Uzasadnienie biznesowe i studium wykonalności

1. **Informacje wstępne**
   a. Temat projektu
      System automatycznego odczytywania wyników prac
   b. Kontekst biznesowy
      Opiekun projektu: dr inż. Krzysztof Ocetkiewicz
      Projekt realizowany w ramach przedmiotu Projekt Grupowy
   c. Ograniczenia projektu
      Czas realizacji projektu: do końca semestru zimowego 2015/2016
      Budżet: projekt realizowany nieodpłatnie, brak budżetu
      Technologia: Brak narzuconej technologii

2. **Stan aktualny**
   a. Procesy i systemy

      Obecnie proces przeprowadzenia i sprawdzenia egzaminów:

      1) Osoba prowadząca przygotowuje szablon egzaminu w wersji elektronicznej (np. dokument Microsoft Word). Zawierający pytania, miejsce na odpowiedź, miejsce na ilość punktów (może być w różnym miejscu dokumentu) oraz obszar na dane osobowe studenta. Następnie szablony są drukowane.

      2) Przeprowadzany jest egzamin, tzn. każdy student otrzymuje swoją kopię arkusza (wersja papierowa) i na podstawie posiadanej wiedzy i umiejętności udziela odpowiedzi. Rozwiązane egzaminy są zbierane przez prowadzącego

      3) Prowadzący ocenia każdą odpowiedź, wprowadzając liczbę punktów w miejsce wyznaczone przez szablon (na papierze)

      4) Prowadzący ręcznie podlicza punkty, samodzielnie wyznacza ocenę wynikającą deterministycznie z liczby zebranych punktów oraz przyjętej strategii oceniania.

      5) Wyznaczoną wprowadza do komputera (np. do osobistych notatek), zaznaczając przy tym jakiego studenta (numer indeksu odczytany z danej pracy) dotyczy ocena.

   b. Problemy i powody podjęcia projektu

      Duża liczba powtarzających się czynności (punkt 4 i 5), z których każda zajmuje niewiele czasu jednak ze względu na ich liczbę znacząco wydłuża czas trwania procesu. Czynności te są bardzo podobne, nie wymagają specjalistycznej wiedzy i mogłyby być realizowane przez system komputerowy.

   c. Główne wymagania na oprogramowanie

      Celem projektu jest wytworzenie aplikacji, która umożliwi zautomatyzowanie procesu sprawdzania prac przez prowadzącego. Jest to czynność wyjątkowo żmudna i wymagająca dużego skupienia. Często wymaga poświęcenia znacznego czasu na realizację czynności, takich jak podliczanie punktów, wystawianie ocen, generowanie listy wyników z egzaminu, czy całego przedmiotu. Podczas, gdy mogłyby one wykonywać się automatycznie.

Głównymi wymaganiami odnośnie oprogramowania są:

- Wykrywanie informacji we wprowadzonych obrazach, którymi są wypełnione arkusze testów, lub obecności. System powinien wykrywać liczbę wprowadzonych przez prowadzącego punktów w określonych wcześniej miejscach oraz przetwarzać dane zgodnie ze zdefiniowanym schematem.

- Definiowanie schematów pozyskiwania informacji z obrazów arkuszy. Każdy rodzaj arkusza wygląda inaczej. Można wyróżnić jego dwa rodzaje test oraz lista wyników. W pierwszym przypadku należy zczytywać punkty dla konkretnego studenta z wskazanych pól przy zadaniach. W drugim przypadku lista zawiera dane wielu studentów wraz z przyznanymi im punktami.

- Dane powinny być grupowane pod względem określonych przez prowadzącego kategorii, którymi są :
    - przedmiot, którego dotyczą testy lub wyniki,
    - student, którego wyniki są zbierane
    - test/egzamin/pojedyncze zajęcia, z których wyniki są zbierane

- Przedstawiane w formie graficznych zestawień. Każda zebrana dana powinna mieć możliwość ręcznej edycji. System powinien też pozwalać na wprowadzenie dodatkowego rodzaju wartości np. dodatkowe plusy poprzez dodanie nowej kolumny, do zgrupowanych wartości.

- Przetwarzanie pobranych danych. Pobrane dane dotyczą studentów oraz ich wyników. System powinien sumować wszystkie wyniki z danego przedmiotu oraz wyliczać na ich podstawie ocenę końcowa.

- Generowanie list wyników z określonego zakresu określonego przez prowadzącego. Zakres wybierany na podstawie kategorii dotyczących studentów,

- Prowadzący powinien mieć możliwość ustalenia progów zaliczeń oraz algorytmu wyznaczającego progi ocen częściowych oraz końcowych.

- Korzystanie z aplikacji powinno być łatwe i wygodne. Ważne jest aby interfejs był intuicyjny, nie posiadał zbędnych funkcji i używanie go było możliwe bez przeczytania instrukcji użytkownika.

- System powinien uruchamiać się na różnych systemach operacyjnych.

d. Gotowe systemy dostępne na rynku

Na rynku jest dość szeroki wybór systemów typu OCR, które teoretycznie powinny umożliwić rozpoznanie wprowadzonych wyników i pozwolić zaimportować je do innego programu. Niestety programy darmowe (np. Free Online OCR) mają niską skuteczność działania, zaś programy komercyjne (np. OminiPage Standard, Readiris Pro) są dość kosztowne. Dodatkowo system OCR może pozwolić jedynie na odczytanie

wyników. Ich dalsza obróbka (wyznaczanie ocen, generowanie list wyników, grupowanie danych) wymaga oddzielnych programów. Ze względu na specyfikę tematyki i zagadnienia jedyną możliwością realizacji takiego programu jest jego implementacja zgodnie z wymaganiami klienta. Kolejnym problemem byłaby integracja takich rozwiązań. Tak więc na rynku istnieją rozwiązania, które częściowo mogłyby rozwiązać problem będący przyczyną rozpoczęcia tego projektu, jednak jakość i kompletność tego rozwiązania byłaby daleka od oczekiwań klienta.

