Niezawodność i Diagnostyka Układów Cyfrowych - Projekt

Temat projektu: Badanie i analiza niezawodności systemu kas sklepowych

Nazwisko i Imię prowadzącego kurs: dr inż. Dominik Żelazny

Imię i Nazwisko nr indeksu, wydział	Filip Tymiński 273023 Jakub Skorupa 272546
Termin zajęć: dzień tygodnia, godzina	wtorek 11:15 – 12:45
Data oddania sprawozdania:	11.06.2024
Ocena końcowa	

Spis treści

Opis Projektu ... 2
Cele projektu ... 2
Badanie niezawodności ... 2
Optymalizacja efektywności ... 2
Identyfikacja obszarów ulepszeń ... 2
Harmonogram projektu ... 2
Oczekiwane rezultaty ... 3
Metodologia ... 3
Przygotowanie środowiska symulacyjnego ... 3
Definiowanie scenariuszy symulacji ... 3
Implementacja logiki symulacji ... 4
Przeprowadzenie symulacji ... 4
Struktura kodu ... 4
Klasa Kasa ... 4
Klasa Klient ... 5
Klasa MyGui ... 6
Napotkane problemy i trudności ... 6
Dobór odpowiednich zmiennych ... 6
Poprawne działanie symulacji ... 7
Testowanie i debugowanie ... 7
Wnioski i poprawki ... 8
Wyniki symulacji ... 8
Przeprowadzone scenariusze symulacji ... 8
Wizualizacja wyników ... 8
Część statystyczna i demograficzna ... 8
Część wydajnościowa ... 12
Analiza zebranych danych ... 14

Wnioski 14
Efektywność i Realizm Symulacji 14
Złożoność Zarządzania Kolejkami i Obsługą Klientów 14
Znaczenie Testowania i Debugowania 15
Wykorzystanie Interfejsu Graficznego 15
Ulepszenia i Przyszły Rozwój 15
Podsumowanie 15

Opis Projektu

Celem niniejszego projektu jest przeprowadzenie symulacji działania systemu kas sklepowych w celu badania ich niezawodności oraz efektywności w różnych warunkach. Symulacja odwzorowuje rzeczywiste funkcjonowanie sklepu, uwzględniając takie czynniki jak liczba klientów, ich zachowanie, obsługa transakcji, awarie sprzętowe oraz inne zdarzenia losowe.

Cele projektu

Badanie niezawodności

Projekt ma na celu ocenę, jak system kas sklepowych zachowuje się pod względem niezawodności w różnych warunkach, takich jak obciążenie klientami, intensywność ruchu w sklepie i częstotliwość awarii.

Optymalizacja efektywności

Analiza wydajności systemu kas będzie przeprowadzona pod kątem czasu obsługi klientów, czasu reakcji na awarie oraz innych czynników wpływających na operacyjną efektywność sklepu.

Identyfikacja obszarów ulepszeń

Projekt ma również na celu wskazanie potencjalnych obszarów, które mogą być ulepszane w celu zwiększenia niezawodności i efektywności systemu kas sklepowych.

Harmonogram projektu

1. **Przygotowanie środowiska symulacyjnego**: Zainicjowanie symulacji, implementacja modelu kas i klientów oraz przygotowanie interfejsu użytkownika do obserwacji symulacji.
2. **Definiowanie scenariuszy symulacji**: Określenie różnych warunków symulacyjnych, takich jak różne poziomy obciążenia klientami, intensywność ruchu w sklepie oraz prawdopodobieństwo wystąpienia awarii.

3. **Implementacja logiki symulacji**: Programowanie logiki symulacji, obejmującej obsługę klientów, zarządzanie kasami, reakcję na awarie oraz inne zdarzenia losowe.
4. **Przeprowadzenie symulacji**: Uruchomienie symulacji dla określonych scenariuszy i zebranie danych dotyczących działania systemu kas w różnych warunkach.
5. **Analiza wyników**: Analiza zebranych danych w celu zrozumienia zachowania systemu kas sklepowych, identyfikacji problemów oraz obszarów ulepszeń.
6. **Przygotowanie raportu**: Opracowanie raportu z wynikami symulacji, w którym zawarte będą wnioski, rekomendacje oraz propozycje dalszych działań w celu poprawy niezawodności i efektywności systemu kas.
7. **Prezentacja wyników**: Prezentacja raportu oraz wyników symulacji przed zespołem projektowym i innymi zainteresowanymi osobami.

Oczekiwane rezultaty

Projekt ma na celu zrozumienie działania systemu kas sklepowych w różnych warunkach symulacyjnych. Przeprowadzenie tej analizy pozwoli na identyfikację czynników wpływających na niezawodność i efektywność systemu kas oraz wskazanie obszarów, które mogą być ulepszane w celu zwiększenia ich niezawodności i efektywności. Wyniki symulacji oraz rekomendacje dotyczące dalszych działań przyczynią się do poprawy jakości obsługi klientów oraz efektywności działania sklepów.

Metodologia

Metodologia projektu obejmuje kilka kluczowych etapów, które są niezbędne do przygotowania, przeprowadzenia i analizy symulacji systemu kas sklepowych. Poniżej opisane są szczegółowo poszczególne etapy metodologii:

Przygotowanie środowiska symulacyjnego

Pierwszym krokiem w realizacji projektu było przygotowanie odpowiedniego środowiska symulacyjnego. Obejmowało to:

1. **Zainicjowanie symulacji**: Tworzenie podstawowych struktur danych i mechanizmów, które umożliwią symulację działania kas i zachowań klientów.
2. **Implementacja modelu kas i klientów**: Zaimplementowano klasy Kasa i Klient, które reprezentują kasy sklepowe oraz klientów sklepu. Klasy te zawierają wszystkie niezbędne właściwości i metody, które pozwalają na symulację interakcji między klientami a kasami.
3. **Przygotowanie interfejsu użytkownika**: Stworzono prosty interfejs (graficzny lub tekstowy), który pozwala na obserwację przebiegu symulacji oraz monitorowanie stanu systemu kas w czasie rzeczywistym.

