Pontrendszer Bsc szakdolgozatok bírálatához

	Szempontok	**Pontszám**
1.	A megoldott programozási feladat nehézsége.	4
2.	A dolgozat felépítése, nyelvezete, külalakja.	4
3.	Felhasználói dokumentáció	4
4.	Megoldási terv (fejlesztői dokumentáció)	4
5.	Megvalósítás (fejlesztői dokumentáció)	4
6.	Tesztelés (fejlesztői dokumentáció)	4
7.	Szoftver futtatása	6
	Összesen:	30

Ha valamelyik szempont nulla pontot kap, akkor a végösszeg is nulla.

Útmutató

Az itt megadott leírást rugalmasan kell kezelni, hiszen egy dolgozaton nem mindig lehet számon kérni az egyes szempontok alatt felsorolt összes tényezőt. Fontos, hogy amelyik szempontra nem adjuk meg a maximális pontszámot, indokoljuk azt meg. Nincs ugyan konszenzus a pontszámok és az osztályzat kapcsolatáról, de talán használhatnánk az alábbi táblázatot:

0 -14: elégtelen

15-18: elégséges

19-22: közepes

23-26: jó

27-30: jeles

1. A megoldott programozási feladat nehézsége

Ha a feladat éppen teljesíti a szakdolgozatként elvárt nehézségi szintet, akkor az 1 pontot ér. Többlet pontok akkor járnak, ha például:

- A feladat megoldása igényelt a kötelező tananyagon kívüli ismereteket.
- A megoldás alkalmaz nem magától értetődő implementációjú algoritmusokat.
- Szükség volt-e a megoldáshoz különböző platformok vegyes alkalmazására.
- Nagy méretű adatbázist kellett létrehozni tárolt eljárásokkal, szerverágensek által kezelt elemekkel, kapcsolatot teremteni különféle egyéb szerverekkel (pl. Notification Server).
- A szoftver egy nagyobb rendszer komponense, ahol a környezetet is meg kellett ismerni a fejlesztés előtt.

2. A dolgozat felépítése, nyelvezete és külalakja

A hallgatónak be kell tartania az általános követelményekben megadott szempontokat. Ezen túlmenően az értékelésnél vegyük figyelembe:

- Tartalmazza-e a diplomamunka a kötelező részeket (felhasználói- és fejlesztői leírás, irodalomjegyzék, tartalomjegyzék)?
- A szöveg szerkezete, fejezetbontása helyénvaló, logikus, érthető-e? Milyen a dolgozat részeinek kapcsolata, összhangja?
- Mennyire áttekinthető a szöveg? Alkalmazza a hallgató a szövegszerkesztési technikákat (lapszámozás, címsorok, tartalomjegyzék, stílusok, futó fejléc, ábraszámozás, stb.)
- Milyen a dolgozat formája esztétikai, nyelvtani és stiláris szempontból?
- Az ábrák és a szövegek aránya megfelelő-e? Olvashatók, helyénvalók az ábrák?
- Megfelelően jelöli-e, hivatkozza-e meg a dolgozat a más forrásokból átvett anyagokat?

3. Felhasználói dokumentáció

Magába foglalja a telepítési- (vagy üzemeltetési-) és a végfelhasználói leírást. Ezek meghatározott célközönséghez szólnak, könnyen és gyorsan kell, hogy eligazítsák a felhasználót a program használatában!

Tartalma:

- A feladat rövid ismertetése (mire való a szoftver)
- Célközönség (kik, mikor, mire használhatják a programot)
- A rendszer használatához szükséges minimális, illetve optimális HW/SW környezet
- Első üzembe helyezés leírása – ha van ilyen –, a program indítása (kivéve, ha nem egy önálló alkalmazásról, hanem egy meglévő rendszer új komponenséről van szó). Itt ellenőrizzük, hogy a telepítési útmutató megfelel-e a valóságos telepítési folyamatnak.
- Általános felhasználói tájékoztató (például a szokásostól eltérő képernyő-, billentyű-, illetve egérkezelés leírása, teendők hibaüzenetek esetén stb.).
- A rendszer funkcióinak ismertetése. A feladat jellegéből fakadóan célszerű lehet ezt folyamatszerűen, képernyőképekkel alátámasztva bemutatni. A funkciókat ajánlatos a felhasználói szintek szerint csoportosítani. Itt vegyük figyelembe, hogy a leírás a fejlesztői dokumentációban meghatározott részfeladathoz illeszkedik-e, az ott meghatározott funkciókat/használati eseteket írja-e le?
- A rendszer futás közbeni üzenetei (hibaüzenetek, figyelmeztető üzenetek, felszólító üzenetek stb.) és azok magyarázata – az esetleges üzemeltetési teendőkkel együtt. Itt vegyük figyelembe, hogy tartalmaz-e biztonsági, illetve hibaelhárítási előírásokat?
- Egyéb, a szoftver használatához szükséges információk.

4. Megoldási terv

Ez a fejlesztői leírás része, a rendszerterv, amelyből az alkalmazás célja, felépítése és működése megérthető, ez alapján az alkalmazás forráskódja lényegében elkészíthető.