3. **Efekty projektu**

a. Produkty (oprogramowanie, dokumentacja, usługi)

Produkt finalny projektu to oprogramowanie wspomagające proces oceny wyników pisemnych prac studentów, spełniający wszystkie wymagania zdefiniowane wcześniej w tym dokumencie.

W ramach projektu powstanie także następująca dokumentacja:

- dokumenty związane z prowadzeniem projektu, min. uzasadnienie biznesowe projektu, harmonogramy prac
- dokumentacja techniczna projektu, np. opis użytego w projekcie algorytmu do analizy prac
- podręcznik użytkownika wytworzonej aplikacji

b. Zakładane korzyści

Wprowadzenie do użytku oprogramowania wytworzonego w ramach projektu przyniesie liczne korzyści. Dzięki zautomatyzowaniu części czynności proces sprawdzania prac ulegnie znacznemu skróceniu. Użytkownik będzie musiał poświęcić dodatkowy czas na tworzenie szablonów prac w programie czy skanowanie wypełnionych przez studentów prac, aby mogły być one opracowane przez program, ale zyska znacznie więcej czasu dzięki temu, że nie będzie musiał ręcznie podliczać punktów i zestawiać wyników. Co więcej, proces sprawdzania prac nie będzie wymagał zaangażowania większej liczby osób, co ma obecnie miejsce, gdy czas, w którym trzeba uzyskać wyniki jest ograniczony.

Korzyści oczekiwane po wdrożeniu projektu:

- przyspieszenie procesu oceny prac o 70%
- zmniejszenie liczby osób zaangażowanych w sprawdzanie prac do 1 osoby

c. Możliwe niepożądane skutki

Zespół projektowy zakłada, że użyty w projekcie algorytm, służący do przetwarzania obrazów prac studentów, nie będzie działał ze stuprocentową skutecznością, jednak błędy przy analizie obrazu nie powinny przekraczać 5%. Może więc dojść do sytuacji, w której praca studenta zostanie oceniona niewłaściwie, gdy błędnie zostaną odczytane punkty przypisane studentowi za poszczególne odpowiedzi lub oceniana praca zostanie przypisana do niewłaściwego studenta, gdy system nie rozpozna poprawnie danych osobowych studenta. Aby zminimalizować skutki błędnych odczytów, użytkownik będzie mógł przejrzeć dane odczytane przez system i w razie potrzeby ręcznie dokonać ich zmiany.

4. Propozycje rozwiązania

a. Architektura/technologia/strategia pozyskania/funkcjonalność - Wariant 1

Pierwszym rozwiązaniem jest przygotowanie systemu jako aplikacji webowej w wariancie "lekkiego" serwera i "ciężkiego" klienta. Koncepcja taka opierałaby się o aplikację serwera wykonaną w technologii skryptowej (język Python) z wykorzystaniem prostego frameworka do tworzenia takiej aplikacji (web2py dla języka Python). Język Python dzięki temu, że posiada bardzo duży zbiór funkcji biblioteki standardowej, liczne moduły dodatkowe, często dedykowane do aplikacji webowych, pozwala na implementację strony serwera niskim nakładem prac i w stosunkowo krótkim czasie. Dodatkowo po stronie serwera umieszczony byłby pojemnik przechowujący dane zrealizowany jako baza danych. Szczegółowy wybór pojemnika i jego konfiguracji jest kwestią dokładniejszej specyfikacji rozwiązania, różnych zalet i wad poszczególnych technologii, jednak optymalnym (ze względu na możliwości, otwartość, darmowość oraz znajomość technologii przez zespół projektowy) wydaje się być PostgreSQL. Strona serwera aplikacji działałaby na zasadzie udostępniania i aktualizacji po stronie klienta danych przechowywanych w sposób trwały w bazie danych. Dostarczałaby również interfejs pozwalający utrwalić aplikacji klienckiej potrzebne dane. Zdecydowana większość odpowiedzialności aplikacji zostałaby zrealizowana po stronie klienta. Podejście takie pozwala uruchomić serwer aplikacji nawet na bardzo słabej sprzętowo maszynie i uzależnia wydajność działania rozwiązania głównie od wydajności komputera klienta. Pozwala również na uruchomienie aplikacji z praktycznie dowolnego miejsca na świecie (pod warunkiem posiadania dostępu do sieci, w której znajduje się włączona maszyna z uruchomionym serwerem aplikacji). W teorii umożliwia działanie aplikacji na urządzeniach mobilnych (rzeczywista przydatność w zależności od stosunku potrzebnej przez aplikację mocy obliczeniowej do dostępnej na danym urządzeniu). Do wykonania strony klienckiej aplikacji wykorzystany zostałby język JavaScript wraz z technologiami AngularJS oraz NodeJS. Dodatkową korzyścią aplikacji zaimplementowanej w tej technologii jest przyjemny w interakcji interfejs użytkownika, w którym przeładowywane są tylko wymagane elementy, zaś do odświeżenia widoku nie jest wymagane jawne (dla użytkownika końcowego) wygenerowanie żądania.

