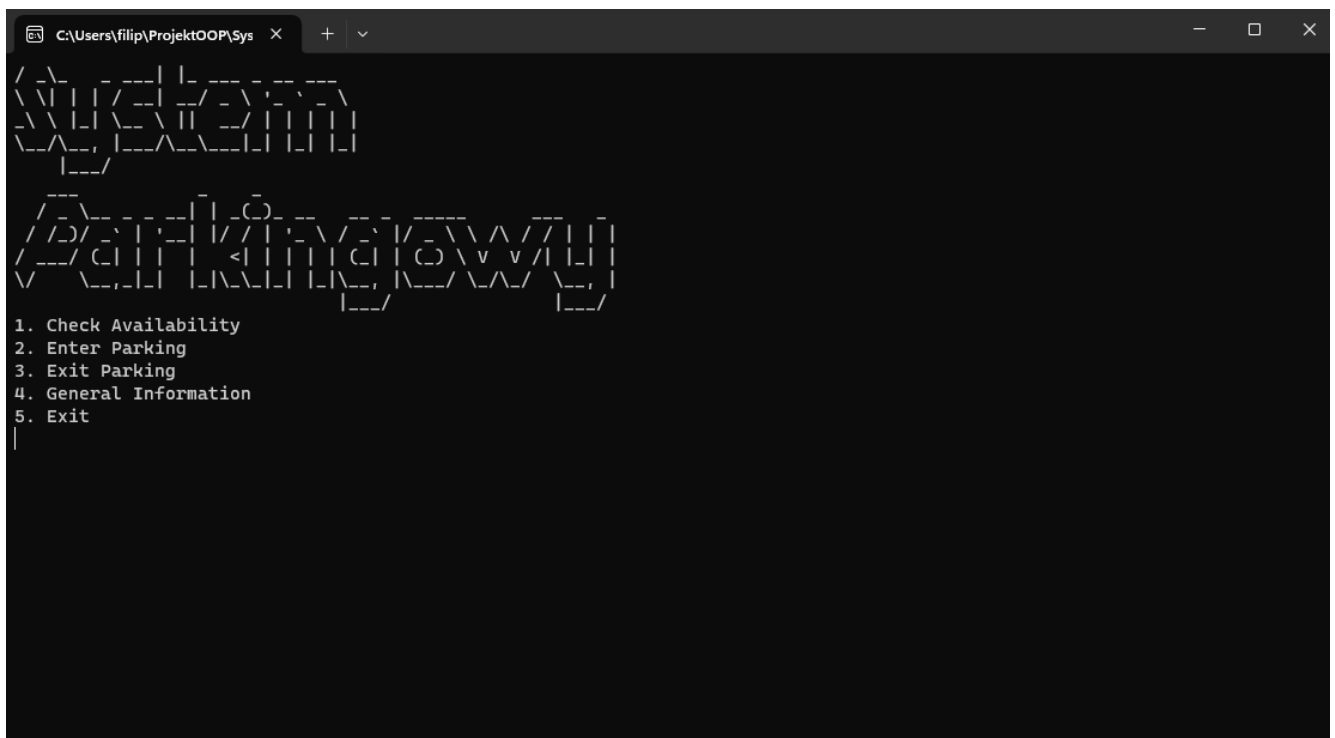
Projekt Systemu Zarządzania Parkingiem oferuje intuicyjny i prosty w obsłudze interfejs użytkownika, który umożliwia szybkie zarządzanie miejscami parkingowymi oraz monitorowanie dostępności przestrzeni parkingowej. Interfejs użytkownika został zaprojektowany z myślą o zapewnieniu maksymalnej użyteczności i dostępności funkcji.

Aplikacja umożliwia użytkownikom wykonanie następujących akcji:

- Sprawdzenie dostępności miejsc parkingowych.
- Rejestracja wjazdu i wyjazdu pojazdów.
- Zarządzanie danymi pojazdów.

Interfejs skupia się na minimalizmie i łatwości nawigacji, co pozwala na szybkie odnalezienie potrzebnych informacji i funkcji.

