REPUBLIKA E SHQIPERISË

UNIVERSITETI I TIRANËS

FAKULTETI I SHKENCAVE TË NATYRËS

DEGA: INFORMATIKE

Mikrotezë për mbrojtjen e gradës "Master i Shkencave" (MSC)

Tema: "*Testimi i automatizuar i aplikacioneve web*"

Punoi **Udhëhoqi**

Ledjon Cili **Dr. Suela Maxhelaku**

Tiranë, më xx/10/2023

ABSTRAKT

Ne ditet e sotme aplikacionet web luajne nje rol shume te rendesishem ne perditshmerine tone. Ato ndihmoje ne shume operacione biznesi dhe per kete arsye siguria dhe funksionaliteti i tyre eshte bere shume i rendesishem.

Kjo teme do te beje një analizë krahasuese të thelluar të tre teknologjive të automatizimit të uebit të njohura: Selenium, Cypress dhe TestComplete.

Selenium, një platformë me burim të hapur, ka qenë gjatë kohës një mjet i rëndësishëm në fushën e automatizimit të uebit. Cypress, një teknologji më e re, ka fituar popullaritet për thjeshtësinë dhe qasjen miqësore të zhvilluesit. Në anën tjetër, TestComplete, një mjet automatizimi komercial, ofron një zgjidhje të përgjithshme me karakteristika të përparuara dhe mbështetje.

Ky studim hulumton shumë aspekte të këtyre teknologjive të automatizimit për të ndihmuar organizatat të marrin vendime të informuara për strategjinë e tyre të testimi automatizimit. Krahasimi përfshin faktorë si lehtësia e përdorimit, ndryshueshmëria, aftësitë e skriptimit, kompatibiliteti me shfletues të ndryshëm dhe mbështetja për teknologjitë moderne të uebit. Po ashtu, studimi merr në konsideratë kostot e licencës, mbështetjen e komunitetit dhe integrimin në ekosistemin e tyre.

Përmes testimeve dhe eksperimenteve te kujdeseshme, kjo analize synon të vlerësojë pikët e forta dhe te dobeta te çdo teknologjie dhe të ofrojë shqyrtime në përputhshmërinë e tyre për skenarë të ndryshëm të testimit. Gjate kesaj analize do të prezantohen shembuj praktikë dhe raste përdorimi për të ilustruar aplikueshmërinë në botën reale të çdo teknologjie.

Gjetjet e kërkimit te kesaj analize synojnë të shërbejnë si një informacion i vlefshëm për ekipet e zhvillimit të softuerit dhe profesionistët e sigurisë së cilësisë që kërkojnë të përmirësojnë praktikat e tyre të testimit te automatizizuar të uebit. Në fund të kësaj analize, lexuesit do të kenë një kuptim më të qartë të aftësive dhe kufizimeve të Selenium, Cypress dhe TestComplete, duke u mundësuar të marrin vendime të informuara kur zgjedhin teknologjinë më të përshtatshme të automatizimit të uebit për nevojat e tyre specifike.

Tabela e permbajtjes

Lista e figurave:

Lista e tabelave

KAPITULLI 1

1.1 Qellimi i Studimit

Teknologjit e automatizimit te testeve po perdoren gjithmone dhe me teper ne procesin e zhvillimit te softuerve, pasi ato permiresojne procesin e testimit te produktit. Kto teknologji lejojne zhvilluesit te automatiozojne testet qe perseriten shpesh dhe shpesh jane TIME-CONSUMING. Ne vitet e fundit ne treg mund te gjejme shume teknologji te ndryshme falas ose me pagese qe ndihmojne ne automatizimin e procesit te testimit.

Pavarsisht avantazheve qe cdo teknologji testmi ka, secila nga ato ka specifikat e veta dhe ne varesi te natyres se projektit nje teknologji mund te jete me e mire se tjetra. Ne kete studim ne do te perpiqemi te krahasojme dy teknologji shume te njohura, qe jane falas si Selenium dhe Cypress dhe te vleresojme gjithashtu si qendrojne ato ne raport me njera tjetren dhe ne raport me Test Complete, nje mjet komercial per automatizin e testeve qe eshte me pagese.