Ryzyko jakie kryje się w tym rozwiązaniu to przede wszystkim duży nakład prac na implementację strony klienta i ograniczenie w tym zakresie do wykorzystania języka JavaScript. Nie jest pewne czy dla tego języka istnieją odpowiednie biblioteki

pozwalające zrealizować oczekiwaną funkcjonalność (w szczególności biblioteki umożliwiające przetwarzanie obrazu). W skrajnym wypadku wymagałoby to implementacji odpowiednich algorytmów lub też rezygnacja z funkcjonalności po stronie klienta i przeniesienie ich na stronę serwera, gdzie można skorzystać z gotowych komponentów dla języka Python.

Strategią pozyskania tego rozwiązania byłaby implementacja przez członków zespołu z wykorzystaniem gotowych darmowych komponentów (biblioteki, framework web2py, AngularJS, NodeJS, baza danych, itd.).

Funkcjonalność zakładana w tym rozwiązaniu to pełna funkcjonalność oczekiwana od systemu opisana w punkcie opisującym główne wymagania stawiane produktowi tego projektu.

b. Architektura/technologia/strategia pozyskania/funkcjonalność - Wariant 2

Drugim rozwiązaniem jest przygotowanie systemu jako aplikacji typu desktop, instalowanej na komputerze użytkownika końcowego. Członkowie zespołu zakładają realizację rozwiązania w języku Python. Wynika to z dostępności wielu bibliotek i rozszerzeń dla tego języka oraz znajomości powiązanych z nim technologii przez członków zespołu. Doświadczenie w implementacji aplikacji w języku Python powinno pozytywnie wpłynąć na koszt oraz poświęcony nakład prac w przygotowywaniu rozwiązania i ostatecznie na jakość produktu. Koncepcja ta polega na przygotowaniu pojedynczej aplikacji, zawierającej część realizującą logikę biznesową użytkowania, odpowiednie algorytmy oraz bazę danych. Całość mogłaby być instalowana na komputerze jako pojedyncze rozwiązanie. Przenośność języka Python oraz prosty sposób instalacji interpretera na różnych systemach operacyjnych pozwala na przygotowanie instalatora możliwego do ewentualnego wykorzystania w przyszłości przez osoby nietechniczne lub też przez osoby techniczne jednak bez poświęcania dodatkowego (poza pobraniem aplikacji i uruchomieniem instalacji) czasu na instalację oprogramowania. Aplikacja ta nie mogłaby być wykorzystywana przez urządzenia mobilne jednak po jednorazowej instalacji nie wymagałaby do swojego działania połączenia z Internetem, dostępu do sieci z uruchomioną aplikacją serwerową ani też posiadania dodatkowej maszyny (chociażby wirtualnej) do hostowania strony serwera aplikacji. Również aspekt dostępności odpowiednich bibliotek i algorytmów zostałby rozwiązany przy wyborze tej opcji, gdyż są one dostępne dla języka Python.

Strategią pozyskania tego rozwiązania byłaby implementacja przez członków zespołu z wykorzystaniem gotowych darmowych komponentów (biblioteki, komponenty graficzne dla języka Python, baza danych, itd.).

Funkcjonalność zakładana w tym rozwiązaniu to pełna funkcjonalność oczekiwana od systemu opisana w punkcie opisującym główne wymagania stawiane produktowi tego projektu.

c. Architektura/technologia/strategia pozyskania/funkcjonalność - Wariant 3

Trzecim rozwiązaniem jest przygotowanie systemu jako aplikacji działającej w formie usługi webowej wraz z prostą aplikacją kliencką. System taki zostałby zrealizowany w oparciu o technologię Python po stronie serwera, gdyż duża dostępność bibliotek i komponentów oraz doświadczenie członków zespołu w wytwarzaniu aplikacji w tej technologii powinny znacząco zmniejszyć koszt dostarczenia oraz korzystnie wpłynąć na czas wytworzenia systemu i jego jakość. Po stronie klienta zastosowano by prostą aplikację uruchamianą w przeglądarce zaimplementowaną z użyciem języka JavaScript. W tym podejściu część kliencka aplikacji odpowiadałaby głównie za komunikację z użytkownikiem końcowym i pełniła rolę interfejsu dostępowego do systemu dla tego użytkownika. Zdecydowana większość logiki biznesowej zostałaby zaimplementowana po stronie serwera. Pozwala to na uruchomienie aplikacji klienckiej nawet na słabym sprzętowo komputerze użytkownika, z praktycznie dowolnego miejsca na świecie (wymagany dostęp do sieci z uruchomioną maszyną realizującą proces serwera), bez instalacji dodatkowego oprogramowania (wystarczy przeglądarka internetowa zazwyczaj dostarczona razem z systemem operacyjnym). Podejście takie daje również realne szanse użycia systemu na urządzeniach mobilnych, gdyż większość ciężaru obliczeń zlokalizowana jest po stronie maszyny serwera. Wymaga ona jednak stosunkowo wydajnego sprzętu do uruchomienia procesu serwera i strony serwerowej aplikacji. Rozwiązanie to nie jest tak dobrze skalowalne jeżeli chodzi o ilość jednoczesnych użytkowników końcowych jak podejście z „ciężkim" klientem. Równocześnie wymagane jest przesyłanie dużo większej ilości danych pomiędzy klientem a serwerem, co może negatywnie wpłynąć na wydajność aplikacji w wypadku niezbyt szybkiego łącza internetowego, czy też korzystania z sieci komórkowej z limitami transferów. Nie występuje tu problem z dostępem do odpowiednich bibliotek, ponieważ część klienta jest stosunkowo prosta i dostępne technologie powinny być całkowicie wystarczające do odpowiedniego jej zrealizowania, zaś dla języka Python, w którym miałaby zostać zaimplementowana strona serwera istnieje wiele gotowych komponentów pozwalających uzyskać algorytmy i narzędzia potrzebne do realizacji funkcjonalności aplikacji.

