WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Wspisujemy po jej wybraniu

Jan Nowak
Nr albumu studenta w9699

System do raportowania przewinień w biurze

Prowadzący: mgr inż. Ewa Żesławska

Praca projektowa programowanie obiekotwe C#

Rzeszów 202X

Spis treści

Wstęp

Przykład wstępu:

Na co dzień wielu współpracowników w biurach spotyka się z pewnymi niedociągnięciami pojawiającymi się na tle drobnych obowiązków domowych, z którymi nie chcieliby spotykać się jeszcze dodatkowo w pracy, a które wynikają z zaniedbań na tym tle innych pracowników. Do takich małych niedociągnięć i zaniedbań obowiązków należy dla przykładu pozostawianie na dłuższy czas naczyń w zlewozmywaku, nie wynoszenie na czas śmieci, które zalegają w koszu przez dłuższy czas. Albo też nieumyty po używaniu ekspres do kawy, niewymienione po wykorzystaniu pojemniki z tuszem w drukarce. Problemem istotnym, a niebędącym bezpośrednio związanym z przedmiotem działalności firmy, jest dbanie o takie szczegóły, a które mogą wpływać na samopoczucie pracowników i atmosferę w firmie. Dlatego, wbrew pozorom, ważne jest odpowiednie zmotywowanie pracowników aby o takie, mimo że drobne obowiązki, dbali na równi z własnym domem, dzięki czemu każdemu współpracownikowi będzie się lepiej pracowało. I tutaj pojawia rozwiązanie naszego zespołu – system, który jednocześnie w jakiś sposób rozwiązuje powyższe zagadnienie, a z drugiej strony jest także formą urozmaicenia i rozrywki w pracy.

Przykłady wstępów do prac projektowych/naukowych/dyplomowych można znaleść w artykułach/pracach naukowych/dyplomowych dostępnych w sieci (artykuły/prace naukowe) lub w repozytorium uczelni (biblioteka WSIZ Kielanrowa).

Rozdział 1

Opis założeń projektu

1.1 Cele projetu

Tutaj powinien zostać umieszczony tekst zawierający następujące informacje:

- Jaki jest cel projektu?
- Jaki jest problem, który będzie rozwiązywany oraz proszę wskazać podstawowe źródło problemu?
- Dlaczego ten problem jest ważny oraz jakie są dowody potwierdzające jego istnienie?
- Co jest niezbędne, aby problem został rozwiązany przez Zespół i dlaczego?
- W jaki sposób problem zostanie rozwiązany? Jak krok po kroku będzie przebiegała realizacja projektu? Co będzie wynikiem prac. (Wynikiem może być np.: aplikacja, system, sposób, metoda, program komputerowy).

Opis założeń projektu ma być ciągłym tekstem, a nie wykupowaną listą.

1.2 Wymagania funkcjonale i niefunkcjonalne

Definicja:
Wymagania funkcjonalne

- opisują funkcje (czynności, operacje, usługi) wykonywane przez system
- Często stosowany sposób opisu wymagań – język naturalny
- Liczba wymagań funkcjonalnych może być bardzo duża; konieczne jest pewnego rodzaju uporządkowanie tych wymagań, które ułatwi pracę nad nimi (złożoność !)
- Opisują, jak funkcja powinna działać.
- Skupiają się na wyniku działania użytkownika.
- Definiuje wymagania użytkownika.
- Posiada funkcje uwzględnione w przypadkach użycia.
- Weryfikuje funkcjonalność system

Wymagania niefunkcjonalne

- opisują ograniczenia, przy zachowaniu których system powinien realizować swe funkcje.

- Opisują, jakie właściwości sprawią, że funkcja będzie działać.
- Skupiają się na uproszczeniu procesu i wykonania wyniku.
- Definiują oczekiwania i doświadczenia użytkownika działania użytkownika.
- Posiadają ograniczenia, które pomogą zredukować czas i koszty rozwoju.
- Weryfikują wydajność systemu.

Wymagania funkcjonalne przykłady:

Lista przykładów wymagań funkcjonalnych obejmuje każde zachowanie systemu IT, zmieniające się pod wpływem zastosowanej funkcji. Jeżeli wymagania funkcjonalne nie zostaną potwierdzone, system nie będzie działał.

