# Zadanie 4: dopracowanie aplikacji lub aplikacja w innym zrębie (UWAGA: termin oddawania do 19:00)

**Otwarto:** niedziela, 21 maja 2023, 00:00
**Wymagane do:** piątek, 16 czerwca 2023, 19:00

Zadanie ma dwie alternatywne ścieżki wykonania. Zrealizowanie każdej pojedynczej ścieżki daje do 10 punktów. Należy wybrać jedną z tych ścieżek. Nie można dostać punktów z obydwu z nich na raz (np. scenariusz 3 punkty w pierwszej i 7 w drugiej, a więc w sumie 10 punktów, nie jest do zrealizowania). Nie należy też liczyć, że zostaną sprawdzone dwie ścieżki i dostanie się z nich maksimum. Sprawdzana będzie praca tylko w jednej z wybranych przez Was ścieżek.

## Ścieżka I - wdrożona aplikacja Django

W tej ścieżce wykonania zadania chodzi o dojście do punktu, w którym zapoznacie się z całością procesu tworzenia aplikacji WWW, łącznie z fazą jej końcowego dopracowywania szczegółów i wdrażania do działania.

### Punktacja wariantu I

- Aplikacja tworzona w ramach zadań 1-3 wdrożona z użyciem serwera nginx, apache lub innego serwera WWW o cechach produkcyjnych. (2 punkty)
- Zintegrowanie z aplikacją działania edytora kodu Codemirror. (2 punkty)
- Zmodyfikowana funkcjonalność Codemirror tak, aby wprowadzone zostało kolorowanie sekcji kodu w C, które zawierają wstawki asemblerowe. Kolorowanie to ma być niezależne od innych kolorowań w edytorze. (1 punkt)
- Przyjemny w obsłudze interfejs użytkownika, pozwalający na logowanie się korzystającego. (1 punkt)
- Pokrycie kodu testami przynajmniej na poziomie 60% (1 punkt)
  pokrycie będzie wyliczane za pomoca komendy
  ```
  > coverage run --source='source_name' manage.py test app_name
  ```
  i pokazywane za pomocą polecenia
  ```
  > coverage report
  ```
- Pokrycie kodu testami przynajmniej na poziomie 70% (1 punkt) mierzone, jak powyżej.
- Pokrycie kodu testami przynajmniej na poziomie 80% (1 punkt) mierzone, jak powyżej.
- Pokrycie kodu testami przynajmniej na poziomie 85% (0,5 punktu) mierzone, jak powyżej.
- Pokrycie kodu testami przynajmniej na poziomie 90% (0,5 punktu) mierzone, jak powyżej.

## Ścieżka II - aplikacja napisana w alternatywnym zrębie

W tej ścieżce wykonania zadania będziecie mogli w praktyce zapoznać się z innym zrębem tworzenia aplikacji WWW. W tej wersji zadania należy przedstawić alternatywne rozwiązanie dotychczas wykonanych zadań 1-3 z użyciem zrębu WWW innego niż Django.

**Uwaga!!! Zrąb, jaki będzie użyty w tym rozwiązaniu, należy uzgodnić z prowadzącym laboratorium. Prowadzący laboratorium ma prawo nie zgodzić się na sprawdzanie rozwiązania w zaproponowanym przez Was zrębie – prowadzący nie są omnibusami i nie muszą znać wszystkich technologii tworzenia aplikacji WWW.**

- Aplikacja powinna w sekcji *Wybór pliku* wyświetlać dostępną w bazie danych strukturę katalogów. Sekcja poiwnna być zaimplementowana jako odpowiedni fragment wzorca strony (lub wzorzec), który jest wypełniony danymi pochodzącymi z bazy danych.
- Aplikacja powinna umożliwiać dodanie pliku do bazy, podzielenie go na sekcje zgodnie z wymienionymi we wcześniejszych zadaniach kategoriami (powinno się to dziać częściowo automatycznie, ale też powinna być możliwość