Strategią pozyskania tego rozwiązania byłaby implementacja przez członków zespołu z wykorzystaniem gotowych darmowych komponentów (biblioteki, framework web2py, AngularJS, NodeJS, baza danych, itd.).

Funkcjonalność zakładana w tym rozwiązaniu to pełna funkcjonalność oczekiwana od systemu opisana w punkcie opisującym główne wymagania stawiane produktowi tego projektu.

5. Analiza porównawcza wariantów i wnioski

a. Zastosowana metoda analizy wariantów (wybór, przedstawienie metody, skale, uzasadnienie)

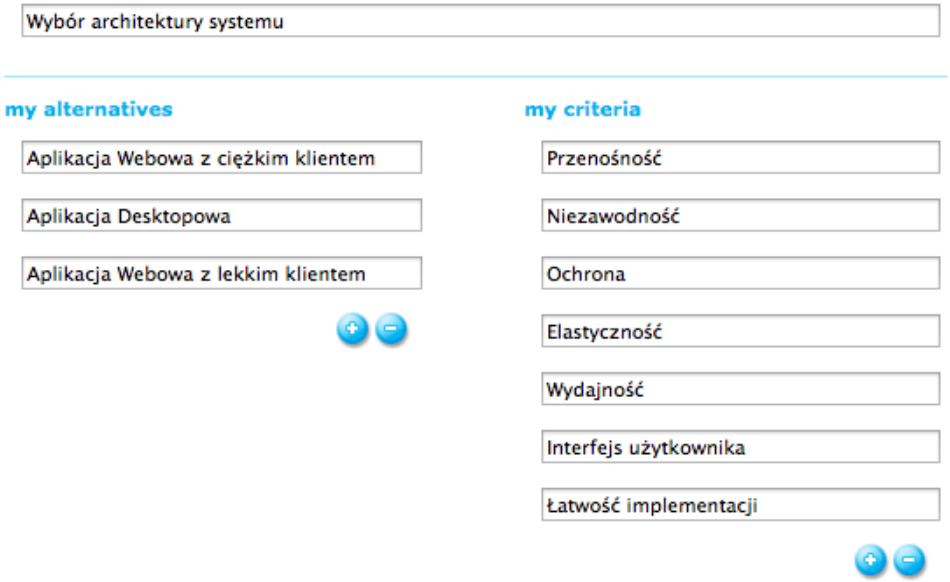
Analiza porównawcza wariantów zostanie wykonana metodą AHP, która polega na porównaniu każdego z wariantów pod kątem tych samych kryteriów, wystawiając im odpowiednią ocenę w porównaniu dwóch względem siebie. Do opisu zostanie wykorzystana następująca skala, w której przyznaje się jednemu z nich liczbę od 1 do 9, gdzie:

- 1 oznacza równoważne obie strony
- 3 oznacza umiarkowaną preferencję jednej
- 5 silną preferencję
- 7 bardzo silną preferencję
- 9 oznacza ekstremalną preferencję jednej.

Wybrana została metoda AHP, ponieważ pozwala ona na porównanie możliwych rozwiązań technologicznych pod tymi samymi kryteriami. Ze względu na fakt, że zaproponowane rozwiązania są podobne, ale posiadają drobne różnice, które mogą mieć kluczowe znaczenie w przyszłości, zdecydowaliśmy, że porównanie ich w ten sposób. Metoda ta ma wiele zalet, jak to, że daje możliwość łącznej analizy kryteriów ilościowych oraz jakościowych. Jest ona oparta o porównanie wielu kryteriów, dzięki czemu można uzasadnić swój wybór. Sama w sobie jest bardzo uniwersalna i prosta.

Porównanie wariantów

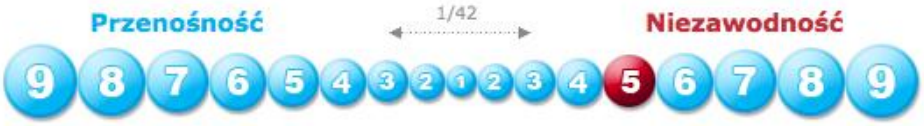
Wybrane warianty oraz kryteria:

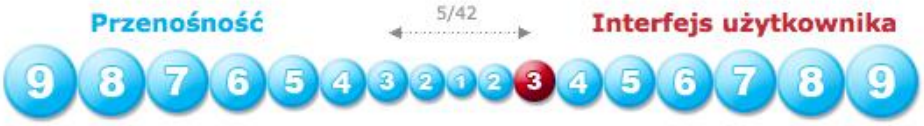


Porównanie kryteriów:

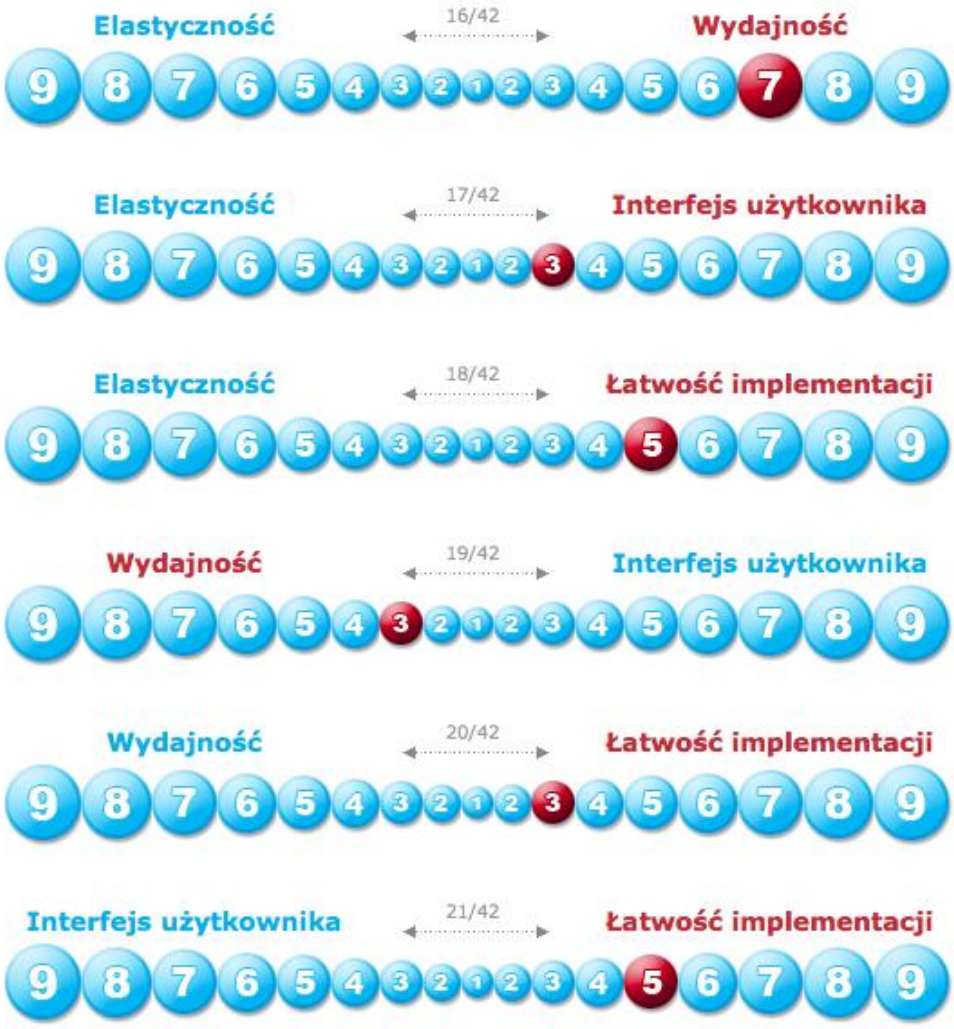
criteria preferences

Use the scale to define importance of criteria.





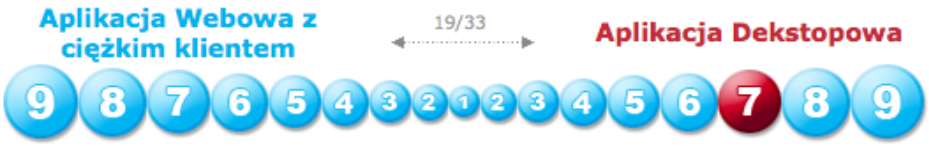

Niezawodność
7/42
Ochrona
Niezawodność
8/42
Elastyczność
Niezawodność
9/42
Wydajność
Niezawodność
10/42
Interfejs użytkownika
Niezawodność
11/42
Łatwość implementacji
Ochrona
12/42
Elastyczność
Ochrona
13/42
Wydajność
Extreme importance
Ochrona
14/42
Interfejs użytkownika
Ochrona
15/42
Łatwość implementacji



Porównanie wariantów architektury względem kryteriów:

criteria Niezawodność

Use the scale to define importance of alternative by criteria Niezawodność ,compared with the other alternative. Continue with comparisons.

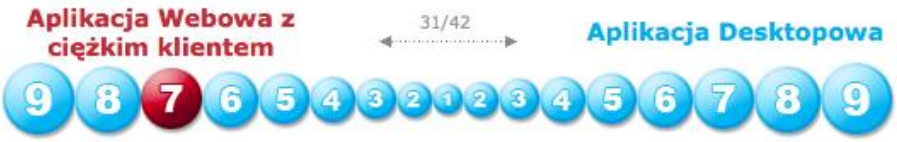


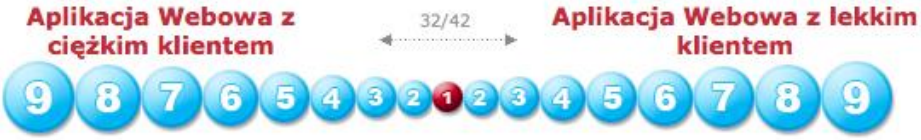
criteria Ochrona

Use the scale to define importance of alternative by criteria Ochrona ,compared with the other alternative. Continue with comparisons.

criteria Elastyczność

Use the scale to define importance of alternative by criteria Elastyczność ,compared with the other alternative. Continue with comparisons.