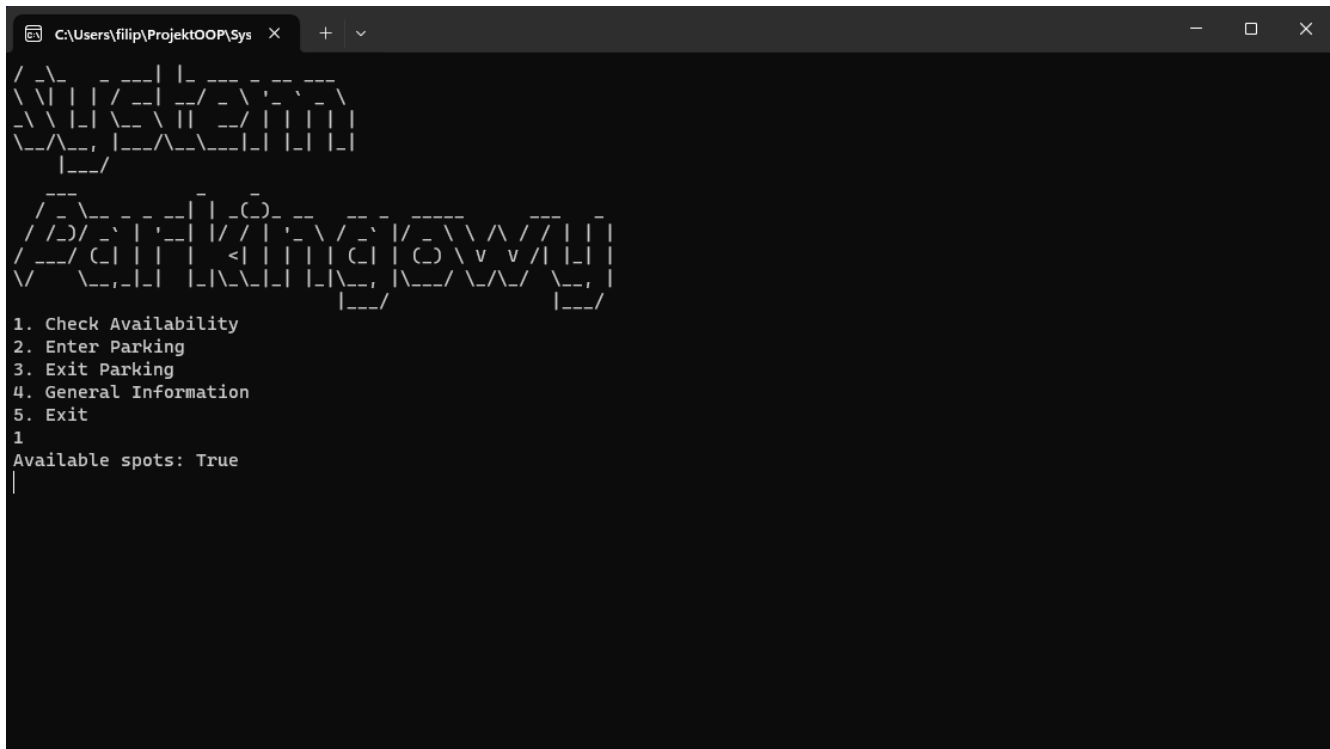
W tej sekcji zostaną umieszczone zrzuty ekranu przedstawiające kluczowe funkcjonalności aplikacji oraz jej interfejs użytkownika.



Rysunek 4.1: Widok głównego menu aplikacji.

W tej sekcji omówione zostaną kluczowe funkcjonalności aplikacji Systemu Zarządzania Parkin-
giem oraz instrukcje dotyczące ich używania.

- **Sprawdzanie dostępności miejsc parkingowych** (*Check Availability*): Użytkownik może spraw-
dzić dostępność miejsc parkingowych dla samochodów, wybierając opcję "1".



