Studia i Materiały Informatyki Stosowanej, Tom 17, Nr 1, 2025
str. 24-29

UNIWERSALNY SKLEP INTERNETOWY

Tobiasz Karbownik, Piotr Prokopowicz

Uniwersytet Kazimierza Wielkiego
Wydział Informatyki
ul. Mikołaja Kopernika 1, 85-074 Bydgoszcz
e-mail: tobiasz.karbownik@student.ukw.edu.pl

Streszczenie: *Niniejszy artykuł opisuje projekt oraz implementację aplikacji webowej przeznaczonej do obsługi uniwersalnego sklepu internetowego. Aplikacja została opracowana z wykorzystaniem frameworka Django opartego na języku Python, uzupełnionego o dodatkowe biblioteki rozszerzające jego bazowe funkcjonalności. Praca przedstawia szczegółową analizę wymagań oraz oczekiwań, jakie aplikacja miała spełnić, a także omówienie procesu projektowania i implementacji rozwiązania.*

Słowa kluczowe: *aplikacja webowa, uniwersalny sklep, sklep internetowy, e-commerce, Django, Python*

UNIVERSAL ONLINE STORE

Abstract: *This article describes the design and implementation of a web application intended for managing a universal online store. The application was developed using the Django framework, based on the Python programming language, and enhanced with additional libraries that extend its core functionalities. The paper presents a detailed analysis of the requirements and expectations that the application was meant to fulfill, as well as a discussion of the design and implementation process.*

Keywords: *web application, universal store, online store, e-commerce, Django, Python*

1. Wstęp

Handel elektroniczny (e-commerce) odgrywa obecnie kluczową rolę w globalnej gospodarce, a jego rozwój jest napędzany dynamicznymi zmianami technologicznymi oraz rosnącym zapotrzebowaniem konsumentów na wygodne i efektywne narzędzia zakupowe. W 2023 roku wartość światowego rynku e-commerce osiągnęła imponujące 6,9 biliona dolarów, co podkreśla nieustannie rosnącą popularność zakupów internetowych. W Polsce odsetek użytkowników regularnie korzystających z tego typu usług także systematycznie wzrasta, a pandemia COVID-19 znacząco przyspieszyła proces transformacji konsumentów z zakupów tradycyjnych na zakupy online. W związku z tym pojawia się rosnące zapotrzebowanie na platformy sprzedażowe, które spełniają oczekiwania nowoczesnych użytkowników w zakresie dostępności, bezpieczeństwa i intuicyjnej obsługi[1-2].

Przedstawiona publikacja koncentruje się na zaprojektowaniu i implementacji uniwersalnego sklepu internetowego, który oferuje szereg funkcjonalności odpowiadających współczesnym standardom rynku e-commerce. Projekt został zrealizowany przy użyciu frameworka Django, bazującego na języku Python, co pozwoliło na stworzenie skalowalnego, bezpiecznego i wydajnego systemu. Aplikacja została wyposażona w kluczowe moduły, takie jak system zarządzania użytkownikami, panel administracyjny, proces rejestracji i logowania, a także interaktywny koszyk zakupowy, który integruje się z systemem płatności Stripe. Ponadto zaimplementowano rozwiązania poprawiające interakcję użytkownika z platformą, w tym możliwość oceniania produktów i sprzedawców, dodawania ofert do ulubionych czy korzystania z czatu w czasie rzeczywistym, realizowanego dzięki bibliotece Django-Channels [3- 4].

W projekcie aplikacji uwzględniono zarówno wymagania funkcjonalne, jak i niefunkcjonalne. System został zoptymalizowany pod kątem szybkości działania, bezpieczeństwa danych użytkowników oraz łatwości rozbudowy w przyszłości. W procesie projektowania baz danych wykorzystano relacyjną bazę SQLite, która zapewniała elastyczność i prostotę w fazie rozwoju, choć w przyszłości platforma może zostać przeniesiona na bardziej zaawansowane systemy, takie jak PostgreSQL czy MySQL, aby sprostać większemu obciążeniu. Diagramy związków encji (ERD) oraz relacyjny model bazy danych pozwoliły na precyzyjne odwzorowanie zależności pomiędzy

obiektami, co umożliwiło efektywne przechowywanie oraz przetwarzanie danych dotyczących produktów, użytkowników i transakcji.[5-8]

