**Házi feladat leírása (ellenőrzési technikák)**

***Áttekintés***

A megismert technikák (pl. kódanalízis, tesztelés, folytonos integráció) gyakorlása egy valós projekten.

* A házi feladat teljesítése *szükséges* feltétele az aláírásnak.
* A házi feladatot 4 vagy 5 fős *csapatokban* kell majd megoldani.
  + Csapatokat a hallgatók alkotnak.
  + Nem szükséges, hogy az összes csapattag ugyanabban a gyakorlati kurzusban legyen.
  + Elvárás, hogy minden csapattag egyenlő módon vegyen részt a feladat megoldásában.
  + Alapesetben a teljes csapat ugyanazt az értékelést kapja a házi feladatra, kivéve, ha a csapat jelzi az oktatóknak, hogy valamelyik csapattag nem vett részt megfelelően a munkában

A feladat során egy **GitHub**-on elérhető nyílt forráskódú projekten kell előre megadott feladatokat elvégezni. A csapat a jelentkezés után kapni fog egy, a választott csapatnévnek megfelelő nevű tárhelyet a BME-MIT-IET GitHub szervezetben. A kiválasztott projekt main ágán lévő kód aktuális állapotát ide kell majd bemásolni, és ezen kell a feladatokat később megoldani. A csapat admin jogot fog kapni a tárhelyen, de tilos átnevezni a tárhelyet vagy megváltoztatni a tárhely hozzáférési jogait. [*Megjegyzés*: A gyakorlatban inkább egy fork lenne a javasolt megoldás ilyenkor, de azért nem fork-ot készítünk a kiválasztott projektről, hogy független legyen a két tárhely, és pl. a csapat pull requestjei ne az eredeti projektnél kössenek ki véletlenül.]

**Lehetséges projektek**

* Vagy erről a listáról lehet választani: <https://share.mit.bme.hu/index.php/s/FS6APf9woLYe6aN>
* Vagy lehet saját ötletet is hozni (pl. saját hobbi-projekt, érdekes nyílt forrású projekt). A feltétel annyi, hogy publikus GitHub projektként elérhető legyen. Ebben az esetben előtte Micskei Zoltánnal emailben egyeztessétek.

**Elvégzendő feladatok**

Első lépésként a csapat ismerkedjen meg az alkalmazással:

* Nézze meg a dokumentációt, tanulmányozza a projekt felépítését, próbálja ki, hajtson végre egyszerűbb kézi teszteket (UI felülettel rendelkező alkalmazás esetén) vagy írjon egy-két egyszerűbb példakódot (könyvtárként felhasználható projekt esetén).
* A tárhely gyökérkönyvtárában egy IET-HF.md fájlban egy-két bekezdésben foglalja össze a csapat a saját szavaival, hogy mi a projekt célja, és nagyobb projekt esetén válasszon ki a funkcióknak egy részhalmazát, amivel később dolgozni fog majd.

A további feladatok két nagy csoportra bonthatók. Az egyes feladatok mellett megadtunk eszközöket, amik segíthetnek, de természetesen lehet bármi más, kapcsolódó eszközt használni.

**Technológia fókusz**

* Build keretrendszer beüzemelése, ha még nincs (Maven, Gradle...) + CI beüzemelése, ha még nincs (Actions, Travis, AppVeyor, Azure Pipelines...)

Mért nem ezt választottuk: Ez egy matematikai algoritmusokat tartalmazó könyvtár, így build keretrendszer beüzemelését nem éreztük kifejezetten a projekthez illő feladatnak.

* Manuális kód átvizsgálás elvégzése az alkalmazás egy részére (GitHub, Gerrit...) + Statikus analízis eszköz futtatása és jelzett hibák átnézése (SonarCloud, SpotBugs, VS Code Analyzer, Codacy, Coverity Scan...).
  + Mivel az eszközök rengeteg hibát és figyelmeztetést találhatnak, ezért elég azok egy részét megvizsgálni és ha a csapat minden tagja egyetért vele, akkor javítani. Törekedjetek arra, hogy különböző típusú, és lehetőleg nem triviális hibajelzéseket vizsgáljatok meg.