1.2 Objektivat e ketij studimi

1. Te rishikojme literaturën mbi kornizat e automatizimit dhe krahasoni kornizat e ndryshme bazuar në veçoritë dhe aftësitë e tyre
2. Te vleresojme pikat e forta dhe të dobëta të Selenium,Cypress dhe Test Complete bazuar në kritere të tilla si lehtësia e përdorimit, pajtueshmëria, besueshmëria, shpejtësia dhe mbështetja për gjuhë të ndryshme programimi
3. Te krahasojme kornizat e automatizimit falas dhe me pagesë bazuar në veçoritë, modelet e çmimeve, mbështetjen e klientit dhe miratimin e përdoruesit
4. Te japim rekomandime për organizatat që po shqyrtojnë përdorimin e kornizave të automatizimit bazuar në analizën dhe vlerësimin e kornizave të ndryshme

Qellimi i ketij studimi eshte qe te bejme nje krahasim midis te githave teknologjive dhe mjeteve komericale te siperpermendura dhe te ndihmojme ne zgjedhjen e teknologjise/mjetit te duhur per projektin tuaj.

KAPITULLI 2

2.1 Cfare eshte testimi?

"Cilësia nuk është kurrë një rastësi; ajo është gjithmonë rezultat i përpjekjeve të mençura." – John Ruskin.

Testimi i software sipas ISTQB CTFL eshte procesi i vleresimit te nje software ose sistemi per te percaktuar nese permbush kerkesat e specifikuar dhe funksionon sic duhet. Procesi perfshin verfikimin dhe validimin e aspekteve te ndryshme te nje softueri si, funksionaliteti, perdorshmeria, besueshmeria, performanca dhe siguria.

Testimi i softureit eshte nje proces shume i rendesishem ne ciklin e zhvillimit te softuerit sepse ndihmon ne identifikimin e erroreve ne softuer dhe sigurohet qe pritshmerite e perdoruesve ne lidhje me cilesine e produktit permbushen. Testimi si proces mund te jete manual, qe do te thote testerat ndjekin nje set me hapa te parapercaktuar ose automatic duke bere ekzukitimin e testeve ne nje kohe me te shpejt dhe rezultate te sakta.

Qellimi i testimit te softuerit eshte te sigurohet qe produkti funksioni sic parashikohet, eshte i lehte per tu perdorur dhe permbush nevojat e perdoruesit. Testimi mund te ndihmoje gjithashtu per te identifikuar permiresime te mundshme ne produkt dhe te sugjeroje modifikime qe mund te permiresojne cilesine e produktit ne pergjithesi.

Shkurtimisht, testimi i softuerit eshte nje proces shume kritik qe ndihmon ne sigurimin e cilesise, funksionalitetit, performances te softuerit dhe sistemeve te ndryshme. Ai ndihmon ne identifikin e defekteve dhe sigurohet qe software permbush nevojat dhe pritshmerite e perdoruesve te tij, duke e bere ate nje aspekt esencial/shume te rendesishem ne ciklin e zhvillimit te nje softueri.

Gjate 5 dekadeve te fundit nje numer i madh sugjerimesh per principet e testimit jane bere per te ofruar disa udhezimie te pergjitheshme per te gjitha llojet e testimit.

2.2 Principet e testimit

1. Testimi tregon pranine e defekteve, por jo mungesen e tyre. -> Testimi redukton mundesine e defekteve ne aplikacion, por edhe nese asnje defekt nuk eshte gjetur, testimi nuk mund të provojë saktësinë e objektit të testit.

2. Te testosh gjithcka eshte e pamundur -> Te testosh gjithcka eshte e pamundur pervec ne disa raste specifike. Fokusi kryesor duhet te jene tekninat e testimit, prioritizimi i test cases, dhe testimi I bazuar ne rrezik sesa testimi i gjithckaje.
3. Testimi i hershem kursen kohe dhe para -> Testimi i hershem qe ne fillim te procesit te zhvillimit te aplikacioneve nuk do te shkaktoje defekte te mevonshme ne produktet e punes qe pasojne. Kostoja do te reduktohet pasi do te kete me pak defekte.
4. Grumbullimi i defekteve -> Shumica e defekteve gjenden në një numër të vogël komponentesh. Sipas parimit Paereto, i njohur gjithashtu si rregulli 80/20, 20% e komponenteve përmbajnë 80% të defekteve të gjetura.
5. Kujdes nga paradoksi i pesticideve -> Nese te njejtet teste perseriten shume here, ato nuk do te jene me efektive per te zbuluar defekte te reja. Prandaj eshte e rendesishme qe tested te rishikohen dhe modifikohen vazhdimisht.
6. Testimi varet nga konteksti -> Lloje të ndryshme të aplikacioneve testohen në mënyra të ndryshme
7. Mungesa e erroreve eshte ide e gabuar -> Nëse aplikacioni ose sistemi nuk është funksional dhe nuk i përmbush kërkesat e klientit dhe nevojat e përdoruesve, atëherë gjetja dhe rregullimi e defekteve është e padobishme.