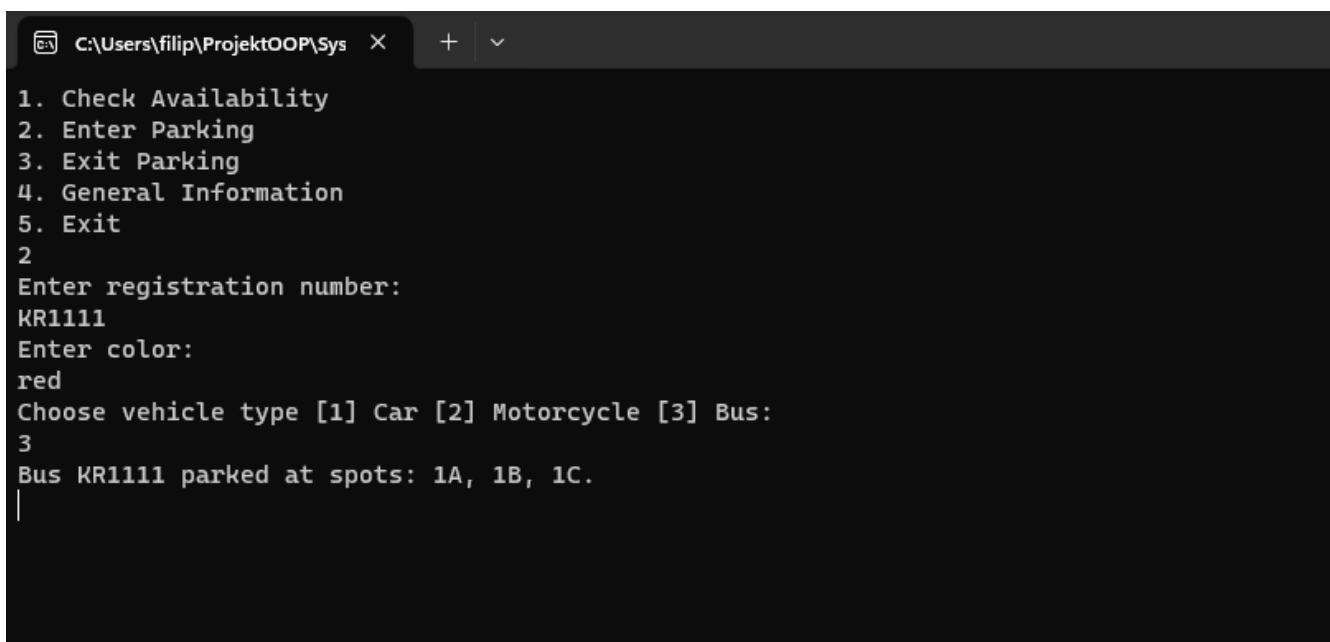
```
C:\Users\filip\ProjektOOP\Sys > 1
/ \_ _ _ _ _ | | _ _ _ _ _
\ \ | | / _ _ | / \_ _ _ _ _
_ \ \ | | _ _ \ \ | | _ _ | | | |
\ \ / _ _ | _ _ \ \ _ _ _ _ | | | |
  | _ _ /
  | _ _ /

/ _ _ _ _ _ | | _ _ _ _ _
/ / \ / \ | | _ _ _ _ _ | / \_ _ _ _ _
_ _ _ / \ | | | | < | | | | \ \ \ \ \ / \ | | | |
\ _ _ \ _ _ | | _ _ \ _ _ \ \ \ \ \ / \ _ _ |
  | _ _ /
  | _ _ /

1. Check Availability
2. Enter Parking
3. Exit Parking
4. General Information
5. Exit
1
Available spots: True
|
```

Rysunek 4.2: Opcja 1 - Check Availability.

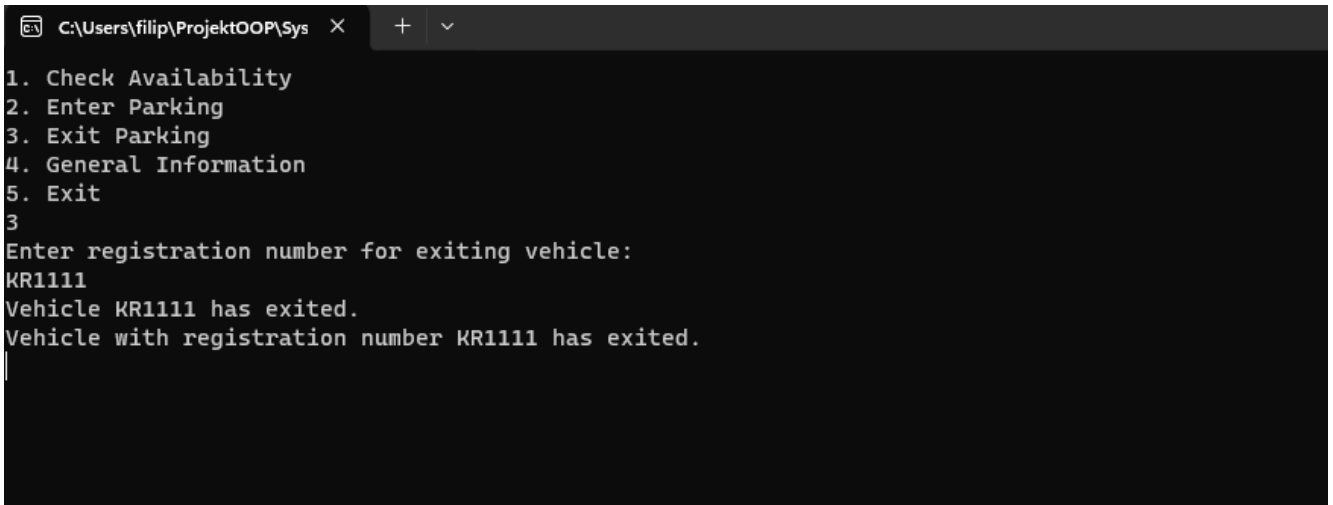
- **Wprowadzanie pojazdów na parking**: Użytkownik może wprowadzić pojazd na parking, wybie-
rając opcję "2". Następnie należy podać numer rejestracyjny pojazdu, jego kolor oraz wybrać typ
pojazdu (1 - Samochód, 2 - Motocykl, 3 - Autobus). Na podstawie podanych informacji, system
tworzy odpowiedni obiekt pojazdu i rejestruje go w systemie parkingowym.



```
C:\Users\filip\ProjektOOP\Sys > 2
1. Check Availability
2. Enter Parking
3. Exit Parking
4. General Information
5. Exit
2
Enter registration number:
KR1111
Enter color:
red
Choose vehicle type [1] Car [2] Motorcycle [3] Bus:
3
Bus KR1111 parked at spots: 1A, 1B, 1C.
|
```

