VILNIUSCODINGSCHOOL

Automatinio testavimo mokymų PRE-KURSAS

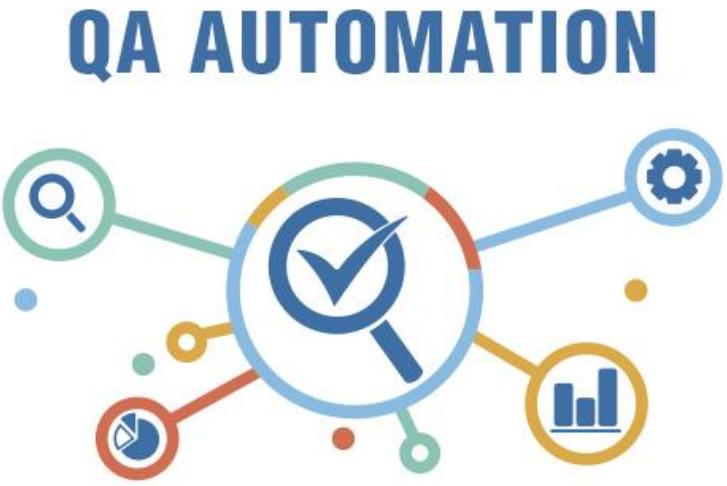
Vilnius Coding School 2021

Įvadas

Prie-kursas skirtas Vilnius Coding School studentams, pasirinkusiems kursą **„Automatinio testavimo mokymai“.**

Tai savaitės trukmės programa, skirta savarankiškam darbui namuose, prieš prasidedant intensyviems testavimo mokymams. Pre-kurso medžiagą sudaro teorinė medžiaga, praktiniai uždaviniai bei programinės įrangos instrukcijos, padėsiančios įsisavinti automatinio testavimo pradmenis bei kuo geriau pasiruošti savarankiškai iki mokymų pradžios.

Prie-kursas orientuotas į jokios programinės įrangos kūrimo/testavimo patirties neturinčius studentus. Turintiems bent jau minimalią patirtį, susijusią automatiniu testavimu, vis tiek pravartu pasikartoti pradmenis. Visų svarbiausia, prisijungus į pirmąją paskaitą užduoti klausimus, kurie kilo skaitant prie-kursą.



Automatinis testavimas

Šiuolaikinis programinės įrangos kūrimo ir paleidimo veikti tempas yra be proto greitas, todėl kiekvienas testuotojas turėtų žinoti bent jau automatinio testavimo pagrindus. Pateiksime, ką turėtumėte žinoti ir kuria kryptimi eiti, jei susidomėjote testavimo automatizavimu.

Testavimo išmanymas ir įrankiai

Pradedant domėtis testavimo automatizavimu, labai svarbu žinoti, kokius įrankius pasirinkti, tačiau tam visų pirma reikia gerai išmanyti testavimą apskritai. Tik tada bus lengva suprasti, kuriuos programinės įrangos testavimo veiksmus norėtum automatizuoti. Be to, nereikia pamiršti, kad kai kurie veiksmai visada bus geriau atlikti rankiniu būdu, o kai kurie kaip tik geriau veiks parinkus tinkamą įrankį.

Programinės įrangos kūrimas

Testavimo automatizavimas tikrai palengvina programinės įrangos kūrimo procesą, tačiau pats automatizavimas irgi yra programinės įrangos kūrimas, nes tu kuri įrankius, kurie tau padės dirbti efektyviau. Taigi, kiekvienam testuotojui verta išmanyti bent vieną, ar dar geriau, kelias programavimo kalbas. Net jei pačiam ir neteks rašyti kodų testavimo automatizavimo įrankiams, išmanant, kaip jie veikia, dirbti bus paprasčiau.

Testavimo automatizavimo strategija

Kaip ir darant bet kokį darbą, kad jis vyktų sėkmingai, pirma reikia turėti strategiją. Jei nežinote, kaip susikurti savo veiksmingą strategiją, pirmiausia pabandykite atsakyti į šiuos klausimus:

- Kodėl man reikia testavimo automatizacijos, t. y. kokią problemą bandau tuo išspręsti?
- Ką konkrečiai automatizuosiu?
- Kas sukurs automatizavimo įrankius?

- Kada reikės turėti rezultatą?
- Kaip įgyvendinsiu savo tikslą?

Ką dar reikia išmanyti?

Kad galėtumėte automatizuoti testavimą, neužteks vien programavimo ir testavimo žinių. Jums taip pat pravers išmanyti šiuos inžinerinius įrankius:

- Projektų kompiliacijų įrankius (pavyzdžiui, „Maven“ arba „Graddle“, skirti „Java“).
- Versijų kontrolės sistemas („Git“ „TFS“, „Subversion“).
- Nenutrūkstamos integracijos platformas („Jenkins“, „GitLab“).
- Imitacijos įrankius („WireMock“, „Hoverfly“).

Labai tikėtina, kad bet kokiam testavimo automatizacijos projekte turėsi panaudoti kažką iš aukščiau išvardintų įrankių, todėl tikrai verta su jais susipažinti.

Pagrindiniai rankinio ir automatinio testavimo skirtumai

Interneto puslapiai ir mobiliosios programėlės turėtų veikti nepriekaištingai, būtent todėl testavimas yra neišvengiamas dalykas programuotojų kasdienybėje. Testavimas gal būti automatinis arba rankinis. Šių dviejų tipų testavimo skirtumus ir aptarsime šiame įraše.

Automatinis ir rankinis testavimas iš esmės skiriasi vienu pagrindiniu dalyku – kas atlieka šį testą – žmogus ar tam skirtas įrankis. Testuojant rankiniu būdu analitikai ir inžinieriai turi būti itin įsitraukę į įvairius testuojamo produkto kūrimo ir paleidimo etapus. Testuojant automatiniu būdu testuotojai turi rašyti scenarijus, kurie testus atlieka automatiškai.

Automatinis testavimas:

Daugiau testavimo per trumpesnį laiką. Tai pats didžiausias automatinio testavimo pranašumas prieš rankinį. Būtent dėl to, pasirinkus automatinį testavimą, didėja produktyvumas. Automatinis testavimas visų pirma buvo sukurtas tam, kad išspręstų pagrindinę rankinio testavimo problemą – laiko trūkumą. O kuo toliau, tuo labiau ši problema darosi aktualesnė, nes IT srityje darbų, kurie reikalauja testavimo, apimtys auga milžinišku greičiu.

Mažiau rankomis atliekamų užduočių. Kam yra tekę testuoti rankiniu būdu, tas puikiai žino, kiek daug pasikartojančių veiksmų tenka atlikti. Visgi automatinis testavimas taip pat reikalauja rankinio darbo (testuotojas turi sukurti testavimo scenarijaus pagrindą), tačiau jo yra nepalyginamai mažiau. Scenarijai paprastai rašomi su tokiomis programavimo kalbomis kaip „JAVA“ ar „Python“ ir, svarbiausia, jie gali būti panaudojami ne vieną kartą.