2.3 Aktivitet dhe detyrat e testimit

Nje proces testimi konsiston ne nje sere aktivitetesh:

1. Plani i testimit – Konsiston ne definimin e objektivave te testimit dhe me pas ne zgjedhjen e nje metodologjie qe na ndihmon me se miri ne arritjen e ketyre objektivave.
2. Monitorimi dhe kontrolli I testimit -> Monitorimi I testimit perfshin kontrollin e vazhdueshem te te gjitha aktiviteve te testimit dhe krahasimin e progresin aktual perkundrejt planit. Kontrolli i testimit konsisiton ne marrjen e veprimeve te nevojshme per te arritur objektivat e testimit.
3. Analiza e testimit -> Identifikimi i **features** qe jane te testueshme dhe prioritizimi i tyre. Gjithashtu dhe vleresimi I rreziqeve.
4. Dizanji I testimit -> Perpunimi i kushteve te testimit ne raste testimi. Identifikimi I mbulimit te testimit, teknikat e testimit, definimi I datave qe do te perdoren per testim, mjediset e testimit, infrastruktura dhe mjetet qe do te perdoren. Pyetja qe ngrihet ne kete aktivitet: Si te testojme?
5. Implementimi I testimit -> konsiston ne krijimin e nje software per ekzekutimin e testeve.

6. Ekzekutimi I testit -> Konsiston ne ekzekutimin manual ose te automatizuar te testeve. Rezultate aktuale te testimit krahason me rezultatet e pritura. Identifikohen anomalite dhe analizohen per te gjetur shkakun e tyre.
7. Perfundimi I testimit -> Ky aktivitet ndodh zakonisht pas cdo release ose iteracioni. Konsiston ne Krijimin e nje raporti I cili u dorezohet paleve te interesuara.

2.4 Nivelet e testimit

1. Testimi i Komponentit (ndryshe Testimi I njesise): Përqendrohet në testimin e komponenteve të një sistemi ose aplikacioni në izolim dhe zakonisht kryhet nga zhvilluesit në mjediset e tyre të zhvillimit.
2. Testimi i Integritetit të Komponenteve (Testimi i Integritetit të Njësive): Përqendrohet në testimin e ndërfaqeve dhe ndërveprimeve midis komponenteve të një sistemi.
3. Testimi i Sistemit: Përqendrohet në sjelljen dhe aftësitë e përgjithshme të një sistemi ose produkti të tërë. Përfshin testimin funksional të detyrave prej fillimi në fund dhe testimin jo-funksional të cilësive të cilësisë. Mund të kryhet nga një ekip i pavarur të testimi dhe është i lidhur me specifikimet për sistemin.
4. Testimi i Integritetit të Sistemit: Përqendrohet në testimin e ndërfaqeve të sistemit nën test dhe sistemeve të tjera dhe shërbimeve të jashtme. Kërkon mjedise të përshtatshme të testimit, të preferueshme të ngjashme me mjedisin operacional.
5. Testimi i Pranimit: Përqendrohet në validim dhe demonstrimin e gatishmërisë së sistemit për implementim, duke u siguruar që të plotësojë nevojat e biznesit të përdoruesit. Mund të kryhet idealisht nga përdoruesit e synuar dhe përfshin formën e testimi të pranimit të përdoruesit, testimin operacional, testimin kontraktual dhe ligjor.

2.5 Llojet e testimit

1. Testimi Funksional: Testimi funksional vlerëson funksionet që një komponent ose sistem duhet të kryejë dhe ka për qëllim të verifikojë kompletësinë funksionale, saktësinë funksionale dhe përputhshmërinë funksionale. Kjo përqendrohet në "çfarë" duhet të bëjë objekti i testimi.
2. Testimi Jo-funksional: Testimi jo-funksional vlerëson atributet që janë të tjera përveç karakteristikave funksionale të një komponenti ose sistemi dhe ka për qëllim të kontrollojë cilësinë jo-funksionale të softuerit, duke përfshirë efikasitetin e performancës, kompatibilitetin, përdorshmërinë, besueshmërinë, sigurinë, mirëmbajtshmërinë, dhe përtëritshmërinë. Kjo përqendrohet në "sa mirë sjell sistemi".

