Teme proiect Testarea sistemelor software

Cerințe

- Pentru realizarea unui proiect se pot forma echipe de maxim 5 studenți. Este de preferat ca temele să fie realizate în echipă, dar nu obligatoriu.

Fiecare echipă va alege una dintre temele precizate mai jos. Fiecare temă poate fi aleasă de cel mult 3 echipe.

- Termene limită de predare:
  - fixare echipa și alegere temă: **24 martie, 23:59**,
  - 1/3 implementare: **14 aprilie, 23:59**,
  - 2/3 implementare: **28 aprilie, 23:59**,
  - tot proiectul: **12 mai, 23:59**.
  - Pentru fiecare termen limită depășit **pierdeți 0,75 puncte** din nota finală.
  - Toti membrii echipei vor primi nota echipei (exceptând cazul în care un membru al echipei a promis ceva, dar nu a realizat până la final).
- Componența echipei, tema aleasă, link către repository (unul din sistemele GitHub/GitLab/BitBucket) și email vor fi adăugate în fișierul (editabil) **echipe_TSS_optional_2024.gdoc**
- Repository-ul va conține:
  - implementarea aplicației și a testelor,
  - un fișier README (sau wiki) cu link-uri către:
    - prezentare (PowerPoint, Keynote, Powtoon, videoclip YouTube et al.).
    - videoclipuri cu demo aplicație și rulare teste (de exemplu, YouTube).
    - diagrame, dacă e cazul. Acestea vor fi realizate cu un utilitar (de exemplu, https://app.diagrams.net, https://www.lucidchart.com/pages/, yEd, Microsoft Visio et al.). Nu se acceptă imagini fotografiate/scanate.
    - raport despre folosirea unui tool de AI care ajută în timpul testării software (de exemplu, GitHub Copilot, chatGPT, Microsoft Copilot). Comparați suita proprie de teste cu cele autogenerate și evidențiați diferențele.
- Pot fi utilizate referințe cum ar fi: documentații oficiale ale tool-urilor, articole științifice și cărți (https://dblp.uni-trier.de, https://scholar.google.ro).

Articolele științifice și cărți care nu sunt free pot fi găsite la https://sci-hub.st, http://libgen.st

- Proiectul trebuie să fie original (a se vedea regulamentul de etică și profesionalism al FMI - pct. 11 de la pagina 3 despre incidente minore), adică NU poate conține *texte preluate mot-à-mot* din alte surse, dar bineînțeles că diverse surse de informare pot fi consultate și vor fi menționate la referințe, inclusiv pt. cele autogenerate. Citate cu ghilimele pot fi de asemenea incluse, dacă e cazul.

Raportul și testele vor fi verificate automat cu tool-ul de similitudine Turnitin. Textele preluate direct din alte surse nu vor fi luate in calcul, astfel ca nota va fi scăzută corespunzător.

- În realizarea proiectului pot fi utilizate instrumente cu IA pentru generarea de conținut text, cod, imagini, filme, pentru transformarea textului în muzică, film (de exemplu, chatGPT, Microsoft Copilot, Gemeni, rikigpt.com, jenni.ai, Microsoft Designer, DALL-E, InVideo), indicând explicit acest lucru.
- **Șablon Referinţe:**

[1] Nume, Prenume autor, *Titlu articol online*, `url`, Data ultimei accesări: …

[2] Nume, Prenume autor, *Titlu articol științific*, Nume revista, vol. X, nr. Y, An publicare, Pagini.

[3] Nume, Prenume autor, *Titlu carte*, Nume editură, An publicare.

Tema 1: Testare unitară în Python

• Utilizați un framework de testare unitară din Python pentru a testa funcționalitățile unei clase.
• Ilustrați strategiile de generare de teste prezentate la curs (partiționare în clase de echivalență, analiza valorilor de frontieră, acoperire la nivel de instrucțiune, decizie, condiție, circuite independente, analiză raport creat de generatorul de mutanți, teste suplimentare pentru a omorî 2 dintre mutanții neechivalenți rămași în viață) pe exemple proprii (create de echipă).