Definiowanie scenariuszy symulacji

Aby zapewnić kompleksową analizę systemu kas sklepowych, zdefiniowano różne scenariusze symulacyjne. Każdy scenariusz uwzględniał różne warunki operacyjne, w tym:

1. **Poziomy obciążenia klientami**: Określono różne liczby klientów odwiedzających sklep w danym czasie, aby zbadać, jak system radzi sobie z różnym natężeniem ruchu.
2. **Intensywność ruchu w sklepie**: Zmieniano częstotliwość pojawiania się nowych klientów oraz ich zachowania, takie jak czas spędzany przy kasie i ilość zakupów.

3. **Prawdopodobieństwo wystąpienia awarii**: Wprowadzono losowe zdarzenia, takie jak awarie kas, aby ocenić, jak system radzi sobie z nieprzewidzianymi problemami.

Implementacja logiki symulacji

Logika symulacji została zaimplementowana tak, aby jak najwierniej odwzorować rzeczywiste działanie systemu kas sklepowych. Kluczowe elementy logiki obejmowały:

1. **Obsługa klientów**: Zaimplementowano mechanizmy przydzielania klientów do kas, obsługi transakcji oraz zarządzania kolejkami klientów.
    - Metoda addClient w klasie Kasa dodaje klienta do kolejki, jeśli jest dostępne miejsce.
    - Metoda serveClient obsługuje klienta przy kasie, zmniejszając jego czas obsługi i aktualizując stan kasy.
2. **Zarządzanie kasami**: Uwzględniono różne stany kas, takie jak aktywność, awarie oraz czas przestoju.
    - Metody setActive, setIncident, setBroken w klasie Kasa zarządzają stanem kasy.
    - Metoda brokenStart symuluje rozpoczęcie awarii, a brokenStop zakończenie awarii.
3. **Reakcja na awarie**: System jest zaprojektowany tak, aby reagować na awarie kas, przenosząc klientów do innych kas lub zarządzając naprawami.
    - Metoda close w klasie Kasa zamyka kasę, przenosząc klientów do globalnej kolejki oczekujących.

Przeprowadzenie symulacji

Po przygotowaniu środowiska symulacyjnego i zdefiniowaniu scenariuszy, przeprowadzono symulacje. Każdy scenariusz został uruchomiony wielokrotnie, aby uzyskać reprezentatywne wyniki. Proces ten obejmował:

1. **Uruchomienie symulacji**: Start symulacji dla różnych scenariuszy, obserwacja działania systemu oraz zbieranie danych o jego wydajności.
2. **Zbieranie danych**: Rejestrowanie danych dotyczących czasu obsługi klientów, liczby obsłużonych klientów, częstotliwości awarii oraz czasu reakcji na awarie.

Struktura kodu

Kod symulacji systemu kas sklepowych składa się z dwóch głównych części: logiki symulacji oraz interfejsu graficznego użytkownika (GUI). Logika symulacji jest zaimplementowana w klasach `Kasa` i `Klient`, podczas gdy GUI jest zaimplementowane w klasie `MyGui`, która umożliwia wizualizację stanu kas oraz klientów w czasie rzeczywistym.

Klasa Kasa

Klasa `Kasa` reprezentuje pojedynczą kasę w sklepie i zawiera metody oraz właściwości pozwalające na symulowanie jej działania.

Zmienne:

- id: Unikalny identyfikator kasy.
- maxCapacity: Maksymalna liczba klientów, którą kasa może obsłużyć jednocześnie.
- service: Czas obsługi kasy (wykorzystywany podczas awarii).

- incident: Flaga wskazująca, czy kasa ma aktualnie awarię.
- downtime: Łączny czas przestoju kasy z powodu awarii.
- active: Flaga wskazująca, czy kasa jest aktywna.
- broken: Flaga wskazująca, czy kasa jest zepsuta.
- totalTransaction: Całkowita liczba transakcji obsłużonych przez kasę.
- queue: Kolejka klientów oczekujących na obsługę.
- cash: Ilość gotówki w kasie.
- current: Aktualnie obsługiwany klient.
- idleTimer: Czas, przez który kasa pozostaje bezczynna.
- serving: Flaga wskazująca, czy kasa obsługuje klienta.

Metody:

- addClient(klient: Klient): Dodaje klienta do kolejki, jeśli jest dostępne miejsce.
- serveClient(): Obsługuje klienta przy kasie, zmniejszając jego czas obsługi i aktualizując stan kasy.
- brokenStart(serviceTime: int): Symuluje rozpoczęcie awarii kasy.
- brokenStop(): Zakończa awarię kasy.
- open(): Aktywuje kasę.
- close(kolejka): Zamyka kasę, przenosząc klientów do globalnej kolejki oczekujących.
- setActive(new: bool): Ustawia stan aktywności kasy.
- setIncident(new: bool): Ustawia stan incydentu (awarii) kasy.
- setBroken(new: bool): Ustawia stan zepsucia kasy.
- getQueue(): Zwraca kolejkę klientów.
- getQueuesize(): Zwraca rozmiar kolejki.
- addCash(cash: int): Dodaje gotówkę do kasy.
- resetQueue(): Resetuje kolejkę klientów.