3. Testimi i Kutise se zeze: Testimi i kutise se zeze është bazuar në specifikime dhe përdor dokumentacion të jashtëm për të krijuar testet. Qëllimi kryesor i testimit të kuq është të kontrollojë sjelljen e sistemit në krahasim me specifikimet e tij.
4. Testimi i kutise se Bardhë: Testimi i kutise se bardhë është bazuar në strukturë dhe përdor strukturën dhe implementimin e sistemit për të krijuar testet. Qëllimi kryesor i testimit të bardhë është të sigurojë që struktura e pasosë testohet në nivelin e duhur.
5. Testimi i Konfirmimit: Testimi i konfirmimit verifikon se një mangësi origjinale është ndrequr me sukses. Sipas riskut, versionin e ndrequr të softuerit mund ta testoni në disa mënyra, përfshirë:
    a. Ekzekutimin e të gjitha rasteve të testimit që më parë kanë dështuar për shkak të mangësisë, ose
    b. Shtimin e testeve të reja për të mbuluar ndonjë ndryshim që është bërë për të ndrequr mangësinë.
    c. Megjithatë, kur koha ose paratë janë të kufizuara për ndreqjen e mangësive, testimi i konfirmimit mund të kufizohet në thjeshtë ekzekutimin e hapat që duhet të riprodhojnë gabimin e shkaktuar nga mangësia dhe verifikimin që gabimi nuk ndodh.
6. Testimi i Regresionit: Testimi i regresionit konfirmon se nuk ka pasur pasojë të dëmshme nga një ndryshim, përfshirë një ndreqje që është tashmë testuar për konfirmim. Këto pasojë të dëmshme mund të prekin të njëjtin komponent ku u bë ndryshimi, komponentë të tjerë në të njëjtin sistem, ose madje sisteme të tjera të lidhura. Testimi i rregullimit mund të mos kufizohet vetëm në objektin e testimit, por mund të ketë lidhje edhe me mjedisin. Është e këshillueshme të bëhet një analizë e ndikimit për të optimizuar shtrirjen e testimit të rregullimit. Analiza e ndikimit tregon cilat pjesë të softuerit mund të ndikohen.
   Seritë e testimit të rregullimit ekzekutohen shpesh dhe në përgjithësi numri i rasteve të testimit të rregullimit rritet me çdo iteracion ose version të ri, kështu që testimi i rregullimit është një kandidat i fortë për automatizim. Automatizimi i këtyre testeve duhet të fillojë herët në projekt.
7. Testimi i mirembajtjes: Mirëmbajtja mund të përfshijë korrigjime, përmirësime të planifikuara, ose zgjidhje të menjëhershme të problemeve (hot fixes). Testimi i mirëmbajtjes përfshin vlerësimin e suksesit të zbatimit të ndryshimeve dhe verifikimin e mundësive të rregullimeve në pjesët e sistemit që nuk janë ndryshuar. Shtrirja e testimi të mirëmbajtjes përcaktohet nga rreziku i ndryshimit, madhësia e sistemit ekzistues dhe ndryshimi i kryer.

2.6 Diferencat midis testimit statik dhe atij dinamik

Testimi Dinamik: Përfshin ekzekutimin e softuerit për të verifikuar sjelljen dhe funksionimin e tij. Rastet e testimi janë të projektuara dhe ekzekutohen për të vlerësuar performancën e softuerit.

Testimi Statik: Përqendrohet në shqyrtimin dhe ekzaminimin e produkteve të punës, si kodi dhe dokumentacioni, pa ekzekutimin e softuerit. Qëllimi është gjetja e mangësive në fazat e hershme të zhvillimit dhe sigurimi i respektimit të standardeve.

	Testimi Statik	Testimi Dinamik
Natyra e Testimit	Është një lloj testimi ku kodi ose softueri nuk ekzekutohet. Përfshin shqyrtimin dhe ekzaminimin e produkteve të punës si kodi, specifikimet dhe dokumentacioni.	Në këtë lloj testimi, softueri ekzekutohet për të verifikuar funksionimin e tij. Rastet e testimi janë të projektuara, dhe kodi testohet gjatë ekzekutimit të tij.
Qëllimi	Qëllimi kryesor është gjetja e mangësive në fazat e hershme të zhvillimit, përmirësimi i cilësisë dhe sigurimi i respektimit të standardeve.	Synon të verifikojë nëse softueri përmbush kërkesat e tij funksionale dhe të performancës dhe zbulon mangësi nëpërmjet ekzekutimit të tij të vërtetë.
Kur Kryhet	Mund të kryhet gjatë të gjithë ciklit të jetës së zhvillimit të softuerit (SDLC), nga analiza e kërkesave deri te shqyrtimi i kodit.	Zakonisht kryhet gjatë fazave më të vona të SDLC, si testimi i sistemit, testimi i integritetit dhe testimi i pranimit nga përdoruesit.
Teknikat	Teknikat përfshijnë shqyrtimet, inspektimet dhe shqyrtimet e dokumentacionit, ku kodi dhe dokumentet ekzaminojnë manualisht.	Teknikat përfshijnë projektimin e rasteve të testimi, krijimin e të dhënave të testimi dhe ekzekutimin e testimi.
Automatizimi	Automatizimi është më i rrallë, megjithëse ekzistojnë mjete për analizën statike.	Automatizimi përdoret gjerësisht, dhe ka shumë mjete testimi për testimin dinamik.