określania sekcji ręcznie; w tym dodawania sekcji, oznaczania fragmentu pliku jako sekcji). Operacje te powinny być dostępne w którymś z menu na pasku menu.
- Aplikacja powinna umożliwiać dodanie katalogu do bazy. Operacja ta powinna być dostępna w którymś z menu na pasku menu.
- Aplikacja powinna umożliwiać skasowanie pozycji w katalogu (pliku, innego katalogu), przy czym plik czy katalog nie powinien być zupełnie usuwany z bazy danych, a jedynie należy go oznaczać jako niedostępny za pomocą znacznika dostępności. Operacja ta powinna być dostępna w którymś z menu na pasku menu oraz być połączona z odpowiednim mechanizmem wskazywania pliku lub katalogu.
- Aplikacja powinna w polu *Fragment kodu* wyświetlać zawartość pliku `<filename>.c` uzyskanego w wyniku działania polecenia

  ```
  # sdcc -S <filename>.c
  ```

  odpowiednio wzbogaconego przez inne opcje, jakie zostały określone w interfejsie użytkownika, przy czym `<filename>.c` odpowiada obecnie wybranemu plikowi. Wynik z pliku `<filename>.asm` powinien zostać podzielony na zakresy oddzielone kreskowanymi liniami. Przy czym najechanie myszką na sekcję powinno spowodować jej wyróżnienie. Osobny rodzaj wyróżnienia ma dotyczyć nagłówka sekcji, a osobny treści. Wyświetlanie tej sekcji powinno być zapewnione przez odpowiedni fragment wzorca lub wzorzec.
- W dolnej części ekranu powinny być udostępnione cztery taby. Oto opis ich zawartości
  - Pierwszy tabulator (zatytułowany STANDARD) powinien wskazywać, z jakim standardem powinna być zachowana zgodność kompilatora (co najmniej C89, C99, C11). Powinno być możliwe wybranie jednego z nich. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny dotyczyć wybranego tutaj standardu.
  - Drugi tabulator (zatytułowany OPTYMALIZACJE) powinien zawierać listę dostępnych rodzajów optymalizacji (co najmniej 3). Powinna być możliwość określenia wybranego zestawu optymalizacji. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny generować kod zgodnie z zestawem optymalizacji.
  - Trzeci tabulator (zatytułowany PROCESOR) powinien zawierać listę dostępnych w SDCC architektur procesorów (co najmniej MCS51, Z80 i STM8). Powinno być możliwe wybranie jednego z nich. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny dotyczyć wybranego tutaj procesora.
  - Czwarty tabulator (zatytułowany ZALEŻNE) powinien zawierać listę opcji kompilatora, które są zależne od procesora. Dla każdego wybranego procesora powinna się tutaj znajdować możliwość wybrania trzech opcji właściwych dla niego (np. dla procesora MCS51 może się tutaj znajdować możliwość wyboru docelowego modelu programu - small, medium, large, huge model programs).
- Menu powinno zawierać opcję pozwalającą na wykonanie pełnej kompilacji obecnie wskazywanego pliku. Przebieg powinien uaktualniać zawartość pola *Fragment kodu* oraz dawać możliwość zapisania wyniku kompilacji na lokalnym dysku.

### Punktacja wariantu II

- Powiązanie wyboru pliku w sekcji Wybór pliku z zawartością sekcji Tekst programu - 1 p.,
- Nawigacja po katalogach i plikach, dodawanie ich i usuwanie - 1 p.,
- Automatyczny i ręczny podział pliku na sekcje - 1 p.,
- Obsługa sekcji Fragment kodu - 1 p.,
- Prawidłowa obsługa wyboru standardu - 0,5 p.,
- Prawidłowa obsługa wyboru optymalizacji - 0,5 p.,
- Prawidłowa obsługa wyboru procesora i opcji zależnych od niego - 1 p.,
- Prawidłowa obsługa kompilacji - 1 p.,
- Brak niepotrzebnego przeładowywania strony - 1 p.,
- Ukrywanie i odkrywanie fragmentów pola *Fragment kodu* - 1 p.,
- Utworzenie infrastruktury do testowania aplikacji i umieszczenie w niej podstawowych testów - 1 p.

Dodaj pracę

## Status przesłanego zadania

| | |
|---|---|
| **Status przesłanego zadania** | Nie przesłano jeszcze zadania |
| **Stan oceniania** | Nieocenione |

| | |
|---|---|
| **Pozostały czas** | Pozostało 15 dni 2 godzin |
| **Ostatnio modyfikowane** | - |
| **Komentarz do przesłanego zadania** | ▸ Komentarze (0) |