Didesnė testavimo apimtis. Automatinis testavimas leidžia pasiekti geresnius rezultatus, nes vieno testo metu gali aptikti daug daugiau klaidų. Skirtingai nei testuojant rankiniu būdu, automatinis testavimas padeda iškart testuoti visą programėlę, prietaisą ar programinę įrangą, o didesnė testavimo apimtis, reiškia ir rezultatyvesnį darbą.

Rankinis testavimas:

Visgi kartais pasitaiko situacijų, kai testuotojui reikia gerai išmanyti visą testavimo procesą, atkreipti dėmesį į mažiausias detales. Tą gali padaryti tik viską atlikdamas pats nuo pradžių iki pat testavimo proceso pabaigos. Kartais testavimo procesas būna toks sudėtingas, kad dar eigoje reikia išbandyti ir keisti įvairius scenarijus.

Automatinio testavimo galimybės dabar yra tikrai didelės ir vis auga, tačiau kartais būna situacijų, kai automatinio testavimo programos aptinka klaidas, kurių iš tiesų nėra, o tada testuotojai turi dar daugiau darbo, bandydami išsiaiškinti, ar rasta klaida yra teisinga ar ne.

Taigi, nors vis dar pasitaiko situacijų, kai rankinis testavimas yra neišvengiamas, automatinis testavimas yra tai, ką renkasi vis daugiau testuotojų, kad galėtų susidoroti su dideliais užduočių kiekiais ir aptikti kuo daugiau klaidų.

Automatinių testų rašymas, vykdymas ir rezultatų analizė

Automatiniai testai rašomi pagal iš anksto sugalvotus testavimo scenarijus. Rašant testų kodą stengiamasi naudoti jau egzistuojančius sukurtus metodus ir funkcijas. Reikėtų vengti kodo dubliavimo.

Kadangi automatinių testų rašymas yra programavimas, reikėtų laikytis gerų programavimo praktikų ir taisyklių.

Nuoroda į gerąsias C# kalbos praktikas: https://vaibhavbhapkarblogs.medium.com/c-best-practices-48e984cfdac0

Automatiniai testai yra vykdomi (angl. Execute, Run) pagal testuotojo poreikį, kai norima patikrinti ar mūsų naujai rašomas testas teisingai vykdo nurodytus žingsnius, arba kai norima ištestuoti puslapio funkcionalumą ir patikrinti, ar neatsirado naujų klaidų. Taip pat reikėtų pabandyti pakeisti testo kodą taip, kad jis specialiai nepraeitų (angl. Failed). Tada galia patikrinti, ar testas išvis gali „nulūžti“ ir parodyti puslapio klaidas.

Analizuojant įvykdytus testus turėtumėme lengvai suprasti, kodėl automatinis testas nepraėjo – ar tai tikrai nauja internetinio puslapio klaida, kurios anksčiau nebuvo, o gal tai „pasenęs“ testas, kurį reikėtų atnaujinti pagal pasikeitusį WEB puslapio funkcionalumą.

Automatiniai testai yra „gyvas“ projektas, kurį reikia prižiūrėti ir atnaujinti kartų su programos ar internetinio puslapio pakeitimais, kitaip po kurio laiko jie taps neaktualus (angl. Outdated) ir nebeturės savo vertės.

WEB elementai

Kiekvienas internetinis puslapis turi savo elementus, tai gali būti mygtukai, įvedimo laukai, nuorodos į kitą puslapį ir pan. Tam, kad būtų galima atskirti ir suprasti, koks elementas yra puslapyje, kiekvienas turi savo žymą (angl. Tag).

Daugiau apie elementus ir jų žymas galima perskaityti čia: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

Rašant testus, dirbama su konkrečiais puslapio elementais, kurie yra unikalūs būtent tam puslapiui. Automatizavimo karkaso pagalba mes galim atkartoti visus veiksmus, kuriuos gali atlikti vartotojas atidaręs norimą puslapį savo naršyklėje: spausti mygtukus, įrašyti tekstą į laukėlį, pasirinkti datą iš kalendoriaus ir pan.

Privaloma tiksliai nurodyti, su kurio elementų mes dirbsim ir kurį veiksmą atliksim. Tam, kad pasiekti norimą elementą, reikia savo testo kode aprašyti to elemento selektorių (angl. Selector) ir vėliau iškviesti norimo veiksmo karkaso funkciją.

Daugiau apie selektorius:

https://www.browserstack.com/guide/css-selectors-in-selenium

Įvadas į programavimą. Kas yra C# ir kam jis naudojamas?

1999 m. kompanija „Microsoft“ pristatė modernią ir greit išpopuliarėjusią programavimo kalbą C#, kad būtų galima patogiai kurti įvairias programas, skirtas tiek serveriams, tiek kompiuteriams, tiek telefonams. Dėl paprastos sintaksės šią kalbą lengva išmokti, be to, ji panaši į kitas objektinio programavimo kalbas, tokias kaip Java ar C++, todėl, prireikus, nesunkiai perprasi ir jas.

C# yra lanksti ir plačiai pritaikoma kalba

Naudojant C# galima programuoti įvairiausias sistemas, išmokęs šią kalbą galėsi kurti:

- *Darbalaukio programas*
- *Windows servisus*
- *Interneto servisus ir programas*
- *Žaidimus*
- *Daiktų interneto programas*
- *Dirbtinio intelekto programas*

C# programavimo kalbos logotipas

C# yra skirta produktyviam darbui

Ši programavimo kalba turi daugybę suderinamų patogių įrankių, tokių kaip programavimo aplinka Visual Studio arba kompiliatorių, kuris padeda rasti klaidas dar prieš paleidžiant veikti kodą. Visi šie dalykai yra labai svarbūs, norint dirbti sklandžiai ir patogiai, pačiam neieškant sprendimų kiekvienai mažai problemai.

C# yra paprasta naudoti ir lengvai suprantama

Ši kalba buvo sukurta remiantis C, C++ ir Java kalbomis, tačiau pagrindinis tikslas buvo sukurti kalbą, kurią naudojant programuoti būtų patogu ir paprasta. Kadangi šios kalbos kūrėjų tikslas buvo sukurti lengviau įsisavinamą kalbą, tad ne tik naudoti, bet ir išmokti šią kalbą taip yra šiek tiek lengviau.

Tinka įvairioms operacinėms sistemoms

Kadangi C# sukūrė kompanija Microsoft, ilgą laiką su šia kalba buvo galima kurti tik Microsoft operacinei sistemai. Tačiau naudojant .NET core galima programuoti ir Linux bei Mac operacinėms sistemoms, o tai dar labiau praplečia C# galimybes.