W artykule omówiono szczegółowy proces analizy, projektowania oraz implementacji platformy. Zaprezentowano zaawansowane funkcje aplikacji, takie jak dynamiczne filtrowanie i wyszukiwanie ofert, dodawanie przedmiotów do koszyka, a także moduły odpowiedzialne za obsługę użytkowników, w tym rejestrację i logowanie oraz zarządzanie profilami. W rozdziale poświęconym testowaniu omówiono także metody walidacji funkcjonalności oraz bezpieczeństwa aplikacji, które przeprowadzono w celu identyfikacji i eliminacji potencjalnych błędów oraz luk w zabezpieczeniach[9-10]

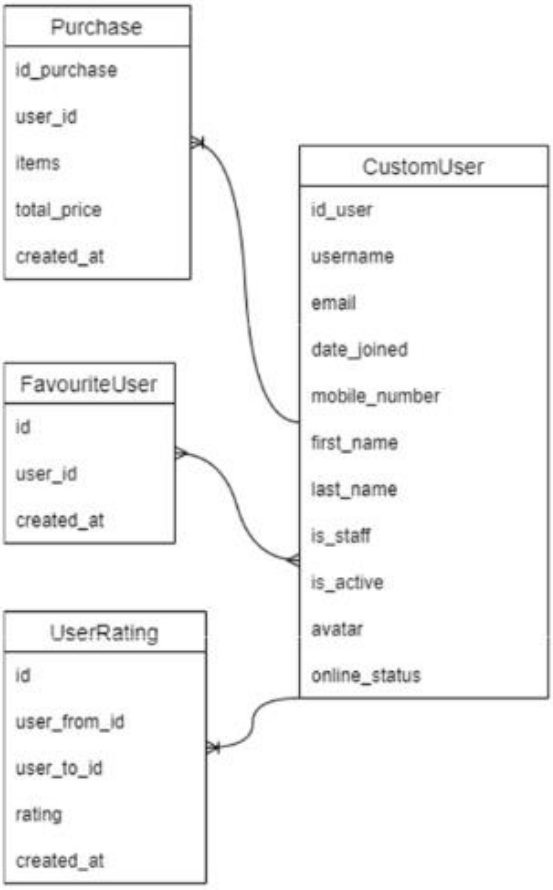
Przedstawiona aplikacja stanowi kompleksowe rozwiązanie e-commerce, które może być wdrożone zarówno przez małe, jak i średnie przedsiębiorstwa, dostosowując się do zmiennych wymagań rynku. Projekt ten nie tylko spełnia współczesne standardy technologiczne, ale także daje możliwość dalszego rozwoju i personalizacji, co czyni go wszechstronnym narzędziem w dynamicznym środowisku handlu elektronicznego [11].

2. Projekt i implementacja uniwersalnego sklepu internetowego

W ramach realizacji projektu polegającego na stworzeniu uniwersalnego sklepu internetowego zastosowano zróżnicowane metody obejmujące analizę wymagań, projektowanie systemu, implementację oraz testowanie

Pierwszym etapem pracy była szczegółowa analiza wymagań funkcjonalnych i niefunkcjonalnych. Wymagania funkcjonalne obejmowały między innymi implementację systemu rejestracji i logowania użytkowników, dodawanie i edycję ofert produktowych, dynamiczne wyszukiwanie oraz filtrowanie produktów, obsługę koszyka zakupowego, integrację z systemem płatności Stripe oraz interaktywny czat w czasie rzeczywistym. Wymagania niefunkcjonalne uwzględniały zapewnienie wysokiej wydajności aplikacji, odpowiedniego poziomu bezpieczeństwa, kompatybilności z różnymi przeglądarkami oraz możliwość skalowania systemu. Na podstawie tych wymagań opracowano szczegółowy plan działań i modele procesów [12].

Na rysunku 1 przedstawiono fragment diagramu związków encji, który ilustruje relacje związane z encją CustomUser. Diagram ten przedstawia, w jaki sposób encja CustomUser jest powiązana z innymi encjami w systemie, wskazując zależności oraz potencjalne interakcje między nimi.