- Reguły biznesowe.
- Poziomy autoryzacji.
- Śledzenie audytów.
- Interfejsy zewnętrzne.
- Funkcje administracyjne.
- Generowanie danych historycznych.
- Uwierzytelnianie użytkownika na żądanie.
- Logi serwera wszystkich istniejących danych.
- Generowanie raportów w określonym czasie.
- Definiowanie poziomów autoryzacji systemu.

Wymagania niefunkcjonalne przykłady:

Na liście wymagań niefunkcjonalnych znajdują się,

- Pojemność.
- Wydajność.
- Środowisko.
- Użyteczność.
- Skalowalność.
- Niezawodność.
- Odzyskiwalność.
- Bezpieczeństwo.
- Utrzymywalność.
- Interoperacyjność.
- Integralność danych.
- 2-poziomowe uwierzytelnianie.

Rozwinięcie wymagań niefunkcjonalnych:

- Aplikacja IT powinna mieć kolor tła wszystkich ekranów #fffaaa.
- Aplikacja IT powinna przestrzegać wymagań regulatora.
- Aplikacja IT powinien rejestrować każdą nieudaną próbę logowania;
- Użytkownicy powinni zmienić hasło po pierwszym udanym logowaniu.
- Dashboard powinien pojawić się w ciągu 3 sekund po zalogowaniu użytkownika.
- Aplikacja IT powinien być w stanie obsłużyć XYZ liczbę użytkowników, zapewniając płynne działanie.

Jak zdefiniować wymagania funkcjonalne?

Jeśli twoje podejście do rozwoju oprogramowania jest zwinne (Agile), prawdopodobnie zdefiniujesz wymagania w dokumencie. Dokument wymagań funkcjonalnych będzie zawierał historie użytkowników, przypadki użycia, a także następujące sekcje.

- Cel: Ta sekcja będzie zawierała całe tło, definicje i przegląd systemu;
- Zakres aplikacji, oczekiwania i zasady biznesowe;
- Wymagania dotyczące bazy danych, atrybuty systemu i wymagania funkcjonalne;
- Przypadki użycia, czyli opisywać, w jaki sposób użytkownik będzie wchodził w interakcję z systemem. Zdefiniuj rolę każdego aktora biorącego udział w interakcji;
- Napisz jasno cel wdrożenia systemu IT.
- Wspomnij o użytkownikach aplikacji, którzy szczegółowo opiszą, jak krok po kroku będą się angażowali w tworzenie aplikacji.
- Opracuj klikalny prototyp aplikacji. To pomoże Ci reprezentować produkt w lepszy i przekonujący sposób dla interesariuszy. Możesz wybrać prototypy do wyrzucenia lub prototypy interaktywne dla swojego projektu.

Jak zdefiniować wymaganie niefunkcjonalne?

Teraz nadchodzi część, w której definiujesz oczekiwania jakościowe aplikacji dedykowanej. Te atrybuty opisują sposoby, w jakie oczekujesz, że aplikacja będzie się zachowywała.

- Zdefiniuj oczekiwania dotyczące użyteczności produktu.
- Opisz, do jakich praw i regulacji aplikacja powinna spełniać.
- Zdefiniuj dostępność aplikacji, czyli czy będzie ona funkcjonować 24/7/365?
- Określ wydajność systemu IT dla różnych funkcjonalności. To znaczy, w jakim czasie użytkownik powinien zobaczyć listę, jak długo użytkownik będzie połączony z aplikacją w przypadku braku połączenia z internetem, itp.
- Zdefiniuj wymagania dotyczące bezpieczeństwa systemu IT.
- Użyj narzędzi do automatycznego testowania, aby upewnić się co do wydajności aplikacji dedykowanej.

1.2.1 Przkłady

Przykłady wymagań funkcjonalnych aplikacji webowej:

- Rejestracja i logowanie użytkowników.
- Bezpieczne uwierzytelnianie i autoryzacja użytkowników.
- Zarządzanie profilami użytkowników.
- Możliwość wyszukiwania treści w aplikacji.
- Funkcjonalność e-commerce, taka jak koszyk i proces kasowy.
- Treści generowane przez użytkowników, takie jak komentarze i oceny.
- Integracja z usługami stron trzecich, takimi jak media społecznościowe i bramki płatności.
- Dynamiczne aktualizacje treści i powiadomienia.
- Pulpit administracyjny do zarządzania aplikacją.