C# yra brandi, populiari ir gerai išvystyta kalba

C# kalbai jau beveik 20 metų. Per tą laiką ji smarkiai patobulėjo ir išpopuliarėjo. Dabar ji yra tarp penkių populiariausių pasaulyje programavimo kalbų. Naujausia, jau aštunta C# versija pasižyminti itin ryškiais patobulinimais, buvo išleista 2019 metų rugsėjį. Šiuo metu programuotojai laukia naujų tiek C#, tiek jo framework‘o .NET patobulinimų.

C# kalbą kuria Microsoft

Microsoft yra tikras IT grandas, kuris stengiasi tobulinti ir gerinti savo kūrinį – C# kalbą. Microsoft yra išsakę savo tikslus visada tobulinti šią kalbą, kad ji atitiktų besikeičiančius visų ar didžiosios dalies programuotojų poreikius, tačiau niekada nepakeisti jos iš esmės, kad ji būtų atpažįstama ir visi žinotų, ko iš jos tikėtis.

C# yra atviro kodo kalba

Paskutinės dvi C# versijos buvo kuriamos naudojant atviro kodo modelį ir tolimesnės versijos bus kuriamos lygiai taip pat. Kiekvienas, dirbantis su C#, gali palikti atsiliepimus ir siūlyti naujas funkcijas specialiai tam skirtame oficialiame C# Github puslapyje. Taip pat idėjomis, sugeneruotos per oficialius kūrėjų susitikimus, visada pasidalinama su bendruomene.

C# turi nuostabią bendruomenę

C# programuotojų bendruomenė yra tikrai didelė. Internete gali rasti įvairių šios kalbos gerbėjų grupių, didžiausios iš jų narių skaičius siekia net 3 milijonus! Bendruomenės nariai noriai dalinasi savo atradimais vieni su kitais, vykdo susitikimus, taip pat padeda vieni kitiems gauti darbą. Be to, Microsoft skiria apdovanojimą tiems nariams, kurių indėlis tobulinant C# yra tikrai didelis.

Gali veikti naršyklėje

Seniau norint dirbti su C# reikėdavo parsisiųsti ir instaliuoti plugin‘us, tačiau dabar tai nėra būtina, nes naujausias .NET papildymas – Blazor framework‘as – leidžia programuoti su C# naudojant naršyklę. Šis papildymas paleistas visai neseniai ir jam žadamas didelis populiarumas, nes programuoti bus galima bet kur, iš bet kurio kompiuterio.

C# paklausi kalba

Daugybė kompanijų visame pasaulyje naudoja C# svarbioms sistemoms kurti. Kadangi ši kalba vis tobulinama ir vis labiau prisitaiko prie programuotojų poreikių, ji bus naudojama dar ilgai nes yra paklausi. Taip pat ši kalba yra naudojama ir kompiuterinių žaidimų kūrimui su U

C# programinė įrangos įdiegimas

Windows vartotojams:

Atsisiųskite Visual Studio Community programavimo aplinkos įdiegimo programą: https://www.visualstudio.com/thank-you-downloading-visual-studio/?sku=Community

Atidarykite atsisiųstą failą ir tęskite įdiegimą tol, kol išvysite šį langą (1 pav.):

Jame pasirinkite ".NET desktop development" ir spauskite "Install" apačioje.

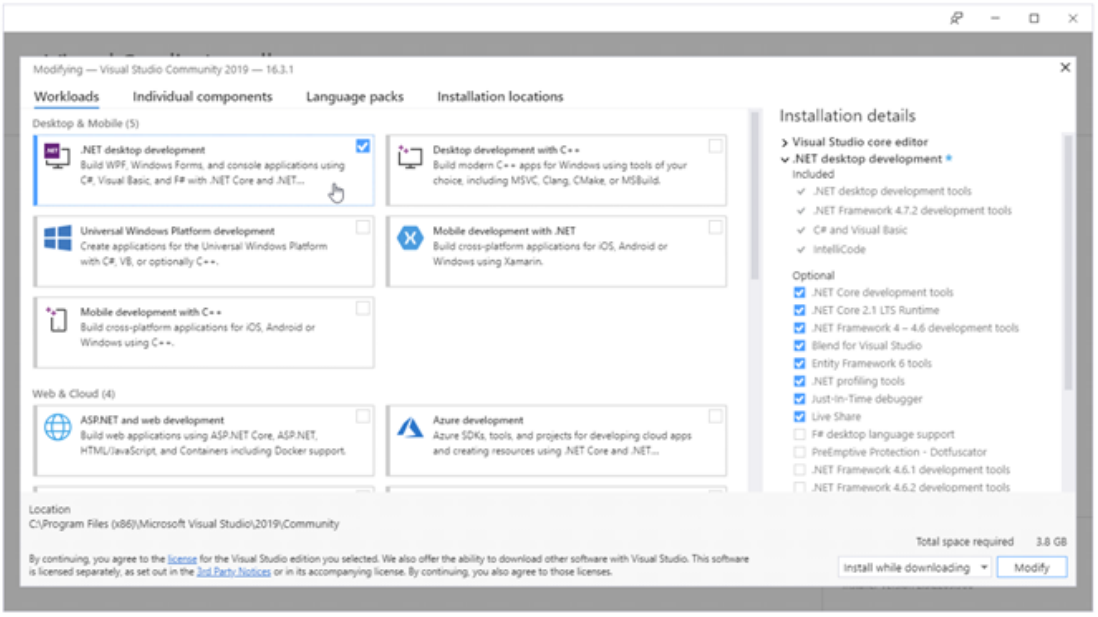
Tęskite įdiegimą iki pabaigos.

Mac vartotojams:

Atsisiųskite Visual Studio for Mac įdiegimo programą iš čia:

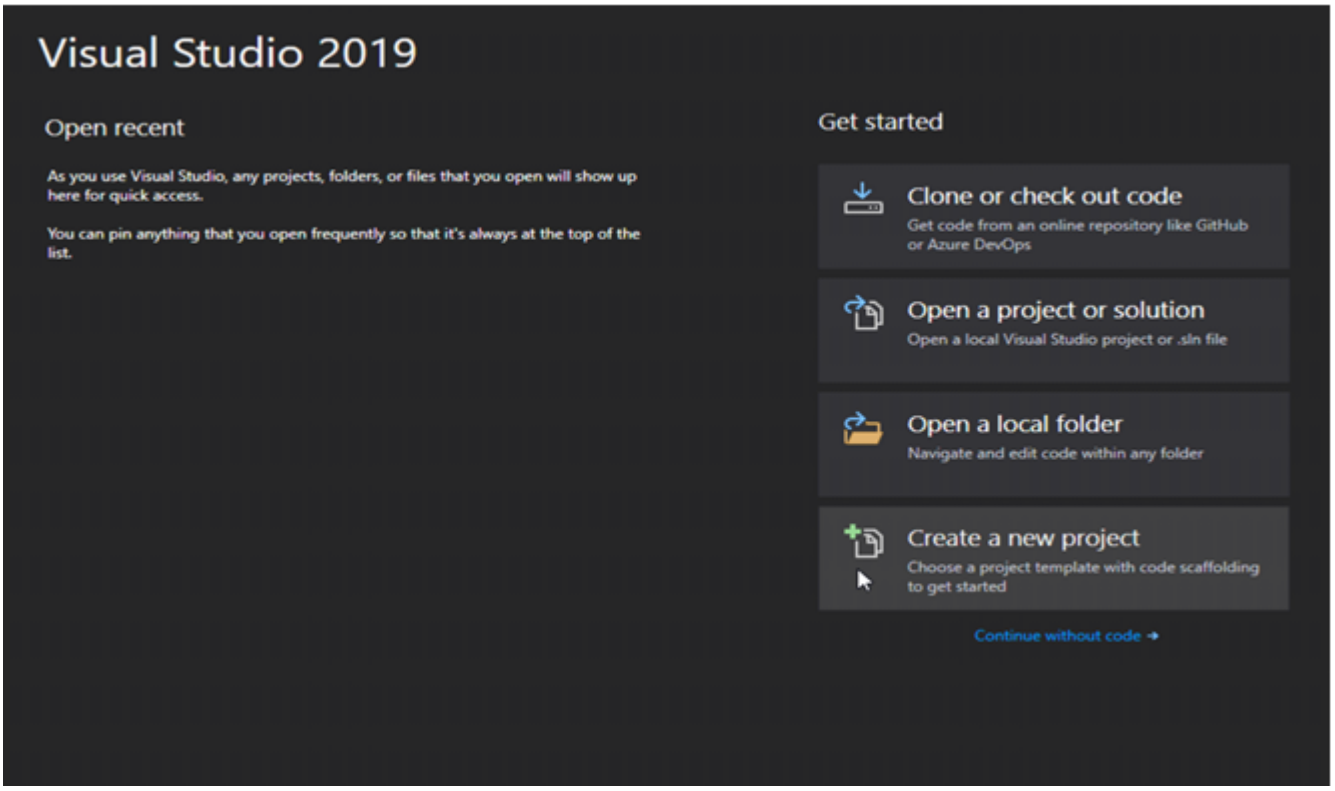
https://www.visualstudio.com/thank-you-downloading-visual-studio-mac/?sku=communitymac

Sekite nurodymus ir įprastai instaliuokite aplinką.

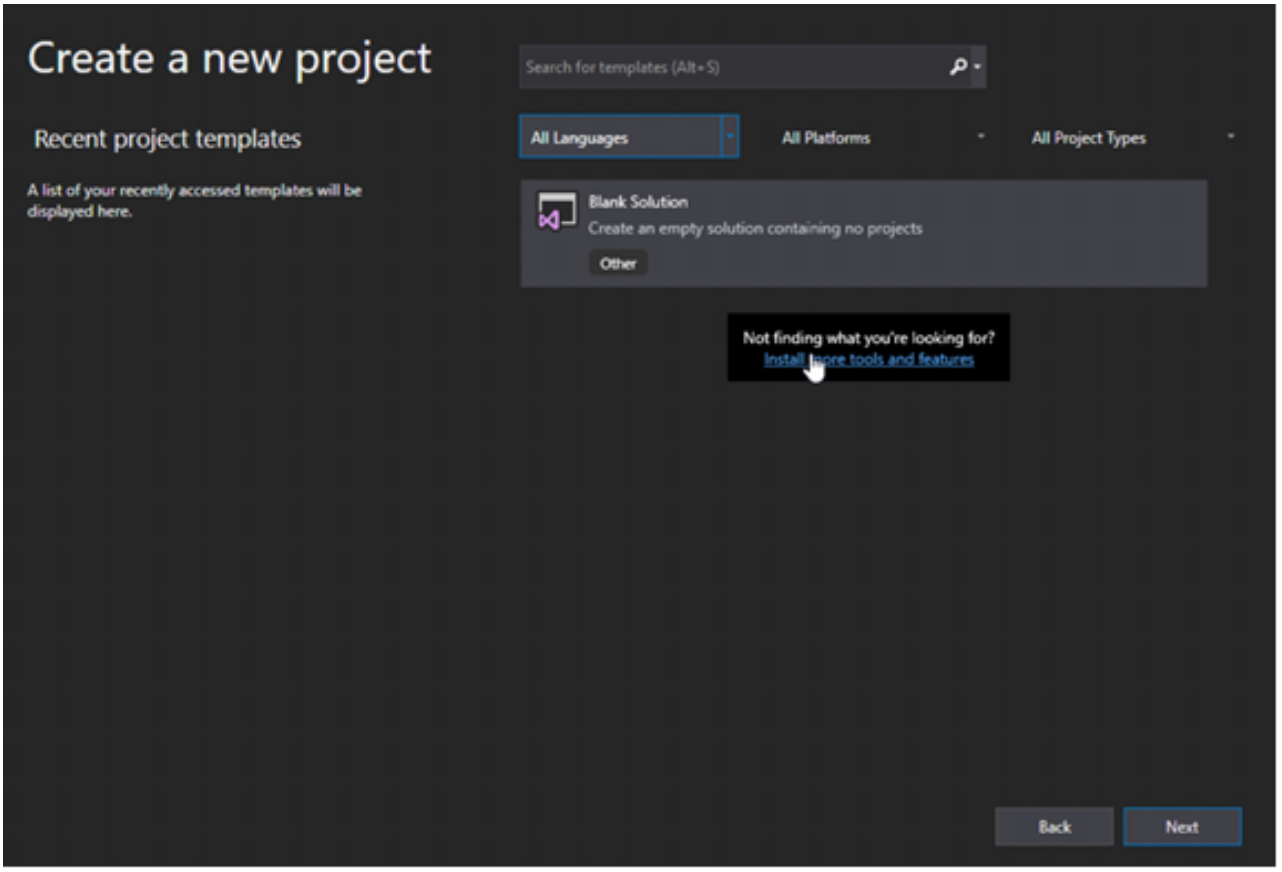


Kai diegimas bus baigtas, spauskite Launch.

