

Projekt z przedmiotu “Inżynieria wymagań i jakości”

Dariusz Cebula, Szymon Miękina, Urszula Stankiewicz

1. Elevator Pitch.....	3
2. Wizja produktu.....	3
2.1. Czym jest produkt.....	3
2.2. Dla kogo?.....	3
2.2. Obecna sytuacja - problemy do rozwiązania.....	3
2.3. Zaspokajane potrzeby.....	3
2.4. Wykorzystane możliwości.....	4
2.5. Cele - korzyści dla użytkowników.....	4
2.6. Rozwiązania konkurencyjne.....	4
2.7. Główni aktorzy systemu i systemy zewnętrzne.....	5
2.7.1. Przykładowa persona: prowadzący.....	5
2.7.2. Przykładowa persona: student.....	7
2.8 Opis i funkcje systemu.....	8
2.9 Przykłady użycia.....	8
2.10 Mierniki użyteczności produktu.....	9
3. Specyfikacja wymagań.....	9
3.1. User Stories.....	9
3.2. Use cases.....	10
3.3. Specyfikacja formalna skrócona.....	13
Konta i użytkownicy.....	13
3.3.1. Logowanie do systemu.....	13
3.3.2. Wylogowanie z systemu.....	14
3.3.3. Dołączenie do kursu.....	15
3.3.4. Stworzenie kursu.....	15
3.3.5. Edytowanie danych użytkownika.....	16
3.3.6. Nadanie użytkownikowi uprawnień prowadzącego.....	16
3.3.7. Dodanie użytkownika do systemu.....	16
Sprawdziany.....	17
3.3.8. Rozwiąż sprawdzian.....	17
3.3.9. Utworzenie nowego sprawdzianu.....	18
3.3.10. Przeglądaj utworzone sprawdziany.....	18
3.3.11. Przeglądaj wyniki sprawdzianu.....	19
3.3.12. Odpowiedz na pytanie na czacie.....	20
3.3.13. Przeglądaj swoje wyniki sprawdzianu.....	20
3.4. Wireframe.....	22
3.4.1.Sprawdziany z perspektywy wykładowcy.....	22
3.4.2.Sprawdziany z perspektywy studenta.....	41

3.4.3. Quizy z perspektywy wykładowcy.....	50
3.4.4. Quizy z perspektywy studenta.....	61
3.4.5. Skrypty.....	70

Unippets

1. Elevator Pitch

Unippets jest aplikacją webową skierowaną do wykładowców akademickich z kierunków informatycznych, którzy potrzebują mówić w łatwy sposób publikować materiały dla studentów. Umożliwia umieszczanie na platformie skryptów zawierających wykonywalne fragmenty kodu, diagramy itp. oraz tworzenie quizów i form sprawdzania wiedzy. W odróżnieniu od innych produktów na rynku Unippets jest szybka i wygodna w obsłudze oraz umożliwia łatwe dodawanie różnorodnych materiałów.

2. Wizja produktu

2.1. Czym jest produkt

Produkt jest aplikacją webową działającą w przeglądarce.

2.2. Dla kogo?

Produkt skierowany jest do wykładowców akademickich kierunków informatycznych, którzy dobrze radzą sobie z różnego rodzaju technologiami oraz ich studentów.

2.2. Obecna sytuacja - problemy do rozwiązania

Dotychczas prowadzący korzystali z systemu UPeL, który jednak ulega awariom i utrudnia sprawne umieszczanie kawałków kodu w skrypcie wysyłanym studentom tak, by formatowanie było czytelne. System oceniania również nie zawsze działa tak, jak wykładowca by sobie tego życzył, a edycja dodawanych materiałów bywa czasochłonna i skomplikowana. Ponadto materiały zostają przesyłane studentom drogą mailową lub umieszczone na stronie www prowadzącego, co utrudnia studentom sprawną naukę i ocenę wymagań przedmiotu.

2.3. Zaspokajane potrzeby

W ostatnim czasie studentów kierunków informatycznych jest coraz więcej, gdyż kierunki te ciągle zyskują na popularności. Dlatego wykładowcy potrzebują aparatu, który umożliwia im sprawną komunikację z wieloma uczestnikami zajęć jednocześnie. Produkt umożliwia wykładowcom szybką publikację skryptu zawierającego przykładowy kod, którego nie trzeba będzie dodatkowo formatować tak, by był czytelny. Ponadto ułatwia przygotowanie quizów utrwalających wiedzę, oraz form sprawdzania wiedzy takich jak kolokwia i egzaminy. Będzie można w nich zawrzeć fragmenty kodu, co pozwoli w dokładny sposób sprawdzić wiedzę studentów. Pytania zamknięte na kolokwiach i egzaminach będą automatycznie sprawdzane, a ich wyniki zapisywane w systemie na podstawie punktów zdobytych przez studenta, co znacząco usprawni pracę prowadzącego, który nie będzie musiał tracić czasu

na sprawdzanie każdej pracy. W przypadku studentów system pozwoli im na kontrolę informacji o obecnym stanie swojego zaliczenia oraz sprawi, że materiały będą łatwo dostępne w jednym miejscu w przejrzysty sposób. Studenci będą mogli również dzielić się tworzonymi przez siebie skryptami, które dzięki swej interaktywnej formie będą ułatwiały im naukę.

2.4. Wykorzystane możliwości

Wykładowcy posiadają własne, bądź służbowe laptopy/komputery oraz biegły się nimi posługują. Nie potrzebują zaawansowanego środowiska graficznego, by poradzić sobie z obsługą systemu - na co dzień korzystają z komend terminala oraz programują - nie sprawia im problemu szybkie nauczenie się jakiegoś języka czy prostego edytora tekstowego. Podobnie jest w przypadku studentów, którzy często sami z siebie uczą się nowych technologii. Posiadają też zazwyczaj dostęp do komputera - swojego bądź kolegi. Mogą też skorzystać z komputera w bibliotece akademickiej.

2.5. Cele - korzyści dla użytkowników

- Usprawnienie pracy wykładowców - dzięki temu będą mieli więcej czasu na pracę naukową. Zyska na tym również uczelnia (większa liczba publikacji).
- Tworzenie różnorodnych materiałów pozwoli wykładowcom lepiej przykuć uwagę studentów
- Przejrzysta komunikacja ze studentami - jasność co do tego, gdzie dodawane są materiały
- Łatwiejsza nauka (studenci) - dostęp do skryptów zawierających różnorodne materiały
- Dzięki quizom i skryptom, studenci z kolei będą mieć jasny sygnał, czego wymaga prowadzący.

2.6. Rozwiązania konkurencyjne

Rozwiązaniem konkurencyjnym jest system UPeL, który umożliwia dodawania materiałów, przeprowadzanie quizów i przechowywanie ocen w systemie. Jego dużą zaletą jest możliwość tworzenia przez prowadzących zahasłowanych kursów, do których dołączają studenci, czy dodawanie skryptów w blokach tematycznych lub w przedziałach czasowych, co ułatwia organizację. Wadą systemu jest jednak jego niejednolitość - każdy kurs może wyglądać inaczej, a niektórzy prowadzący mają problem z edycją quizów i ustalaniem sposobu ich punktacji. Ponadto UPeL jest systemem skomplikowanym, jeśli chodzi o dodawanie skryptów. Jest to czasochłonne, a dodane materiały nie zawsze są przyjemne w obsłudze. Szczególnie fragmenty kodu ciężko łatwo edytować, a ich wykonywanie jest niemożliwe. System uniemożliwia też dodawanie dużych plików, co może być uciążliwe dla wykładowcy przedmiotu związanego z programowaniem.

Innym takim systemem jest MS Teams, który także umożliwia tworzenie i udostępnianie materiałów oraz przeprowadzanie sprawdzianów. Posiada jednak dużą ilość kanałów w obrębie danego kursu, co może być mylące dla użytkowników oraz utrudnia sprawne

poruszanie się po platformie. Podobnie jak UPeL nie dostarcza możliwości łatwego tworzenia materiałów zawierających fragmenty kodu. Jego zaletą są własne wbudowane edytory, które działają w sposób znajomy większości użytkowników (podobieństwo do pakietu Office).

Warto też wspomnieć o języku Markdown, w którym wygodnie można dodawać fragmenty kodu. Sam język jest prosty do nauczenia, a ponadto istnieją narzędzia (jak chociażby rozszerzenia do VSCode), które umożliwiają live preview tworzonego dokumentu, a także pozwalają na stworzenie pliku pdf. Istnieją platformy, które wykorzystują prostotę i wygodę języka na swoje potrzeby. Przykładem jest tu github, który posiada edytor Markdowna, a nawet własną wersję tego języka. Żaden edytor nie jest jednak zarazem platformą e-learningową pozwalającą na udostępnianie stworzonych materiałów w grupach, czy tworzenie testów wiedzy.

2.7. Główni aktorzy systemu i systemy zewnętrzne

- Prowadzący - może dodawać własne skrypty i materiały, pisać krótkie wiadomości opisujące dodane materiały, tworzyć quizy oraz weryfikować, czy student do nich podszedł, dodawać formy sprawdzania wiedzy, ustalać ich punktację oraz przeglądać oceny studentów
- Student - może przeglądać materiały i skrypty oraz pobierać skrypty umieszczone przez prowadzącego, podchodzić do quizów i form sprawdzania wiedzy oraz śledzić swoje postępy oraz wyniki. Może też tworzyć własne skrypty oraz dzielić się nimi z innymi użytkownikami systemu.

2.7.1. Przykładowa persona: prowadzący



Imię: Andrzej

Nazwisko: Kowalski

Wiek: 40 lat

Wykształcenie: wyższe, skończył kierunek "Informatyka"

Zawód: asystent uczelni technicznej, pracownik naukowy, wykładowca - koordynator przedmiotu "Programowanie obiektowe", prowadzi także przedmiot "Bazy danych"

Stan cywilny: żonaty, ma 2 dzieci

Zainteresowania: Interesuje się nowymi językami programowania oraz technologiami pojawiającymi się na rynku, w wolnym czasie tworzy własne projekty IoT, jeździ na rowerze i fotografuje

Marzenia i cele życiowe: wyjazd na wakacje z rodziną na Malediwy, stworzenie własnej aplikacji "Inteligentny dom"

Umiejętności techniczne: sprawnie posługuje się komputerem, na co dzień korzysta z systemy ArchLinux bez środowiska desktopowego, więc radzi sobie biegły z obsługą komend terminala i konfiguracją narzędzi, kiedy czegoś nie potrafi, z łatwością znajduje rozwiązanie w internecie

Doświadczenie i nawyki związane z projektem: Dotychczas korzystanie z systemu UPeL uznawał za zbyt czasochłonne i niewygodne - nie pozwalało mu w wygodny sposób umieszczać dużych plików z kodem, ani przykładów ze świata programistycznego. Dlatego też materiały dla studentów umieszczał na własnej stronie www, gdzie też pojawiały się informacje o planowanych kolokwiach oraz pliki Excel z numerami indeksów i ocenami za kolokwia i egzaminy. Z tego powodu często otrzymywał prośbę o konsultacje, gdyż studenci nie byli pewni, dlaczego otrzymali daną liczbę punktów. Studenci często dopytywali też mailowo, które materiały obowiązują na dane zajęcia. Konsultacje oraz odpowiadanie na maile powodowało u niego frustrację i niechęć do prowadzenia zajęć na uczelni.

Używany sprzęt: średniej półki laptop dostarczany przez uczelnię, prywatnie posiada komputer stacjonarny o dobrym procesorze i dużej pamięci RAM. Dzięki pracy na uczelni ma licencję na wiele płatnych platform i programów.

Nadzieje związane z projektem: liczy na zmniejszenie się liczby maili w jego skrzynce - będzie mógł wreszcie w pełni skupić się na rozwijaniu swojej aplikacji "Inteligentny dom". Ma nadzieję, że wklejane przez niego wycinki programu nie będą wymagały dodatkowego formatowania po skopiowaniu ich z IDE do skryptu, a także że dodawanie materiałów będzie szybkie i przyjemne, a jednocześnie konfigurowalne tak, jak będzie sobie tego życzył.

Obawy: Boi się, że powstanie kolejny monumentalny system, w którym w każdym semestrze będzie się powiększać liczba stworzonych przez niego kursów dla studentów i że będzie musiał co roku dodawać te same materiały dla innego grona osób. Ponadto obawia się, że będzie musiał poświęcać więcej czasu na wstawianie materiałów i przygotowanie quizów, a dodawanie fragmentów kodu i diagramów nadal nie będzie wygodne.

2.7.2. Przykładowa persona: student



Imię: Jan

Nazwisko: Nowak

Wiek: 21 lat

Wykształcenie: średnie, absolwent technikum elektronicznego, student kierunku "Informatyka"

Zawód: student, dorabia pracując w piekarni i udzielając korepetycji z fizyki

Stan cywilny: niezamężny

Zainteresowania: Interesuje się programowaniem niskopoziomowym oraz algorytmiką, w wolnym czasie spotyka się ze znajomymi i jeździ na nartach, chodzi na siłownię

Marzenia i cele życiowe: Praca w firmie Google, zostanie instruktorem narciarstwa, wyjazd za granicę na stałe

Umiejętności techniczne: sprawnie posługuje się komputerem, chętnie poznaje nieznane mu obszary IT

Doświadczenie i nawyki związane z projektem: Jak dotąd denerwowało go to, że każdy wykładowca ma swój sposób organizacji plików i ocen w systemie UPeL, a niektórzy korzystają z innych systemów takich jak Microsoft Teams lub własne strony internetowe wykładowców. Przysparzało mu to wielu problemów, gdyż musiał pamiętać, który przedmiot na której platformie jest realizowany. Dodatkowo materiały dodane przez prowadzących rzadko były dla niego pomocne, więcej dawały mu materiały znalezione w internecie czy filmiki na platformie YouTube. Z systemu korzystał tylko wtedy, kiedy musiał, najintensywniej przed kolokwiami i egzaminami.

Używany sprzęt: średniej półki laptop gamingowy

Nadzieje związane z projektem: liczy na jednolity sposób umieszczania informacji o przedmiotach przez wszystkich prowadzących na jednej platformie oraz na różnorodność i interaktywność materiałów. Ma nadzieję, że materiały te wystarczą mu, by opanować dany materiał i nie będzie musiał szukać pomocy gdzie indziej. Obiecuje sobie, że wtedy będzie się uczył regularnie i rozwiązywał zadania domowe.

Obawy: Boi się, że powstanie jeszcze jeden system, który spodoba się tylko części wykładowców, przez co nadal będzie musiał korzystać z wielu platform na raz. Obawia się też, że quizy i kolokwia będą trudniejsze, gdyż prowadzącym będzie łatwiej je edytować i dodawać, przez co zaliczenie przedmiotu może się stać bardziej wymagające.

2.8 Opis i funkcje systemu

Wysoka jakość materiałów na zajęciach przynosi zarówno dla prowadzącego, jak i studentów wiele korzyści - ułatwia zrozumienie tematu, jest bardziej atrakcyjnym podejściem do nauki niż większość tradycyjnych sposobów nauczania, oraz ze względu na ważny aspekt automatyzacji, która w połączeniu z swobodą, prostotą tworzenia notatek i multimedialnych pozwała skrócić znacznie czas przygotowania do zajęć.

Dlatego, uwzględniając potrzeby dzisiejszych uczelni technicznych, aplikacja udostępnia wbudowane edytory do tworzenia materiałów na zajęcia - skryptów, quizów, sprawdzianów. Ważnym aspektem jest duża integracja środowiska, umożliwiająca tworzenie konspektów bogatych w multimedia i elementy interaktywne.

Materiały będą tworzone przez wykładowcę w języku Markdown, który świetnie nadaje się do umieszczania w nim fragmentów kodu. Podczas tworzenia skryptu, sprawdzianu lub quizu, będzie wyświetlany na żywo podgląd tworzonego materiału. Dodatkową funkcją systemu jest możliwość umieszczania w materiałach fragmentów wykonywalnych kodu, które po opublikowaniu danego materiału będzie można wykonać na platformie (twórca będzie zobowiązany zaznaczyć, w jakim języku jest dany fragment kodu, a platforma sama znajdzie kompilator online, w którym wykona kod, a następnie umieści jego wynik w odpowiednim okienku na platformie).

Materiały prowadzący będzie mógł udostępniać odpowiednim grupom studentów oraz ustalać konkretne daty sprawdzianów oraz punktację za zadania i próg zdawalności testu.

Ponadto podobne opcje będzie można udostępnić studentom, którzy podczas odpowiadania na pytania otwarte quizów i sprawdzianów, również będą zobowiązani posługiwać się językiem Markdown. Zestaw narzędzi do tworzenia skryptów również będzie dostępny dla studentów, chcących usprawnić swoją pracę w projektach przedmiotowych. Skrypty te studenci będą mogli udostępniać sobie nawzajem.

Dodatkową funkcją jest możliwość zadania wykładowcy pytania dotyczącego treści na czacie.

Prowadzący mają również możliwość uzyskania raportów o postępie studentów z quizów i testów. Dostarczają one informacji o najczęściej popełnianych przez studentów błędach.

2.9 Przykłady użycia

Aby lepiej zilustrować dostępna funkcjonalność, przedstawimy kilka przykładów użycia aplikacji:

1. Prowadzący wykład chce lepiej zwiastować koncepty wprowadzone na poprzednim wykładzie. Jego obecny skrypt jest pełen tekstu, i prostych schematów, które nie są wystarczające dla studentów, wpływając na tempo prowadzenia ćwiczeń i będące przyczyną małego zaangażowania podczas wykładu. Jako że temat materiałów obejmuje wiele małych kawałków kodu, prowadzący decyduje się umieścić konspekt na Unippets, dodając do istniejącego skryptu kod, z opcja

uruchomienia w środowisku w przeglądarce, znacząco ułatwiające przyswojenie materiału.

2. Grupa studentów rozplanowuje projekt z zajęć z programowania. Jednym z problemów jest sprawne komunikowanie poszczególnych idei rozwiązania ważniejszych algorytmów i innych konceptów, które będą potrzebne innym uczestnikom projektu. W tym celu tworzą konspekt, który udostępniają całej grupie. Każdy członek projektu jest w ten sposób komentować rozwiązania projektowe, bez konieczności spotykania się w celu omówienia postępów.