Przykłady wymagań niefunkcjonalnych aplikacji webowej:

- Użyteczność aplikacji i dostępność, takie jak responsywny design i dostępność klawiatury.
- Wydajność aplikacji i skalowalność, np. szybkie czasy ładowania i zdolność do obsługi dużej liczby użytkowników jednocześnie.
- Bezpieczeństwo aplikacji i prywatność, takie jak szyfrowanie wrażliwych danych i ochrona przed atakami.
- Niezawodność aplikacji i dostępność, np. kopie zapasowe i plany odzyskiwania danych po awarii.
- Zgodność aplikacji z wymogami prawnymi i regulacyjnymi, takimi jak GDPR i przepisy dotyczące dostępności.
- Interoperacyjność aplikacji, taka jak zgodność z różnymi przeglądarkami i systemami operacyjnymi.
- Utrzymanie i wsparcie aplikacji, takie jak łatwość aktualizacji i dokumentacja dla programistów.
- Efektywność kosztowa aplikacji, taka jak minimalizacja kosztów serwera i hostingu.

System zarządzania treścią (CMS) umożliwiający edycję i usuwanie treści.
Przykłady wymagań funkcjonalnych aplikacji webowej:

- Integracja aplikacji z zewnętrznymi API w celu wymiany danych lub rozszerzenia funkcjonalności.
- Funkcje optymalizacji aplikacji pod kątem wyszukiwarek (SEO) w celu poprawy widoczności w wyszukiwarkach internetowych.
- Obsługa wielu języków w aplikacji w celu dostosowania do użytkowników posługujących się różnymi językami.
- Narzędzia współpracy, takie jak czat w czasie rzeczywistym i udostępnianie plików dla zespołów.
- Narzędzia analizy danych do śledzenia zachowań użytkowników i wydajności aplikacji.

Projektowanie doświadczeń użytkownika (UX), takich jak łatwy w użyciu interfejs i przejrzysta nawigacja.
Przykłady wymagań niefunkcjonalnych aplikacji webowej:

- Optymalizacja mobilna aplikacji, np. responsywny design i podejście mobile-first.
- Integracja systemu, np. kompatybilność z dotychczasowymi systemami i narzędziami stron trzecich.
- Bezpieczeństwo aplikacji i prywatność danych, takie jak szyfrowanie danych, kopie zapasowe i kontrola dostępu.
- Zgodność aplikacji z normami branżowymi, takimi jak PCI-DSS dla aplikacji e-commerce.
- Wsparcie dla użytkowników aplikacji, takie jak help desk, podręczniki użytkownika i samouczki.
- Monitorowanie aplikacji i raportowanie, takie jak śledzenie błędów i metryki wydajności.

To tylko kilka przykładów funkcjonalnych i niefunkcjonalnych wymagań aplikacji internetowych. Konkretne wymagania będą zależały od celu i charakteru aplikacji internetowej, a także potrzeb użytkowników i zainteresowanych stron.

Rozdział 2

Opis struktury projektu

W tym rozdziale mają pojawić się informacje odnośnie zaprojektowanej struktury oraz jej opis wraz z opisem technicznym. Należy umieścić informacje odnośnie wykorzystywanego języka, narzędzi oraz minimalnych wymagań sprzętowych. Opisać zarządzanie danymi oraz BD, umiescić informacje odnośnie zaprojektowanej hierarchni klas wraz z krótkkim opisem najważniejszych metod.

Rozdział 3

Harmonogram realizacji projektu

W rozdziale tym nalezy umieścić harmonogram realizacji projektu – diagram Ganta. Rysunek oraz krótki opis do niego. Można napisać jakie problemy trudności wystąpiły w trakcie realizacji projektu. Rozdział ten musi zawierać informacje o repozytorium i systemie kontroli wersji.

UWAGA!!!

Należy pamiętać że pliki do projektu na repozytorium musza być dsotepne przed rok od dnia złożenia końcowej pracy. W przypadku gdy ktoś chciałby usunąć pliki z repozytorium do projektu należy dołączyć załącznik z plikami źódłowymi.

