Požiadavky na semestrálnu prácu

# Všeobecné informácie

## Základné požiadavky na semestrálnu prácu

1. Práca musí byť vytvorená ako aplikácia v technológii Java alebo Kotlin (nie webová aplikácia). Prijateľná je aj aplikácia pre iOS. V prípade, že aplikáciu nie je možné skompilovať v nástroji **Android Studio**, študent pri obhajobe musí predviesť kompiláciu na svojom počítači.
2. Zdrojové kódy a dokumentácia sa odovzdávajú cez **Github, GitLab, BitBucket**, alebo iný GIT systém. Vyučujúci **musí mať** prístup k zdrojovým kódom aplikácie. Android aplikácia musí byť spustiteľná na **emulátore**, ako aj na **fyzickom zariadení**. Android aplikácia, ktorá nespĺňa tieto podmienky, nebude hodnotená.

## Hodnotenie priebežného stavu sem. práce

1. V **7. týždni** semestra bude **priebežná kontrola** stavu semestrálnej práce.
2. Študent odovzdá vytvorenú prezentáciu **výhradne vo formáte PDF** v publikovanom zadaní MS Teams (záložka *Assignments*). Nie je potrebná osobná obhajoba.
3. Hodnotiť sa bude (max. 10 bodov):
   * Spracovanie prehľadu dostupných aplikácií podobného zamerania,
   * Analýza navrhovanej aplikácie (napr. tvorba prípadov použitia),
   * Návrh architektúry aplikácia (popis, návrh procesov, atď.),
   * Ukážka návrhu obrazoviek aplikácie (napr. *wireframe* modely).

## Termín obhajoby semestrálnej práce

1. Obhajoba hotovej práce je plánovaná na **13. týždeň** semestra.
2. Študent do zadania v *MS Teams* **vloží odkaz** na prácu umiestnenú na niektorom z GIT systémov.
3. Po dohode s vyučujúcim bude stanovený termín obhajoby práce. Práca musí byť obhájená **online** a **osobne**. Obhajoba semestrálnej práce v prihlásenom termíne bude pozostávať z prezentácie aplikácie, popisu tvorby aplikácie, orientácie v zdrojovom kóde, odpovedí na otázky, diskusie, atď.

## Git repozitár

1. Repozitár musí byť verejne dostupný.
2. Git repozitár musí obsahovať **master** vetvu (*branch*)**,** ktorá bude považovaná za stabilnú.
3. Pred odovzdaním práce, je potrebnéoznačiť **jeden** **commit** v *master* vetve značkou (*tag*) **final**. Tento *commit* bude skúšajúci považovať za finálnu verziu práce. Ak je aplikácia projekt vytvorený v *Android Studiu*, musí byť skompilovateľná a spustiteľná aj u hodnotiaceho vyučujúceho. Pri nedodržaní tohto pravidla nebude práca hodnotená.
4. Doporučujeme preto robiť aktívny vývoj v ďalšej, napr. **dev** vetve a následne robiť *merge* z *dev* do *master***,** až keď vytvoríte stabilnú, funkčnú verziu.