Tema 2: Testare unitară în C#

• Utilizați un framework de testare unitară din C# pentru a testa funcționalitățile unei clase.
• Ilustrați strategiile de generare de teste prezentate la curs (partiționare în clase de echivalență, analiza valorilor de frontieră, acoperire la nivel de instrucțiune, decizie, condiție, circuite independente, analiză raport creat de generatorul de mutanți, teste suplimentare pentru a omorî 2 dintre mutanții neechivalenți rămași în viață) pe exemple proprii (create de echipă).

Tema 3: Testare unitară în Java

• Utilizați un framework de testare unitară din Java pentru a testa funcționalitățile unei clase.
• Ilustrați strategiile de generare de teste prezentate la curs (partiționare în clase de echivalență, analiza valorilor de frontieră, acoperire la nivel de instrucțiune, decizie, condiție, circuite independente, analiză raport creat de generatorul de mutanți, teste suplimentare pentru a omorî 2 dintre mutanții neechivalenți rămași în viață) pe exemple proprii (create de echipă).

Tema 4: Testare unitară în JavaScript

• Utilizați un framework de testare unitară din JavaScript pentru a testa componentele JavaScript.

Tema 5: Testare unitară în PHP

• Implementați teste unitare pentru un backend PHP. Acoperiți cazuri de testare pentru diferite funcționalități ale serviciului.

Tema 6: Testarea interfeței grafice a unei aplicații web

• Să se realizeze un studiu comparativ a cel puţin 2 framework-uri de testare, evidenţiindu-se utilitatea şi uşurinţa în folosire a fiecăruia. Pe baza unor exemple de cod, se vor evidenția diferențele dintre tool-uri.

- Folosiți un framework pentru a realiza teste ale interfeței grafice a unei aplicații web (teste de funcționalitate, navigare, validare formular, compatibilitate browser et al.). Motivați alegerea făcută.

Tema 7: Testarea unei aplicații mobile

- Să se realizeze un studiu comparativ a cel puţin 2 framework-uri de testare, evidenţiindu-se utilitatea şi uşurinţa în folosire a fiecăruia. Pe baza unor exemple de cod, se vor evidenția diferenţele dintre tool-uri.
- Folosiți un framework pentru a realiza teste ale unei aplicații mobile (teste de funcționalitate, UI, performanță, securitate, compatibilitate diferite sisteme de operare et al.). Motivați alegerea făcută.

Tema 8: Testarea unei rețele Blockchain

- Implementați teste unitare, teste de integrare, teste de performanță și teste de securitate pentru o rețea Blockchain.

Tema 9: Testare automată folosind un robot

- Dezvoltați scripturi de testare (automatizare teste de regresie, teste de performanță, teste de securitate et al.) care să fie rulate de către roboți.

Tema 10: Îmbunătățirea testării unitare cu IA

- Utilizarea IA pentru optimizarea și îmbunătățirea testelor unitare existente, asigurând o acoperire cât mai eficientă a codului sursă.
- Implementarea unui sistem care să identifice automat punctele critice ale codului și să prioritizeze testele în funcție de acestea.

Tema 11: Analiza și testarea strategiilor în jocuri bazate pe decizii

- Dezvoltarea unui sistem care simulează jocuri bazate pe decizii, utilizând concepte din teoria jocurilor.
- Implementarea algoritmilor pentru strategii de joc.
- Crearea unui mediu de testare pentru a evalua performanța strategiilor.

Tema 12: Studiul și prezentarea unui articol științific

- Alegeți un articol științific publicat în ultimii 4 ani pe o temă de testare software (https://dblp.uni-trier.de, https://scholar.google.ro).
- Studiați și înțelegeți conceptele și metodele prezentate în articol.
- Discutați cu profesorul pentru a primi acordul.

- Elaborați un raport care să descrie pe scurt conceptele și metodele abordate în articol.
- Creați un demo care să ilustreze, prin exemple practice, conceptele și metodele din articol. Folosiți exemple proprii, nu cele din articol.

Tema 13: Teme conexe cu testarea sistemelor software

- Echipele pot propune alte teme de actualitate (strategii de generare teste, instrumente; de exemplu, automatizarea testelor, testarea continuă a integrării (CI), testarea securității sistemelor, testarea performanței sistemelor, testarea în cloud) sau testarea unor sisteme software dezvoltate anterior la alte discipline. Discutați cu profesorul pentru a primi acordul.