Tartalmazza a következő elemeket:

- Rendszer architektúrájának leírását (alrendszerek, rétegek bemutatása, az alkalmazott szabványok, technológiák, fejlesztő módszerek megadása, felhasznált eszközök és kész

komponensek definiálása). Az értékelésnél vegyük figyelembe, hogy mennyire válnak szét az alkalmazás rétegei (például felhasználói felület, logika, adatforrás)?

- Az adatbázis – feltéve, hogy van – leírását. Érdemes egy áttekintő diagammal szemléltetni a táblákat és a köztük levő kapcsolatokat, majd külön táblázatokban megadni az egyes táblák mezőszerkezeti leírását, az esetleges tárolt eljárások, függvények, triggerek, stb leírását.
- Modul és/vagy osztályszerkezet (fontosabb modulok és/vagy osztályok és azok metódusai, továbbá ezek kapcsolatának) leírását. Az egyes csomagok fő eljárásait illetve a fontos osztályok fő metódusait bemenő-, kimenőadat, tevékenység hármassal jellemezni kell.
- A felhasználói felület – feltéve, hogy van – tervét (a képernyő- és listaterveket, valamint a menütervet). Legyen egy áttekintő ábra, amely mutatja a képernyők (ablakok, weblapok) közti navigálási lehetőségeket, irányokat. Ki kell emelni a fontosabb felhasználói eseménykezeléseket.

5. Megvalósítás

A fejlesztői leírásnak a megvalósításról szóló része bemutatja, hogy milyen döntéseket kellett hozni a terv megvalósítása során (adatábrázolás, felhasznált komponensek, kódban alkalmazott nyelvi elemek, stb). A dokumentáció ne tartalmazza a forrásprogramot (legfeljebb csak fontosnak ítélt részleteit), elég azt a mellékelt adathordozón elhelyezni. A megvalósítás a fentieken kívül tartalmazza a komponens tervet (az alkalmazás fizikai komponenseinek kapcsolatrendszerét) és azok telepítésének módját.

Az értékelésnél vegyük figyelembe:

- A forráskód tartalma, szerkezete megfelel-e a tervnek?
- Mennyire ismeri a hallgató az adott fejlesztő eszközt (pl. korszerű, hatékony nyelvi elemek vannak-e túlsúlyban, vagy ehelyett bonyolult, nehézkes, körülményes és leginkább terjengős forráskódot eredményező nyelvi elemek jellemzik a kódot)? Indokoltak-e a választott nyelvi elemek használata?
- Milyen a forráskód külalakja, mennyire áttekinthető (strukturáltság, bekezdések, tagolások, kommentezés stb.)?
- Mennyire módosítható a kód. Alkalmazza-e a hallgató a kód-újrafelhasználás nyelvi eszközeit (függvények, származtatás, generikus elemek)?
- Törekszik-e a hatékony adatábrázolásra?
- Mennyire öndokumentáló a kód, vagyis a választott azonosítók (pl. változónevek) mennyire beszédesek, konvencionálisak, a megjegyzések mennyire segítik a kódértést?
- Tartalmazza a szükséges ellenőrzési, hibakezelési funkciókat, általában megoldott-e a kivételkezelés?
- Mennyire gazdálkodik jól az emberi és gépi erőforrásokkal, így például a felhasználó idejével és türelmével, a lemezkapacitással és a memóriakapacitással?

6. Tesztelés

Ez is a fejlesztői leírás része, amelynek a tesztelési szempontokat kell bemutatnia, és a tesztelés során szerzett tapasztalatokat összegeznie valamint a szoftver skálázhatóságáról készített elemzést kell tartalmaznia.

Az értékelésnél vegyük figyelembe, hogy a dokumentáció:

- Tartalmaz-e tesztelési terveket, teszteseteket (Ezeket csoportosíthatja rendszerteszt és modultesztek szerint illetve fekete és fehérdoboz megközelítéssel)?
- Beszámol-e olyan tanulságokról, amelyek alapján meg kellett változtatni a korábbi implementációs döntéseket, esetleg a terv egyes elemeit (az ilyen tapasztalatok nem rontják a dolgozat értékét)?
- Tartalmazza-e nagy adattömeg melletti futtatások értékelését?
- Elemzi-e a program által adott eredmény helyességét (különösen olyan optimalizációs feladatok esetén, ahol több helyes megoldást valamilyen célfüggvénnyel lehet rangsorolni)?
- Elemzi-e a program futásának hatékonyságát?

7. Szoftver futtatása

Helyesség:

- A feladat-meghatározásnak, illetve a tervnek megfelelően működik-e a program?

Robusztusság:

- Mennyire védett a felhasználói hibákkal szemben a program (elronthatatlan/hibatűrő)?
- Életszerű (nagy mennyiségű) adatok esetén is hatékony munkavégzést biztosít-e a felhasználó számára a program?
- Ha a rendszer „jóindulatú” tesztelés esetén elszáll, akkor a munka nem fogadható el.

Felhasználó-barátság (szabványos felület, kényelem):

- A program könnyen használható? Áttekinthető? Rugalmas? Esztétikus?
- A felhasználói felület támogatja a szakterületi feladat elvégzését?
- A feliratok konvencionálisak? Egyértelműek a megfogalmazások? Jó a helyesírás?
- Van-e segítség (helyzet-érzékeny súgó)?
- Indokolt esetben a műveletek megszakíthatók-e, illetve védettek-e megszakítás ellen?
- Ismeri, illetve figyelembe veszi a felhasználói tradíciókat?