2.10 Mierniki użyteczności produktu

- Wykładowca powinien być w stanie tworzyć nowe konspekty w systemie z minimalnym czasem, zakładając, że czas ten powinien być o 50% krótszy niż czas tworzenia konspektów przed wdrożeniem systemu.
- Przynajmniej 75% wykładowców powinno korzystać z edytora konspektów dostępnego w nowym systemie
- Zwiększenie średniej ocen z quizów zdobytych przez studentów, co będzie oznaczało zwiększenie skuteczności nauki w nowym systemie
- Student powinien być w stanie szybciej nauczyć się nowych zagadnień korzystając z konspektów dodawanych przez wykładowcę

3. Specyfikacja wymagań

3.1. User Stories

Jako wykładowca:

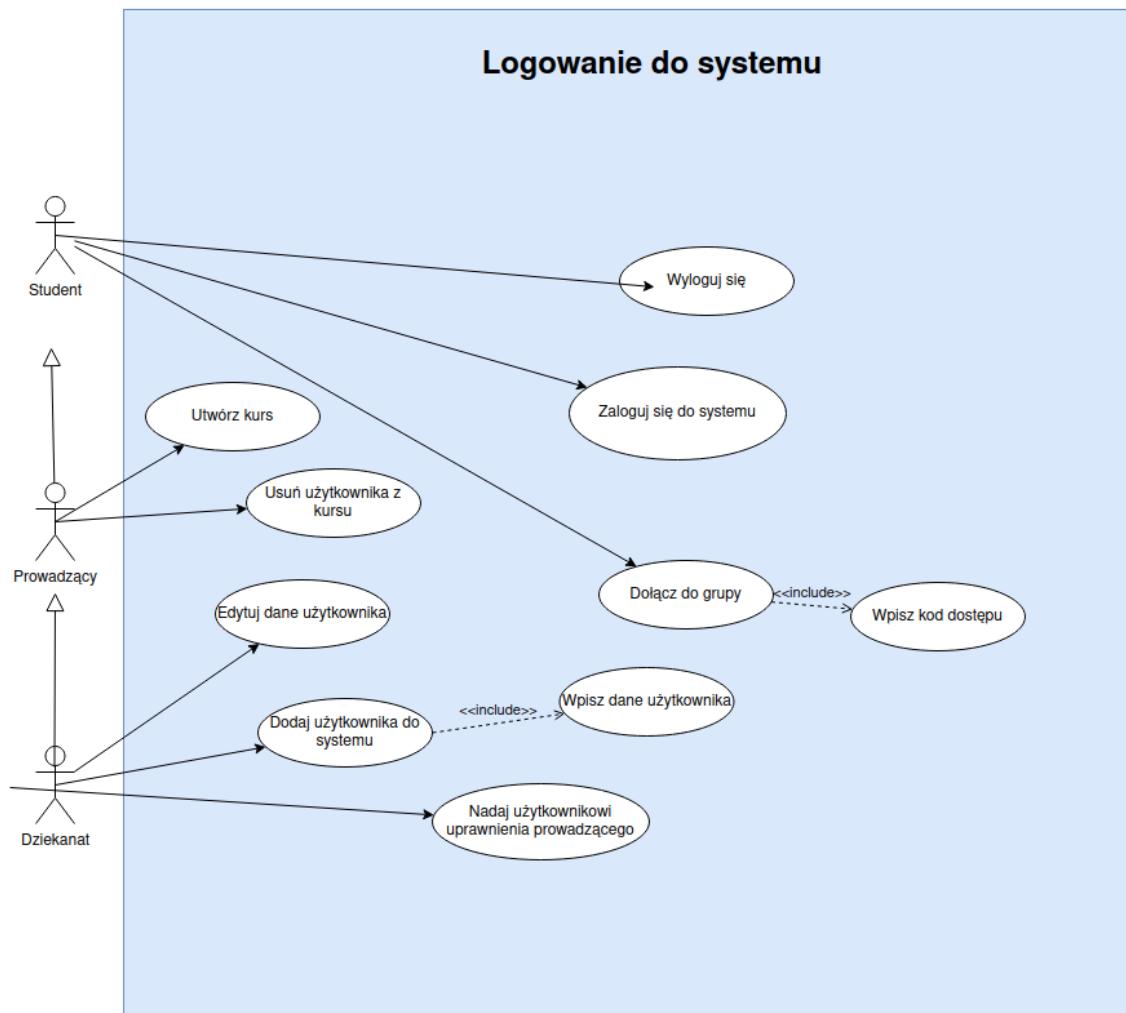
- chcę móc utworzyć nową grupę zajęciową, aby efektywnie zarządzać materiałami i aktywnościami dla danej klasy.
- chcę móc dodawać studentów do mojej grupy zajęciowej, aby zapewnić dostęp do materiałów edukacyjnych i aktywności.
- chcę móc tworzyć konspekty w formacie Markdown, aby dostarczyć klarowne i estetyczne materiały edukacyjne.
- chcę móc edytować istniejący konspekt, aby zaktualizować materiały w dowolnym momencie
- chcę móc tworzyć quizy i sprawdziany do konspektu, aby oceniać postępy studentów i dostarczać im dodatkowe ćwiczenia
- chcę móc przypisywać zadania do grupy zajęciowej, aby umożliwić studentom praktyczne zastosowanie zdobytej wiedzy

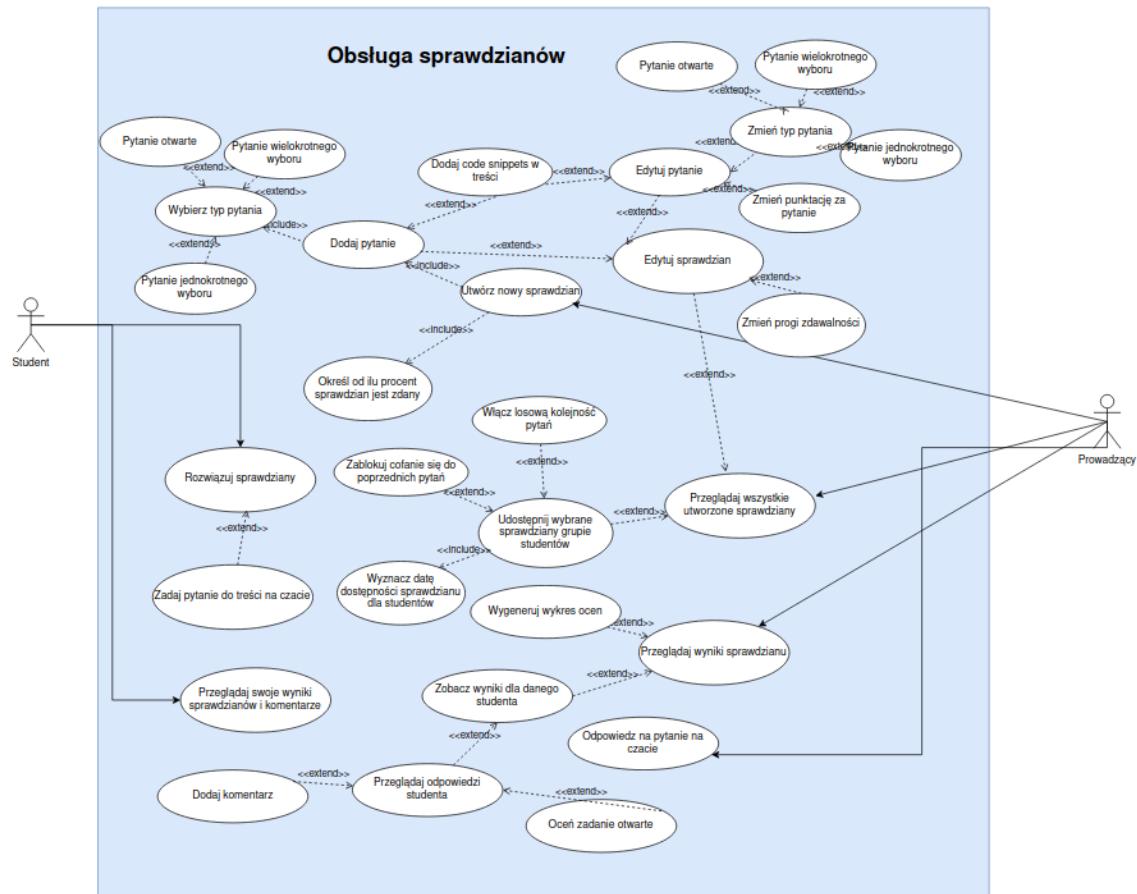
Jako student:

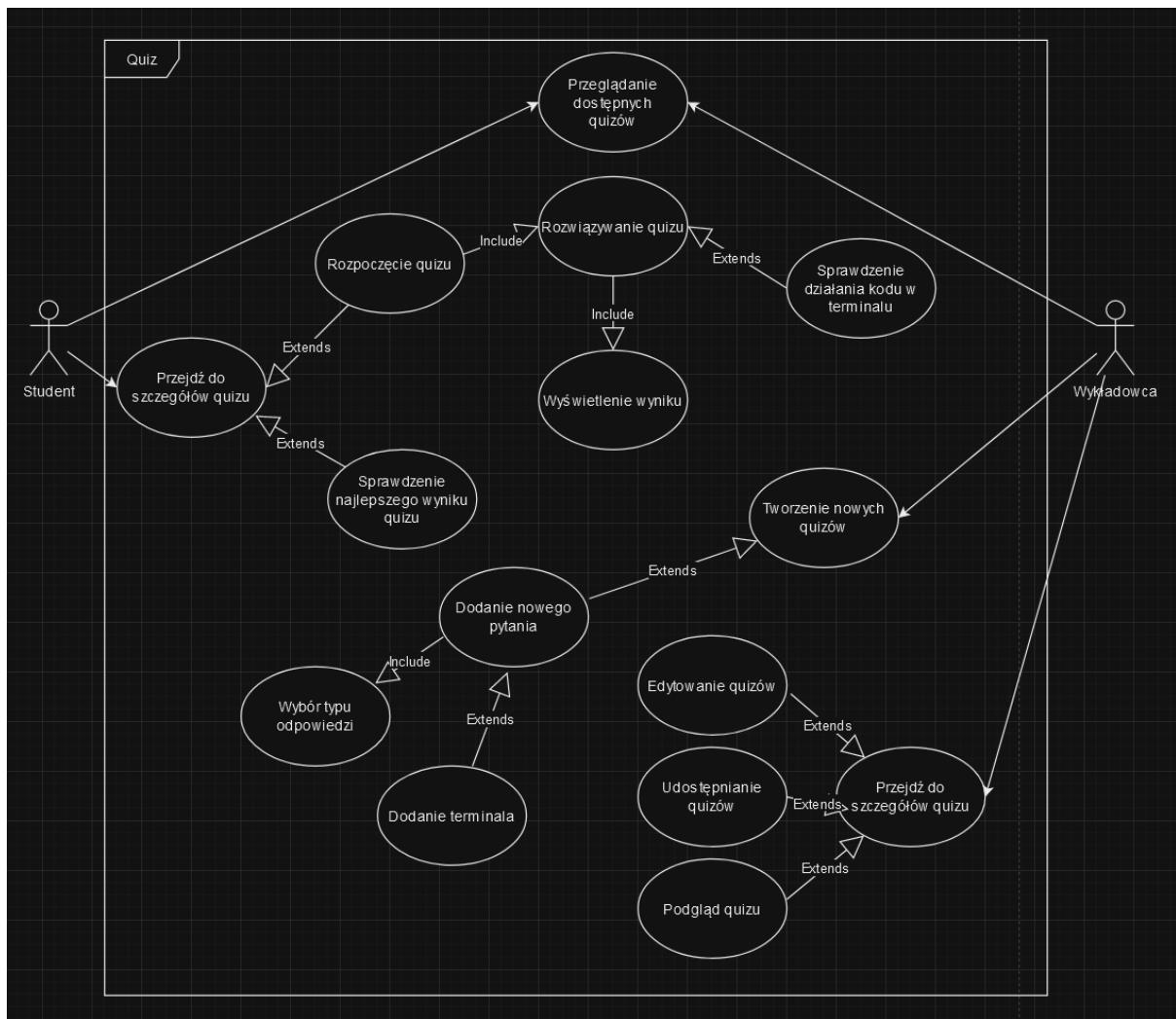
- chcę móc dołączyć do grupy zajęciowej, aby mieć dostęp do materiałów edukacyjnych i zadań
- chcę móc przeglądać konspekty, aby skutecznie się uczyć
- chcę móc rozwiązywać zadania przypisane do grupy, aby praktycznie zastosować zdobytą wiedzę

- chcę móc uczestniczyć w quizach i sprawdzianach, aby oceniać swoje umiejętności
- chcę otrzymywać informacje zwrotne od wykładowcy dotyczące rozwiązań zadań i quizów, aby lepiej zrozumieć swoje postępy
- chcę móc zgłaszać problemy związane z konspektami i zadaniami, aby otrzymać pomoc od wykładowcy

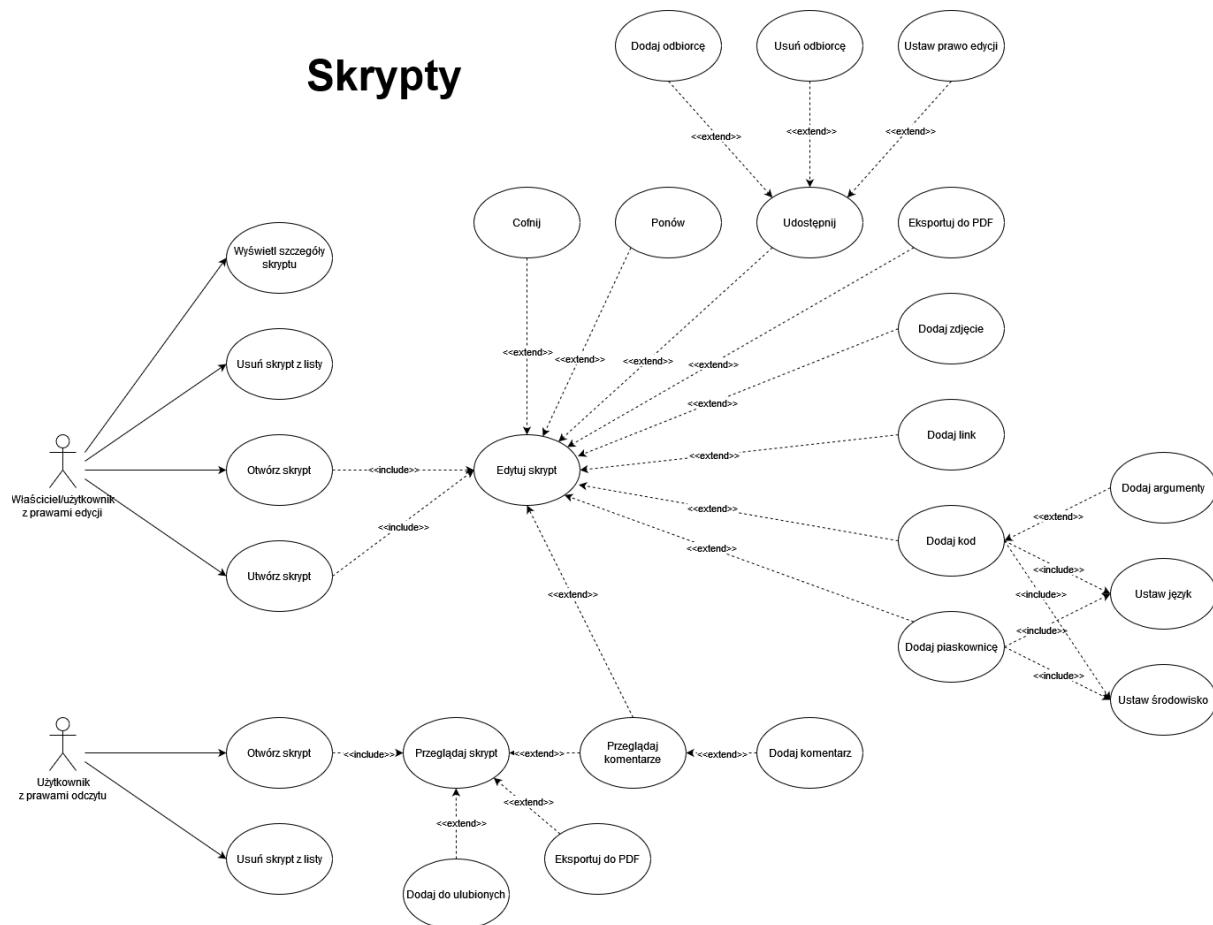
3.2. Use cases







Skrypty



3.3. Specyfikacja formalna skrócona

Konta i użytkownicy

3.3.1. Logowanie do systemu

Cel: Zalogowanie się do systemu

Główny aktor: Użytkownik

Warunek początkowy (warunki wstępne): użytkownik niezalogowany, wyświetla się strona główna systemu (home)

Zdarzenie początkowe (wyzwalaacz): Użytkownik kliknął "Zaloguj się"

Główny scenariusz sukcesu:

1. Wyświetlane jest okno logowania z miejscem na wpisanie loginu i hasła
2. Użytkownik wpisuje login i hasło oraz kliką "Zaloguj"/Enter
3. Wprowadzone dane są poprawne (pozytywnie zweryfikowane przez system) i użytkownik zostaje przekierowany do swojego dashboardu (wyświetlane są kursy, na które jest zapisany)

Scenariusze alternatywne:

- 1.1. Wprowadzone przez użytkownika hasło nie odpowiada loginowi
- 1.2. Wyświetla się komunikat o błędzie logowania, użytkownik jest proszony o ponowną próbę logowania. Oprócz tego wyświetla się też możliwość kliknięcia "Nie pamiętam hasła".
 - 1.2.1. Użytkownik podejmuje kolejną próbę logowania, wpisując login i hasło
 - a) Użytkownik wpisał hasło poprawnie, przechodzimy do punktu 3 głównego scenariusza
 - b) Użytkownik ponownie podał złe hasło. Jeżeli ilość błędnych prób logowania przekroczyła 3, konto jest blokowane, wyświetla się informacja o konieczności kontaktu z dziekanatem w celu odblokowania konta i zmiany hasła
 - 1.2.2. Użytkownik kliką "Nie pamiętam hasła" - zostaje przekierowany na stronę, gdzie może wprowadzić swojego maila. Kliką "Potwierdź", zostaje wysłany do niego mail zawierający link umożliwiający zmianę hasła oraz jednorazowy kod
 - 1.2.2.1. Użytkownik loguje się do swojego konta email, otwiera odpowiedniego maila, klikając link.
 - 1.2.2.2. Wyświetla się prośba o podanie adresu email oraz zawartego w mailu kodu
 - a) Użytkownik podaje poprawny adres email i kod - wyświetla się okno umożliwiające wpisanie nowego hasła, system dokonuje walidacji hasła - czy zawiera przynajmniej 10 znaków, w tym przynajmniej 2 cyfry, 2 znaki i 3 duże litery
 - i) jeśli tak, użytkownik jest przekierowywany na stronę logowania
 - ii) jeśli nie, wyświetla się komunikat "Hasło zbyt słabe" - użytkownik jest ponownie proszony o wprowadzenie poprawnego hasła
 - b) Użytkownik podaje błędna parę email - kod, wyświetlany jest komunikat o błędzie, użytkownik ma możliwość próby wpisania poprawnych danych jeszcze raz. Jeśli ilość niepoprawnych prób przekroczy 3, konto jest blokowane, wyświetla się informacja o konieczności kontaktu z dziekanatem w celu zmiany hasła i odblokowania konta

2.1. Użytkownik nie posiada konta

2.2. Użytkownik kliką "Utwórz konto", wyświetla się możliwość kontaktu z dziekanatem w celu utworzenia konta

3.3.2. Wylogowanie z systemu

Cel: Wylogowanie się z systemu

Główny aktor: Użytkownik

Warunek początkowy (warunki wstępne): użytkownik zalogowany, wyświetla się dowolna podstrona systemu, na górze menu

Zdarzenie początkowe (wyzwalacz): Użytkownik wybrał w menu "Wyloguj"

Główny scenariusz sukcesu:

1. Wyświetla się okienko z możliwością potwierdzenia chęci wylogowania
2. Użytkownik kliką "Tak, chcę się wylogować"
3. Zostaje wylogowany i przekierowany do strony głównej systemu (Home)

Scenariusze alternatywne:

- 1.1. Po wyświetleniu okienka z możliwością potwierdzenia chęci wylogowania użytkownik kliką “Anuluj”
- 1.2. Okienko zostaje zamknięte, a użytkownik widzi stronę, którą przeglądał przed kliknięciem “wyloguj”

3.3.3. Dołączenie do kursu

Cel: Dołączenie do kursu

Główny aktor: Użytkownik

Warunek początkowy (warunki wstępne): użytkownik zalogowany

Zdarzenie początkowe (wyzwalacz): Użytkownik kliknął w menu “Dołącz do kursu”

Główny scenariusz sukcesu: Wyświetla się strona z polem umożliwiającym wpisanie kodu dostępu do kursu

1. Kod jest zweryfikowany jako istniejący, użytkownik zostaje dodany do kursu
2. Wyświetla się pierwsza strona kursu, zawierająca bieżące informacje od prowadzącego oraz odnośniki do quizów, sprawdzianów i materiałów

Scenariusze alternatywne:

- 1.1. Podany kod jest niepoprawny - nie istnieje kurs o takim kodzie
 - a) Użytkownik wpisuje kod ponownie
 - b) Użytkownik kliką “anuluj” - zostaje przekierowany do swojego dashboardu

3.3.4. Stworzenie kursu

Cel: Stworzenie kursu

Główny aktor: Prowadzący

Warunek początkowy (warunki wstępne): prowadzący zalogowany

Zdarzenie początkowe (wyzwalacz): Prowadzący kliknął w menu “Utwórz kurs”

Główny scenariusz sukcesu:

1. Wyświetla się strona z polem umożliwiającym nadanie nazwy kursu, opisu oraz wprowadzenia nazwy przedmiotu
2. Prowadzący kliką “zatwierdź”
3. Generowany i wyświetlany jest kod dostępu do kursu wraz z polem “kopij do schowka”
4. Prowadzący kliką “kopij do schowka” - kod zostaje skopiowany
5. Prowadzący kliką “Powrót” - jest przekierowany do swojego dashboardu

Scenariusze alternatywne:

- 1.1. Prowadzący kliką “Anuluj” - wszystkie zmiany są kasowane, a prowadzący przekierowany do swojego dashboardu

3.3.5. Edytowanie danych użytkownika

Cel: Uaktualnienie danych użytkownika

Główny aktor: Pracownik dziekanatu

Warunek początkowy (warunki wstępne): pracownik zalogowany

Zdarzenie początkowe (wyzwalacz): Pracownik wybrał istniejącego użytkownika i kliknął "Edytuj dane"

Główny scenariusz sukcesu:

1. Wyświetlana jest strona z aktualnymi danymi użytkownika, które da się edytować
2. Pracownik edytuje dane i kliką "zatwierdź"
3. Dane są zapisywane w systemie, a pracownik przekierowany z powrotem do listy użytkowników

Scenariusze alternatywne:

- 1.1. Pracownik kliką "Anuluj" - wszystkie zmiany są kasowane, a pracownik przekierowany z powrotem do listy użytkowników

3.3.6. Nadanie użytkownikowi uprawnień prowadzącego

Cel: Nadanie użytkownikowi uprawnień prowadzącego

Główny aktor: Pracownik dziekanatu

Warunek początkowy (warunki wstępne): pracownik zalogowany

Zdarzenie początkowe (wyzwalacz): Pracownik wybrał istniejącego użytkownika i kliknął "Nadaj uprawnienia prowadzącego"

Główny scenariusz sukcesu:

1. wyświetlane jest okienko z możliwością potwierdzenia nadania uprawnień lub ich anulowania
2. Pracownik kliką "zatwierdź"
3. Wybranemu użytkownikowi przyznawane są przez system uprawnienia prowadzącego
4. Pracownik przekierowany z powrotem do listy użytkowników

Scenariusze alternatywne:

- 1.1. Pracownik kliką "Anuluj" - wszystkie zmiany są kasowane, a pracownik przekierowany z powrotem do listy użytkowników

3.3.7. Dodanie użytkownika do systemu

Cel: Dodanie użytkownika do systemu

Główny aktor: Pracownik dziekanatu

Warunek początkowy (warunki wstępne): pracownik zalogowany

Zdarzenie początkowe (wyzwalacz): Pracownik kliknął "Dodaj użytkownika"

Główny scenariusz sukcesu:

1. Wyświetlane jest okno umożliwiające wpisanie danych użytkownika (imię, nazwisko, adres email, numer indeksu/numer pracownika)
2. Pracownik uzupełnia dane
3. Pracownik kliką zatwierdź - użytkownikowi wysyłany jest na podany adres email login i hasło tymczasowe
4. Pracownik przekierowany do swojego dashboardu

Scenariusze alternatywne:

- 1.1. Pracownik kliką “Anuluj” - wszystkie zmiany są kasowane, a pracownik przekierowany do swojego dashboardu

Sprawdziany

3.3.8. Rozwiąż sprawdzian

Cel: Rozwiązywanie sprawdzianu

Główny aktor: Student

Warunek początkowy (warunki wstępne): zalogowany student, należy do kursu, w którym udostępniony jest sprawdzian, wybrany sprawdzian jest aktualnie aktywny

Zdarzenie początkowe (wyzwalacz): Student wybrał sprawdzian i kliknął rozpoczęij

Główny scenariusz sukcesu:

1. Wyświetla się pierwsze pytanie, a w prawym górnym rogu czas pozostały do końca sprawdzianu
2. Student zaznacza odpowiedź (pytanie zamknięte) lub wpisuje ją (pytanie otwarte)
3. Student scrolluje w dół - zostaje wyświetcone następne pytanie
4. Jeśli sprawdzian ma aktywne wracanie do pytań poprzednich, student scrollując w górę, może też wrócić do poprzedniego pytania
5. Po odpowiedzeniu na wszystkie pytania student kliką “zakończ” - wyświetlany jest jego wynik z pytań zamkniętych.
6. Po kliknięciu “Wróć do sprawdzianów” jest przekierowywany na stronę ze wszystkimi sprawdzianami w kursie

Scenariusze alternatywne:

- 1.1. Czas pisania sprawdzianu kończy się przed kliknięciem przez studenta “zakończ”
- 1.2. Zmiany wprowadzone dotąd przez studenta są zapisywane i wyświetlana jest strona z wynikiem studenta z pytań zamkniętych
 - 2.1. Jeśli student przy danym pytaniu ma wątpliwości co do treści, kliką “zadaj pytanie”
 - 2.2. Pojawia się okienko z możliwością wpisania pytania
 - 2.3. Student wpisuje pytanie i kliką “wyślij”
 - 2.4. Jeśli prowadzący odpowiedział na pytanie lub inny student je zadał, z prawej strony ekranu pojawia się o tym informacja
 - 2.5. Student kliką w ikonkę czatu

2.6. Wyświetlane jest okienko z czatem sprawdzianu oraz odpowiedzią prowadzącego

3.3.9. Utworzenie nowego sprawdzianu

Cel: Utworzenie nowego sprawdzianu

Główny aktor: Prowadzący

Warunek początkowy (warunki wstępne): prowadzący zalogowany

Zdarzenie początkowe (wyzwalacz): Prowadzący kliknął "utwórz sprawdzian"

Główny scenariusz sukcesu:

1. Wyświetlana jest strona pozwalająca na określenie, od ilu procent sprawdzian jest zdany, a także określenia tytułu i tematyki sprawdzianu
2. Prowadzący wprowadza wybrane przez siebie dane i kliką "zatwierdź"
3. Pojawia się strona z szablonem pierwszego pytania
4. Prowadzący wybiera typ pytania (wiekokrotny/jednokrotny wybór/otwarte)
5. Prowadzący wpisuje treść pytania, opcjonalnie dodając odpowiednie fragmenty kodu, posługując się językiem Markdown (np. kod w Javie jest dodawany następująco: ``java jakiś program``) oraz możliwe odpowiedzi, jeśli pytanie jest zamknięte
6. Prowadzący zaznacza poprawne odpowiedzi jeśli pytanie jest zamknięte
7. Określa punktację za pytanie
8. Scrolluje w dół
9. Wyświetlany jest szablon kolejnego pytania (wracamy do punktu 3)
10. Jeśli prowadzący chce zakończyć dodawanie pytań, kliką "Zatwierdź i zakończ"
11. Zmiany są zapisywane w systemie, a prowadzący przekierowany do widoku swoich sprawdzianów

Scenariusze alternatywne:

1.1. Prowadzący kliką "Anuluj" - wszystkie zmiany są kasowane, a prowadzący przekierowany do swojego dashboardu

2.1. Prowadzący kliką "dodaj kod wykonywalny"

2.2. Wyświetla się okienko z wyborem języka programowania

2.3. Prowadzący wybiera odpowiedni język, linijkę kodu, w której powinien się znaleźć fragment możliwy kodu do uruchomienia oraz wkleja kod i kliką "zatwierdź"

2.4. Okienko z możliwością wykonania kodu jest dodane do treści pytania

3.3.10. Przeglądaj utworzone sprawdziany

Cel: Przeglądnięcie utworzonych przez siebie sprawdzianów

Główny aktor: Prowadzący

Warunek początkowy (warunki wstępne): prowadzący zalogowany

Zdarzenie początkowe (wyzwalacz): Prowadzący kliknął "Przeglądaj sprawdziany"

Główny scenariusz sukcesu:

1. Wyświetlana jest strona zawierająca wszystkie tytuły i tematyki sprawdzianów utworzonych przez danego prowadzącego

2. Prowadzący klika sortuj po tytule/tematyce/dacie dodania
3. Sprawdziany są ustawiane w podanej kolejności
4. Prowadzący klika dany tytuł sprawdzianu
5. Wyświetlane są kolejne pytania sprawdzianu oraz ich punktacja
6. Prowadzący klika "powrót" - przekierowanie na stronę z wszystkimi sprawdzianami

Scenariusze alternatywne:

- 1.1. Prowadzący klika przy danym sprawdzianie "udostępnij sprawdzian studentom"
- 1.2. Wyświetla się strona z kursami prowadzącego
- 1.3. Prowadzący zaznacza wybraną grupę, której chce udostępnić sprawdzian oraz termin sprawdzianu i czas rozwiązywania, klika "dalej"
- 1.4. Wyświetlają się opcje sprawdzianu: włącz losową kolejność pytań, zablokuj możliwość wracania do pytań
- 1.5. Prowadzący zaznacza wybrane opcje i klika "zatwierdź"
- 1.6. Zostaje przekierowany do strony ze sprawdzianami, a sprawdzian jest udostępniany w wybranych kursach z zaznaczonymi opcjami i terminem
 - 2.1. Prowadzący po wybraniu danego sprawdzianu (4) klika "Edytuj"
 - 2.2. Wyświetlana jest strona z tytułem oraz tematyką sprawdziany oraz programami procentowymi na kolejne oceny, a także możliwość ich zmiany
 - 2.3. Prowadzący zmienia progi zdawalności (lub nie), tytuł, opis lub zdjęcie i klika "dalej"
 - 2.4. Wyświetlana jest lista pytań z możliwością zmiany ich treści, punktacji oraz typu pytania
 - 2.5. Prowadzący wprowadza zmiany treści/punktacji/typu pytania, ewentualnie dodaje code snippets (tak jak w dodawaniu pytania przy tworzeniu sprawdzianu (punkty 4-7, par. 3.3.9.) lub fragmenty kodu, które można wykonać (scenariusz alternatywny 2 dodania sprawdzianu (par. 3.3.9.) i klika "zatwierdź")
 - 2.5.1. Opcjonalnie prowadzący dodaje pytanie scrollując w dół i uzupełniając szablon pytania (punkty 4-7, par. 3.3.9.)
 - 2.6. Zmiany są zapisywane, a prowadzący przekierowany na stronę przeglądania sprawdzianów

3.3.11. Przeglądaj wyniki sprawdzianu

Cel: Przeglądnięcie wyników danego sprawdzianu w danym kursie, dodanie ewentualnych komentarzy dla studentów oraz ocena zadań otwartych

Główny aktor: Prowadzący

Warunek początkowy (warunki wstępne): prowadzący zalogowany

Zdarzenie początkowe (wyzwalacz): Prowadzący w odpowiednim kursie kliknął przy wybranym sprawdzianie "Zobacz wyniki"

Główny scenariusz sukcesu:

1. Wyświetla się lista studentów zapisanych do kursu wraz z informacją czy wzięli udział w sprawdzianie oraz procentem uzyskanych punktów

2. Prowadzący kliką w pole z wynikiem danego studenta
3. Wyświetla się lista odpowiedzi i punktów, które student zdobył za poszczególne pytania
4. Prowadzący czyta odpowiedzi na pytania otwarte i wybiera w odpowiednim polu liczbę przyznanych punktów za pytanie
5. Prowadzący dodaje komentarze do zadań w polu obok pytania
6. Kliką zatwierdź
7. Zmiany są zapisywane w systemie, a prowadzący przekierowany do listy wyników studentów
8. Prowadzący może kliknąć "wygeneruj wykres ocen" - wtedy generowany jest wykres i pojawia się okienko zapisu wykresu na komputerze prowadzącego - prowadzący wybiera lokalizację i kliką "zapisz". Wykres zapisywany jest na dysku prowadzącego w wybranej lokalizacji
9. Po zakończeniu przeglądania i edycji wyników prowadzący kliką "zakończ" i jest przekierowany do strony głównej kursu, w którym przeglądał wyniki sprawdzianu

3.3.12. Odpowiedz na pytanie na czacie

Cel: Wyjaśnienie niejasności co do treści pytania w czasie rozwiązywania przez studentów sprawdzianu

Główny aktor: Prowadzący

Warunek początkowy (warunki wstępne): prowadzący zalogowany, znajduje się na stronie z kursami

Zdarzenie początkowe (wyzwalacz): Prowadzący kliknął ikonkę czatu w odpowiednim kursie i aktywnym sprawdzianie

Główny scenariusz sukcesu:

1. Wyświetla się strona z czatem
2. Prowadzący kliką "odpowiedź" przy wybranym pytaniu studenta i wpisuje odpowiedź
3. Kliką "wyślij" - odpowiedź zostaje wysłana do wszystkich studentów biorących udział w sprawdzianie

3.3.13. Przeglądaj swoje wyniki sprawdzianu

Cel: Zobaczenie własnych wyników sprawdzianów, komentarzy prowadzącego oraz średniej ocen w kursie

Główny aktor: Student

Warunek początkowy (warunki wstępne): Student zalogowany, znajduje się na stronie odpowiedniego kursu

Zdarzenie początkowe (wyzwalacz): Student kliknął "Przeglądaj wyniki sprawdzianów"

Główny scenariusz sukcesu:

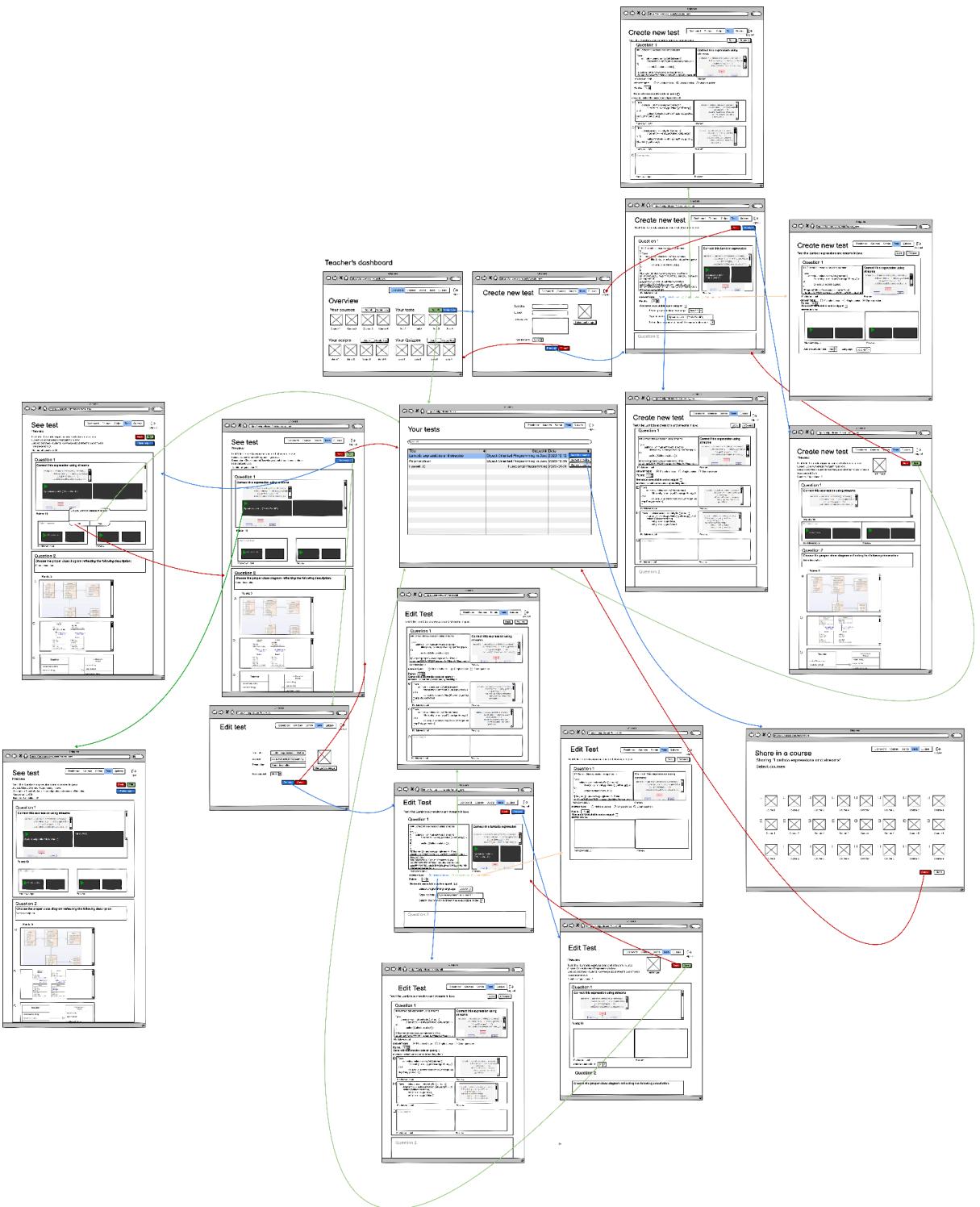
1. Wyświetla się lista wszystkich napisanych przez studenta sprawdzianów wraz z ich wynikiem procentowym, na górze średni procent zdobytych punktów
2. Student scrolluje wyniki i kliką w wybrany sprawdzian

3. Jeśli pytania otwarte zostały ocenione, wyświetla się strona zawierająca wszystkie pytania otwarte wraz z punktami otrzymanymi za nie oraz komentarzami prowadzącego. W przeciwnym razie wyświetlane są wyniki z pytań zamkniętych oraz pytania otwarte i odpowiedzi studenta na nie
4. Zakończywszy przeglądanie, student kliką "zakończ"
5. Zostaje przekierowany na stronę z listą sprawdzianów

3.4. Wireframe

3.4.1. Sprawdziany z perspektywy wykładowcy

- wireflow ([PDF z schematami w wysokiej jakości](#))



- Powiększenie wireframe'ów

Teacher's dashboard

The wireframe shows a web browser window for 'Unippets' at the URL <https://unippets.com/dashboard>. The dashboard has a navigation bar with tabs: Dashboard (selected), Courses, Scripts, Tests, and Quizzes. A 'Log out' button is also in the top right. The main area is titled 'Overview'.

Your courses: Shows four items, each with a delete icon. Buttons for 'See all' and 'Create New' are above them.

Course 1	Course 2	Course 3	Course 4
Course 1	Course 2	Course 3	Course 4

Your tests: Shows four items, each with a delete icon. Buttons for 'See all' and 'Create New' are above them.

Test 1	Test 2	Test 3	Test 4
Test 1	Test 2	Test 3	Test 4

Your scripts: Shows four items, each with a delete icon. Buttons for 'See all' and 'Create New' are above them.

Script 1	Script 2	Script 3	Script 4
Script 1	Script 2	Script 3	Script 4

Your quizzes: Shows four items, each with a delete icon. Buttons for 'See all' and 'Create New' are above them.

Quiz 1	Quiz 2	Quiz 3	Quiz 4
Quiz 1	Quiz 2	Quiz 3	Quiz 4

Unippets

https://unippets.com/tests/share

Dashboard Courses Scripts Tests Quizzes Log out

Share in a course

Sharing: 'Lambda expressions and streams'

Select a course

search

 Course 1  Course 2  Course 3  Course 4  Course 1  Course 2  Course 3  Course 4

 Course 1  Course 2  Course 3  Course 4  Course 1  Course 2  Course 3  Course 4

 Course 1  Course 2  Course 3  Course 4  Course 1  Course 2  Course 3  Course 4

Unippets

https://unippets.com/tests/create_new

See test

Preview

Test title: Lambda expressions and streams in java
 Subject: Object Oriented Programming in Java
 Description: Check students' knowledge about streams and lambdas
 Pass percent: 50%
 Number of questions: 12

Question 1
Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        .getEnergy() > 0)
    .collect(Collectors.toList());
```

System.out.println("Hello World");

Points: 10

Submit answer

Write code...

Markdown input

Preview

Question 2
Choose the proper class diagram reflecting the following description:

Some description

Points: 5

Unippets
https://unippets.com/tests/create_new

See test

Preview

Test title: Lambda expressions and streams in java
 Subject: Object Oriented Programming in Java
 Description: Check students' knowledge about streams and lambdas
 Pass percent: 50%
 Number of questions: 12

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        .getEnergy() > 0)
    .collect(Collectors.toList());
```

System.out.println("Hello World");

Hello World

Points: 10

Submit answer

Write code...

Markdown input

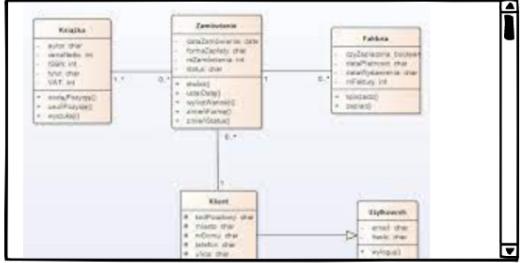
Preview

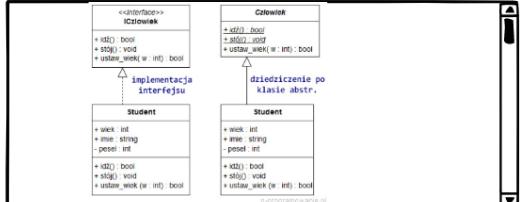
Question 2

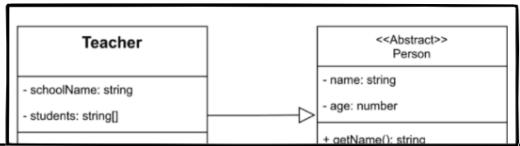
Choose the proper class diagram reflecting the following description:

Some description

Points: 5







Unippets
https://unippets.com/tests/create_new

See test

Preview

Test title: Lambda expressions and streams in java
 Subject: Object Oriented Programming in Java
 Description: Check students' knowledge about streams and lambdas
 Pass percent: 50%
 Number of questions: 12

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        ().getEnergy() > 0)
    .collect(Collectors.toList());
```

System.out.println("Hello World");

Points: 10

Do you want to delete this test?

Submit answer

No Yes

Write code... Preview

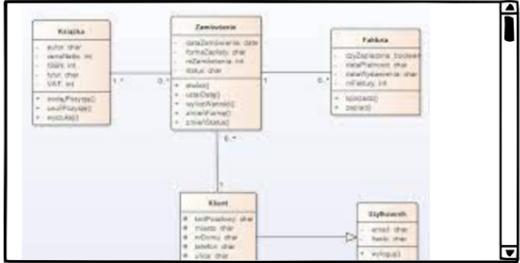
Markdown input

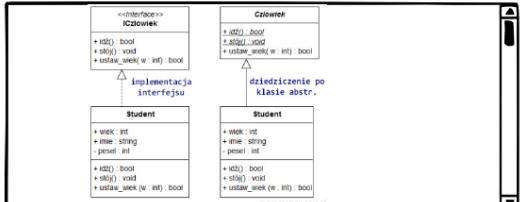
Question 2

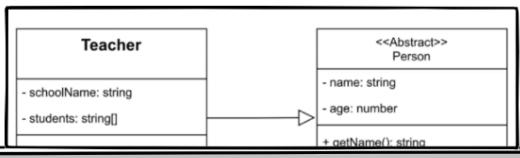
Choose the proper class diagram reflecting the following description:

Some description

Points: 5







Unippets

https://unippets.com/tests/create_new

Create new test

Dashboard Courses Scripts Tests Quizzes Log out

Test title:

Subject:

Description:

Pass percent: 

 Change test image

Proceed Cancel

Unippets

https://unippets.com/tests/create_new

Create new test

Dashboard Courses Scripts Tests Quizes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

```
1 ## Correct this expression using streams
2
3 ``java
4     animals = animals.entrySet().stream()
5         .filter(entry -> entry.getValue().getEnergy() >
0)
6         .collect(Collectors.toList());
7``
```

q![Diagram](https://www.google.com/url?sa=i
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-1
0diagram%2F
psig=AOvVaw3sOGlpNsnPPdal4wVHS2sw
ust=170325117444000source=images11cd=vfe
opi=89978449&ved=0CBEQjRxqFwoTCPxxyoHPoIMD
FQAAAAAAdAAAAABAD)

Markdown input

Answear type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Select programming language:

Type in code:

Select line in which to insert the executable code:

Correct this lambda expression

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        ().getEnergy() > 0)
    .collect(Collectors.toList());
```

Preview

Question 2

Unippets

https://unippets.com/tests/create_new

Dashboard Courses Scripts Tests Quizes Log out

Create new test

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

Correct this expression using streams

```
``java
animals = animals.entrySet().stream()
.filter(entry -> entry.getValue().getEnergy() >
0)
.collect(Collectors.toList());
``
```

![[Diagram](https://www.google.com/url?sa=i
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-di

Markdown input Preview

Answer type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Answers - select correct ones by marking them

``java
animals = animals.entrySet().stream()
.filter(entry -> entry.getValue().getEnergy()
> 0)
.collect(Collectors.toList());

```
animals = animals.entrySet().stream()
.filter(entry -> entry.getValue()
().getEnergy() > 0)
.collect(Collectors.toList());
```

Markdown input Preview

``java
animals = animals.entrySet().stream()
.filter(entry -> entry.getValue().getEnergy() > 0)
.collect(Collectors.toList(
entry -> entry.getKey(),
entry -> entry.getValue()
``

```
animals = animals.entrySet().stream()
.filter(entry -> entry.getValue()
().getEnergy() > 0)
.collect(Collectors.toList(
entry -> entry.getKey(),
entry -> entry.getValue()
``
```

Markdown input Preview

New answer...

Markdown input Preview

Question 2

Unippets

https://unippets.com/tests/create_new

Create new test

Preview

Test title: Lambda expressions and streams in java

Subject: Object Oriented Programming in Java

Description: Check students' knowledge about streams and lambdas

Pass percent: 50%

Number of questions: 12

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        ()>0)
    .collect(Collectors.toList());
```

Submit answer

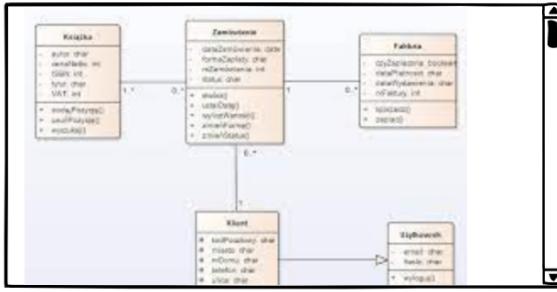
Markdown input Preview

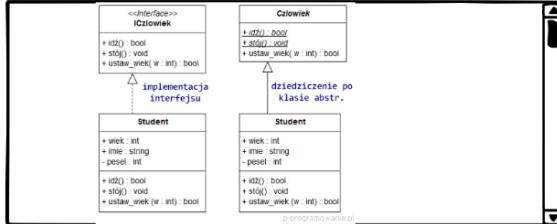
Question 2

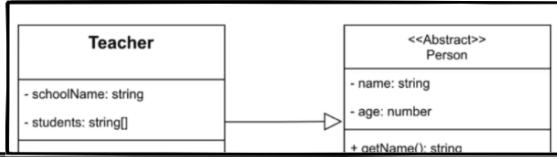
Choose the proper class diagram reflecting the following description:

Some description

Points: 5







Unippets

https://unippets.com/tests/create_new

Create new test

Dashboard Courses Scripts Tests Quizes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

Correct this expression using streams

```
``java
    animals = animals.entrySet().stream()
        .filter(entry -> entry.getValue().getEnergy() >
0)
        .collect(Collectors.toList());
``
```

!Diagram](https://www.google.com/url?sa=i
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-di

Markdown input Preview

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
    .getEnergy() > 0)
    .collect(Collectors.toList());
```

Answer type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Answer preview:

Markdown input Preview

Add executable field: Yes Language: Java 9.0

Unippets

https://unippets.com/tests/create_new

Create new test

Dashboard Courses Scripts Tests Quizzes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

Correct this expression using streams

```
``java
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue().getEnergy() >
0)
    .collect(Collectors.toList());
``
```

![[Diagram](https://www.google.com/url?sa=i
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-diagram%2FStream-Transformation-and-Filtering-with-Java-Lambda-Expressions))

Markdown input

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        .getEnergy() > 0)
    .collect(Collectors.toList());
```

Preview

Answer type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Answers - select the correct one by marking it

``java
`animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy()
> 0)
 .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

Markdown input

Preview

``java
`animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy()
> 0)
 .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

Markdown input

Preview

New answer...

Markdown input

Preview

Unippets

https://unippets.com/tests/create_new

Create new test

Preview

Test title: Lambda expressions and streams in java
 Subject: Object Oriented Programming in Java
 Description: Check students' knowledge about streams and lambdas
 Pass percent: 50%
 Number of questions: 12

Dashboard Courses Scripts Tests Quizzes Log out

Back Save Test image

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        (.getEnergy() > 0)
    .collect(Collectors.toList());
```

Points: 10

Submit answer

Write code... Preview

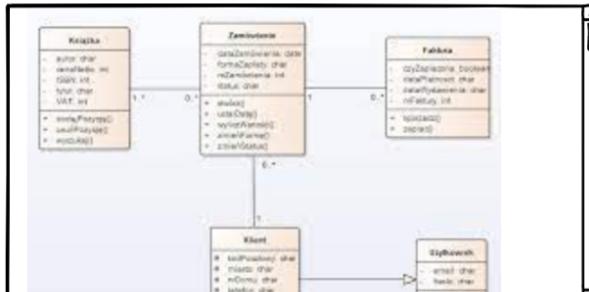
Markdown input

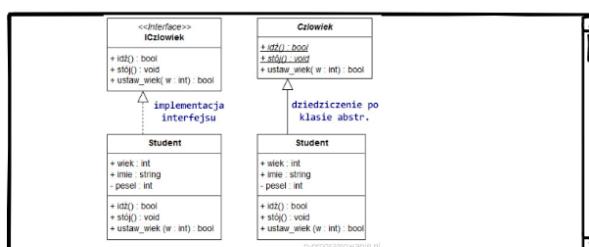
Question 2

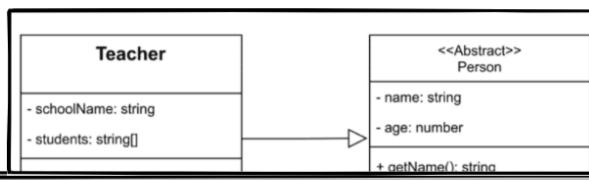
Choose the proper class diagram reflecting the following description:

Some description

Points: 5

○ 

○ 

○ 

Unippets

https://unippets.com/tests/edit

Edit test

Dashboard Courses Scripts Tests Quizzes Log out

Test title: Lambda expressions and streams

Subject: Object Oriented Programming

Description: Some description

Pass percent: 50%

Change test image

Proceed Cancel

The screenshot displays a web-based application interface for editing a test. At the top, there's a header bar with navigation icons (back, forward, search, etc.) and the URL 'https://unippets.com/tests/edit'. Below the header, the title 'Edit test' is prominently displayed. A navigation menu includes 'Dashboard', 'Courses', 'Scripts', 'Tests' (which is highlighted in blue), 'Quizzes', and 'Log out'. The main content area contains four input fields: 'Test title' with the value 'Lambda expressions and streams', 'Subject' with 'Object Oriented Programming', 'Description' with 'Some description', and 'Pass percent' set to '50%' with an up/down arrow. To the right of these fields is a square placeholder image containing a large 'X', with a 'Change test image' button next to it. At the bottom of the form are two buttons: 'Proceed' (blue) and 'Cancel' (red). The entire page has a clean, modern design with a light gray background.