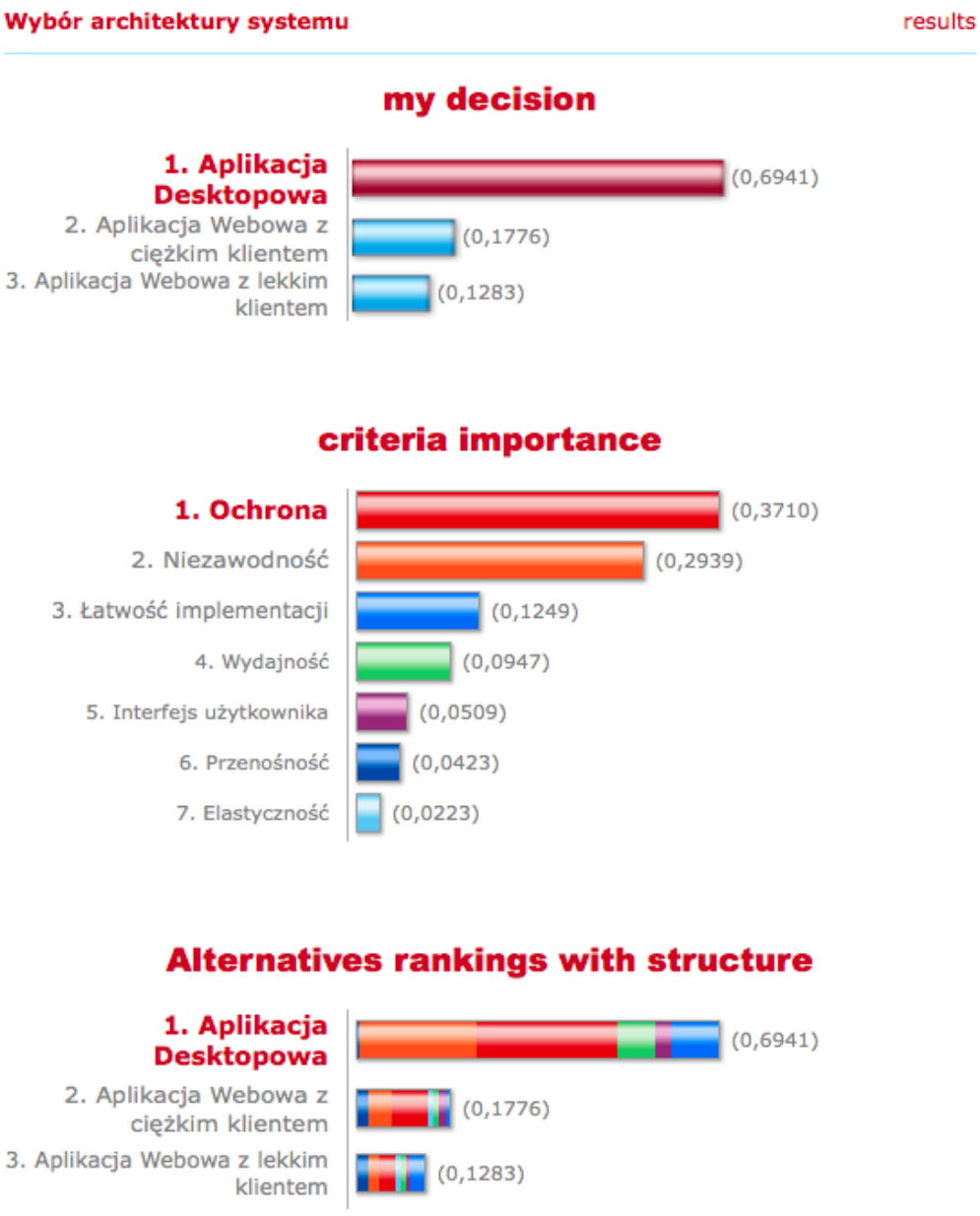




criteria Wydajność

Use the scale to define importance of alternative by criteria Wydajność ,compared with the other alternative. Continue with comparisons.

3. Wnioski i zalecenia (wskazanie najlepszego wariantu, zalecenia realizacji projektu)



Dokonaliśmy porównania metodą AHP trzech możliwych wariantów wyboru architektury aplikacji, jakimi są:

- Aplikacja Desktopowa
- Aplikacja Webowa z ciężkim klientem
- Aplikacja Webowa z lekkim klientem

Najlepiej wypadła w porównaniu architektura aplikacji desktopowej. Okazała się być ona najlepszym wyborem, gdy zależy nam na wysokiej ochronie danych oraz niezawodności. Są one kluczowe dla działania aplikacji, ponieważ dane osobowe studentów, ich wyniki nie mogą być dostępne dla osób trzecich ze względu na prawo o ochronie danych osobowych. Nieakceptowalne jest również, aby wczytywane z arkuszy dane, generowały błędne wyniki. Wytworzony system nie może powodować, że studenci otrzymają niesprawiedliwe oceny. Równie ważnymi kryteriami wyboru są wydajność oraz łatwość implementacji. Wynika to z faktu, że wykonywane przez komputer obliczenia nie mogą trwać dłużej, niż ręczne wprowadzanie danych przez prowadzącego. Jeżeli tak działoby się, produkt nie byłby

przydatny. Łatwość implementacji jest równie ważna ze względu na ograniczony czas realizacji projektu oraz zasoby ludzkie pracujące przy nim.