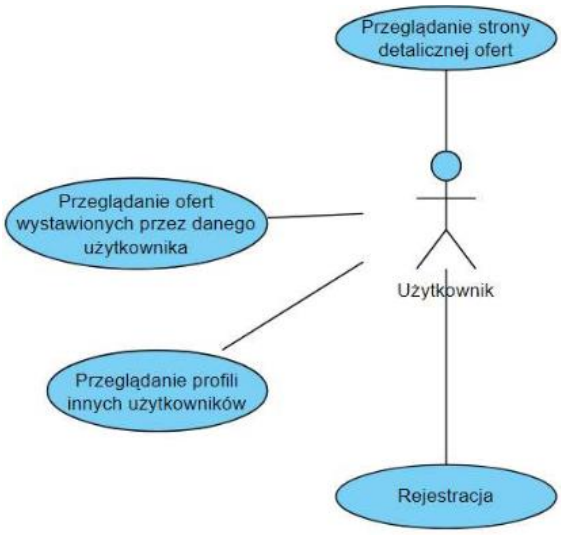


Rysunek. 1 *Wycinek diagramu związków encji przedstawiający relacje CutomUser.*

Proces projektowania systemu obejmował stworzenie diagramów przypadków użycia, diagramów związków encji oraz prototypów interfejsu użytkownika. Model bazy danych został zaprojektowany zgodnie z modelem relacyjnym, co pozwoliło na dokładne odwzorowanie zależności między obiektami, takimi jak użytkownicy, produkty, transakcje oraz koszyk. Relacyjna baza danych SQLite została wybrana jako silnik bazodanowy na etapie prototypowania, z możliwością migracji do bardziej zaawansowanego systemu w przyszłości. W celu zapewnienia intuicyjności i atrakcyjności wizualnej aplikacji opracowano makiety graficzne kluczowych podstron za pomocą narzędzia Figma. Makiety uwzględniały takie elementy jak strona główna, formularze rejestracji i logowania, szczegóły produktów oraz widok koszyka. Wykorzystano wzorzec architektoniczny Model-View-Template, wbudowany w framework Django, który pozwala na efektywne zarządzanie logiką biznesową, interfejsem użytkownika oraz danymi w bazie[13].

Na rysunku 2 przedstawiono fragment schematu przypadków użycia, który dotyczy

działań dostępnych dla niezalogowanego użytkownika. Schemat ten przedstawia możliwe interakcje użytkownika z systemem w stanie, gdy nie posiada on aktywnej sesji ani uprawnień wynikających z zalogowania.



Rysunek. 2 *Wycinek schematu przypadków użycia dla niezalogowanego użytkownika.*

Implementacja systemu została przeprowadzona z użyciem nowoczesnych narzędzi i technologii. Framework Django, oparty na języku Python, został użyty do implementacji logiki aplikacji oraz obsługi bazy danych. Wykorzystano wbudowany mechanizm ORM, który umożliwia efektywne zarządzanie danymi w bazie. Do obsługi protokołu WebSocket, co umożliwiło implementację czatu w czasie rzeczywistym, wykorzystano bibliotekę Django-Channels. Integracja z platformą Stripe pozwoliła na obsługę płatności online, a użycie HTML, CSS i JavaScript umożliwiło stworzenie interaktywnego interfejsu użytkownika. W tym celu wykorzystano również bibliotekę jQuery oraz technologię AJAX do asynchronicznej komunikacji z serwerem. Do implementacji mapy wskazującej lokalizację odbioru produktu użyto biblioteki Folium. Zarządzanie kodem źródłowym odbywało się za pomocą systemu kontroli wersji Git, a repozytorium zostało hostowane na platformie GitHub, co umożliwiło efektywne śledzenie zmian i współpracę w środowisku deweloperskim.
Na rysunku 3 przedstawiono model przedmiotu, który odzwierciedla strukturę i właściwości obiektu w kontekście analizowanego systemu. Model ten przedstawia kluczowe cechy przedmiotu, pomagając w zrozumieniu jego roli i funkcji w całościowej architekturze systemu.