Unippets

https://unippets.com/tests/create_new

Edit Test

Dashboard Courses Scripts Tests Quizzes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

```
1## Correct this expression using streams
2
3``java
4    animals = animals.entrySet().stream()
5        .filter(entry -> entry.getValue().getEnergy() >
0)
6        .collect(Collectors.toList());
7``
```

8
[!Diagram\]\(https://www.google.com/url?sa=i&url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-10diagram%2Fpsig=AOvVaw3sOGIpNsnPPdal4wVHS2sw_ust=170325117444000source=images11cd=vfe_opi=89978449&ved=0CBEQjRxqFwoTCPChyoHPoIMDFQAAAAAdAAAAABAD\)](#)
 Markdown input

Correct this lambda expression

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        .getEnergy() > 0)
    .collect(Collectors.toList());
```

Preview

The diagram illustrates the execution flow of the code. It starts with a Stream input, which is then processed by a Data block. Finally, the output goes to a System.out.println("Hello World") block.

Answer type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Select programming language:

Type in code:

Select line in which to insert the executable code:

Question 2

Unippets

https://unippets.com/tests/edit

Edit Test

Dashboard Courses Scripts Tests Quizes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

Correct this expression using streams

```
``java
    animals = animals.entrySet().stream()
        .filter(entry -> entry.getValue().getEnergy() >
0)
    .collect(Collectors.toList());
``
```

!Diagram](https://www.google.com/url?sa=i
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-di

Markdown input Preview

Answer type: Multiple choice Single choice Open question

Points:

Generate executable code snippet:

Answer preview:

Markdown input Preview

Add executable field:

Unippets

https://unippets.com/tests/edit

Edit Test

Dashboard Courses Scripts Tests Quizes Log out

Preview

Test title: Lambda expressions and streams in java
Subject: Object Oriented Programming in Java
Description: Check students' knowledge about streams and lambdas
Pass percent: 50%
Number of questions: 12

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
    .filter(entry -> entry.getValue()
        ().getEnergy() > 0)
    .collect(Collectors.toList());
```

Diagram illustrating the flow of data from a **Dataset** to an **External Reservation System** through a **Booking** step.

Points: 10

Markdown input Preview

Question 2

Choose the proper class diagram reflecting the following description:

Unippets

<https://unippets.com/tests/edit>

Edit Test

Dashboard Courses Scripts Tests Quizes Log out

Test title: Lambda expressions and streams in java

Back Proceed

Question 1

Correct this expression using streams

```
``java
    animals = animals.entrySet().stream()
        .filter(entry -> entry.getValue().getEnergy() >
0)
        .collect(Collectors.toList());
```


![Diagram](https://www.google.com/url?sa=i&url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-diagram)

Markdown input Preview

Answer type: Multiple choice Single choice Open question

Points: 5

Generate executable code snippet:

Answers - select correct ones by marking them

``java
 animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy()
> 0)
 .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```


Markdown input Preview



``java
    animals = animals.entrySet().stream()
        .filter(entry -> entry.getValue().getEnergy() >
0)
        .collect(Collectors.toMap(
            entry -> entry.getKey(),
            entry -> entry.getValue()
```


Markdown input Preview

New answer...

Markdown input Preview

Question 2


```

Unippets

<https://unippets.com/tests/edit>

# Edit Test

Dashboard Courses Scripts Tests Quizzes Log out

Test title: Lambda expressions and streams in java

Back Proceed

## Question 1

## Correct this expression using streams

```
``java
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy() >
0)
 .collect(Collectors.toList());
``
```

!Diagram](https://www.google.com/url?sa=i  
url=https%3A%2F%2Fmiro.com%2Fblog%2Fcontext-di

Markdown input Preview

Answer type:  Multiple choice  Single choice  Open question

Points: 3

Generate executable code snippet:

Answers - select the correct one by marking it

``java
`animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy()
> 0)
 .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

Markdown input Preview

``java
`animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue().getEnergy()
> 0)
 .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

Markdown input Preview

New answer...

Markdown input Preview

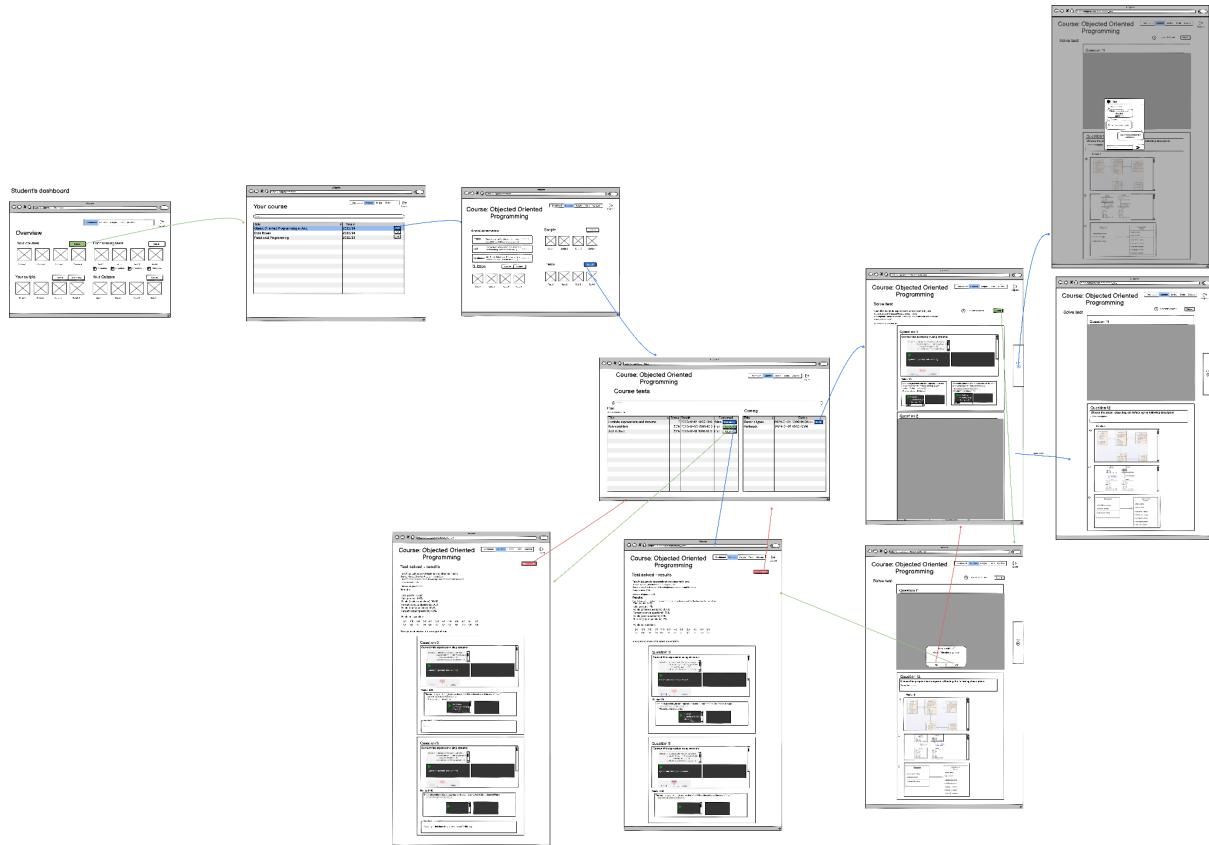
Correct this expression using streams

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 .getEnergy() > 0)
 .collect(Collectors.toList());
```

The diagram illustrates the process of generating code. It starts with an 'Output' box at the bottom, which points up to a 'Stream' box. From the 'Stream' box, an arrow points up to a 'Generated' box, which contains the corrected Java code.

### 3.4.2. Sprawdziany z perspektywy studenta

- wireflow ([PDF z schematami w wysokiej jakości](#))



- Powiększenie wireframe'ów

# Student's dashboard

The screenshot shows a web-based student dashboard titled "Unippets". At the top, there are navigation icons (back, forward, search, etc.) and a URL bar containing <https://unippets.com/dashboard>. Below the header is a navigation menu with tabs for "Dashboard" (which is highlighted in blue), "Courses", "Scripts", "Tests", and "Quizzes". To the right of the menu are a "Log out" button and a search icon.

## Overview

**Your courses**

|                         |          |          |          |
|-------------------------|----------|----------|----------|
|                         |          |          |          |
| Course 1                | Course 2 | Course 3 | Course 4 |
| <a href="#">See all</a> |          |          |          |

**Forthcoming tests**

|            |            |            |            |
|------------|------------|------------|------------|
|            |            |            |            |
| Test 1     | Test 2     | Test 3     | Test 4     |
| 2024-01-04 | 2024-01-04 | 2024-01-04 | 2024-01-04 |

**Your scripts**

|                                                    |          |          |          |
|----------------------------------------------------|----------|----------|----------|
|                                                    |          |          |          |
| Script 1                                           | Script 2 | Script 3 | Script 4 |
| <a href="#">See all</a> <a href="#">Create New</a> |          |          |          |

**Your Quizzes**

|                         |        |        |        |
|-------------------------|--------|--------|--------|
|                         |        |        |        |
| Quiz 1                  | Quiz 2 | Quiz 3 | Quiz 4 |
| <a href="#">See all</a> |        |        |        |



Unippets

<https://unippets.com/tests>

Course: Objected Oriented Programming

Dashboard Courses Scripts Tests Quizzes Log out

### Announcements

- Project** Dear Students, I'd like to inform you that you should prepare a project... 2023-12-20
- Lab8** Please finish exercises 1 and 4 for our next meeting. Next time we will(...) 2023-12-17
- My Absence** On 15th of December I'm attending a conference and I will be absent (...) 2023-12-10

### Scripts

See all

Script 1 Script 2 Script 3 Script 4

### Quizzes

See all

Quiz 1 Quiz 2 Quiz 3 Quiz 4

### Tests

See all

Test 1 Test 2 Test 3 Test 4

Unippets

<https://unippets.com/tests>

Course: Objected Oriented Programming

Dashboard Courses Scripts Tests Quizzes Log out

### Course tests

search

Post Whole course: 73%

| Title                          | Date | Result                | Evaluated |
|--------------------------------|------|-----------------------|-----------|
| Lambda expressions and streams | ?    | 2023-12-12 13:00-14:3 | false     |
| Polymorphism                   | 70%  | 2023-12-05 15:00-16:0 | true      |
| GUI in Java                    | 75%  | 2023-12-01 15:00-16:3 | true      |

Coming

| Title         | Date                       |
|---------------|----------------------------|
| Generic types | 2024-01-04 13:00-14:30 NOW |
| Optionals     | 2024-01-07 15:00-16:00     |

Unippets

https://unippets.com/tests/create\_new

# Course: Objected Oriented Programming

Solve test

Test title: Lambda expressions and streams in java  
Subject: Object Oriented Programming in Java  
Description: Check students' knowledge about streams and lambdas  
Pass percent: 50%

Number of questions: 12

Time left: 01:03:44

Submit

Log out

Question 1

Correct this expression using streams

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 .getEnergy() > 0)
 .collect(Collectors.toList());
```

System.out.println("Hello World");

Points: 10

The main problem with this expression is that it is collected to List instead of Map ("java.collect(Collectors.toList());")  
The correction should be:

```
animals = animals.entrySet()
 .stream()
```

The main problem with this expression is that it is collected to List instead of Map ("collect(Collectors.toList());").  
The correction should be:

```
animals = animals.entrySet()
 .stream()
```

Question 2

Unippets

[https://unippets.com/tests/create\\_new](https://unippets.com/tests/create_new)

Course: Objected Oriented Programming

Solve test

Question 11

Chat

Student 1234: What should we do in the 2nd question?

Teacher: Write the proper code

Me: How to understand this sentence:

following description:

Ask a question... ➤

Points: 5

①

```

classDiagram
 class Klient {
 -id: string
 -imie: string
 -nazwisko: string
 -ulazek: string
 -wylazek: string
 +zaloguj()
 +wyslij_email()
 +wyslij_wiadomosc()
 +zaloguj()
 }
 class Zamek {
 -id: string
 -lokalizacja: string
 -firmaZakupy: string
 -status: string
 +zamknij()
 +otwrij()
 +wyslij_email()
 +wyslij_wiadomosc()
 +zamknij()
 }
 class Falka {
 -lokalizacja_boss: string
 -lokalizacja_kierowcy: string
 -status: string
 +zamknij()
 +otwrij()
 }
 class Kierowca {
 -email: string
 -haslo: string
 +wyslij_email()
 +wyslij_wiadomosc()
 }
 Klient "1..>" Zamek
 Klient "1..>" Falka
 Zamek "1..>" Falka
 Klient "1..>" Kierowca
 Klient "1..>" Kierowca

```

②

```

classDiagram
 class Czlowiek {
 -imie: string
 -nazwisko: string
 -ulazek: string
 -wylazek: string
 +zaloguj()
 +wyslij_email()
 +wyslij_wiadomosc()
 }
 class Student {
 -imie: string
 -nazwisko: string
 -ulazek: string
 -wylazek: string
 +zaloguj()
 +wyslij_email()
 +wyslij_wiadomosc()
 }
 class Person {
 -imie: string
 -nazwisko: string
 -ulazek: string
 -wylazek: string
 +zaloguj()
 +wyslij_email()
 +wyslij_wiadomosc()
 }
 Student "1..>" Czlowiek
 Student "1..>" Person
 Czlowiek "1..>" Person

```

③

```

classDiagram
 class Teacher {
 -schoolName: string
 -students: string[]
 +getSchool(): string
 ...
 }
 class Person {
 -name: string
 -age: number
 +getName(): string
 +setName(): string
 +getAge(): number
 +setAge(): number
 +isAdult(): boolean
 }
 Teacher --> Person

```

Unippets

[https://unippets.com/tests/create\\_new](https://unippets.com/tests/create_new)

Course: Objected Oriented Programming

Solve test

Question 11

Time left: 01:03:44

Submit

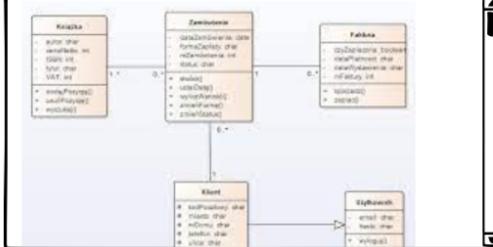
Chat

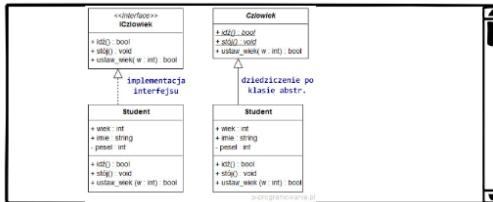
Question 12

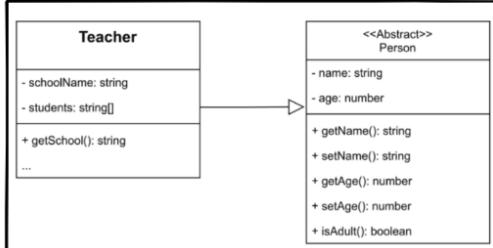
Choose the proper class diagram reflecting the following description:

Some description

Points: 5







Unippets

[https://unippets.com/tests/create\\_new](https://unippets.com/tests/create_new)

# Course: Objected Oriented Programming

Dashboard Courses Scripts Tests Quizzes Log out

## Test solved - results

Test title: Lambda expressions and streams in java  
 Subject: Object Oriented Programming in Java  
 Description: Check students' knowledge about streams and lambdas  
 Pass percent: 50%

Number of questions: 12

**Results:**  
 Result does not include your score from open questions - wait for the teacher to check them  
 Total points: ?/68  
 Total percent: ??%  
 Points (choice questions): 35/50  
 Percent (choice questions): 70%  
 Points (open questions): ?/18  
 Percent (open questions): ??%

Points per question:

|     |     |     |     |      |     |     |     |     |     |     |     |
|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|
| 2/3 | 5/5 | ?/8 | 3/3 | ?/10 | 2/3 | 4/5 | 1/8 | 2/6 | 1/3 | 3/4 | 2/3 |
| Q1  | Q2  | Q3  | Q4  | Q5   | Q6  | Q7  | Q8  | Q9  | Q10 | Q11 | Q12 |

See your answers to open questions

**Question 3**

**Correct this expression using streams**

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 ().getEnergy() > 0)
 .collect(Collectors.toList());
```

System.out.println("Hello World");

**Points: ?/8**

The main problem with this expression is that it is collected to List instead of Map (.collect(Collectors.toList()); ). The correction should be:

```
animals = animals.entrySet()
 .stream()
```

**Question 5**

**Correct this expression using streams**

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 ().getEnergy() > 0)
 .collect(Collectors.toList());
```

System.out.println("Hello World");

**Points: ?/10**

The main problem with this expression is that it is collected to List instead of Map (.collect(Collectors.toList()); )

Unippets [https://unippets.com/tests/create\\_new](https://unippets.com/tests/create_new)

# Course: Objected Oriented Programming

Dashboard Courses Scripts Tests Quizzes Log out

## Test solved - results

Test title: Lambda expressions and streams in java  
 Subject: Object Oriented Programming in Java  
 Description: Check students' knowledge about streams and lambdas  
 Pass percent: 50%

Number of questions: 12  
**Results:**

Total points: 45/68  
 Total percent: 65%  
 Points (choice questions): 35/50  
 Percent (choice questions): 70%  
 Points (open questions): 10/18  
 Percent (open questions): 60%

Points per question:

|     |     |     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 2/3 | 5/5 | 8/8 | 3/3 | 2/1 | 2/3 | 4/5 | 1/8 | 2/6 | 1/3 | 3/4 | 2/3 |
| Q1  | Q2  | Q3  | Q4  | Q5  | Q6  | Q7  | Q8  | Q9  | Q10 | Q11 | Q12 |

See your answers to open questions

**Question 3**

**Correct this expression using streams**

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 ()>0)
 .collect(Collectors.toList());
```

System.out.println("Hello World");

**Points: 8/8**

The main problem with this expression is that it is collected to List instead of Map ( .collect(Collectors.toList()); ).  
 The correction should be:

```
animals = animals.entrySet()
 .stream()
 .filter(entry -> entry.getValue()
 ()>0)
 .collect(Collectors.toMap());
```

Teacher's comment \_\_\_\_\_

None

**Question 5**

**Correct this expression using streams**