Rysunek 4.3: Opcja 2 - Enter Parking.

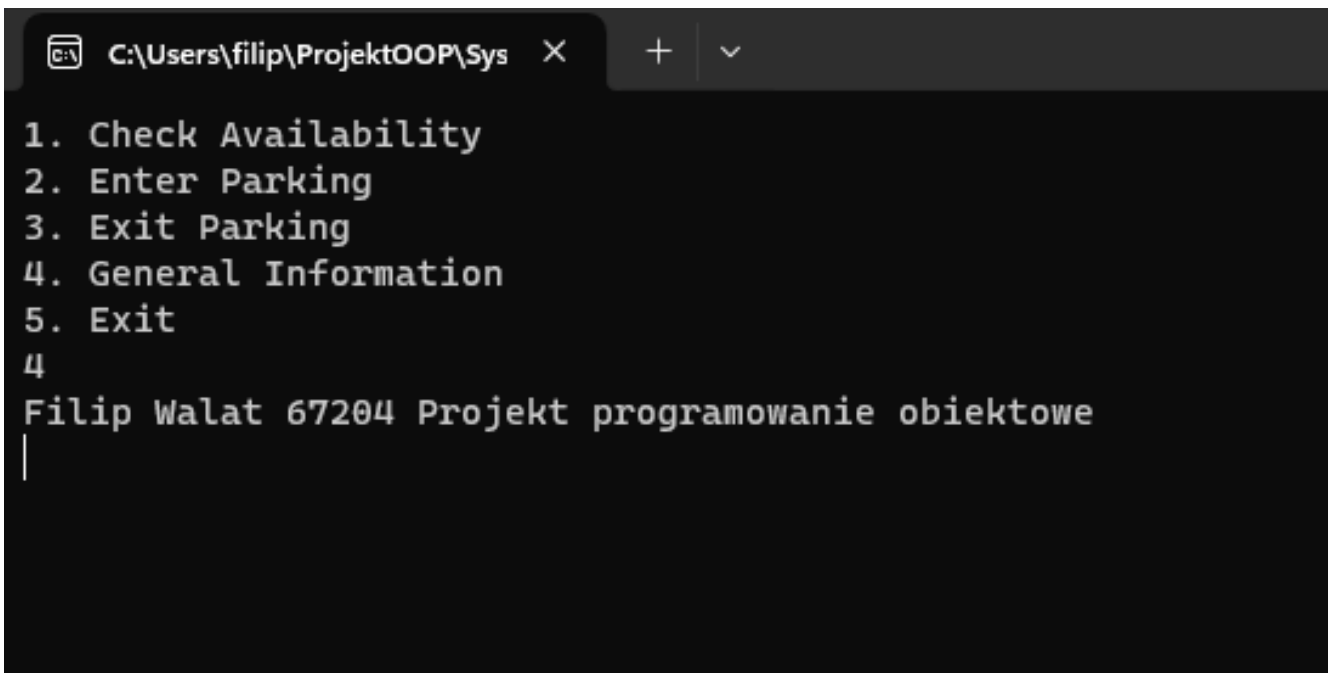
- **Wyjazd pojazdów z parkingu:** Użytkownik może wyjechać pojazdem z parkingu, wybierając opcję "3". Następnie należy podać numer rejestracyjny pojazdu. Na podstawie podanej informacji, system usuwa pojazd z rejestru parkingowego.



```
C:\Users\filip\ProjektOOP\Sys X + v
1. Check Availability
2. Enter Parking
3. Exit Parking
4. General Information
5. Exit
3
Enter registration number for exiting vehicle:
KR1111
Vehicle KR1111 has exited.
Vehicle with registration number KR1111 has exited.
```

Rysunek 4.4: Opcja 3 - Exit Parking.