Klasa Klient

Klasa Klient reprezentuje pojedynczego klienta w sklepie i zawiera właściwości oraz metody pozwalające na symulowanie jego zachowania.

Zmienne:

- id: Unikalny identyfikator klienta.
- totalTime: Całkowity czas obsługi klienta.
- family: Liczba członków rodziny klienta.
- local: Zmienna określająca lokalność klienta.
- wroclaw: Zmienna określająca czy klient pochodzi z Wrocławia.
- total: Całkowita kwota zakupów klienta.
- age: Wiek klienta.
- card: Flaga wskazująca, czy klient posiada kartę rabatową.
- exists: Flaga wskazująca, czy klient istnieje (jest obecny).

Metody:

- getTotaltime(): Zwraca całkowity czas obsługi klienta.
- getId(): Zwraca unikalny identyfikator klienta.
- getFamily(): Zwraca liczbę członków rodziny klienta.

- getCash(): Zwraca całkowitą kwotę zakupów klienta.

Klasa MyGui

Klasa MyGui odpowiada za reprezentację graficznego interfejsu użytkownika (GUI) symulacji. Umożliwia ona wizualizację stanu kas oraz klientów w czasie rzeczywistym.

Zmienne:

- root: Główne okno aplikacji Tkinter.
- squares_big: Lista obiektów Canvas reprezentujących kasy.
- squares_small: Lista list obiektów Canvas reprezentujących klientów.
- status_legend: Etykieta legendy statusów kas.
- status_active: Etykieta statusu aktywnej kasy.
- status_closed: Etykieta statusu zamkniętej kasy.
- status_problem: Etykieta statusu kasy z problemem.
- status_repair: Etykieta statusu kasy w naprawie.
- time_label: Etykieta wyświetlająca aktualną godzinę.
- day_label: Etykieta wyświetlająca aktualny dzień tygodnia.

Metody:

- __init__(self, size: int, length: int): Inicjalizuje główne okno GUI oraz tworzy elementy interfejsu.
  - Tworzy kwadraty reprezentujące kasy (squares_big) oraz klientów (squares_small).
  - Tworzy etykiety legendy statusów kas oraz aktualnej godziny i dnia tygodnia.
- klient_change_color(self, row, column, color): Zmienia kolor kwadratu reprezentującego klienta na podany kolor.
- klienci_change_color(self, column, numberOfClients, length: int): Zmienia kolor kwadratów reprezentujących klientów w danej kolumnie, w zależności od liczby klientów.
- kasa_change_color(self, col, color): Zmienia kolor kwadratu reprezentującego kasę na podany kolor.
- display_numbers(self, num1, num2): Aktualizuje etykietę wyświetlającą aktualną godzinę.
- display_days(self, num1): Aktualizuje etykietę wyświetlającą aktualny dzień tygodnia w zależności od numeru dnia (0 - Poniedziałek, 6 - Niedziela).
- run(self): Uruchamia główną pętlę aplikacji Tkinter.

Kod symulacji systemu kas sklepowych jest dobrze zorganizowany, z wyraźnym podziałem na logikę symulacji i interfejs graficzny użytkownika. Klasa `Kasa` i `Klient` umożliwiają realistyczne symulowanie działania kas sklepowych oraz zachowań klientów, podczas gdy klasa `MyGui` zapewnia wizualizację tych symulacji w sposób intuicyjny i łatwy do zrozumienia.

Napotkane problemy i trudności

Dobór odpowiednich zmiennych

Jednym z pierwszych wyzwań napotkanych podczas pracy nad projektem było odpowiednie dobranie i zdefiniowanie zmiennych, które miały odzwierciedlać rzeczywiste cechy kas i klientów w sklepie. W szczególności trudność sprawiło nam:

- **Określenie właściwych zakresów dla zmiennych:** Musieliśmy dokładnie przemyśleć i przetestować, jakie wartości powinny przyjmować zmienne takie jak `maxCapacity` dla kas czy `total` i `totalTime` dla klientów. Na przykład, ustalenie, ile czasu zajmuje obsługa klienta w zależności od liczby członków rodziny i posiadania karty rabatowej wymagało kilku iteracji testów.
- **Wprowadzenie losowości:** Zmienność takich parametrów jak liczba członków rodziny klienta (`family`) czy jego wiek (`age`) została wprowadzona przy użyciu funkcji losowych. Wprowadzenie losowości było niezbędne do realistycznej symulacji, ale jednocześnie wymagało starannego doboru zakresów, aby symulacja była realistyczna, a jednocześnie stabilna.