Rozdział 4

Prezentacja warstwy użytkowej projektu

W rozdziale tym należy przedstawić opis warstwy użytkowej projektu w tym celu należy umieścić opis aplikacji oraz PrtSc o których jest mowa.

Rozdział 5

Podsumowanie

W rozdziale należy opisać zrealizowane prace oraz ewentulane planowane dalsze prace rozwojowe projektu.

Rozdział 6

Przydatne informacje do pisania w LaTeX

6.1 Pierwszy dokument

W rozdziale tym przedstawiono podstawowe informacje dotyczące struktury prostych plików LaTeXa. Omówiono również metody kompilacji plików z zastosowaniem programów *latex* oraz *pdflatex*.

6.2 Struktura dokumentu

Plik LaTeXowy jest plikiem tekstowym, który oprócz tekstu zawiera polecenia formatujące ten tekst (analogicznie do języka HTML). Plik składa się z dwóch części:

1. Preambuły – określającej klasę dokumentu oraz zawierającej m.in. polecenia dołączającej dodatkowe pakiety;

2. Części głównej – zawierającej zasadniczą treść dokumentu.

6.3 Sztuczne życie

Nie ma żadnych przeciwskazań do tworzenia dokumentów w LaTeXu w języku polskim. Plik źródłowy jest zwykłym plikiem tekstowym i do jego przygotowania można użyć dowolnego edytora tekstów, a polskie znaki wprowadzać używając prawego klawisza `Alt`. Jeżeli po kompilacji dokumentu polskie znaki nie są wyświetlane poprawnie, to na 95% źle określono sposób kodowania znaków (należy zmienić opcje wykorzystywanych pakietów).

6.4 Kompilacja

Załóżmy, że przygotowany przez nas dokument zapisany jest w pliku `test.tex`. Kolejno wykonane poniższe polecenia (pod warunkiem, że w pierwszym przypadku nie wykryto błędów i kompilacja zakończyła się sukcesem) pozwalają uzyskać nasz dokument w formacie pdf:

```
latex test.tex
dvips test.dvi -o test.ps
ps2pdf test.ps
```

lub za pomocą PDFLaTeX:

```
pdflatex test.tex
```

Przy pierwszej kompilacji po zmiane tekstu, dodaniu nowych etykiet itp., LaTeX tworzy sobie spis rozdziałów, obrazków, tabel itp., a dopiero przy następnej kompilacji korzysta z tych informacji.

W pierwszym przypadku rysunki powinny być przygotowane w formacie eps, a w drugim w formacie pdf. Ponadto, jeżeli używamy polecenia `pdflatex test.tex` można wstawiać grafikę bitową (np. w formacie jpg).

6.5 Narzędzia

Istnieje wiele narzędzi do pisania w LaTeX, które mogą pomóc w tworzeniu dokumentów i zarządzaniu nimi. Oto lista popularnych narzędzi i edytorów LaTeX, które są dostępne zarówno w wersji online, jak i offline.

Narzędzia online do pisania w LaTeX.

- Overleaf – jeden z najpopularniejszych edytorów LaTeXonline. Umożliwia współpracę z innymi użytkownikami w czasie rzeczywistym oraz zapewnia automatyczną kompilację dokumentów. Overleaf oferuje szereg gotowych szablonów (np. prace dyplomowe, CV, artykuły naukowe). Wsród funkcji można wyróżnić m.in.: kompilacja LaTeXw czasie rzeczywistym, wsparcie dla XeLaTeX, LuaLaTeX, BibTeX, Biber, współpraca z wieloma użytkownikami, gotowe szablony oraz integracja z GitHub. Zobacz: `https://www.overleaf.com/`.

- Papeeria – edytor LaTeXonline podobny do Overleaf. Umożliwia współpracę w czasie rzeczywistym i ma integrację z GitHub. Oferuje bezpłatną wersję z podstawowymi funkcjami. Funkcje: wsparcie dla współpracy online, kompilacja LaTeXw chmurze. Zobacz: `https://www.papeeria.com/`.

Edytory offline do pisania w LaTeX.