Tabela 1: Diferencat midis testimit statik dhe atij dinamik

2.7 Teknikat e dizenjemit te testeve

1. Teknikat e Testimit të Kutise se Zeze (Black-box): Njihen gjithashtu si teknika të bazuar në specifikacion dhe janë të bazuar në një analizë të sjelljes të specifikuar të objektit të testimi pa referuar në strukturën e tij të brendshme. Prandaj, rastet e testimi janë të pavarura nga mënyra se si është implementuar softueri. Si rrjedhojë, nëse ndryshon implementimi, por sjellja e kërkuar mbetet e njëjtë, atëherë rastet e testimi vazhdojnë të jenë të dobishme.
2. Teknikat e Testimit të Kutise se Bardhë (White-box): Njihen gjithashtu si teknika të bazuar në strukturë dhe janë të bazuar në një analizë të strukturës dhe procesimit të brendshëm të objektit të testimi. Pasqyrohen në mënyrë të ngushtë me mënyrën se si është projektuar softueri, dhe ato mund të krijohen vetëm pas projektimit ose implementimit të objektit të testimi.
3. Teknikat e Testimit të Bazuar në Përvojë: Përdorin efektivisht njohuritë dhe përvojën e testuesve për projektimin dhe implementimin e rasteve të testimi. Efektiviteti i këtyre teknikave varet shumë nga aftësitë e testuesit. Teknikat e bazuar në përvojë mund të zbulojnë mangësi që mund të humbasin duke përdorur teknikat e testimit të kuq dhe të bardhë.

2.8 Menaxhimi i procesit te testimit

Kapitulli "Menaxhimi i Veprimeve të Testimit" është një pjesë e rëndësishme e programit të nivelit bazë të ISTQB, sepse mbulon shkathtësitë dhe njohuritë themelore të nevojshme për të planifikuar, ekzekutuar dhe menaxhuar një projekt testimi të suksesshëm.

Këtu janë disa detaje shtesë për secilën nga temat e mbuluara në këtë kapitull:

Planifikimi i Testimit

Planifikimi i testim-it është një dokument që përshkruan shtrirjen, qasjen dhe burimet e nevojshme për testim. Është e rëndësishme të keni një plan të mirëdefinuar të testim-it para se testimi të fillojë, sepse kjo do të ndihmojë në sigurimin që testimi të kryhet me efikasitet dhe efektivitet.

Monitorimi dhe Kontrolli i Testimit

Monitorimi dhe kontrolli i testim-it është procesi i ndjekjes së përparimit të testim-it, identifikimit dhe menaxhimit të rreziqeve, dhe bërjes së ndryshimeve në planin e testim-it sipas nevojës. Ky është një proces i rëndësishëm, sepse ndihmon në sigurimin që testimi është në rrugë të duhur dhe që çdo problem potencial identifikohet dhe adresohet në fillim.

Analiza e Testimit

Analiza e testim-it është procesi i rishikimit të kërkesave, dizajnit dhe kodit për të identifikuar mangësitë potenciale dhe për të zhvilluar rastet e testim-it. Ky është një proces kyç, sepse siguron që rastet e testim-it janë të plotësueshme dhe efektive.

Dizajni i Testimit

Dizajni i testim-it është procesi i krijoj të rasteve të testim-it që do të verifikojnë që softueri përmbush kërkesat dhe objektivat e testim-it. Ka një shumëllojshmëri teknikash të dizajnit të testim-it që mund të përdoren, si testimi i kutisë së zi, testimi i kutisë së bardhë, dhe testimi i bazuar në rreziqe.

Implementimi i Testimit

Implementimi i testim-it përfshin zhvillimin dhe ekzekutimin e rasteve të testim-it. Ky proces mund të jetë manual ose i automatizuar. Testimi manual është qasja tradicionale e testim-it, ku testuesi ekzekuton rastet e testim-it manualisht. Testimi i automatizuar përdor mjete për të ekzekutuar automatikisht rastet e testim-it.