```
animals = animals.entrySet().stream()
 .filter(entry -> entry.getValue()
 ()>0)
 .collect(Collectors.toList());
```

System.out.println("Hello World");

**Points: 2/10**

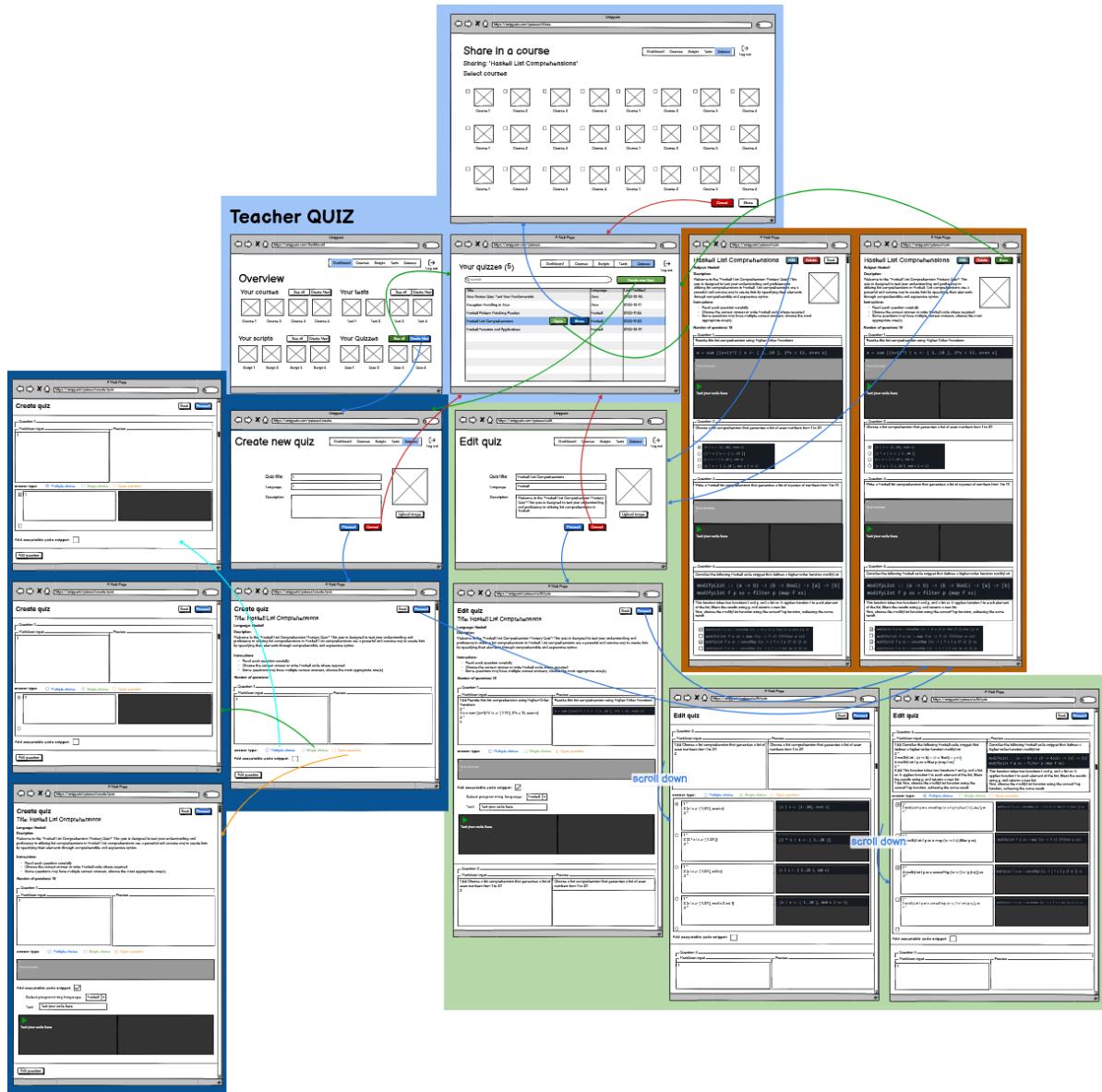
The main problem with this expression is that it is collected to List instead of Map ( .collect(Collectors.toList()); ).

Teacher's comment \_\_\_\_\_

You're right, but there is no answer how to fix this bug

### 3.4.3. Quizy z perspektywy wykładowcy

- wireflow ([PDF z schematami w wysokiej jakości](#))



- Powiększenie wireframe'ów

Unippets

https://unippets.com/dashboard

Dashboard Courses Scripts Tests Quizzes Log out

## Overview

Your courses

See all Create New

Course 1 Course 2 Course 3 Course 4

Your tests

See all Create New

Test 1 Test 2 Test 3 Test 4

Your scripts

See all Create New

Script 1 Script 2 Script 3 Script 4

Your Quizzes

See all Create New

Quiz 1 Quiz 2 Quiz 3 Quiz 4

This screenshot shows the Unippets dashboard. At the top, there are navigation icons and a URL bar. Below that is a header with tabs for Dashboard, Courses, Scripts, Tests, Quizzes, and Log out. The main area is titled 'Overview' and contains four sections: 'Your courses' (Course 1, Course 2, Course 3, Course 4), 'Your tests' (Test 1, Test 2, Test 3, Test 4), 'Your scripts' (Script 1, Script 2, Script 3, Script 4), and 'Your Quizzes' (Quiz 1, Quiz 2, Quiz 3, Quiz 4). Each section has 'See all' and 'Create New' buttons.

A Web Page

https://unippets.com/quizzes

double-click to edit

Dashboard Courses Scripts Tests Quizzes Log out

Your quizzes (5)

search

Create new Quiz

| Title                                    | Language           | Last Modified |
|------------------------------------------|--------------------|---------------|
| Java Basics Quiz: Test Your Fundamentals | Java               | 2023-12-04    |
| Exception Handling in Java               | Java               | 2023-12-11    |
| Haskell Pattern Matching Puzzles         | Haskell            | 2023-11-24    |
| Haskell List Comprehensions              | Open Share Haskell | 2023-11-22    |
| Haskell Functors and Applicatives        | Haskell            | 2023-12-01    |

This screenshot shows the Unippets quizzes page. At the top, there are navigation icons and a URL bar. Below that is a header with tabs for Dashboard, Courses, Scripts, Tests, Quizzes, and Log out. A search bar and a 'Create new Quiz' button are also present. The main area is titled 'Your quizzes (5)' and displays a table of quizzes. The columns are 'Title', 'Language', and 'Last Modified'. The table rows are: 'Java Basics Quiz: Test Your Fundamentals' (Java, 2023-12-04), 'Exception Handling in Java' (Java, 2023-12-11), 'Haskell Pattern Matching Puzzles' (Haskell, 2023-11-24), 'Haskell List Comprehensions' (Haskell, 2023-11-22), and 'Haskell Functors and Applicatives' (Haskell, 2023-12-01). The 'Haskell List Comprehensions' row has 'Open' and 'Share' buttons.

A Web Page  
 https://unippets/quizzes/quiz

## Haskell List Comprehensions

**Language:** Haskell

**Description:**  
 Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

**Instructions:**

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

**Number of questions: 15**

**Question 1**  
 Rewrite this list comprehension using Higher-Order Functions

```
a = sum [(x+1)^3 | x <- [1..10], 2*x < 13, even x]
```

Your answer

Test your code here

**Question 2**  
 Choose a list comprehension that generates a list of even numbers from 1 to 20

[x | x <- [1..20], even x]  
 [2 \* x | x <- [ 1..20 ]]  
 [x | x <- [ 1..20 ], odd x]  
 [x | x <- [ 1..20 ], mod x 2 == 1]

**Question 3**  
 Write a Haskell list comprehension that generates a list of squares of numbers from 1 to 10.

Your answer

Test your code here

**Question 4**  
 Consider the following Haskell code snippet that defines a higher-order function modifyList

```
modifyList :: (a -> b) -> (b -> Bool) -> [a] -> [b]
modifyList f p xs = filter p (map f xs)
```

This function takes two functions f and p, and a list xs. It applies function f to each element of the list, filters the results using p, and returns a new list.  
 Now, choose the modifyList function using the concatMap function, achieving the same result.

modifyList f p xs = concatMap (\x -> if p (f x) then [f x] else []) xs  
 modifyList f p xs = map (\x -> f x) (filter p xs)  
 modifyList f p xs = concatMap (\x -> [ f x | p (f x) ]) xs  
 modifyList f p xs = concatMap (\x -> [ f x | not (p x) ]) xs

Unippets

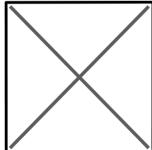
[Dashboard](#) [Courses](#) [Scripts](#) [Tests](#) [Quizzes](#) [Log out](#)

Create new quiz

Quiz title:

Language:

Description:



A Web Page

[Back](#) [Proceed](#)

Create quiz

**Title:** Haskell List Comprehensions

**Language:** Haskell

**Description:**

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

**Instructions:**

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

**Number of questions:** 1

Question 1

Markdown input

Preview

**answer type:**  Multiple choice  Single choice  Open question

Add executable code snippet:

A Web Page  
<https://unippets/quizzes/create/quiz>

## Create quiz

Question 1

Markdown input

Preview

1

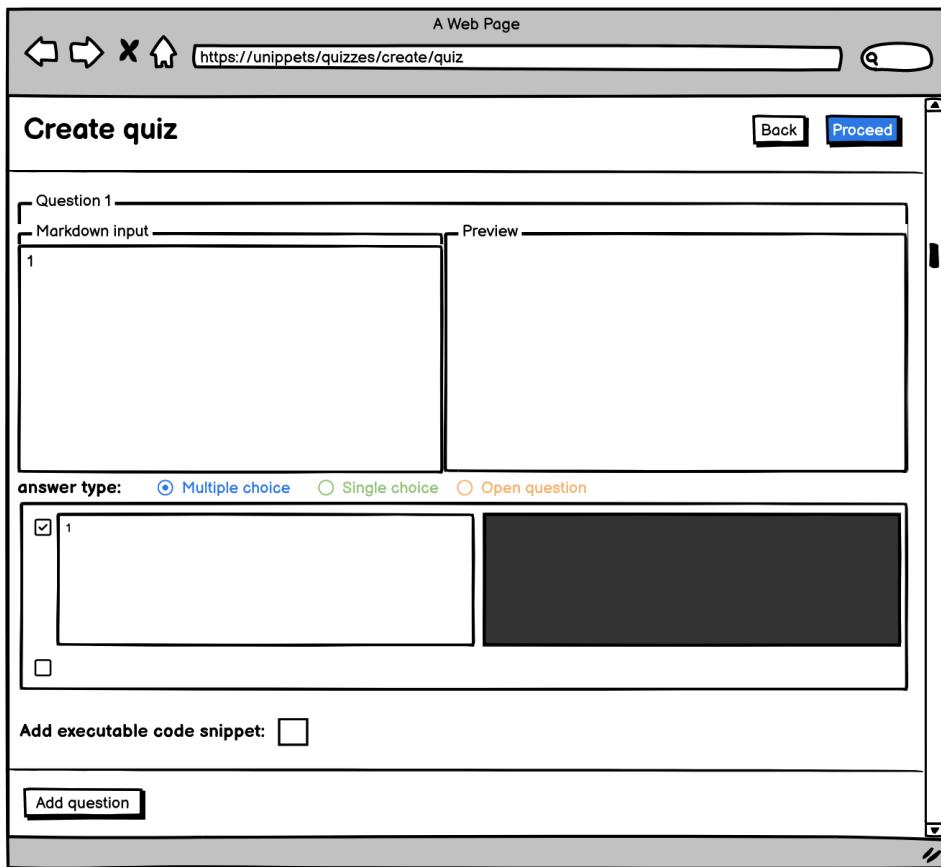
answer type:  Multiple choice  Single choice  Open question

1

Add executable code snippet:

Add question

Back Proceed



This screenshot shows a web-based quiz creation tool. At the top, there's a header with navigation icons (back, forward, search) and a URL bar showing the address. Below the header is a title 'Create quiz'. A question card for 'Question 1' is displayed, containing a 'Markdown input' field with the number '1' and a 'Preview' field. Below the question card are answer type selection buttons: 'Multiple choice' (selected), 'Single choice', and 'Open question'. Underneath these buttons is a list of options: '1' (with a checked radio button) and an empty square checkbox. There's also a checkbox for adding executable code snippets. At the bottom of the page are 'Back' and 'Proceed' buttons.

A Web Page  
<https://unippets/quizzes/create/quiz>

## Create quiz

Question 1

Markdown input

Preview

1

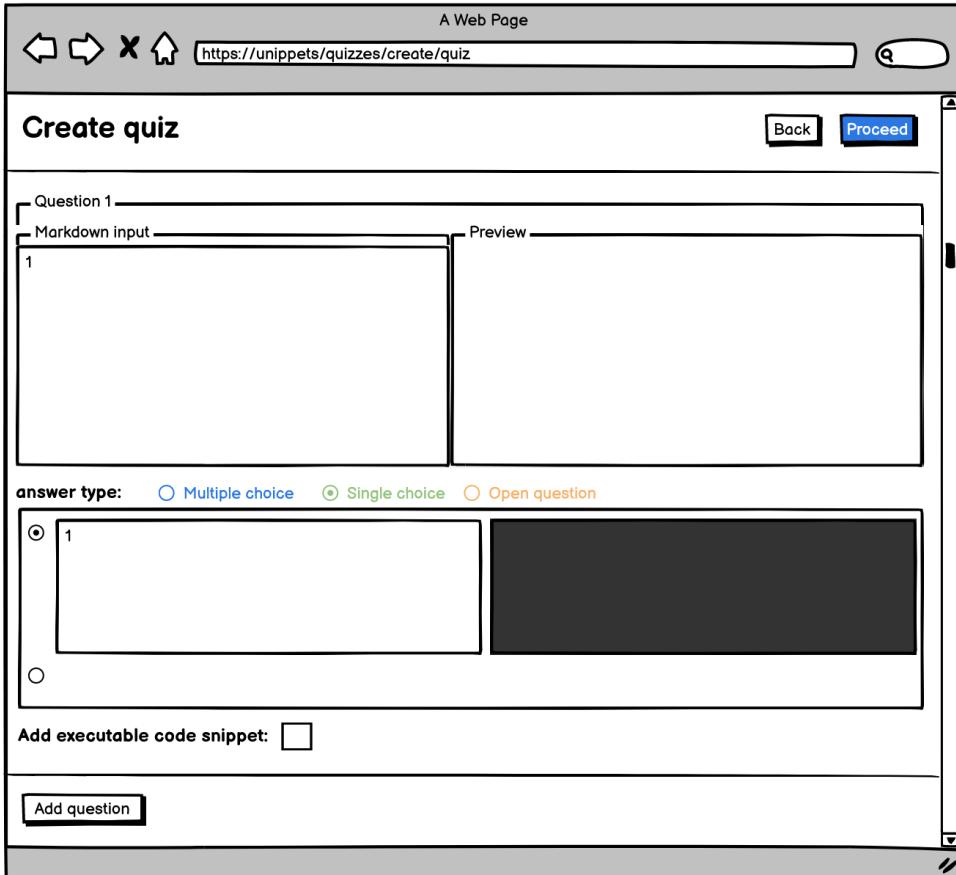
answer type:  Multiple choice  Single choice  Open question

1

Add executable code snippet:

Add question

Back Proceed



This screenshot is identical to the one above it, except for the 'answer type' selection. In this version, 'Single choice' is selected instead of 'Multiple choice'. The rest of the interface, including the question text, preview, options list, and footer buttons, remains the same.

A Web Page  
<https://unippets/quizzes/create/quiz>

## Create quiz

**Title:** Haskell List Comprehensions

**Language:** Haskell

**Description:**  
Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

**Instructions:**

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

**Number of questions:** 15

Question 1

Markdown input

Preview

1

**answer type:**  Multiple choice  Single choice  Open question

Your answer

Add executable code snippet:

Select programming language: Haskell

Text: Test your code here

Test your code here

Add question

Unippets

https://unippets.com/quizzes/edit

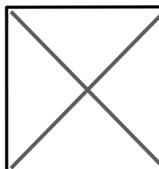
## Edit quiz

Dashboard Courses Scripts Tests Quizzes Log out

Quiz title: Haskell List Comprehensions

Language: Haskell

Description: Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell.



Upload image

Proceed Cancel

A Web Page

https://unippets/quizzes/edit/quiz

## Edit quiz

Title: Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Instructions:

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

Number of questions: 15

Question 1

Markdown input

```
1 ## Rewrite this list comprehension using Higher-Order Functions
2 ""
3 a = sum [(x+1)^3 | x <- [1..10], 2*x < 13, even x]
4 ""
5
```

Preview

```
Rewrite this list comprehension using Higher-Order Functions
a = sum [(x+1)^3 | x <- [1..10], 2*x < 13, even x]
```

answer type:  Multiple choice  Single choice  Open question

Your answer

Add executable code snippet:

Select programming language: Haskell

Text: Test your code here

Test your code here

Question 2

Markdown input

```
1 ## Choose a list comprehension that generates a list of even numbers from 1 to 20
2
```

Preview

```
Choose a list comprehension that generates a list of even numbers from 1 to 20
```

A Web Page  
<https://unippets/quizzes/edit/quiz>

## Edit quiz

Question 2

Markdown input

```
1 ## Choose a list comprehension that generates a list of even numbers from 1 to 20
2
3
```

Preview

```
Choose a list comprehension that generates a list of even numbers from 1 to 20
```

answer type:  Multiple choice  Single choice  Open question

|                                                                                                   |                                  |
|---------------------------------------------------------------------------------------------------|----------------------------------|
| <input checked="" type="radio"/> 1 <sup>“</sup><br>2 [x   x <- [1..20], even x]<br>3 <sup>“</sup> | [x   x <- [1..20], even x]       |
| <input type="radio"/> 1 <sup>“</sup><br>2 [2 * x   x <- [1..20]]<br>3 <sup>“</sup>                | [2 * x   x <- [1..20]]           |
| <input type="radio"/> 1 <sup>“</sup><br>2 [x   x <- [1..20], odd x]<br>3 <sup>“</sup>             | [x   x <- [1..20], odd x]        |
| <input type="radio"/> 1 <sup>“</sup><br>2 [x   x <- [1..20], mod x 2 == 1]<br>3 <sup>“</sup>      | [x   x <- [1..20], mod x 2 == 1] |
| <input type="radio"/>                                                                             |                                  |

Add executable code snippet:

Question 3

Markdown input

```
1
```

Preview

A Web Page

<https://unippets/quizzes/edit/quiz>

## Edit quiz

Question 4

Markdown input