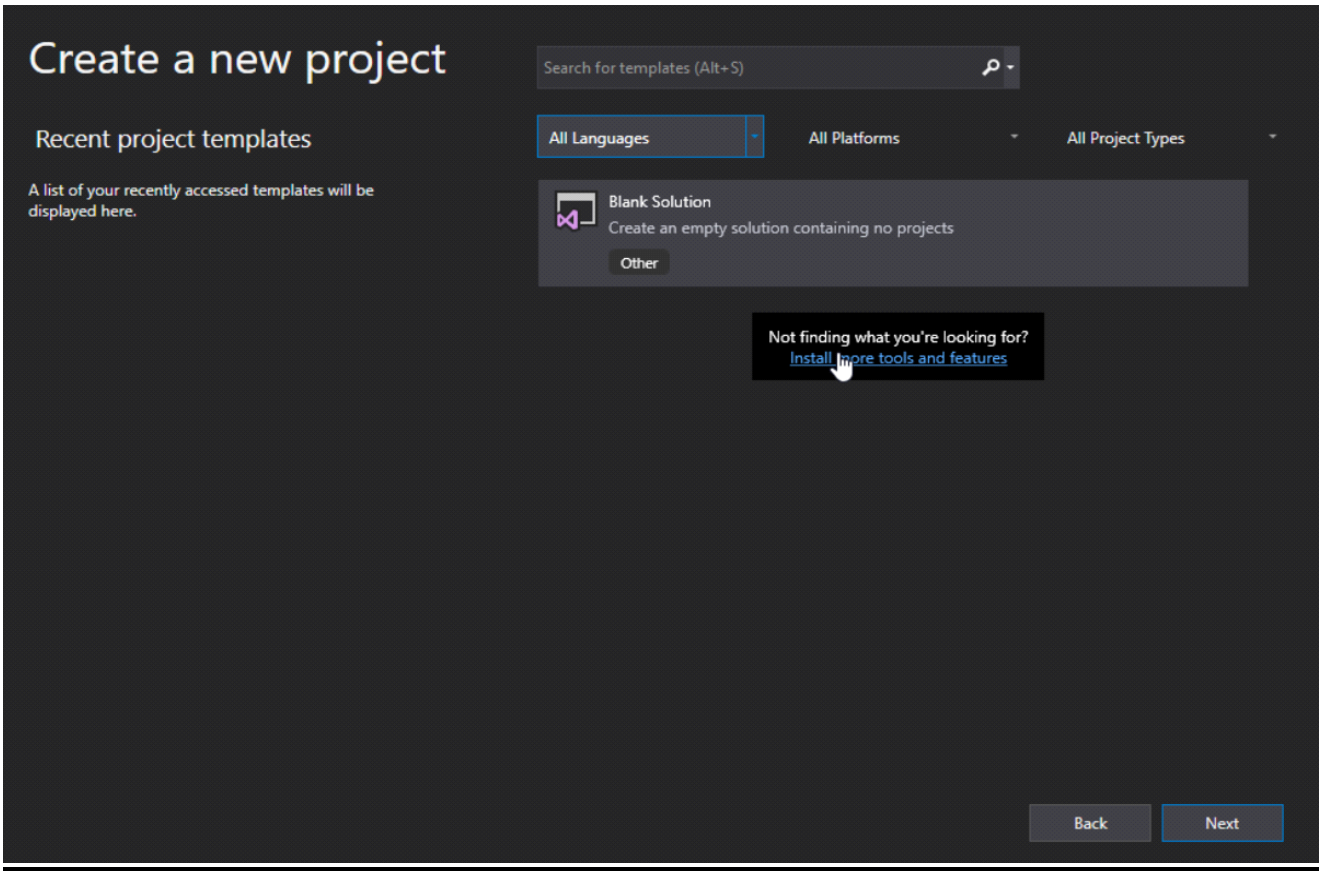
Atsidariusiame lange pasirinkite **Create a new project**:



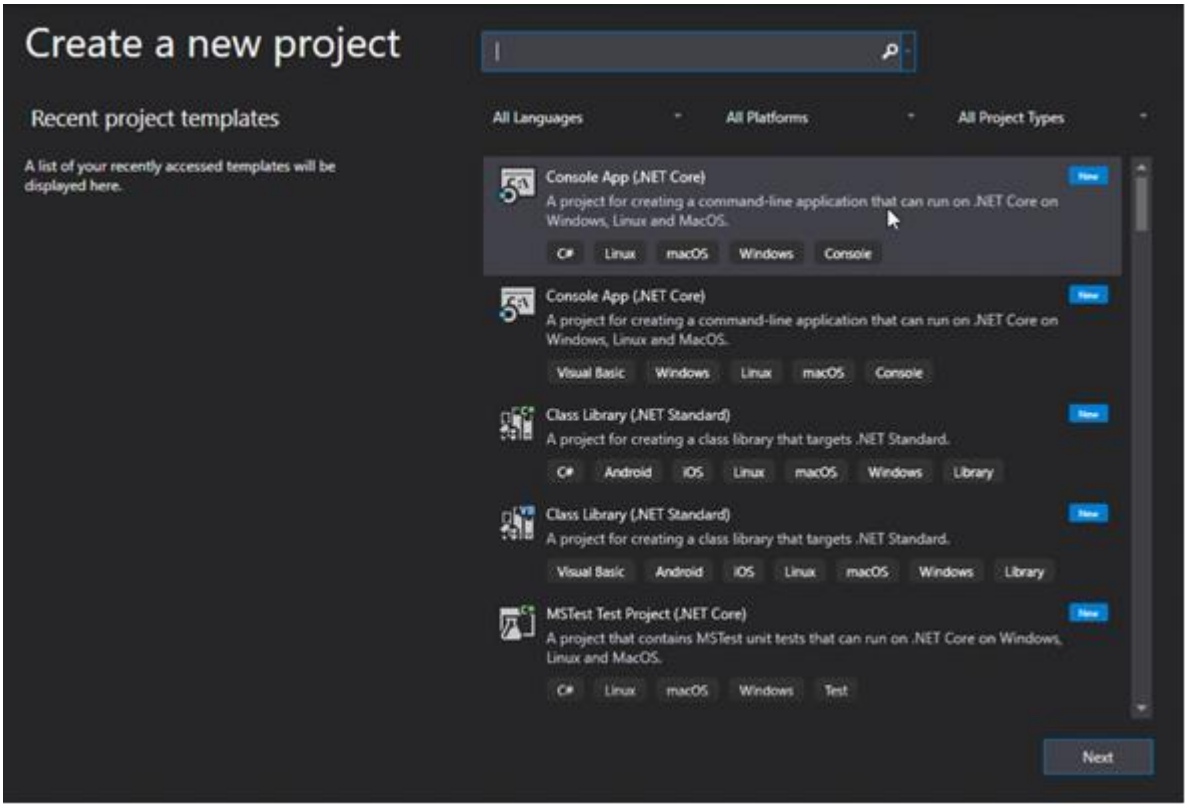
Paspauskite "**Install more tools and features**" nuorodą:



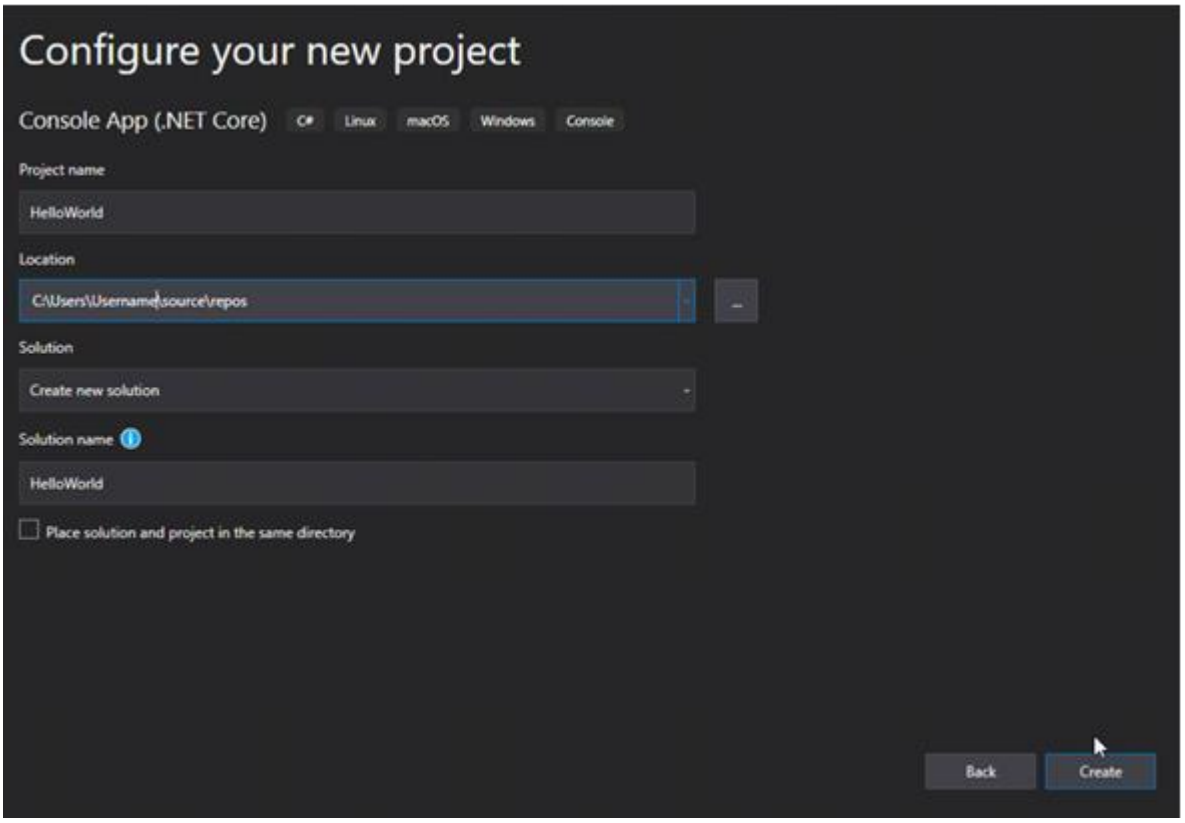
Paspauskite "**Install more tools and features**" nuorodą:



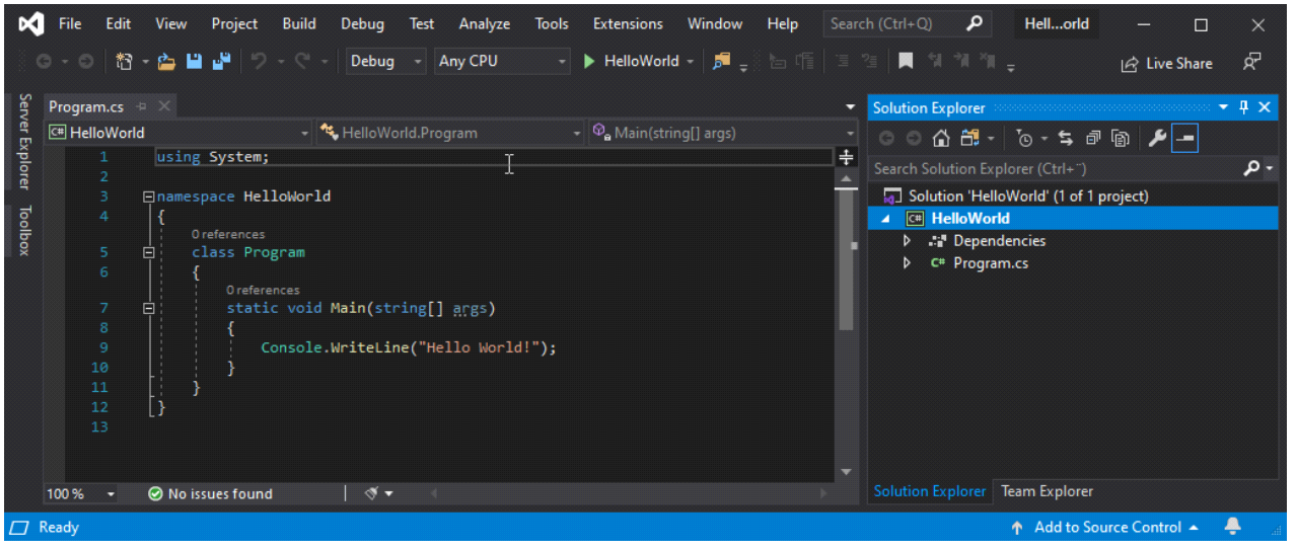
Iš sąrašo pasirinkite "**Console App (.NET Core)**" ir spauskite **Next**:



Įveskite sugalvotą projekto pavadinimą ir spauskite **Create**:



Visual Studio automatiškai sugeneruos dalį kodo:



Nesijaudinkite jeigu nesuprantate kodo viršuje – mes apie tai kalbėsime ir mokysimės paskaitų metu.