Poprawne działanie symulacji

Kolejnym istotnym wyzwaniem było zapewnienie poprawnego działania całej symulacji, co obejmowało zarówno logikę obsługi klientów przez kasy, jak i synchronizację z graficznym interfejsem użytkownika. Poniżej przedstawiono główne trudności, z jakimi się spotkaliśmy:

- **Zarządzanie kolejką klientów:** Jednym z trudniejszych aspektów było zarządzanie kolejkami klientów przy kasach. Konieczne było zapewnienie, aby klienci byli poprawnie dodawani do kolejki, obsługiwani przez kasę i usuwani z kolejki po zakończeniu obsługi. Początkowo napotkaliśmy problemy z prawidłowym przetwarzaniem klientów w kolejkach, co wymagało przemyślenia algorytmów obsługi.
- **Obsługa awarii kas:** Implementacja obsługi awarii kas i ich wpływu na symulację była również wyzwaniem. Konieczne było wprowadzenie mechanizmów do symulowania awarii (`brokenStart`), ich zakończenia (`brokenStop`) oraz przestoju (`setDowntime`, `resetDowntime`). Złożoność tego procesu wymagała dokładnego przemyślenia i testów, aby uniknąć sytuacji, w której kasa pozostaje w stanie awarii na zawsze lub jest naprawiana bez końca.
- **Synchronizacja z GUI:** Synchronizacja stanu symulacji z interfejsem graficznym użytkownika była kolejnym wyzwaniem. Konieczne było zapewnienie, aby zmiany stanu kas i klientów były natychmiast odzwierciedlane w GUI. Początkowo mieliśmy problemy z opóźnieniami w aktualizacji interfejsu oraz z błędnym wyświetlaniem stanu kas i klientów. Problemy te zostały rozwiązane przez dokładne przemyślenie momentów aktualizacji GUI i użycie odpowiednich metod `update`.

Testowanie i debugowanie

Testowanie i debugowanie całej symulacji zajęło nam sporo czasu, głównie ze względu na złożoność interakcji między różnymi elementami systemu. Poniżej przedstawiono główne wyzwania związane z testowaniem:

- **Symulowanie różnych scenariuszy:** Konieczne było przetestowanie wielu różnych scenariuszy działania sklepu, w tym sytuacji z dużą liczbą klientów, częstymi awariami kas oraz różnymi godzinami i dniami tygodnia. Każdy z tych scenariuszy wymagał innego podejścia do testowania i debugowania.
- **Śledzenie błędów logicznych:** W trakcie testów często napotykaliśmy błędy logiczne, takie jak nieprawidłowe dodawanie klientów do kolejki, błędne obliczenia czasu obsługi

czy niepoprawne zmiany stanu kas. Identyfikacja i naprawa tych błędów wymagała dokładnej analizy kodu oraz wykorzystania narzędzi do debugowania.

Wnioski i poprawki

Na podstawie napotkanych problemów wprowadziliśmy szereg poprawek i usprawnień, które pozwoliły na osiągnięcie stabilnej i realistycznej symulacji. Kluczowe zmiany obejmowały:

- **Ulepszenie logiki obsługi klientów i kas:** Poprawiliśmy algorytmy zarządzania kolejkami oraz obsługi awarii, co pozwoliło na bardziej płynne i realistyczne działanie symulacji.
- **Optymalizacja GUI:** Udoskonaliliśmy sposób aktualizacji interfejsu graficznego, aby zapewnić szybsze i bardziej responsywne działanie aplikacji.

Podsumowując, pomimo napotkanych trudności, projekt zakończył się sukcesem dzięki wprowadzeniu odpowiednich poprawek i dokładnemu testowaniu. Symulacja działa poprawnie i realistycznie odzwierciedla działanie systemu kas sklepowych oraz interakcji z klientami.