## Dokumentácia

1. Súčasťou semestrálnej práce musí byť aj dokumentácia. Dokumentáciu uložte do adresára **dokumentacia** vo vašom repozitári.
2. Dokumentácia musí obsahovať nasledovné časti:
   * Popis a analýza riešeného problému
     + Špecifikácia zadania, definovanie problému
     + Nájdenie min. dvoch podobných aplikácií a popis, v čom sa vaša aplikácia líši.
   * Návrh riešenia problému
     + Krátka analýza (napr. diagramy prípadov použitia)
     + Návrh aplikácie (napr. diagram tried)
   * Popis implementácie – **klaďte dôraz na bodovo hodnotené časti** aplikácie, tak, aby bolo z dokumentácie zrejmé, ktoré hodnotené časti ste využili a akým spôsobom (ukážka kódu, UML diagramy a pod.).
   * Zoznam použitých zdrojov – okrem štandardnej Android developer dokumentácie a kurzu predmetu.
3. Maximálna dĺžka dokumentácie je 15 strán. Bodové zrážky pri nedodržaní pravidiel:
   * Chýbajúca dokumentácia -20 bodov,
   * Zlá úroveň dokumentácie (neúplnosť, neprehľadnosť, veľké množstvo chýb, atď.) až do -10 bodov,
   * Aplikácia je iná ako bola navrhnutá na *Checkpoint1* -10 bodov
   * Zlé umiestnenie dokumentácie -3 body.
4. V prípade použitia jazykov *Java* alebo *Kotlin* kód musí byť okomententovaný v súlade s **javadoc** alebo **kdoc**. Projekt musí obsahovat adresár “**doc**”, do ktorého umiestnite **vygenerovaný** **javadoc** alebo **kdoc**. Bodové zrážky pri nedodržaní pravidiel:
   * Chýbajúca dokumentácia -10 bodov,
   * Nedostatočné dokumentačné komentáre v zdrojových kódoch do -5 bodov,
   * Zlé umiestnenie dokumentácie v rámci projektu -3 body.

# Hodnotenie Android aplikácie

1. Toto bodové hodnotenie sa vzťahuje na Android aplikácie. Aplikácie pre iné platformy (iOS) bude hodnotené individuálne.
2. Hodnotenie aplikácie sa bude skladať z nasledovných častí:
   * Zložitosť aplikácie – max. 45 bodov
   * Použitá programátorská technika – max. 25 bodov
   * Dizajn, UX aplikácie a výsledný dojem z aplikácie – max. 20 bodov
3. Pri hodnotení sa budú brať do úvahy len časti kódu, ktoré sú bodovo relevantné. Za zbytočnú implementáciu sa považuje niečo, čo je doplnené len, aby boli zarátané body, t.j. daná časť nemá užitočný prínos pre výslednú aplikáciu. Príkladom je aktivita so statickým obsahom napr. aktivity typu “*O aplikácii*”.
4. Časť aplikácie, ktorá je implementovaná **triviálne** napr. *webview* a iba do neho načítaná webová stránka, sa pri hodnotení nebude brať do úvahy.
5. Identické časti aplikácie s minimálnymi odlišnosťami nie sú brané do úvahy.

## Zložitosť aplikácie

1. **Povinne:**
   * Aplikácia musí správne reagovať na otočenie displeja z polohy na výšku na polohu na šírku (*portrait / landscape*) a naopak (pozor na uchovávanie stavu aktivity pred otočením, napr. pokiaľ je niekde použitý formulár, po otočení musia zostať polia vyplnené). V prípade nekorektného správania sa aplikácie, bodová zrážka do -10 bodov.
   * Zdroje ako texty, obrázky, multimédiá a pod. v projekte by mali uvedené ako *resources* (ak to nie je možné, je potrebné iné riešenie obhájiť), bodová zrážka do -5 bodov.
2. Aktivity a fragmenty – max. 6 bodov za každú aktivitu s prihliadnutím na jej spracovanie a komplexnosť.
   * Aktivity musia mať korektne implementovaný životný cyklus, korektne ukladať priebežný stav.
   * Maximálne je možné získať 24 bodov.
3. Za využitie komponentov *service, broadcast receiver, content provider* – max. 4 body za každý podľa ich spracovania a komplexnosti.
   * Maximálny počet je 16 bodov.
4. Využitie *AndroidX* komponentov:
   * *Data Binding, LifeCycles, LiveData(Mutable), Navigation, Paging, Room, ViewModel, WorkManager* – max. 4 body za každý použitý komponent (nie za viacnásobné použitie toho istého komponentu).
5. *Widget miniaplikácia*2 body za každú.
6. Notifikácia2 body za každú (max. 6 bodov).
7. Dynamické rozloženie (*dynamic layout*) – napr. zoznamy, mriežky, tabuľky:
   * Využitie zoznamu, mriežky alebo podobného *widgetu* s adaptérom:2 body za každý *widget*,
   * Využitie zoznamu, mriežky alebo podobného widgetu s použitím *RecyclerView*: 3 body za každý *widget.*
8. Použitie externého frameworku / knižnice – max. 4 body za každý.
9. Použitie senzora – max. 4 body za každý.
10. Použitie sieťovej komunikácie – max. 5 bodov.
11. Používateľské rozhranie – vytvorenie vlastných štýlov a tém – max. 3 body.
12. Dodatočné body za použitie technológie mimo rozsahu predmetu. Podmienkou je, že daná technológia musí byť zdokumentovaná v priloženej dokumentácii – max. 6 bodov spolu.