Bet pabandykime paleisti šią programą. Ją paleisti galite spausdami klaviatūroje F5 arba pasirinkdami iš Visual Studio meniu "Debug" -> "Start Debugging". Visa tai sukompiliuos ir paleis kodą. O rezultatas turėtų atrodyti taip:

```
Hello World!
C:\Users\Username\source\repos\HelloWorld\HelloWorld\bin\Debug\netcoreapp3.0\HelloWorld.exe (process 13784) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Sveikiname! Jūs paleidote savo pirmąja programą

Informacija savarankiškai pasimokyti:
https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/intro-to-csharp/hello-world

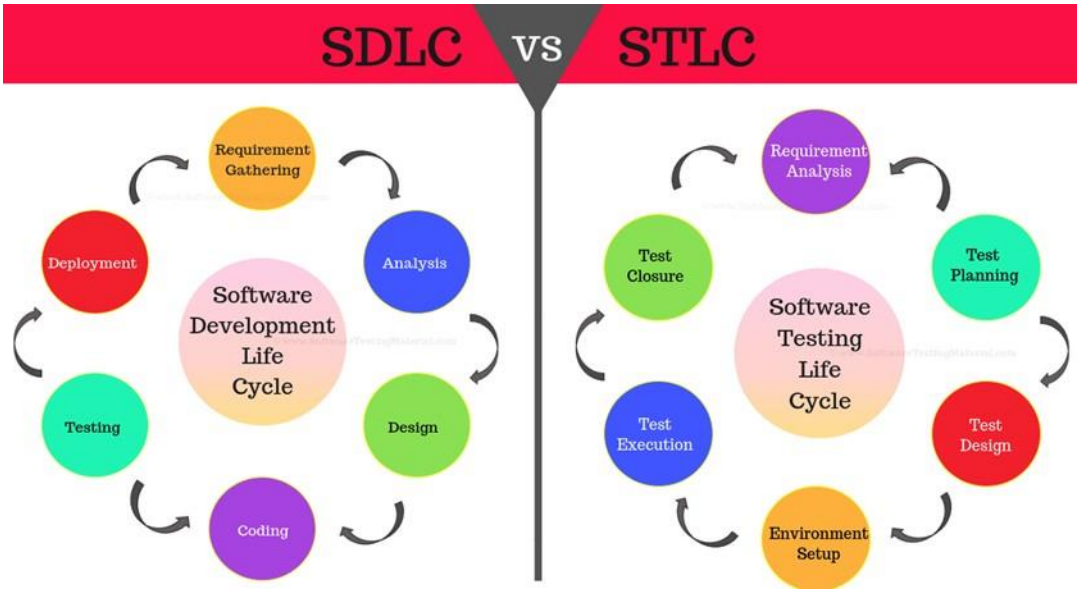
Pagrindinių sąvokų žodynas

- Programinė įranga (angl. software) – tai kompiuterio vykdomų instrukcijų seka, skirta tam tikriems veiksmams atlikti. Trumpinys – PĮ;
- Techninė įranga (angl. hardware) – tai fizinių komponentų visuma arba tos visumos dalis. Tai fizinės kompiuterio dalys;
- Testavimas – tyrimas kurio pagrindinė užduotis patikrinti ar programinės įrangos funkcionalumas atitinka jai iškeltus reikalavimus, naudojant testavimo atvejus;
- Kokybės užtikrinimas – procesas, tobulindamas PĮ kūrimą įvairiuose etapuose, siekiant minimizuoti galimų klaidų tikimybę;
- Bug – PĮ kodo klaidą kurios pasekmė neteisingas funkcionalumas;
- Funkcinis testavimas – programos funkcines dalies patikrinimas, atsakantis į klausimą
  „ar programos funkcionalumas veikia?";
- Nefunkcinis testavimas – programos nefunkcines dalies patikrinimas, atsakantis į klausimą "kaip veikia funkcionalumas?". Sąvoka apima tokius kriterijus kaip: greitaveika, saugumas, patogumas ir t.t.;
- Performance test – našumo arba greitaveikos testavimas, siekiantis nustatyti ar PĮ atitinka nefunkcinius reikalavimus. (pvz: visi programos puslapiai užsikrauna greičiau
  negu per 2 sekundes);
- Security test – saugumo testavimas. Patikrinimas ar PĮ naudojami duomenys yra saugus, nėra pasiekiami neautorizuotam vartotojui arba pametami.
- Black Box testing – testavimas naudojant tik įeities ir išeities duomenis, neturint informacijos apie PĮ vidinį veikimą;

- White Box testing - testų kūrimas ir atlikimas, remiantis žiniomis apie konkrečios PĮ įgyvendinimą ir vidinius veikimo principus.

- STLC - PĮ testavimo ciklas. Plačiau: https://www.guru99.com/software-testing-life-cycle.html

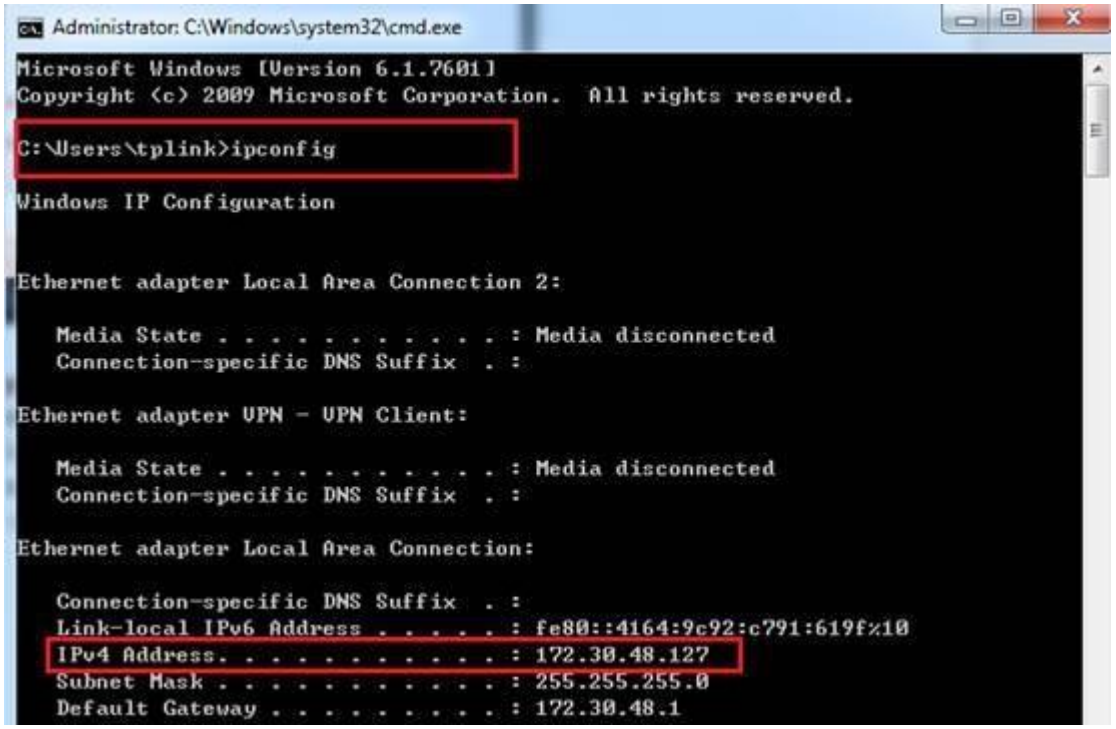
- SDLC – PĮ kūrimo ciklas. Plačiau: https://www.guru99.com/software-development-life-cycle-tutorial.html



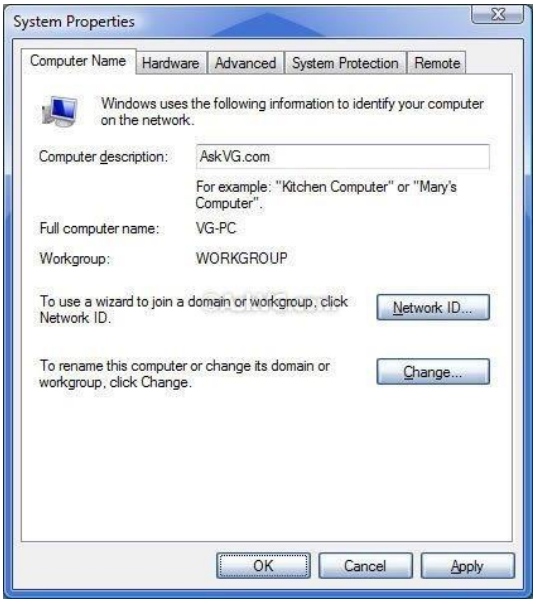