Item	
item_id	UUIDField
category	ForeignKey (id)
subcategory	ForeignKey (id)
user	ForeignKey (id_user)
address	CharField
city	CharField
condition	CharField
country	CharField
created_at	DateTimeField
discription	TextField
image	FileField
latitude	DecimalField
longitude	DecimalField
name	CharField
price	DecimalField
price_negotiable	BooleanField
quantity	PositiveIntegerField
type_of_purchase	CharField
zip_code	CharField

Rysunek. 3 *Model przedmiotu.*

Przed oddaniem końcowej wersji aplikacji przeprowadzono kompleksowe testy, które obejmowały testy funkcjonalne, testy bezpieczeństwa, testy interfejsu użytkownika, testy integracyjne oraz testy wydajnościowe. Testy funkcjonalne zweryfikowały działanie kluczowych modułów aplikacji, takich jak rejestracja użytkowników, dodawanie ofert, przetwarzanie płatności oraz czat w czasie rzeczywistym. Testy bezpieczeństwa pozwoliły na zidentyfikowanie i wyeliminowanie potencjalnych zagrożeń, takich jak ataki typu SQL injection czy Cross-Site Scripting. Framework Django zapewnił wbudowane mechanizmy ochrony przed tego typu atakami. Testy interfejsu użytkownika upewniły, że aplikacja działa poprawnie na różnych urządzeniach i w różnych przeglądarkach, co osiągnięto dzięki zastosowaniu technik responsywnego projektowania CSS. Testy integracyjne zweryfikowały poprawność współdziałania modułów aplikacji, takich jak komunikacja między front-endem a serwerem oraz interakcja z zewnętrznymi systemami, np. Stripe. Wreszcie testy wydajnościowe pozwoliły na ocenę czasu odpowiedzi aplikacji i wydajności bazy danych w scenariuszach obejmujących jednoczesne zapytania od wielu użytkowników [14].

Na rysunku 4 przedstawiono formularz strony rejestracji, który umożliwia nowym użytkownikom utworzenie konta w systemie. Formularz ten zawiera pola wymagane do wprowadzenia danych rejestracyjnych, takich jak imię, adres e-mail czy hasło.

Rysunek 4 *Formularz strony rejestracji.*

Zastosowane metody pozwoliły na stworzenie funkcjonalnej i skalowalnej aplikacji e-commerce, która spełnia wymagania współczesnych użytkowników. Dzięki wykorzystaniu sprawdzonych technologii, takich jak Django i Stripe, oraz przeprowadzeniu wszechstronnych testów zapewniono wysoką jakość oraz bezpieczeństwo aplikacji.

3. Funkcjonalność uniwersalnego sklepu internetowego

W niniejszej pracy zaprezentowano projekt uniwersalnego sklepu internetowego, który wykazał dużą przydatność zastosowanych technologii w kontekście współczesnych wymagań stawianych platformom e-commerce. Kluczowym osiągnięciem projektu jest stworzenie kompletnej aplikacji internetowej, której funkcjonalność obejmuje rejestrację i logowanie użytkowników, zarządzanie produktami, obsługę koszyka zakupowego, integrację z systemem płatności Stripe oraz system ocen i komentarzy. Wykorzystanie frameworka Django przyspieszyło proces tworzenia aplikacji, zapewniając jednocześnie wysoką stabilność i możliwość integracji różnych modułów. Implementacja bazy danych SQLite umożliwiła natomiast szybkie prototypowanie i zarządzanie danymi na potrzeby wczesnych etapów projektu. Przeprowadzone testy funkcjonalne oraz integracyjne dowiodły, że aplikacja działa zgodnie z założeniami, minimalizując błędy wpływające na doświadczenia użytkownika.

Mimo sukcesów projektu, należy zauważyć pewne ograniczenia. Wybór SQLite jako silnika bazy danych, choć uzasadniony prostotą i łatwością implementacji, może ograniczać skalowalność aplikacji oraz jej zdolność do obsługi dużych zbiorów danych i równoczesnych połączeń. Ponadto, wykorzystanie Django ORM, choć przyjazne dla deweloperów, wiąże się z dodatkowym narzutem wydajnościowym, co może stanowić wyzwanie przy bardziej skomplikowanych zapytaniach do bazy danych. Brak zaawansowanych modułów analitycznych i funkcji marketingowych, takich jak analiza zachowań użytkowników czy dynamiczne rekomendacje produktów, ogranicza konkurencyjność aplikacji w porównaniu z komercyjnymi platformami e-commerce.