## Použitá programátorská technika

1. Návrh aplikácie, architektúra.
2. Dodržiavanie zásad kódovania (*coding standards*).
   * Logické pomenovanie identifikátorov,
   * Dostatočné komentáre v aplikácii,
   * Ošetrenie výnimiek,
   * Žiadny duplicitný kód a pod.
3. Správne uplatnenie princípov OOP.

## Dizajn a UX aplikácie

1. Výsledná aplikácia musí byť intuitívna na používanie.
2. Jednotlivé prvky a vstupné polia musia byť jasne označené.
3. Aplikácia musí mať primerane zložité ovládanie podľa typu riešeného problému.
4. Výsledný dojem z aplikácie – vzhľad, prepracovanosť, pohľad používateľa.

# Odporúčania pri tvorbe zdrojového kódu

1. Zdrojový kód by mal byť okomentovaný (dokumentačné komentáre nad každým verejným typom a štandardné komentáre všade tam, kde je to potrebné na pochopenie funkcionality)
2. Vhodná voľba názvoslovia identifikátorov a súborov (príklady zlých pomenovaní *textView1, textView2, button1, int K, int a, b, c, d,* ...)
3. Nepoužívať metódy s rovnakým alebo podobným kódom (vytvoriť univerzálny kód, napr. zoskupiť rovnaké bloky kódu do jednej metódy a pod.)
4. Dobrý objektový návrh (oddelenie vrstiev, nie všetko vložiť do jednej triedy). Dodržiavanie princípu oddelenia zodpovednosti, tj. nepísať celý kód do jednej triedy. Triedy reprezentujúce UI by mali obsahovať iba logiku, ktorá spracováva interakcie UI a operačného systému.
5. Optimálny kód, odstránenie nepoužitých premenných, importov, a pod.
6. Ošetrovanie výnimiek (aplikácia nesmie „spadnúť“, ani vyhodiť neošetrenú výnimku).
7. Formátovanie a prehľadnosť zdrojového kódu – používať úpravu formátovania pomocou vývojového prostredia.
8. Žiadne vulgarizmy v kóde, nielen v komentároch, dialógových oknách, notifikáciách, logovacích súboroch a pod.
9. Pomenovanie metód a tried stručné, ale výstižné podľa toho, čo je ich úlohou.
10. Ak je nejaký kód skopírovaný z internetu, uvádzať v komentári zdroj (URL odkaz).

# Záverečné upozornenia

1. Študent odovzdaním práce prehlasuje, že celý projekt (prácu) vypracoval samostatne.
2. Ak pri vypracovaní projektu použijete nejaký tutoriál, je to potrebné v dokumentácii uviesť.
3. Ak bude odhalený **plagiát**, bude udelená známka **Fx**, **bez možnosti opravy**. Za plagiát sa bude považovať i časť zdrojových kódov, ktoré budú obsahovať rovnaký alebo refaktorovaný kód inej práce, resp. dostupný z internetu (zmenené názvy identifikátorov, poradie deklarácie a definície premenných, metód, atď.) – výnimku tvoria časti známych kódov a návrhové vzory (v tomto prípade uveďte zdroj, napr. URL, z ktorého ste čerpali).
4. Práca pred hodnotením prejde kontrolou antiplagiátorskym systémom.
5. V prípade akýchkoľvek nejasností alebo neistoty nás kontaktujte.