Ekzekutimi i Testimit

Ekzekutimi i testim-it përfshin ekzekutimin e rasteve të testim-it dhe regjistrimin e rezultateve. Është e rëndësishme të regjistroni me saktësi rezultatet e çdo rasti të testim-it, që ato të mund të analizohen dhe raportohen.

Përfundimi i Testimit

Përfundimi i testim-it përfshin vlerësimin e rezultateve të testim-it, raportimin e gjetjeve dhe mbylljen e projektit të testim-it. Rezultatet e testim-it duhet të vlerësohen për të përcaktuar nëse softueri përmbush kërkesat dhe objektivat e testim-it. Gjetjet e testim-it duhet të raportohen te palët e interesuara relevante. Projekti i testim-it duhet të mbyllet pasi testimi të përfundojë dhe gjetjet të raportohen.

KAPITULLI 3

3.1 Cfare eshte testimi i automatizuar dhe cilat jane disa nga teknologjite dhe mjetet me te perdorura?

Testimi i automatizuar i softuerit është procesi i përdorimit të mjeteve të specializuara softuerike për të ekzekutuar teste të parashkruara në një aplikacion ose sistem softueri, në vend të testimit manual. Ai përfshin shkrimin e skripteve ose rastet e testimit që mund të ekzekutohen automatikisht për të testuar softuerin, në vend që testuesit të kryejnë çdo rast testimi manualisht. Qëllimi i testimit të automatizuar është të përmirësojë efikasitetin, shpejtësinë dhe saktësinë e procesit të testimit, duke reduktuar përpjekjen e kërkuar për testimin manual.

Mjetet e automatizuara të testimit mund të simulojnë ndërveprimet e përdoruesve me softuerin, si futja e të dhënave, klikimi i butonave dhe lundrimi nëpër ekrane. Mjetet mund të verifikojnë më pas rezultatet kundrejt rezultateve të pritshme, të tilla si kontrolli nëse shfaqen të dhënat e sakta ose nëse mesazhet e gabimit shfaqen kur është e përshtatshme. Testimi i automatizuar mund të përdoret për një sërë llojesh testimi, duke përfshirë testimin funksional, testimin e regresionit, testimin e performancës dhe testimin e ngarkesës.

Përfitimet e testimit të automatizuar përfshijnë përmirësimin e efikasitetit të testimit, rritjen e mbulimit të testeve, reagime më të shpejta dhe reduktimin e rrezikut të gabimit njerëzor. Megjithatë, testimi i automatizuar kërkon ekspertizë në mjetet dhe kornizat e automatizimit të testimit, si dhe aftësi programimi dhe skriptimi. Është gjithashtu e rëndësishme që të dizajnohen dhe mirëmbahen me kujdes skriptet e testimit për t'u siguruar që ato të mbeten efektive dhe të sakta ndërsa softueri evoluon.

3.2 Perfitimet dhe Rreziqet e testimit te automatizuar - ISTQB

Testimi I automatizuar i softuereve eshte i rendesishem per shume arsye, si:

1. Kursimi i kohës: Testimi i automatizuar mund të ekzekutojë testet shumë më shpejt se testimi manual, duke i lejuar testuesit të ekzekutojnë më shumë teste në më pak kohë. Kjo mundëson reagime më të shpejta dhe mund të zvogëlojë kohën në treg për produktet softuerike.
   Konsistenca: Testet e automatizuara kryejnë të njëjtat hapa dhe kontrolle sa herë që ekzekutohen, duke siguruar qëndrueshmëri në procesin e testimit. Kjo zvogëlon gjasat e gabimit njerëzor dhe përmirëson besueshmërinë e rezultateve të testit.

2. Mbulimi: Testimi i automatizuar mund të mbulojë një numër të madh rastesh testimi, duke përfshirë ato komplekse që mund të jenë të vështira për t'u ekzekutuar me dorë. Kjo siguron që të gjitha aspektet e softuerit të testohen tërësisht dhe redukton rrezikun e defekteve të pazbuluara.
3. Ripërdorshmëria: Testet e automatizuara mund të ripërdoren nëpër versione dhe ndërtime të ndryshme të softuerit, duke kursyer kohë dhe përpjekje në krijimin e testeve të reja për çdo version.
4. Me kosto efektive: Ndërsa investimi fillestar në testimin e automatizuar mund të jetë më i lartë se testimi manual, testimi i automatizuar përfundimisht mund të kursejë kostot në afat të gjatë duke reduktuar nevojën për testues manualë dhe duke zvogëluar kohën dhe përpjekjen e kërkuar për testim.
5. Sigurimi i cilësisë: Testimi i automatizuar mund të ndihmojë në përmirësimin e cilësisë së softuerit duke identifikuar defektet në fillim të ciklit të zhvillimit, duke reduktuar gjasat që klientët të zbulojnë defekte pas lëshimit.