Készíti: Csaba

Ezért választottuk ezt: Nem találtuk nyomát, hogy kifejezetten alkalmaztak volna bármilyen statikus analízis eszközt, így kifejezetten érdekesnek találtuk a projektre bevezetni és megnézni mennyire sok dolgot fedez fel.

* Deployment segítése (Docker, Vagrant, felhő szolgáltatásba telepítés...)

Mért nem ezt választottuk: Továbbra is ez egy matematikai algoritmusokat tartalmazó library, így ennek a deploy-olásának nem látjuk az értelmét.

* Egységtesztek készítése/kiegészítése (xUnit...) + tesztek kódlefedettségének mérése és ez alapján tesztkészlet bővítése (JaCoCo, OpenCover, Coveralls, Codecov.io...)

Készíti: Márk

Ezt választottuk mert: A projekt már eleve tartalmazott ugyan UNIT teszteket, viszont ezek sok helyen nem értek el teljes lefedettséget és még egyesek hibába is futottak.

**Termék/felhasználó fókusz**

* Nem-funkcionális jellemzők vizsgálata (teljesítmény, stresszteszt, biztonság, használhatóság...)

Készíti: Moha

Azért ezt választottuk: Mivel algoritmusokról van szó, így a teljesítmény kifejezetten érdekes vizsgálati területnek találtuk, amely egyaránt fontos is.

* UI tesztek készítése (Selenium, Tosca, Appium...)

Mért nem ezt választottuk: Az általunk vizsgált projektnek nincsen felhasználói felülete, így ezen tesztek elkészítése nem lehetséges.

* BDD tesztek készítése (Cucumber, Specflow...)

Készíti: Zoli

Azért ezt választottuk: BDD tesztek készítésével még nem foglalkoztunk, így szerettük volna megismerni, és konzultáció után alkalmasnak is tekintettük erre a feladatra.

* Manuális tesztek megtervezése, végrehajtása és dokumentálása vagy exploratory testing

Mért nem ezt választottuk: Az általunk vizsgált projektnek nincsen felhasználói felülete, így ezen tesztek elvégzése és dokumentálásának nincs értelme.

Mindkét csoportból**legalább 2-2 feladat** elvégzése kötelező. A csapat eldöntheti, hogy melyik feladatokat akarja elvégezni (a projekt adottságait és az egyéni érdeklődéseket figyelembe véve). Javasolt, hogy egy-egy feladaton többen is dolgozzanak. A feladat elvégzése után **kötelező**, hogy legalább egy másik, a feladaton nem dolgozó csapattag ellenőrizze a munkát (pull request review keretében).

**FONTOS**A házi feladatra csapattagonként kb. 15 órát kell fordítani. Ezt az időt a csapat jól ossza be, és koncentráljatok azokra a feladatokra, amik az adott projekt szempontjából hasznosak, és amivel a legtöbbet tudtok meg az adott szoftver minőségéről vagy nagy eséllyel tudtok javítani rajta.

**Feladatok dokumentálása**

* Minden egyes kiválasztott feladathoz egy-egy **issue**-t kell felvenni, amiben nyomon lehet majd követni a feladat megoldásának alakulását. Az issue-ban egyeztessék a csapattagok, hogy ki mit csinál majd, és legyenek belinkelve a kapcsolódó főbb commit-ok vagy pull request-ek, és látszódjon, hogy ki ellenőrizte a végén az adott feladatot.
* Az elvégzett munka rövid dokumentációját a saját GitHub tárhely **doc** könyvtárába rakott fájlba kell írni. A fájlokhoz a [Markdown](https://guides.github.com/features/mastering-markdown/) vagy [AsciiDoc](http://asciidoc.org/) formátumot használjuk. Minden egyes kiválasztott feladatról legyen egy-egy külön fájl: egy-két bekezdés az elvégzett munkáról, képernyőkép (ha releváns), eredmények és tanulságok összefoglalása.