Pagrindiniai skirtumai tarp SDLC ir STLC.

- Automatinis testavimas - PĮ pertestavimas naudojant automatinius įrankius (testavimo skriptus). Plačiau https://www.guru99.com/automation-testing.html

- IP adresas - kitaip žinomas kaip interneto protokolo adresas, iš esmės yra skaičius priskiriamas kiekvienam tinklo įrenginiui, įskaitant maršrutizatorius, spausdintuvus, mobiliuosius įrenginius ir kompiuterius. Pagrindinė IP adreso funkcija - kompiuterių identifikavimas tinkle. Interneto protokolas reglamentuoja kompiuterių adresavimą, duomenų skaidymą į paketus prieš išsiunčiant ir surinkimą juos atsiuntus, paketų maršruto. Kaip sužinoti savo IP adresą žiūrėti čia.

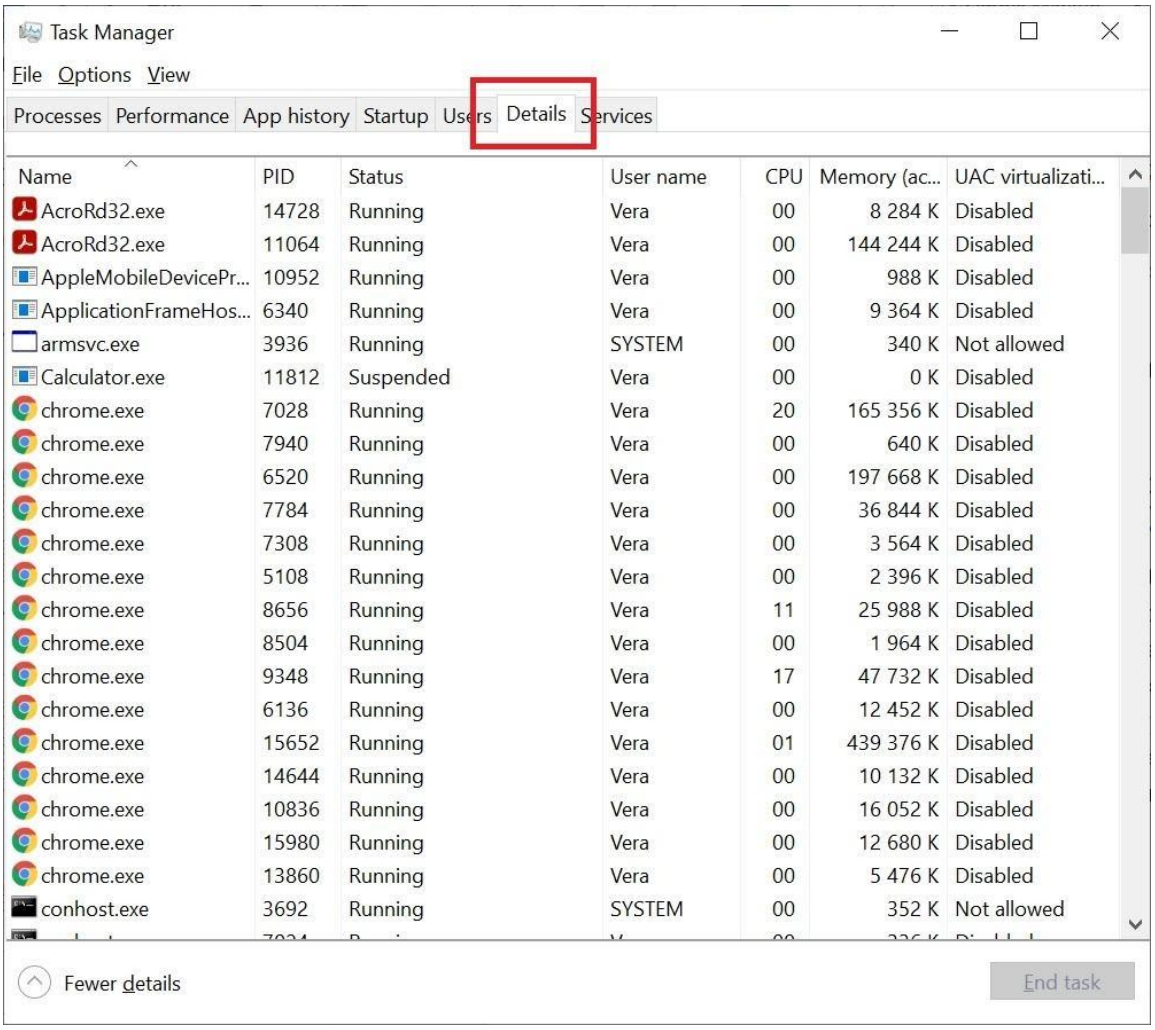


- Kompiuterio vardas – tai unikalus vardas, kuris suteikiamas kompiuteriui. Kaip sužinoti savo kompiuterio vardą - Win + r tada įveskite control sysdm.cpl, kad pereitumėte į reikiamą ekraną.



- F12 (Developer tools) - naršyklės įrankis, kuris leidžia pamatyti web puslapio elementus, slapukus ir kitą naudingą informaciją. Norit atidaryti „Developer tools“ reikėtų paspausti F12 klaviatūros mygtuką, arba [CTRL] + [SHIFT] + [I] klaviatūros mygtukų kombinaciją, arba dešinę pelytes mygtuką ir „Inspect“ opciją. Apačioje yra paveiksliukas Developer tools įrankio Chrome naršyklėje:

- GIT – atviro kodo paskirstyto versijų valdymo sistema, kontroliuojanti projekto versijas ir skirta sekti bet kokią failų aibę. Ši sistema buvo sukurta padėti programuotojams dalintis kodo pakeitimais programinės įrangos vystymo metu siekiant duomenų vientisumo, greičio ir patogaus nevientiso darbo (daugiau informacijos git ir github);
- Slapukai (angl. Cookies) - mažas tekstinis dokumentas, turintis unikalų identifikacijos numerį, kuris yra perduodamas iš interneto tinklalapio į lankytojo kompiuterio kietąjį diską, kad tinklalapio administratorius galėtų atskirti lankytojo kompiuterį ir matyti jo veiklą internete.
- Task manager – programa, kuri leidžia pamatyti, kokie procesai šio metu yra vykdomi jūsų kompiuteryje. Šio įrankio pagalba galima sustabdyti nereikalingus procesus arba išjungti pakibusias programas. Windows komanda: [CTRL] + [ALT] + [DELETE], Mac [CMD] + [ALT] + [ESC]. Norint pamatyti visus procesus, reikėtų nueiti į „Details“ skilti, kaip parodyta paveiksliuke.



- Versijų kontrolės sistema (angl. Version Control System, VCS) - yra programinė įranga, kuri išsaugo itin detalius failo arba failų pakeitimus, laiko šių pakeitimų istoriją atstatymui ir analizei ateityje.