```
1 ## Consider the following Haskell code snippet that defines a higher-order function modifyList
2 "
3 modifyList :: (a -> b) -> (b -> Bool) -> a -> b
4 modifyList f p xs = filter p (map f xs)
5 "
6 ## This function takes two functions f and p, and a list xs. It applies function f to each element of the list, filters the results using p, and returns a new list.
7 ## Now, choose the modifyList function using the concatMap function, achieving the same result.
```

Preview

Consider the following Haskell code snippet that defines a higher-order function modifyList

```
modifyList :: (a -> b) -> (b -> Bool) -> [a] -> [b]
modifyList f p xs = filter p (map f xs)
```

This function takes two functions f and p, and a list xs. It applies function f to each element of the list, filters the results using p, and returns a new list.

Now, choose the modifyList function using the concatMap function, achieving the same result.

answer type:  Multiple choice  Single choice  Open question

|                                     |                                                                                 |                                                                        |
|-------------------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> | 1" 2 modifyList f p xs = concatMap (\x -> if p (f x) then [f x] else []) xs 3 " | modifyList f p xs = concatMap (\x -> if p (f x) then [f x] else []) xs |
| <input type="checkbox"/>            | 1" 2 modifyList f p xs = map (\x -> f x) (filter p xs) 3 "                      | modifyList f p xs = map (\x -> f x) (filter p xs)                      |
| <input checked="" type="checkbox"/> | 1" 2 modifyList f p xs = concatMap (\x -> [ f x   p (f x) ]) xs 3 "             | modifyList f p xs = concatMap (\x -> [ f x   p (f x) ]) xs             |
| <input type="checkbox"/>            | 1" 2 modifyList f p xs = concatMap (\x -> [ f x   not (p x) ]) xs 3 "           | modifyList f p xs = concatMap (\x -> [ f x   not (p x) ]) xs           |
| <input type="checkbox"/>            |                                                                                 |                                                                        |

Add executable code snippet:

Question 5

Markdown input

Preview

```
1
```

A Web Page  
 https://unipetts/quizzes/quiz

## Haskell List Comprehensions

**Language:** Haskell

**Description:**

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

**Instructions:**

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

**Number of questions: 15**

**Question 1**

Rewrite this list comprehension using Higher-Order Functions

```
a = sum [(x+1)^3 | x <- [1..10], 2*x < 13, even x]
```

Your answer

 Test your code here

**Question 2**

Choose a list comprehension that generates a list of even numbers from 1 to 20

[x | x <- [1..20], even x]  
 [2 \* x | x <- [1..20]]  
 [x | x <- [1..20], odd x]  
 [x | x <- [1..20], mod x 2 == 1]

**Question 3**

Write a Haskell list comprehension that generates a list of squares of numbers from 1 to 10.

Your answer

 Test your code here

**Question 4**

Consider the following Haskell code snippet that defines a higher-order function modifyList

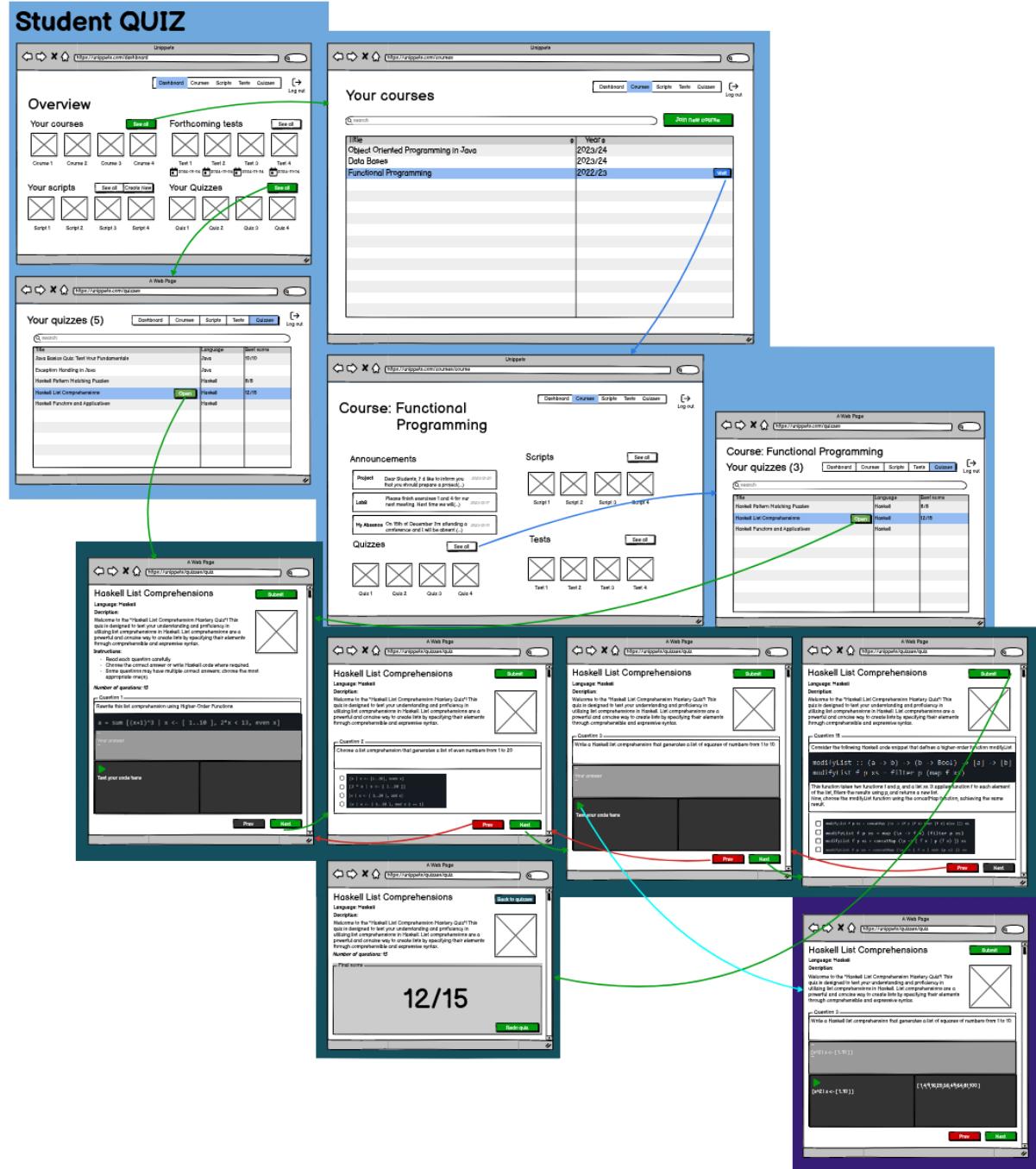
```
modifyList :: (a -> b) -> (b -> Bool) -> [a] -> [b]
modifyList f p xs = filter p (map f xs)
```

This function takes two functions f and p, and a list xs. It applies function f to each element of the list, filters the results using p, and returns a new list.  
 Now, choose the modifyList function using the concatMap function, achieving the same result.

modifyList f p xs = concatMap (\x -> if p (f x) then [f x] else []) xs  
 modifyList f p xs = map (\x -> f x) (filter p xs)  
 modifyList f p xs = concatMap (\x -> [f x | p (f x)]) xs  
 modifyList f p xs = concatMap (\x -> [f x | not (p x)]) xs

### 3.4.4. Quizy z perspektywy studenta

- wireflow ([PDF z schematami w wysokiej jakości](#))



- Powiększenie wireframe'ów

Unippets

<https://unippets.com/dashboard>

Dashboard Courses Scripts Tests Quizzes Log out

## Overview

Your courses

Course 1 Course 2 Course 3 Course 4 See all

Course 1 Course 2 Course 3 Course 4

Forthcoming tests

Test 1 Test 2 Test 3 Test 4 See all

Test 1 Test 2 Test 3 Test 4

2024-01-04 2024-01-04 2024-01-04 2024-01-04

Your scripts

Script 1 Script 2 Script 3 Script 4 See all Create New

Script 1 Script 2 Script 3 Script 4

Your Quizzes

Quiz 1 Quiz 2 Quiz 3 Quiz 4 See all

Quiz 1 Quiz 2 Quiz 3 Quiz 4

A Web Page

<https://unippets.com/quizzes>

Dashboard Courses Scripts Tests Quizzes Log out

## Your quizzes (5)

search

| Title                                    | Language | Best score |
|------------------------------------------|----------|------------|
| Java Basics Quiz: Test Your Fundamentals | Java     | 10/10      |
| Exception Handling in Java               | Java     |            |
| Haskell Pattern Matching Puzzles         | Haskell  | 8/8        |
| Haskell List Comprehensions              | Haskell  | 12/15      |
| Haskell Functors and Applicatives        | Haskell  |            |

Unippets

https://unippets.com/courses/course

# Course: Functional Programming

Dashboard Courses Scripts Tests Quizzes Log out

## Announcements

**Project** Dear Students, I'd like to inform you that you should prepare a project(...) 2023-12-20

**Lab8** Please finish exercises 1 and 4 for our next meeting. Next time we will(...) 2023-12-17

**My Absence** On 15th of December I'm attending a conference and I will be absent (...) 2023-12-10

## Scripts

See all

 Script 1    Script 2    Script 3    Script 4

## Quizzes

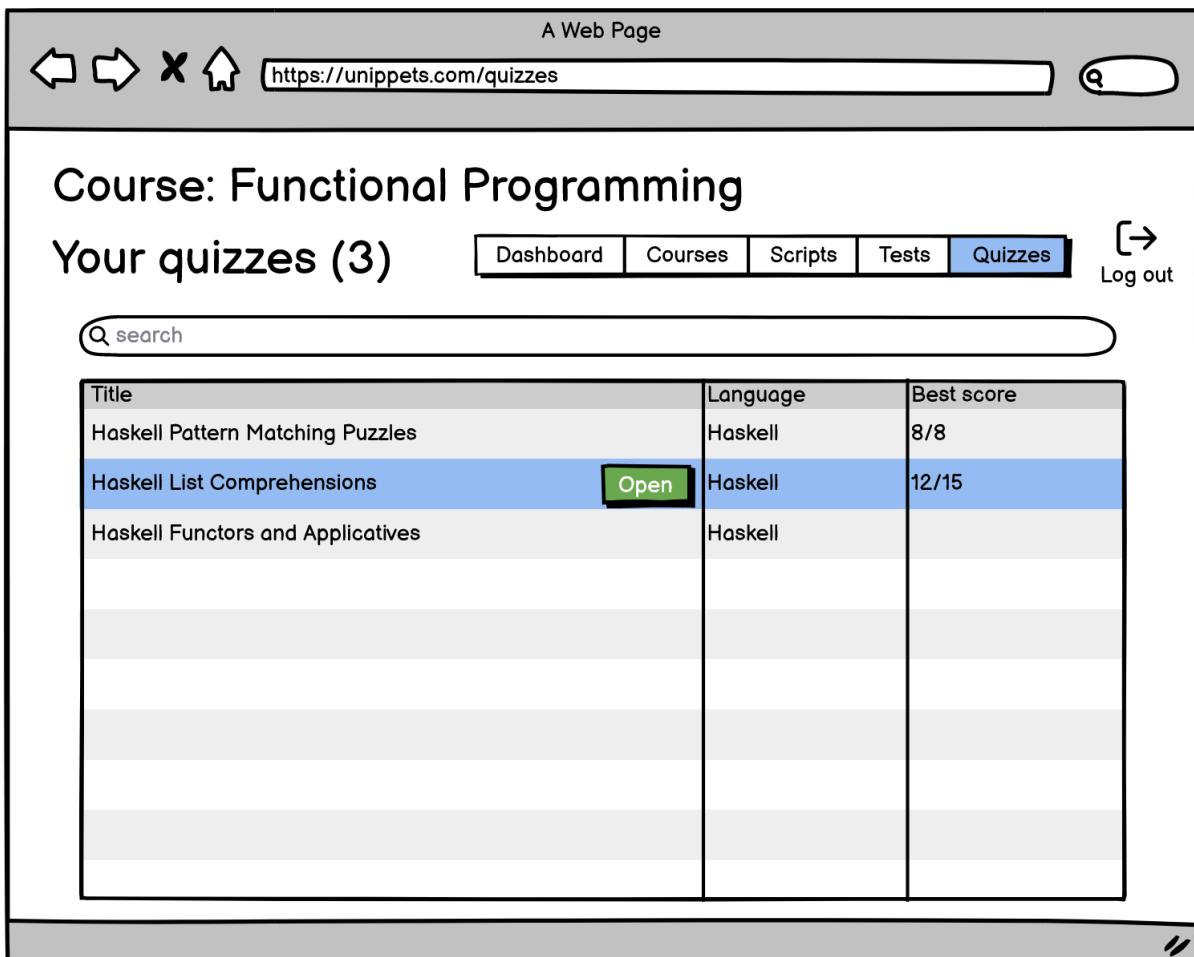
See all

 Quiz 1    Quiz 2    Quiz 3    Quiz 4

## Tests

See all

 Test 1    Test 2    Test 3    Test 4



A Web Page

https://unippets/quizzes/quiz

# Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Instructions:

- Read each question carefully.
- Choose the correct answer or write Haskell code where required.
- Some questions may have multiple correct answers; choose the most appropriate one(s).

Number of questions: 15

Question 1

Rewrite this list comprehension using Higher-Order Functions

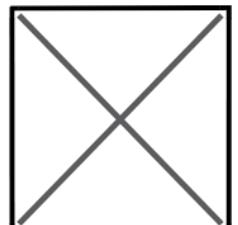
```
a = sum [(x+1)^3 | x <- [1..10], 2*x < 13, even x]
```

Your answer

Test your code here

Prev

Next



A Web Page

https://unippets/quizzes/quiz

# Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Submit

Question 2

Choose a list comprehension that generates a list of even numbers from 1 to 20

- [x | x <- [1..20], even x]
- [2 \* x | x <- [ 1..20 ]]
- [x | x <- [ 1..20 ], odd x]
- [x | x <- [ 1..20 ], mod x 2 == 1]

Prev

Next

A Web Page

https://unippets/quizzes/quiz

## Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Question 3

Write a Haskell list comprehension that generates a list of squares of numbers from 1 to 10.

Your answer

Test your code here

Submit

Prev Next

A Web Page

https://unippets/quizzes/quiz

## Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Question 3

Write a Haskell list comprehension that generates a list of squares of numbers from 1 to 10.

[ $x^2 \mid x <- [ 1..10 ]$ ]

[ $x^2 \mid x <- [ 1..10 ]$ ] [ 1,4,9,16,25,36,49,64,81,100 ]

Submit

Prev Next

A Web Page

https://unippets/quizzes/quiz

## Haskell List Comprehensions

Language: Haskell

Description:

Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Submit

Question 15

Consider the following Haskell code snippet that defines a higher-order function modifyList