Në përgjithësi, testimi i automatizuar i softuerit mund të përmirësojë efikasitetin, efektivitetin dhe cilësinë e procesit të testimit, duke e bërë atë një mjet thelbësor për ekipet moderne të zhvillimit të softuerit.

Ekzistojne shume teknologji dhe mjete per testimin e automatizuar, falas si dhe me pagese. Me poshte permendim disa prej tyre.

3.3 Teknologji falas per automatizimin e softuereve

1. Selenium WebDriver: Selenium WebDriver është një teknologji e hapur i përdorur për automatizimin e shfletuesve të internetit. Ai mbështet shumë gjuhë programimi dhe ka një komunitet të madh të përdoruesve dhe kontribuuesve.
2. Appium: Appium është një teknologji e hapur, e përdorur për automatizimin e aplikacioneve mobile. Ai mbështet platformat iOS, Android dhe Windows.
3. Robot Framework: Robot Framework është një teknologji e automatizimit të testim-it që mund të përdoret për një larmi të llojeve të testim-it. Ai ka një sintaksë të lehtë për t'u përdorur dhe mbështet shumë gjuhë programimi.
4. Cypress: Cypress është një teknologji me burim te hapur, e përdorur për testimin e aplikacioneve web. Ai ofron aftësi testimi të shpejta, të besueshme dhe të lehta për t'u përdorur, me një fokus në testimin end-to-end.

3.4 Teknologji komerciale per automatizimin e softuereve

1. TestComplete: TestComplete është një mjet komercial i përdorur për testimin funksional automatik të aplikacioneve web, desktop dhe mobile. Ai mbështet shumë gjuhë scriptimi dhe ka aftësi të integruara për menaxhimin e testeve.
2. Ranorex: Ranorex është një mjet komercial i përdorur për testimin automatik të aplikacioneve desktop, web dhe mobile. Ai ka një ndërfaqe të lehtë për t'u përdorur dhe mbështet shumë gjuhë programimi.
3. Telerik Test Studio: Telerik Test Studio është një mjet komercial i përdorur për testimin funksional automatik të aplikacioneve web dhe desktop. Ai ka një ndërfaqe të lehtë për t'u përdorur dhe mbështet shumë gjuhë programimi.
4. Katalon Studio: Katalon Studio është një mjet komercial i përdorur për testimin automatik të aplikacioneve web, mobile dhe API. Ai ka një sistem të integruar për menaxhimin e testeve dhe mbështet shumë gjuhë scriptimi.

KAPITULLI 4

4.1 Selenium

Selenium është një kuadër i hapur burimor për automatizimin e testim-it që përdoret gjerësisht për automatizimin e shfletuesve të internetit. Ai u krijua fillimisht nga Jason Huggins në vitin 2004 si një mjet i brendshëm në ThoughtWorks, një firmë konsulence për softuer. Mjeti shpejt u bë i njohur dhe u lëshua si një projekt i hapur burimor në vitin 2008. Që atëherë, është bërë një nga kuadret më të njohura të automatizimit të testim-it në industrinë e softuerit, me një komunitet të madh të përdoruesve dhe kontribuesve.

Selenium ofron një suitemë mjetesh për testimin automatik të aplikacioneve web, përfshirë Selenium IDE, Selenium WebDriver dhe Selenium Grid. Selenium IDE është një mjet për regjistrimin dhe ekzekutimin e testeve që lejon përdoruesit të krijojnë teste automatike në një ndërfaqe të lehtë për t'u përdorur. Selenium WebDriver është një ndërfaqe programuese për ndërveprimin me shfletuesit e internetit, duke lejuar përdoruesit të krijojnë teste më të sofistikuara duke përdorur gjuhë programimi si Java, Python dhe C#. Selenium Grid është një mjet për testimin e shpërndarë, duke lejuar teste të ekzekutohen nëpër disa kompjuterë dhe shfletues të njëkohshëm.

Një nga avantazhet kyçe të Selenium është fleksibiliteti i tij. Ai mbështet një gamë të gjërë gjuhësh programimi dhe mund të integrohet me një varietet të mjeteve dhe kuadreve të tjera për testim. Ai gjithashtu mbështet shumë shfletues dhe sisteme operimi, duke lejuar ekzekutimin e testeve në platforma të ndryshme. Gjithashtu, Selenium është i hapur burimor dhe i lirë për përdorim, duke e bërë atë të arritshëm për një spektër të gjerë organizatash dhe individësh.