6. **Wstępny plan projektu**

Zespół projektu

1) Kamil Kolonko
2) Mateusz Szerszyński
3) Bartłomiej Lewandowski
4) Adrian Boguszewski

a. Terminy

- 01.03.2015 – Rozpoczęcie projektu
- 01.29.2016 – Zakończenie projektu

- Zebranie i specyfikacja wymagań – ok. 2.5 miesiąca
- Analiza wymagań – ok. 1.5 miesiąca
- Opracowanie architektury projektu – ok. 2.5 miesiąca
- Implementacja – ok. 3 miesiące
- Testy i wdrożenie – ok. 3 miesiące

b. Strategia prowadzenia projektu

Przy wyborze strategii realizowania projektu, zespół kierował się głównym założeniem stworzenia produktu o wysokiej jakości. Istotnym czynnikiem wyboru metodyki projektowej była również częściowa swoboda zespołu wytwórczego dotycząca częstotliwości dostarczania i prezentowania postępów prac nad produktem, zapewniona przez klienta. W konsekwencji, do zarządzania procesem projektowym, zespół postanowił zaadaptować Model V z wykorzystaniem elementów metodologii Scrum. Wybór ten dokonany został ze względu na chęć przeznaczenia stosunkowo długiego czasu na testy produktu. Specyfika projektu jasno wskazuje także modułową konstrukcję oprogramowania, co nie działało pozytywnie na rzecz metodyk zwinnych. Elementami metodyki Scrum, które zespół chciałby zaadoptować w swoich pracach są częste spotkania zespołu deweloperskiego podczas fazy implementacji z omawianiem postępów prac oraz retrospektywa każdej z faz po jej zakończeniu. Kontrola jakości przeprowadzana po zakończeniu każdego z etapów projektu pozwoli na zminimalizowanie ryzyka dostarczenia produktu o niskiej jakości i użyteczności.

c. Etapy projektu

Przy projektowaniu procesu wytwórczego oprogramowania wyszczególniono następujące etapy prac:

- Zebranie i specyfikacja wymagań

Celem etapu jest określenie wymagań projektu zgodnie z oczekiwaniem klienta. W tym celu postanowiono wyznaczyć kilka terminów spotkań gdzie owe wymagania zostaną przekazane zespołowi oraz szczegółowo doprecyzowane. Etap ten obejmuje stworzenie dokumentu pt. „Specyfikacja wymagań systemowych", w którym zawarte zostaną wszelkie ustalenia dotyczące wymagań projektu. Dokument ten zostanie zatwierdzony przez klienta i będzie stanowił podstawę do dalszych prac projektowych.

- Analiza wymagań

  Celem etapu jest opracowanie wstępnego modelu systemu ze względu na wymagania wyspecyfikowane w etapie poprzedzającym. Warunki projektu zgromadzone w dokumencie SWS zostaną poddane analizie ze względu na możliwości realizacji oraz założenia technologiczne. Istotnym czynnikiem będzie również opracowanie diagramu ryzyk z wyszczególnionymi zdarzeniami mogącymi wpłynąć na pomyślność realizacji projektu. Możliwym produktem etapu będą również makiety funkcjonalności produktu w prosty sposób przedstawiające interfejs oraz zasadę działania aplikacji.

- Architektura projektu

  Celem etapu jest stworzenie prototypu architektonicznego aplikacji realizowanej w projekcie oraz wyszczególnienie modułów i interfejsów programu. Wykonanie etapu w znaczny sposób usprawni wytworzenie aplikacji oraz pozwoli na wczesną identyfikację błędów koncepcyjnych. Produktem etapu jest diagram modułów aplikacji wraz z zależnościami między nimi. W prototypie architektury projektu uwzględniony zostanie także wybór technologii, bibliotek oraz usług, które zostaną wykorzystane w projekcie.

- Implementacja

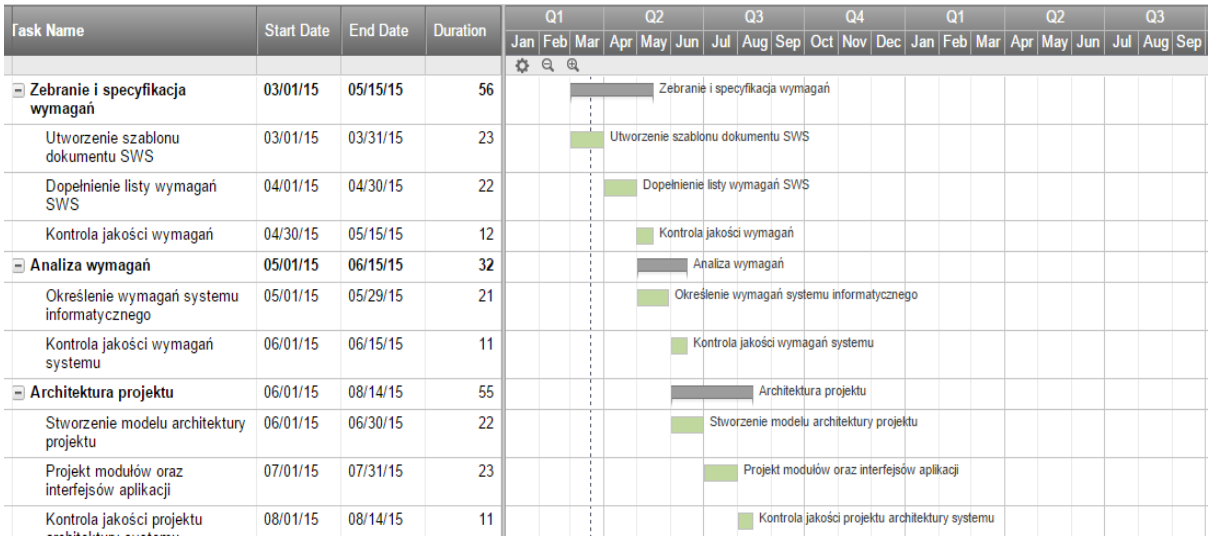
  Celem etapu jest wytworzenie aplikacji zgodnie z założeniami opracowanymi w poprzednich częściach procesu wytwórczego. Wyznaczono trzy iteracje podczas których zaimplementowane zostaną poszczególne moduły produktu oraz w fazie finalnej gotowa aplikacja. Zakłada się wytworzenie kilku wersji aplikacji już w trakcie wytwarzania części składowych (spojenie nie w pełni sprawnych modułów) tak, aby zminimalizować ryzyko wydłużenia prac implementacyjnych. Produktem etapu jest w pełni funkcjonalna aplikacja komputerowa.