```
modifyList :: (a -> b) -> (b -> Bool) -> [a] -> [b]
modifyList f p xs = filter p (map f xs)
```

This function takes two functions f and p, and a list xs. It applies function f to each element of the list, filters the results using p, and returns a new list.  
Now, choose the modifyList function using the concatMap function, achieving the same result.

`modifyList f p xs = concatMap (\x -> if p (f x) then [f x] else []) xs`

`modifyList f p xs = map (\x -> f x) (filter p xs)`

`modifyList f p xs = concatMap (\x -> [ f x | p (f x) ]) xs`

`modifyList f p xs = concatMap (\x -> [ f x | not (p x) ]) xs`

Prev

Next

A Web Page

← → X ⌂ https://unippets/quizzes/quiz

## Haskell List Comprehensions

Language: Haskell

Description:

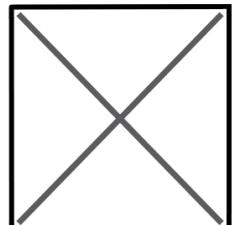
Welcome to the "Haskell List Comprehension Mastery Quiz"! This quiz is designed to test your understanding and proficiency in utilizing list comprehensions in Haskell. List comprehensions are a powerful and concise way to create lists by specifying their elements through comprehensible and expressive syntax.

Number of questions: 15

Final score —

12/15

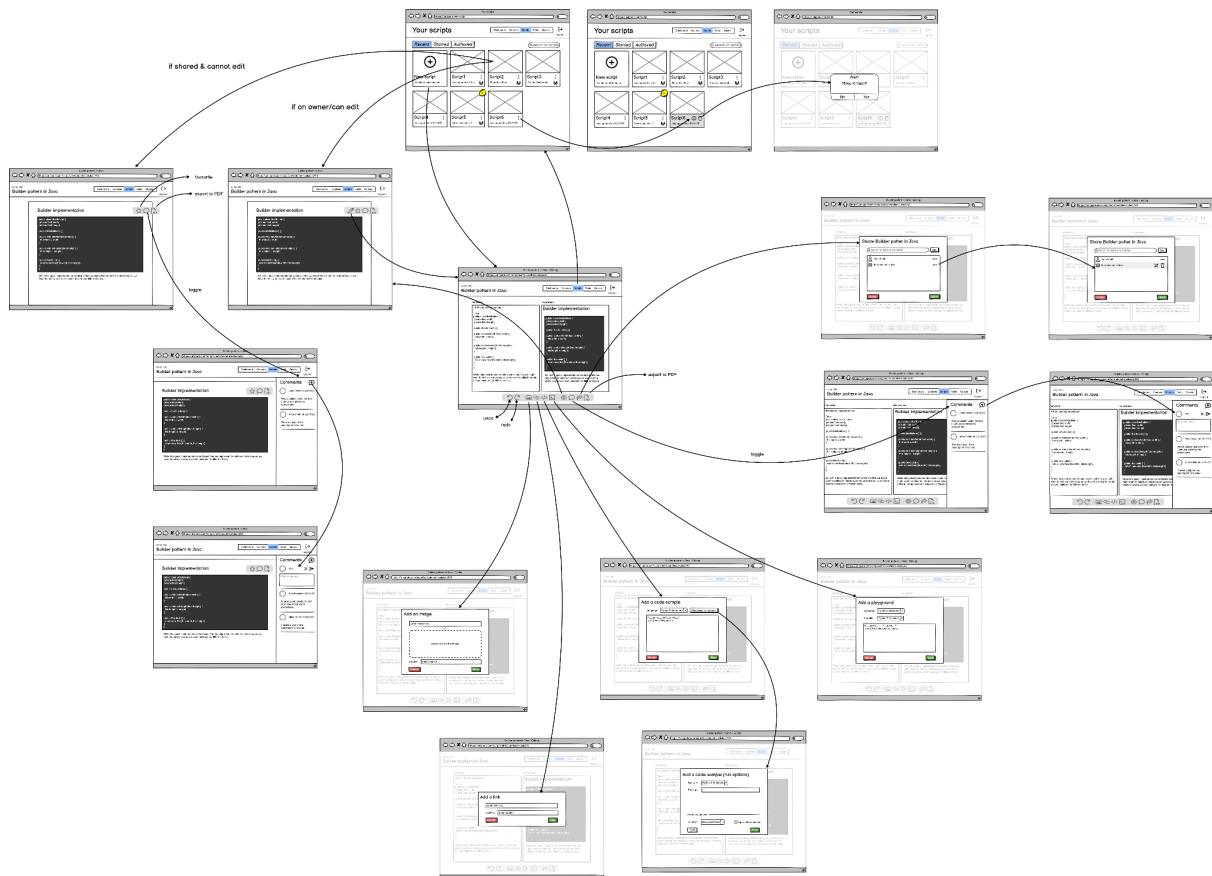
Redo quiz



69

### 3.4.5. Skrypty

- (wireflow) [PDF z schematami w wysokiej jakości](#)



Builder pattern in Java

https://unippets.com/script/view/abcdefksaskdsds243

script title

Builder pattern in Java

Dashboard Courses Scripts Tests Quizzes Log out

## Builder implementation

public class BoxBuilder {  
 private float width;  
 private float height;  
  
 public BoxBuilder() {}  
  
 public void setWidth(float width) {  
 this.width = width;  
 }  
  
 public void setHeight(float height) {  
 this.height = height;  
 }  
  
 public Box build() {  
 return new Box(this.width, this.height);  
 }  
}

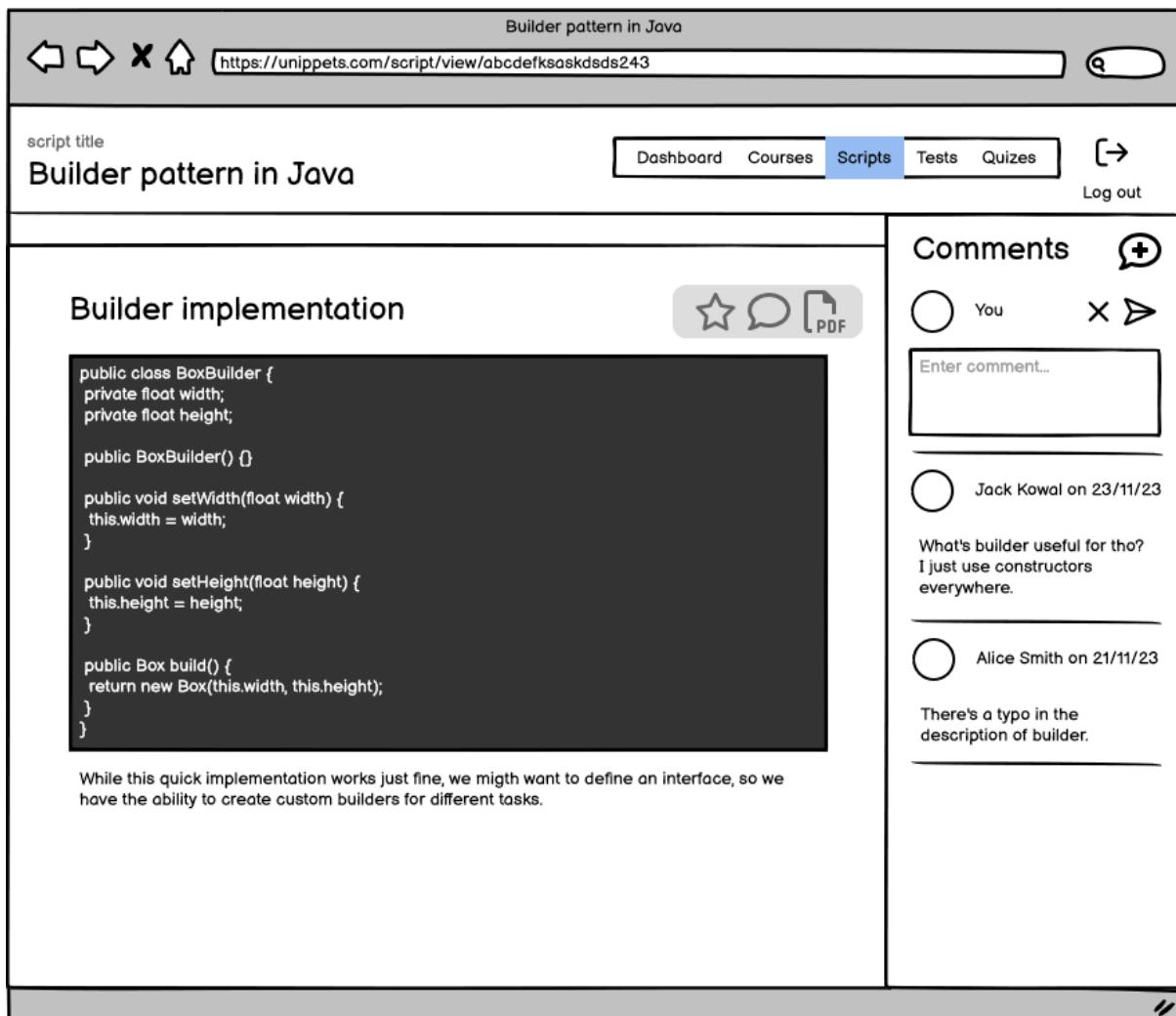
Comments +

You X ➤

Enter comment...

Jack Kowal on 23/11/23  
What's builder useful for tho?  
I just use constructors  
everywhere.

Alice Smith on 21/11/23  
There's a typo in the  
description of builder.



Widok czytnika skryptu - dodawanie komentarza

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
Builder pattern in Java

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation
```

```
java
public class BoxBuilder {
 private float width;
 private float height;

 public BoxBuilder() {}

 public void setWidth(float width) {
 this.width = width;
 }

 public void setHeight(float height) {
 this.height = height;
 }

 public Box build() {
 return new Box(this);
 }
}
```

Add a playground

Language Python3 (detected)  
Run with Python 3.12 (latest)

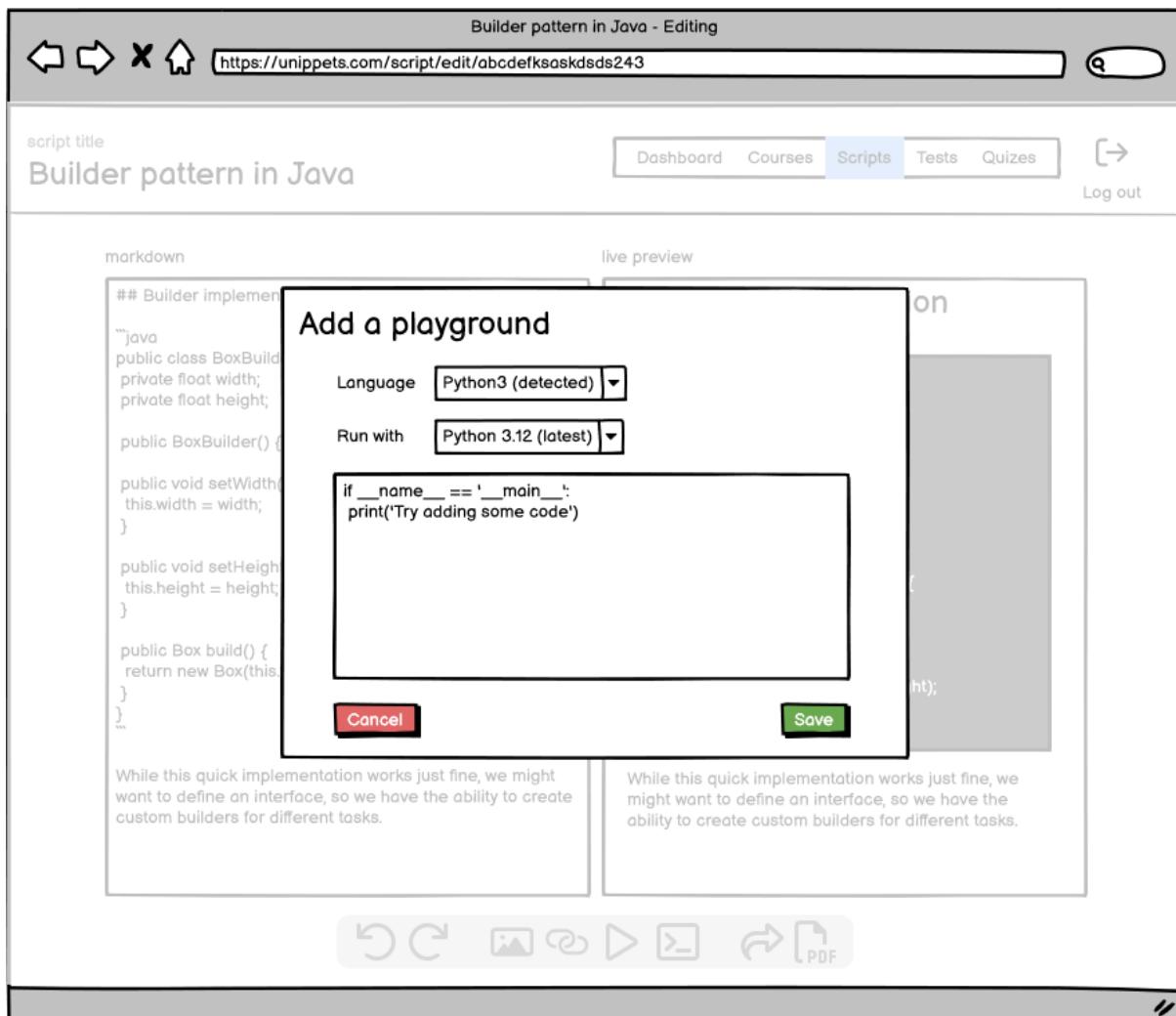
```
if __name__ == '__main__':
 print('Try adding some code')
```

Cancel Save

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

Back Forward Image Refresh Print PDF



Dodawanie interaktywnego kodu

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
Builder pattern in Java

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation

```java
public class BoxBuilder {
    private float width;
    private float height;

    public BoxBuilder() {}

    public void setWidth(float width) {
        this.width = width;
    }

    public void setHeight(float height) {
        this.height = height;
    }

    public Box build() {
        return new Box(this);
    }
}
```

Add a code sample
```

Language Python3 (detected) Customize run options

```
if __name__ == '__main__':
 print("Hello Unippets!")
```

Cancel Save

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

Back Forward Image Refresh Print PDF

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
Builder pattern in Java

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation

```java
public class BoxBuilder {
    private float width;
    private float height;

    public BoxBuilder() {}

    public void setWidth(float width) {
        this.width = width;
    }

    public void setHeight(float height) {
        this.height = height;
    }

    public Box build() {
        return new Box(this.width, this.height);
    }
}
```

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.
```

### Builder implementation

public class BoxBuilder {

Add a link

Enter link URL

Caption Enter caption

Cancel Save

public Box build() {  
 return new Box(this.width, this.height);  
}

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

Back Forward Home Search

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
Builder pattern in Java

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation
```

```
java
public class BoxBuilder {
 private float width;
 private float height;

 public BoxBuilder() {}

 public void setWidth(float width) {
 this.width = width;
 }

 public void setHeight(float height) {
 this.height = height;
 }

 public Box build() {
 return new Box(this);
 }
}
```

Add an image

Enter image URL or, Drag and drop the image

Caption Enter caption

Cancel Save

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

Back Forward Image Refresh Print PDF

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
**Builder pattern in Java**

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation

```java
public class BoxBuilder {
    private float width;
    private float height;

    public BoxBuilder() {}

    public void setWidth(float width) {
        this.width = width;
    }

    public void setHeight(float height) {
        this.height = height;
    }

    public Box build() {
        return new Box(this.width, this.height);
    }
}
```

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.
```

### Builder implementation

```
public class BoxBuilder {
 private float width;
 private float height;

 public BoxBuilder() {}

 public void setWidth(float width) {
 this.width = width;
 }

 public void setHeight(float height) {
 this.height = height;
 }

 public Box build() {
 return new Box(this.width, this.height);
 }
}
```

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.



The diagram shows the Unipetts editor's toolbar with several icons. Two specific icons are highlighted with arrows pointing to them from below: the 'undo' icon (a left-pointing arrow inside a circle) and the 'redo' icon (a right-pointing arrow inside a circle). These icons are located at the bottom of the toolbar, just above the main content area.

Your scripts

https://unippets.com/script

Dashboard Courses Scripts Tests Quizzes Log out

Recent Starred Authored

search for scripts

New script Create empty template

Script1 Last opened on Tue

Script2 Shared on Mon

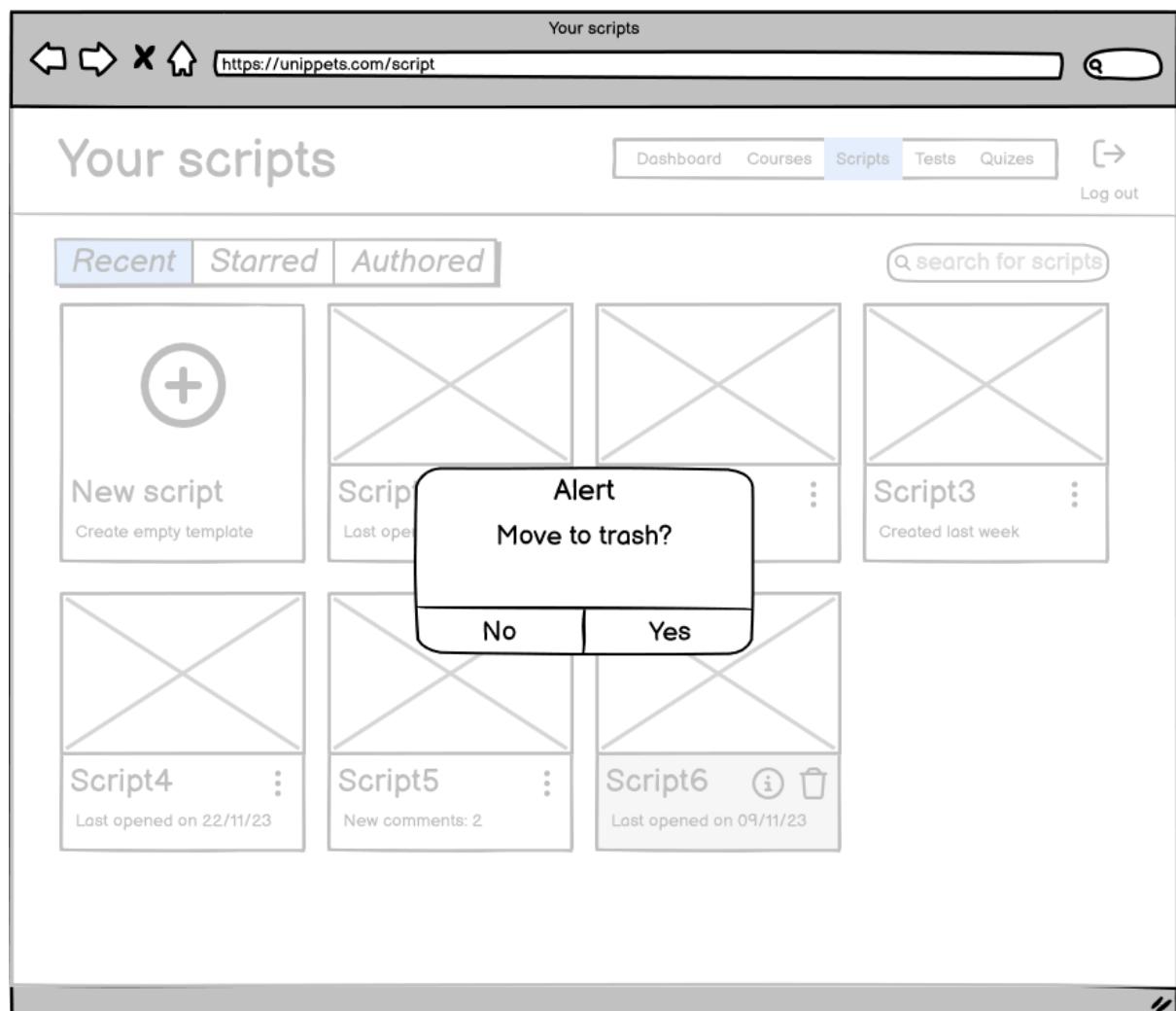
Script3 Created last week

Script4 Last opened on 22/11/23

Script5 New comments: 2

Script6 Last opened on 09/11/23

77



Your scripts

https://unippets.com/script

Dashboard Courses Scripts Tests Quizzes Log out

Recent Starred Authored

search for scripts

New script Create empty template

Script1 Last opened on Tue

Script2 Shared on Mon

Script3 Created last week

Script4 Last opened on 22/11/23

Script5 New comments: 2

Script6 Last opened on 09/11/23

Builder pattern in Java - Editing  
<https://unippets.com/script/edit/abcdefksaskdsds243>

script title  
Builder pattern in Java

Dashboard Courses Scripts Tests Quizes Log out

markdown live preview

```
Builder implementation
```

```
java
public class BoxBuilder {
 private float width;
 private float height;

 public BoxBuilder() {}

 public void setWidth(float width) {
 this.width = width;
 }

 public void setHeight(float height) {
 this.height = height;
 }

 public Box build() {
 return new Box(this);
 }
}
```

**Share Builder pattern in Java**

search for users and courses Add

John Smith

Introduction to Java

Cancel Share

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

While this quick implementation works just fine, we might want to define an interface, so we have the ability to create custom builders for different tasks.

undo redo image link email PDF