Kierunki dalszego rozwoju aplikacji obejmują między innymi migrację do bardziej zaawansowanego systemu bazy danych, takiego jak PostgreSQL lub MySQL, co umożliwi lepszą obsługę dużych ilości danych i zapewni wyższą wydajność w środowiskach o zwiększonym obciążeniu. Rozszerzenie funkcjonalności administracyjnych o raporty sprzedażowe oraz statystyki dotyczące użytkowników mogłoby zwiększyć wartość aplikacji dla właścicieli sklepów. Dalsza optymalizacja interfejsu użytkownika, uwzględniająca responsywność oraz zwiększenie dostępności na różnych urządzeniach, pozwoliłaby na poprawę doświadczeń użytkownika. Warto również rozważyć integrację z systemami logistycznymi oraz wdrożenie zaawansowanych funkcji marketingowych, takich jak system rekomendacji produktów oparty na danych o preferencjach użytkowników.

Podsumowując, projekt uniwersalnego sklepu internetowego stanowi dowód na to, że współczesne technologie open source mogą być skutecznie wykorzystane do tworzenia funkcjonalnych i skalowalnych aplikacji e-commerce. Pomimo pewnych ograniczeń technicznych, aplikacja ta może służyć jako solidna baza do dalszego rozwoju i adaptacji w dynamicznie zmieniającym się środowisku rynkowym. Wyzwania związane z jej dalszym rozwojem, takie jak zwiększenie skalowalności czy rozszerzenie funkcjonalności, stanowią jednocześnie obszary o dużym potencjale do innowacji.

4. Podsumowanie i wnioski

Podstawową funkcjonalnością systemu jest zarządzanie produktami, rejestracja użytkowników, realizacja zakupów, a także integracja z systemem płatności. W ramach przeprowadzonych testów systemu stwierdzono, że wszystkie te funkcje

działają stabilnie i zgodnie z założeniami projektowymi. Proces zarządzania produktami jest łatwy w obsłudze i pozwala na szybkie wprowadzanie nowych artykułów, ich edytowanie i usuwanie. Funkcja rejestracji użytkowników działa bezbłędnie, a integracja z systemem płatności została zrealizowana zgodnie z wymaganiami, umożliwiając bezpieczne i szybkie realizowanie transakcji.

Aplikacja oferuje także szereg innowacyjnych funkcji, które wyróżniają ją na tle konkurencyjnych rozwiązań. System ocen użytkowników i produktów pozwala na dynamiczną interakcję, co skutkuje większym zaangażowaniem ze strony użytkowników. Funkcja czatu na żywo zapewnia bezpośrednią komunikację między użytkownikami a obsługą sklepu, co poprawia jakość obsługi klienta i przyspiesza proces rozwiązywania problemów. Dodatkowo, możliwość negocjacji cen stanowi unikalną funkcję, która przyciąga użytkowników szukających elastyczności i lepszych warunków zakupu, co zwiększa konkurencyjność aplikacji na rynku.

Testy aplikacji pod kątem wydajności wykazały, że system działa płynnie przy średnim obciążeniu użytkowników. Niemniej jednak, przeprowadzona analiza wydajności ujawniła pewne ograniczenia w zakresie obsługi dużej liczby użytkowników, szczególnie w kontekście użycia bazy danych SQLite. Choć SQLite jest efektywnym narzędziem w fazie prototypowania, nie jest to optymalne rozwiązanie w przypadku aplikacji o dużym ruchu, co może wpłynąć na skalowalność systemu w przyszłości. Przeprowadzono również testy obciążeniowe, które potwierdziły, że w warunkach dużego ruchu aplikacja może doświadczać spowolnienia w procesach zapisu i odczytu danych [15].

W aspekcie bezpieczeństwa, aplikacja wykazała pełną zgodność z wymaganiami dotyczącymi ochrony danych użytkowników oraz bezpieczeństwa transakcji online. System uwierzytelniania użytkowników działa bezbłędnie, zapewniając bezpieczny dostęp do konta i danych. Testy penetracyjne, przeprowadzone w ramach oceny bezpieczeństwa, nie wykazały poważnych luk w zabezpieczeniach aplikacji. Ponadto, zapobiegawcze środki ochrony, takie jak szyfrowanie danych, sprawiają, że aplikacja spełnia standardy bezpieczeństwa wymagane dla systemów e-commerce[16].