Wyniki symulacji

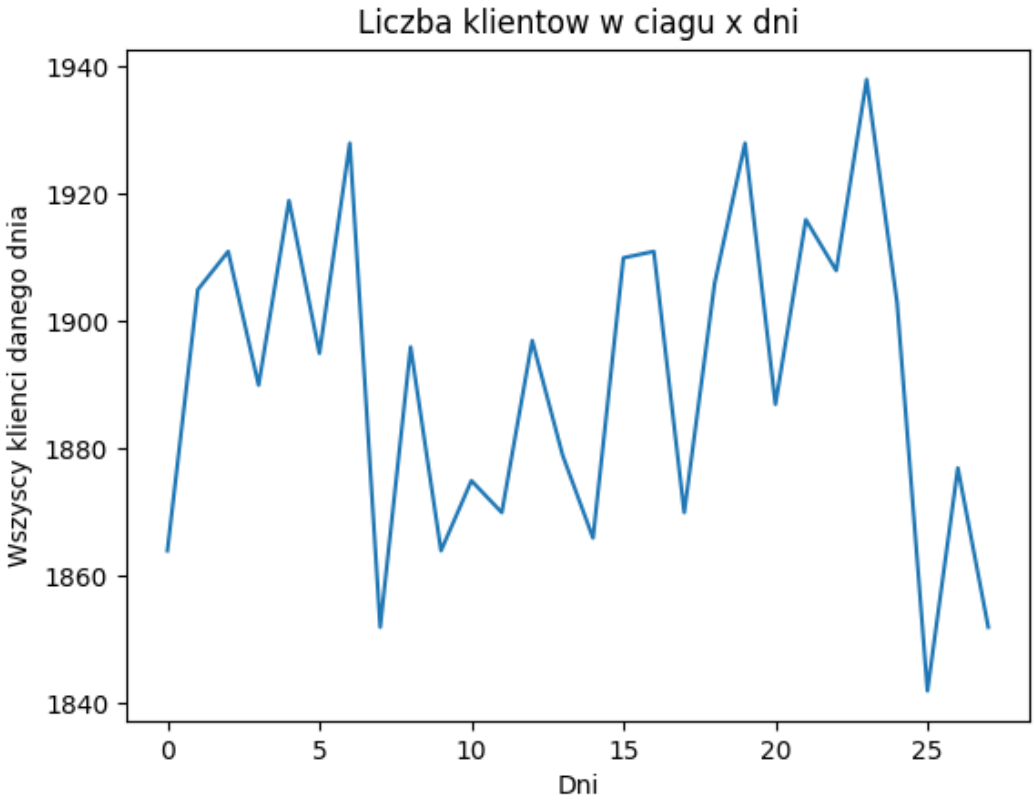
Przeprowadzone scenariusze symulacji

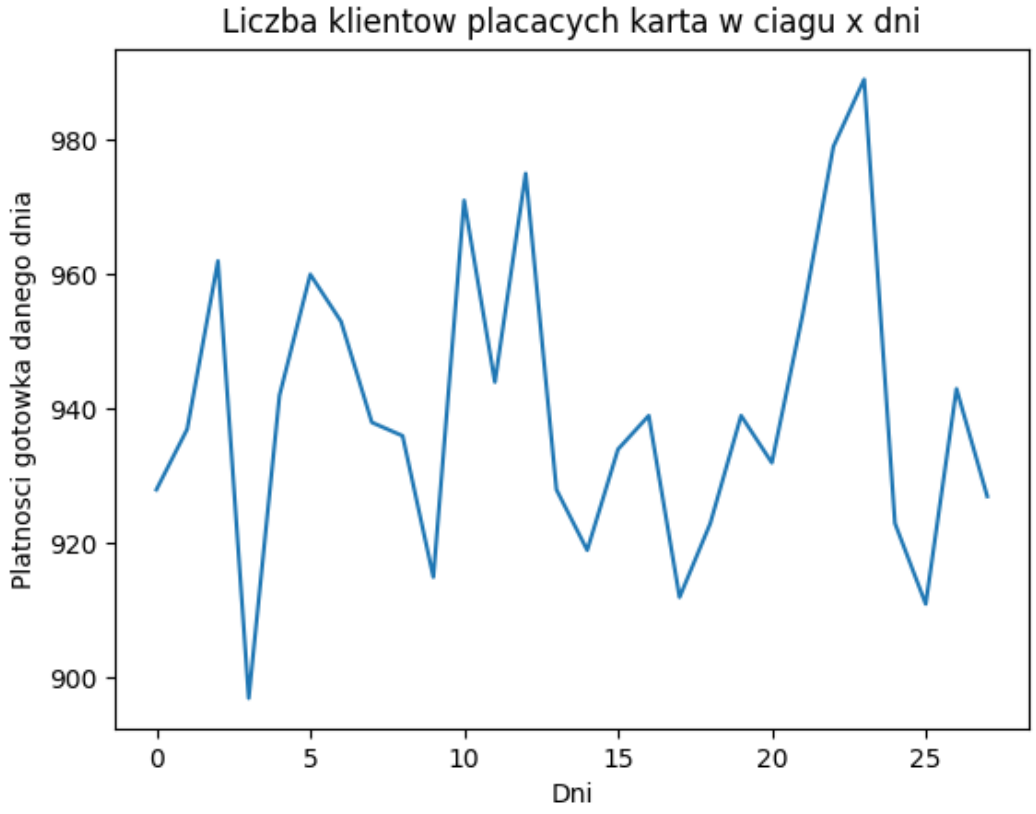
Symulacja przeprowadzona została wobec następujących parametrów; 8 kas o wielkości kolejki 8. Czas symulacji wynosił 1 miesiąc, czyli 28 dni w celu uproszczenia. W każdy dzień sklep otwarty był przez 12 godzin, od godziny 8 do godziny 20. Parametry te dawały możliwość zauważenia wpływu pory dnia na liczbę klientów, a także pozwalały sklepowi obsłużyć większość klientów pojawiających się w sklepie. Dodatkowo przeprowadzona została symulacja o długości 4 dni, w celu uzyskania wykresów ilustrujących godziny szczytu.