- **Dodatkowe informacje:** Użytkownik wybierając opcję "4" wyświetla dodatkowe informacje o systemie.



```
C:\Users\filip\ProjektOOP\Sys X + v
1. Check Availability
2. Enter Parking
3. Exit Parking
4. General Information
5. Exit
4
Filip Walat 67204 Projekt programowanie obiektowe
```

Rysunek 4.5: Opcja 4 - General Information.

- **Koniec:** Użytkownik wybierając opcję "5" zamyka aplikację.

4.2 Opis interakcji z użytkownikiem

Sekcja ta zawiera informacje na temat interakcji z użytkownikiem, w tym otrzymywanych komunikatów, alertów oraz sposobu obsługi błędów. Poniżej przedstawiono wybrane komunikaty wyświetlane przez aplikację:

- `"Available spots: [True lub False]"` - informacja mówiąca czy znajdują się obecnie wolne miejsca parkingowe.
- `"Enter registration number:"` - monit o wprowadzenie numeru rejestracyjnego pojazdu.
- `"Enter color:"` - prośba o podanie koloru pojazdu.
- `"Choose vehicle type [1] Car [2] Motorcycle [3] Bus:"` - wybór typu pojazdu do wprowadzenia na parking.
- `"Invalid vehicle type selected."` - komunikat o błędnym wyborze typu pojazdu.
- `"Vehicle entered the parking."` - informacja o pomyślnym wprowadzeniu pojazdu na parking.

Rozdział 5

Podsumowanie

W ramach projektu Systemu Zarządzania Parkingiem wykonano szereg prac, w tym:

- Projektowanie i implementacja klas reprezentujących pojazdy (samochody, motocykle, autobusy) oraz parking.
- Rozwój funkcjonalności zarządzania miejscami parkingowymi, w tym sprawdzanie dostępności, rejestracja wjazdu i wyjazdu pojazdów.
- Przygotowanie interfejsu użytkownika w konsoli do interakcji z systemem.
- Wykorzystanie plików tekstowych do prostego zarządzania danymi.

W dalszej kolejności planowane są następujące prace rozwojowe:

- Zastosowanie klasycznej bazy danych SQL, np. PostgreSQL, dla lepszego zarządzania danymi i wydajności.
- Konteneryzacja aplikacji z wykorzystaniem Docker, co ułatwi wdrożenie i skalowalność.
- Implementacja systemu płatności.
- Usprawnienia interfejsu użytkownika, w tym rozwój graficznego interfejsu użytkownika (GUI) dla lepszej dostępności i ergonomii.
- Rozbudowa obsługi wyjątków dla zwiększenia stabilności i niezawodności aplikacji.
- Implementacja możliwości generowania raportów w formacie CSV z danych przechowywanych w systemie dla ułatwienia analizy i zarządzania.

Bibliografia

- [1] Scott Chacon, Ben Straub. *Pro Git*. Apress, 2014. Dostępne online: <https://git-scm.com/book/en/v2>
- [2] Dokumentacja Microsoft .NET. Microsoft, dostęp online: <https://docs.microsoft.com/en-us/dotnet/>
- [3] Albahari, Joseph, Albahari, Ben. *C# 7.0 in a Nutshell: The Definitive Reference*. O'Reilly Media, 2017.

Spis rysunków

2.1	Diagram klas systemu zarządzania parkingiem.	7
3.1	Diagram Gantta projektu System Zarządzania Parkingiem.	9
3.2	Diagram Gantta projektu System Zarządzania Parkingiem.	9
3.3	Diagram Gantta projektu System Zarządzania Parkingiem.	10
4.1	Widok głównego menu aplikacji.	11
4.2	Opcja 1 - Check Availability.	12
4.3	Opcja 2 - Enter Parking.	12
4.4	Opcja 3 - Exit Parking.	13
4.5	Opcja 4 - General Information.	13