- Visual Studio Code (VSC) – to potężne narzędzie do pisania dokumentów w LaTeX, szczególnie po zainstalowaniu rozszerzenia LaTeXWorkshop. Zapewnia ono pełne wsparcie dla kompilacji, podglądu PDF, inteligentnego uzupełniania składni oraz współpracy z narzędziami bibliograficznymi jak BibTeX i Biber. Dzięki integracji z systemem zarządzania plikami i narzędziami do wersjonowania (jak Git), VS Code staje się bardzo elastycznym narzędziem do pisania profesjonalnych dokumentów, artykułów naukowych i prac dyplomowych w LaTeX. Szczegółowe informacje odnośnie konfiguracji środowiska do pisania LaTeXw VSC można znelźć a instrukcji *"How to write LaTeX documents using Visual Studio Code"* pod adresem

- TeXstudio – to zintegrowane środowisko do tworzenia dokumentów LaTeX. Celem jest uczynienie pisania w LaTeXu tak łatwym i wygodnym, jak to tylko możliwe. Dlatego TeXstudio ma wiele funkcji, takich jak podświetlanie składni, zintegrowana przeglądarka, sprawdzanie referencji i różni asystenci. Więcej szczegółów można znaleźć w funkcjach. Zobacz: `https://www.texstudio.org/`

Dodatkowe narzędzia wspierające pisanie w LaTeX.

- Mendeley – popularne narzędzie do zarządzania bibliografią i organizacji dokumentów naukowych. Obsługuje format BibTeX i pozwala na łatwą integrację z LaTeX. Zobacz: `https://www.mendeley.com`.

- JabRef – darmowe narzędzie do zarządzania bibliografią i referencjami, które wspiera format BibTeX i BibLaTeX. Ułatwia organizowanie źródeł bibliograficznych i integrację z LaTeX. Zobacz: `https://www.jabref.org/`.

- Zotero – kolejne narzędzie do zarządzania referencjami naukowymi, które wspiera BibTeX i BibLaTeX. Zotero jest popularnym wyborem wśród studentów i badaczy. Zobacz: `https://www.zotero.org`.

Wśród środowisk online wyrżnić można: Overleaf, Papeeria i Authorea oferują wygodną współpracę i szybki dostęp do narzędzi LaTeXz dowolnego miejsca. Środowiska offline: TeXmaker, TeXworks, Kile i LyX to doskonałe edytory dla pracy lokalnej z rozbudowanymi funkcjami. Dodatkowe narzędzia: JabRef, Mendeley i Zotero ułatwiają zarządzanie bibliografią i cytowaniami w LaTeX. Wybór odpowiedniego narzędzia zależy od Twoich potrzeb, czy preferujesz pracę lokalną, czy online, a także jak bardzo zaawansowane funkcje są Ci potrzebne.

6.6 Przygotowanie dokumentu

Plik źródłowy LaTeXa jest zwykłym plikiem tekstowym. Przygotowując plik źródłowy warto wiedzieć o kilku szczegółach:

- Poszczególne słowa oddzielamy spacjami, przy czym ilość spacji nie ma znaczenia. Po kompilacji wielokrotne spacje i tak będą wyglądały jak pojedyncza spacja. Aby uzyskać *twardą spację*, zamiast znaku spacji należy użyć znaku *tyldy*.
- Znakiem końca akapitu jest pusta linia (ilość pusty linii nie ma znaczenia), a nie znaki przejścia do nowej linii.
- LaTeX sam formatuje tekst. **Nie starajmy się go poprawiać**, chyba, że naprawdę wiemy co robimy.

Przydatna pomoc do pracy w LaTeX:

- Dokumentacja dostęna w Overleaf `https://www.overleaf.com/learn`.
- Edytor równań dla matematyki online`https://editor.codecogs.com/`.
- Edytor/generator tabel `https://www.tablesgenerator.com/`.

Bibliografia

[1] http://www.potaroo.net/tools/ipv4/ z dnia 8.12.2018

[2] http://bgp.potaroo.net/iso3166/v6dcc.html z dnia 8.12.2018

[3] Autorzy1, *Tytuł1*, Wydawnictwo1, Miasto1 Rok1.

[4] Autorzy2, *Tytuł2*, Wydawnictwo2, Miasto2 Rok2.

[5] Autorzy3, *Tytuł3*, Wydawnictwo3, Miasto3 Rok3.

[6] Autorzy4, *Tytuł4*, Wydawnictwo4, Miasto4 Rok4.

Spis rysunków

Spis tabel