Wizualizacja wyników

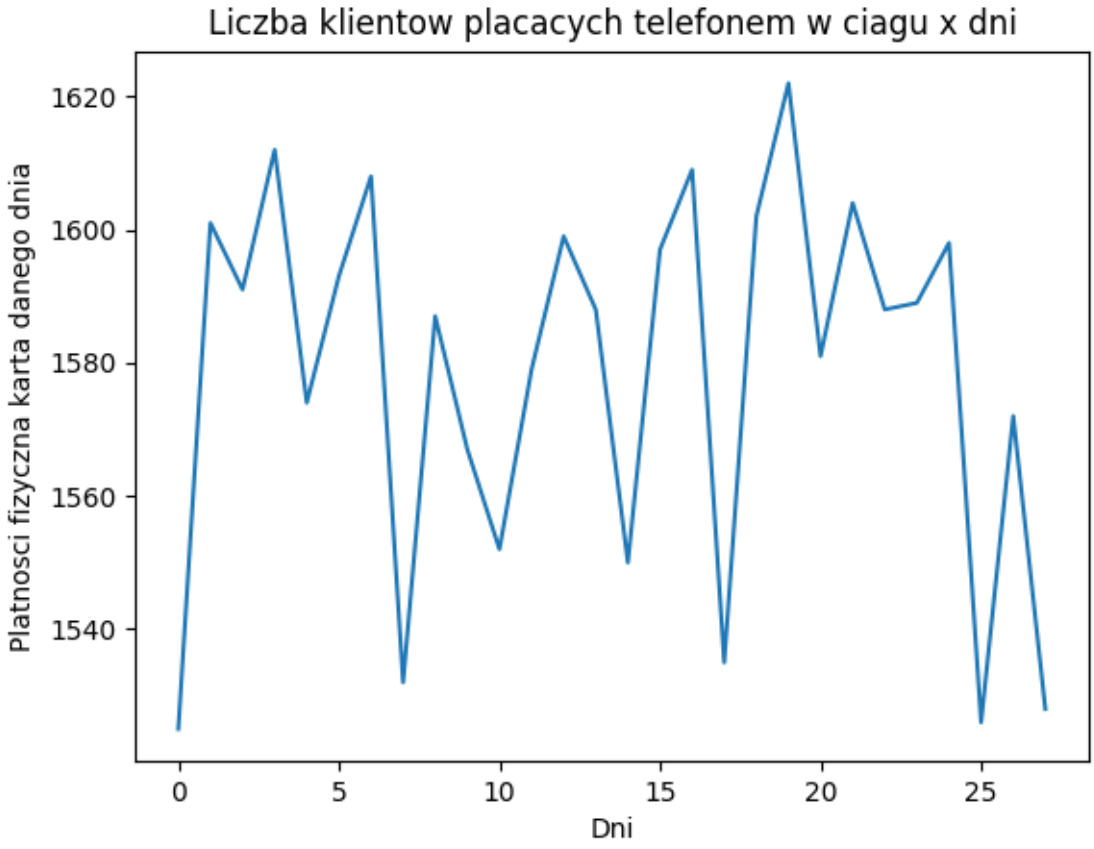
Część statystyczna i demograficzna

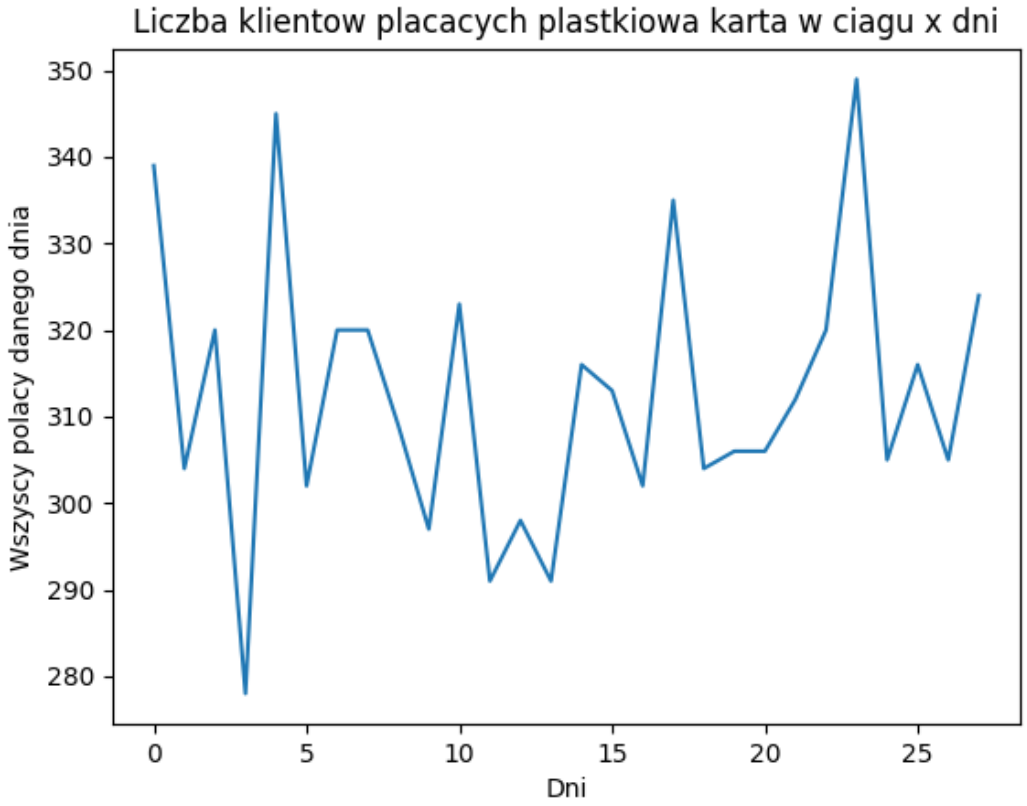
W tej części zliczane i prezentowane są dane co do klientów, ich płci, sposobu płatności oraz narodowości. Uznaliśmy to za informacje istotne, gdyż w sytuacji prawdziwej dane te są wykorzystywane w celu poprawy sprawności oraz przygotowanie sklepu na potrzeby klientów. Miały one bezpośredni wpływ na długość obsługi i co za tym idzie, wyniki symulacji.


Liczba klientow w ciagu x dni
Wszyscy klienci danego dnia
1940
1920
1900
1880
1860
1840
0
5
10
15
20
25
Dni


Liczba klientow placacych karta w ciagu x dni
Platnosci gotowka danego dnia
980
960
940
920
900
0
5
10
15
20
25
Dni


Liczba klientow placacych gotowka w ciagu x dni
Platnosci telefonem danego dnia
980
960
940
920
900
0
5
10
15
20
25
Dni