**Értékelés**

* A házi feladatot a félév végén be kell majd mutatni. (Ennek pontos beosztását majd a 14. héten intézzük.)
* Az értékelés nem mennyiségi, hanem minőségi alapon történik. Nem az a lényeg, hogy 15 vagy 20 teszt készült el, hanem, hogy mit tanult a feladatból a csapat, és sikerült-e a kiválasztott projekt minőségén javítani valamit. Hasonlóan figyeljetek arra, hogy hiába készít a csapat 20 egységtesztet, ha azok mind az egyszerű getter és setter metódusok működését ellenőrzik, akkor túl sok hasznuk nincs.
* Törekedjen mindenki arra, hogy jó minőségű munkát végezzen (szép kód, értelmes commit üzenetek, rendezett repository, áttekinthető dokumentáció, követhető issue-k...). Tehát nem csak az elkészült kód, hanem minden egyéb termék része az értékelésnek.

**Kérdések**

* Technikai jellegű kérdéseket a Teams csoport HF csatornájában lehet feltenni (így más is tud segíteni, és a válaszokat más is csapatok is láthatják).
* Egyéb, személyes kérdésekkel Micskei Zoltánt keressétek emailben.

**Határidők**

* 2022.05.19. feladatok elvégzése és dokumentálása
* 2022.05.26. pótlás határideje

**Bemutatás (2021-es információk)**

A munkát a félév végén a csapatnak be kell mutatni (az erre való jelentkezést majd Teams üzenetben publikáljuk):

* Ismertetni kell az elvégzett munkát, az egyes csapattagok hozzájárulását, az elért eredményeket, a kiválasztott projekt értékelését (mit tudtunk meg a végrehajtott ellenőrzési feladatok során a projekt állapotáról és minőségéről) és az elvégzett munka értékelését (mi sikerült, mit tanultak belőle, mi hiányzik még, mivel érdemes később bővíteni...).
* A bemutatóra nem szükséges fóliasorozatot készíteni, egy képernyőmegosztással kell egy Teams meetingben az egyes részfelatatokat bemutatni (tehát a tesztek lefuttathatók, statikus ellenőrző eszköz eredményei megnézhetők...).
* A bemutató jellegében olyasmi, mintha a csapat a főnökének vagy megrendelőjének mutatná be a munkáját, tehát készüljetek fel rá adatokkal, összesítésekkel és érvekkel, hogy kellő információt tudjatok adni kérdések esetén, és meg tudjátok győzni az elvégzett munka minőségéről.
* Alapesetben elvárt, hogy az összes csapattag részt vegyen. Ha bemutatási alkalom valakinek órarendi elfoglaltsággal ütközik, akkor neki nem kötelező megjelenni, de ezt is figyelembe véve a csapatnak olyan időpontot kell választani, amin legalább 3 csapattag részt vesz.
* Egy-egy csapatra kb. 10 perc fog jutni.
* A bemutatásra készíteni kell egy legfeljebb 2 oldalas dokumentumot, ami a végleges, részletes munkamegosztást tartalmazza (melyik csapattag pontosan milyen részfeladatokat végzett el, kb. mennyi időben). Figyelem, ennek konzisztensnek kell lennie a csapat GitHub tárhelyén lévő commit történettel! A dokumentum ezen kívül tartalmazza a tárgy nevét, a csapat nevét és a csapattagok nevét, NEPTUN-kódját és GitHub azonosítóját. Használható [ez a sablon](https://docs.google.com/document/d/15JoUoHTxSsW-VtG443a2YOY-uEaExV0vu3nS7ceuFmo/edit?usp=sharing). A dokumentumot Moodle-ra kell feltölteni majd PDF formátumban.