W zakresie użyteczności, aplikacja została oceniona pozytywnie zarówno przez testujących, jak i przez grupę docelową użytkowników. Interfejs użytkownika jest intuicyjny, a kluczowe funkcje są łatwo dostępne, co zapewnia komfort podczas zakupów online. Testy użyteczności wykazały, że aplikacja jest przyjazna dla użytkowników o różnym poziomie zaawansowania technologicznego. Pomimo to, zidentyfikowano obszary, w których możliwa jest optymalizacja, szczególnie w kontekście responsywności aplikacji na urządzeniach mobilnych. Użytkownicy zgłosili sugestie dotyczące poprawy wyglądu koszyka zakupowego oraz usprawnienia nawigacji w obrębie kategorii produktów, co może zostać uwzględnione w przyszłych wersjach systemu.

W odniesieniu do zaawansowanych funkcji analitycznych, system nie zawierał w wersji testowej funkcji do analizy danych dotyczących zachowań użytkowników, takich jak zaawansowane systemy rekomendacji produktów czy analiza preferencji klientów. Testy wykazały, że wprowadzenie takich funkcji mogłoby znacząco poprawić doświadczenie użytkownika i wartość aplikacji zarówno dla końcowych odbiorców, jak i właścicieli sklepów. Analiza ścieżek zakupowych użytkowników, na przykład, mogłaby pozwolić na lepsze dopasowanie oferty do potrzeb konsumentów [17].

Podsumowując, wyniki testów potwierdziły, że aplikacja spełnia kluczowe wymagania funkcjonalne, wydajnościowe oraz użytkowe. Pomimo pewnych ograniczeń związanych z wyborem bazy danych oraz brakiem zaawansowanych funkcji analitycznych, projekt wykazuje duży potencjał w kontekście dalszego rozwoju. Testy i analiza wykonane w trakcie realizacji projektu wskazują na potrzebę migracji do bardziej zaawansowanego silnika bazy danych, a także rozwoju funkcji analitycznych, które pozwolą na jeszcze lepsze dopasowanie oferty do potrzeb użytkowników. Wnioski te stanowią podstawę do planowania dalszych prac nad optymalizacją systemu, mających na celu zwiększenie jego skalowalności, wydajności oraz atrakcyjności w oczach użytkowników.

Literatura

1. C. Jolaosoi Main, „What Is E-Commerce? Definition, Types&Getting Started,” 2023.
2. B. Cox, „Ecommerce statistics to get you ahead in 2024,” Dash, 2024.
3. Django Software Foundation, „Django Documentation,” [Online]. Available: https://docs.djangoproject.com/pl/4.2/. [Data uzyskania dostępu: Marzec 2024].
4. Stripe, „Stripe Documentation,” [Online]. Available: https://stripe.com/docs. [Data uzyskania dostępu: Marzec 2024].
5. International Requirements Engineering Board, „Słownik Terminów Inżynierii Wymagań,”

2023. [Online]. Available: https://sjsi.org/ireb-do-pobrania/. [Data uzyskania dostępu: Marzec 2024].
6. B. Thalheim, Entity-relationship modeling: foundations of database technology, Springer, 2000.
7. J. Beatty i K. Wiegers, Specyfikacja oprogramowania. Inżynieria wymagań. Wydanie III, Helion, 2014.
8. Prajapati, V. (2013). Big Data Analytics with R and Hadoop. Packt Publishing.
9. J. Forcier, P. Bissex i W. Chun, Python i Django: programowanie aplikacji webowych, Helion, 2009.
10. Sale, D. (2014). Testing Python: Applying Unit Testing, TDD, BDD, and Acceptance Testing. Wiley.
11. Chaffey, D. (2022). Digital Marketing: Strategy, Implementation, and Practice. Pearson Education.
12. Sommerville, I. (2020). Software Engineering (10th Edition). Pearson.
13. S. Wrycza, B. Marcinkowski i K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, 2006.
14. Hunt, A., & Thomas, D. (1999). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley.
15. Huang, J. (2021). Database Performance Tuning and Optimization: Strategies for Scalability. Springer.
16. Howard, M., & Lipner, S. (2006). The Security Development Lifecycle. Microsoft Press.
17. Zhao, Z., & Cao, Q. (2019). "Analyzing Consumer Behavior in E-commerce: A Path Analysis." Journal of Business Research.