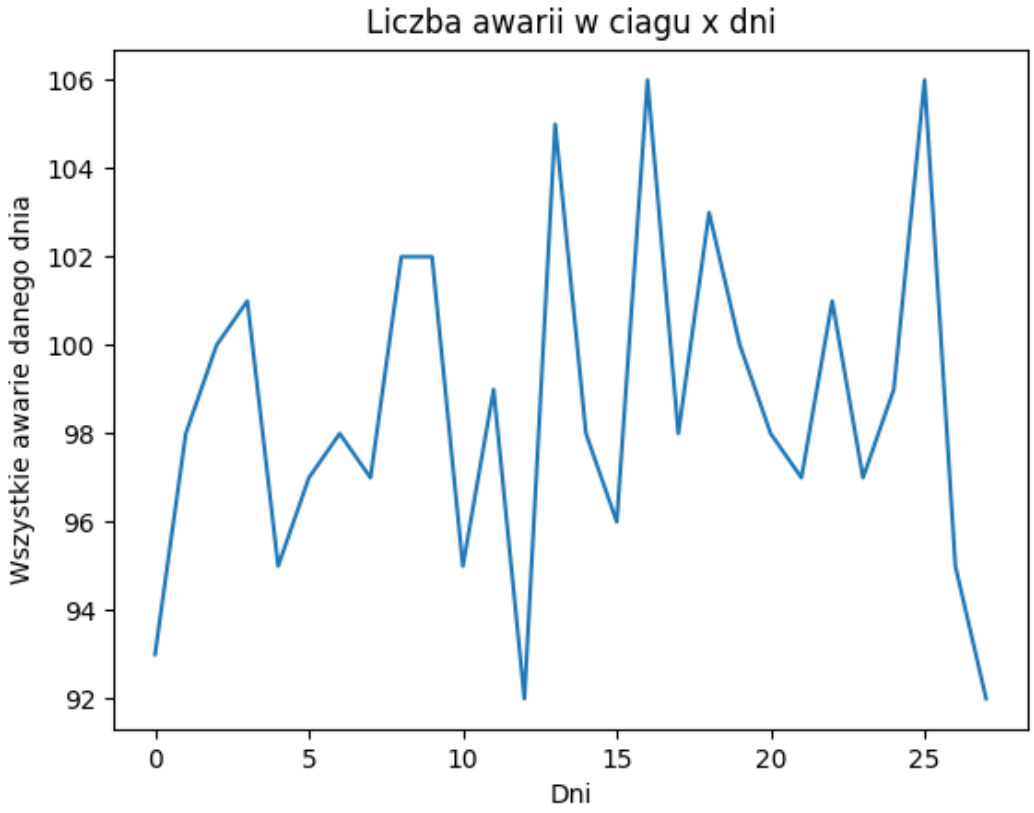

Liczba klientow placacych telefonem w ciagu x dni
Platnosci fizyczna karta danego dnia
1620
1600
1580
1560
1540
0
5
10
15
20
25
Dni

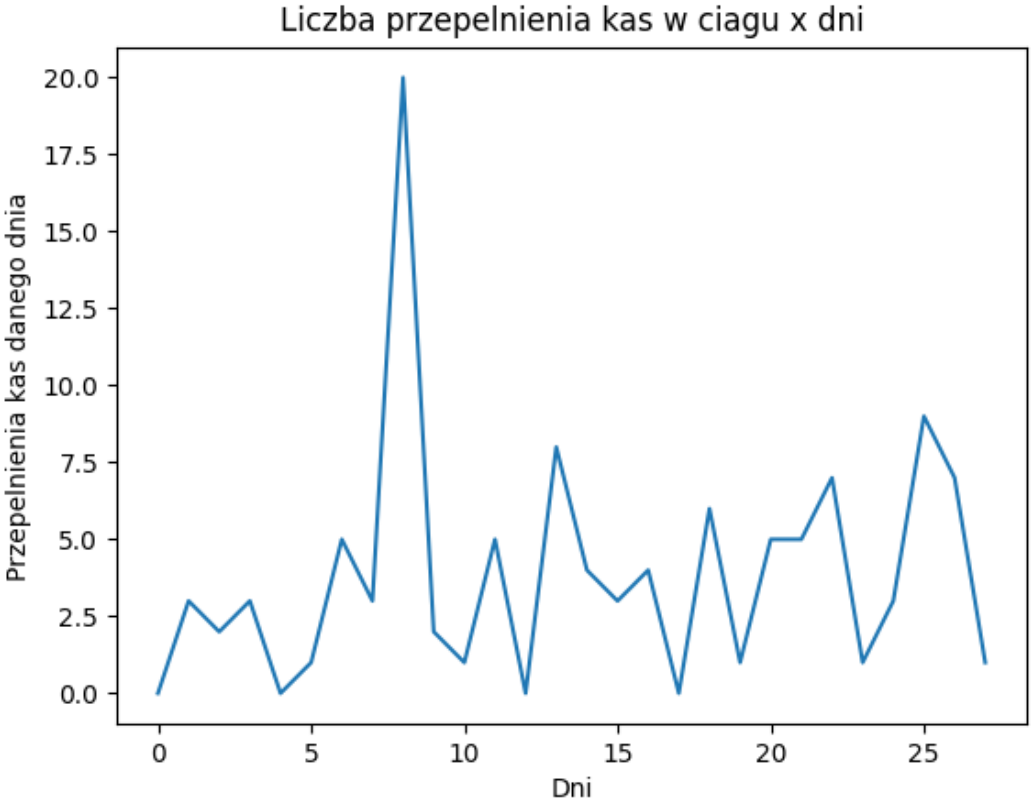

Liczba klientow placacych plastkiowa karta w ciagu x dni
Wszyscy polacy danego dnia
Dni
350
340
330
320
310
300
290
280
0
5
10
15
20
25


Liczba wszystkich klientow w x godzin
Liczba klientow
Godziny
300
250
200
150
100
50
0
5
10
15
20
25
30
35

Część wydajnościowa

W tej części zebrane są dane związane z awariami występującymi przez zdarzenia losowe, wypełnienia kas, oraz wynikające z eksploatacji kas. Dodatkowo zbierane są dane na temat przepełnień występujących, gdy wszystkie kasy mają pełną kolejkę, lub niektóre kasy są niesprawne z powodu awarii. Notowana jest także ilość klientów, która nie jest obsłużona do końca dnia.