- Faza testów

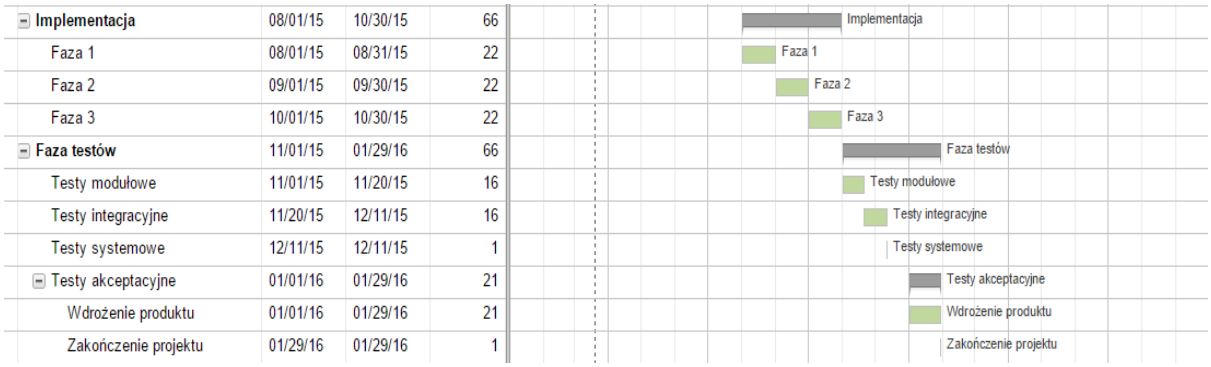
  Celem etapu jest wyeliminowanie wszelkich błędów w aplikacji klienckiej mogących wpłynąć na nieefektywne użytkowanie bądź całkowitą utratę funkcjonalności programu. Aby uzyskać założony efekt, planuje się dokonania szeregu testów modułowych, integracyjnych oraz systemowych przed dostarczeniem produktu do klienta oraz testów akceptacyjnych już po instalacji aplikacji w środowisku docelowym. Produktem etapu jest wdrożenie stworzonego systemu oraz zakończenie projektu z sukcesem.

d. Ogólny harmonogram etapów projektu

Harmonogram projektu został przedstawiony poniżej na diagramie Gantta:



Task Name	Start Date	End Date	Duration
Zebranie i specyfikacja wymagań	**03/01/15**	**05/15/15**	56
Utworzenie szablonu dokumentu SWS	03/01/15	03/31/15	23
Dopełnienie listy wymagań SWS	04/01/15	04/30/15	22
Kontrola jakości wymagań	04/30/15	05/15/15	12
Analiza wymagań	**05/01/15**	**06/15/15**	32
Określenie wymagań systemu informatycznego	05/01/15	05/29/15	21
Kontrola jakości wymagań systemu	06/01/15	06/15/15	11
Architektura projektu	06/01/15	08/14/15	55
Stworzenie modelu architektury projektu	06/01/15	06/30/15	22
Projekt modułów oraz interfejsów aplikacji	07/01/15	07/31/15	23
Kontrola jakości projektu	08/01/15	08/14/15	11



Implementacja	08/01/15	10/30/15	66
Faza 1	08/01/15	08/31/15	22
Faza 2	09/01/15	09/30/15	22
Faza 3	10/01/15	10/30/15	22
Faza testów	11/01/15	01/29/16	66
Testy modułowe	11/01/15	11/20/15	16
Testy integracyjne	11/20/15	12/11/15	16
Testy systemowe	12/11/15	12/11/15	1
Testy akceptacyjne	01/01/16	01/29/16	21
Wdrożenie produktu	01/01/16	01/29/16	21
Zakończenie projektu	01/29/16	01/29/16	1

Harmonogram projektu został dostosowany do wymagań przedstawionych w ramach przedmiotu Projekt Grupowy. Tym samym, datę zakończenia projektu ustalono na dzień 29 stycznia 2016 roku, co oznacza końcową część semestru zimowego 2015/2016. Proponowana strategia wytwarzania aplikacji, nie wymaga pośrednich faz dostarczania. W planowaniu procesu wytwórczego postanowiono położyć szczególny nacisk na otrzymanie wysokiej jakości produktu końcowego co objawia się około trzymiesięcznym okresem przeprowadzania testów oprogramowania.