Liczba przepelnienia kas w ciagu x dni
Przepelnienia kas danego dnia
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
5
10
15
20
25
Dni


Liczba klientow nieobsluzonych w x dni
Nieobsluzeni klienci
20
15
10
5
0
5
10
15
20
25
Dni

Analiza zebranych danych

Z uzyskanych wyników widzimy, że liczba klientów waha się w przedziale od 1800 do 1950, daje to średnio 150-160 klientów na godzinę. Wykres godzinowy popiera to założenie, można z niego odczytać ok. 100 klientów w godzinę „normalną", widać także skok do ok, 300 klientów w godzinach szczytu, które mają miejsce przez 4 z 12 godzin otwarcia sklepu. Ilość awarii uśrednia się w okolicy 100 dziennie. Wynika z tego 12 awarii na kasę w przeciągu 12 godzin. Daje to więc średnio jeden problem na jedną kasę co godzinę. Jednak, jako że założona definicja awarii jest bardzo obszerna i zawiera oprócz awarii sprzętu, zdarzenia losowe oraz z winy klienta, nie jest to liczba nierealistyczna i wskazuje dobrze dobrane parametry w kodzie.

Liczba przepełnień była niska z wyjątkiem dnia 8, w którym wyniosła ona 20. Liczba awarii tego dnia nie była elementem odstającym statystycznie. Powodem tej sytuacji mogło więc być nałożenie się na siebie kilku awarii w jednym momencie oraz klientów z wysokim czasem obsługi.

Liczba klientów nieobsłużonych nie była zależna od ilości awarii. Zależała ona bardziej od losowej liczby klientów pojawiającej się w sklepie tuż przed ostatnimi 30 minutami otwarcia sklepu, podczas których nie przychodzili już nowi klienci. W celu uzyskania innego wyniku należałoby zmienić długość czasu zamknięcia w kodzie.

Wnioski

Efektywność i Realizm Symulacji

Realizacja projektu symulacji kas w sklepie pozwoliła na stworzenie modelu, który efektywnie odzwierciedla rzeczywiste procesy zachodzące w punkcie sprzedaży. Dzięki odpowiedniemu doborowi zmiennych i wprowadzeniu losowości w parametrach klientów, symulacja stała się realistyczna i umożliwia analizę różnych scenariuszy. Możliwość symulowania awarii kas i zarządzania nimi dodała dodatkowy poziom realizmu, pokazując, jak takie sytuacje wpływają na obsługę klientów.

Złożoność Zarządzania Kolejkami i Obsługą Klientów

Jednym z głównych wniosków płynących z realizacji projektu jest zrozumienie złożoności zarządzania kolejkami i obsługą klientów. Początkowe problemy z zarządzaniem kolejkami pokazały, jak ważne jest dokładne przemyślenie logiki biznesowej oraz implementacja mechanizmów obsługi wyjątków. Poprawne działanie systemu wymagało iteracyjnego podejścia do projektowania i testowania algorytmów.

Znaczenie Testowania i Debugowania

Proces testowania i debugowania był kluczowy dla osiągnięcia stabilności i poprawności symulacji. Testowanie różnych scenariuszy działania sklepu pozwoliło na identyfikację i naprawę wielu błędów logicznych. Testy jednostkowe, choć jeszcze nie zaimplementowane, są niezbędnym elementem, który należy dodać, aby zapewnić ciągłość poprawnego działania systemu przy kolejnych zmianach i rozszerzeniach funkcjonalności.

Wykorzystanie Interfejsu Graficznego

Zastosowanie interfejsu graficznego (GUI) znacząco poprawiło interaktywność i użyteczność symulacji. Wizualne reprezentowanie stanu kas i kolejek klientów ułatwiło zrozumienie i analizę działania systemu. Jednakże, synchronizacja GUI z logiką symulacji okazała się być wyzwaniem, które wymagało szczególnej uwagi i licznych poprawek.

Ulepszenia i Przyszły Rozwój

Projektowanie i implementacja symulacji pozwoliły na zidentyfikowanie kilku obszarów, które można by jeszcze ulepszyć. Przyszły rozwój systemu mógłby obejmować:

- **Zaawansowane scenariusze:** Wprowadzenie bardziej zaawansowanych scenariuszy, takich jak różne promocje, zmieniające się godziny pracy czy specjalne wydarzenia wpływające na liczebność klientów.
- **Lepsze zarządzanie danymi:** Ulepszenie zarządzania danymi, np. poprzez zapis wyników symulacji do plików i ich późniejszą analizę.
- **Optymalizacja wydajności:** Przeprowadzenie analizy wydajnościowej i optymalizacja kodu, aby symulacja mogła obsłużyć większe liczby kas i klientów bez spadku wydajności.
- **Rozszerzenie testów:** Implementacja pełnego zestawu testów jednostkowych oraz testów integracyjnych, aby zapewnić jeszcze większą pewność co do poprawności działania systemu.

Podsumowanie

Realizacja projektu symulacji kas w sklepie była wymagającym, ale bardzo pouczającym przedsięwzięciem. Praca nad nim pozwoliła na zdobycie cennego doświadczenia w zakresie projektowania systemów symulacyjnych, zarządzania złożonymi strukturami danych oraz synchronizacji z interfejsem graficznym. Pomimo napotkanych trudności, ostateczny rezultat spełnia założenia i stanowi solidną podstawę do dalszych prac rozwojowych i optymalizacyjnych.