Në përgjithësi, Selenium është një mjet i fuqishëm dhe i fleksueshëm për testimin automatik të aplikacioneve web. Popullariteti dhe mbështetja e komunitetit bëjnë nga një pasuri e çmuar për çdo ekip testimi.

4.1.1 Selenium IDE

Selenium IDE (Integrated Development Environment) është një mjet për regjistrimin dhe ekzekutimin e testeve automatike në Selenium. Është një ndërfaqe e thjeshtë dhe e lehtë për t'u përdorur që lejon testuesit të regjistrojnë ndërveprimet e tyre me një aplikacion web dhe pastaj t'i ekzekutojnë ato ndërveprime si teste automatike. Selenium IDE gjithashtu lejon testuesit të ndryshojnë dhe të

kustomizojnë manualisht testet e regjistruara, duke e bërë atë një mjet fleksibël dhe fuqishëm për testimin automatik.

Një nga avantazhet kyçe të Selenium IDE është thjeshtësia e tij. Ai është një mjet i lehtë që është i lehtë për tu instaluar dhe kërkon konfigurim minimal. Ai gjithashtu përfshin një numër karakteristikash të dobishme si lokalizues të elementeve, vërtetime dhe variabla, duke e bërë të lehtë krijimin dhe kustomizimin e testeve.

Megjithatë, është e vlefshme të theksohet se Selenium IDE është kryesisht i projektuar për krijimin e testeve të thjeshta dhe të drejtpërdrejta. Për teste më komplekse, testuesit mund të kenë nevojë të përdorin mjetet e tjera në setin e Selenium, si Selenium WebDriver.

4.1.2 Selenium Grid

Selenium Grid është një mjet në setin e Selenium që lejon përdoruesit të ekzekutojnë teste nëpër disa kompjuterë dhe shfletues të njëkohshëm. Është projektuar për të ofruar aftësi të shpërndara për testimin, duke lejuar teste të ekzekutohen nëpër disa kompjuterë dhe shfletues të njëkohshëm.

Me Selenium Grid, testuesit mund të krijojnë një qendër që vepron si një pikë qendrore për shpërndarjen e testeve te nodet e ndryshme, të cilat janë kompjuterë ose kompjuterë virtuale që kanë shfletues të instaluar. Qendra mund të konfigurohet për të shpërndarë teste bazuar në një varietet kriteresh, përfshirë llojin e shfletuesit, sistemin operativ dhe burimet e disponueshme.

Një nga avantazhet kyçe të Selenium Grid është shkallëzueshmëria e tij. Ajo lejon testuesit të ekzekutojnë teste në paralel nëpër disa kompjuterë dhe shfletues, gjë që mund të reduktojë në mënyrë signifikative kohën e ekzekutimit të testeve. Ajo gjithashtu ofron një zgjidhje me kosto efektive për testimin në platforma dhe shfletues të ndryshëm, pasi lejon testuesit të ekzekutojnë teste në kompjuterë me konfigurime të ndryshme pa nevojën për pajisje fizike.

Selenium Grid është i përputhshëm me Selenium WebDriver, çka do të thotë se testet e krijuara duke përdorur WebDriver mund të ekzekutohen në Grid. Kjo e bën të lehtë për testuesit të shkallëzojnë përpjekjet e tyre për automatizimin e testeve dhe të ekzekutojnë teste në një shumëllojshmëri mjedise.

Në përgjithësi, Selenium Grid është një mjet i fuqishëm për testimin e shpërndarë dhe mund të përmirësojë në mënyrë të ndjeshme efikasitetin dhe shkallëzueshmërinë e automatizimit të testeve. Përshtatshmëria e tij me Selenium WebDriver dhe aftësia për të ekzekutuar teste në paralel nëpër disa kompjuterë dhe shfletues bëjnë nga një pasuri e çmuar për çdo ekip testimi.

4.1.3 Selenium WebDriver

Selenium WebDriver është një mjet i njohur i hapur burimor për automatizimin e testeve që lejon testuesit të automatizojnë aplikacionet web. Ai ofron një ndërfaqe të thjeshtë dhe të fuqishme për ndërveprimin me elementet e internetit si butonat, format dhe dropdown-et.

Një nga avantazhet kyçe të WebDriver është aftësia për të ndërvepruar me elementet e internetit duke përdorur shfletues të vërtetë, duke lejuar testuesit të kryejnë teste më realiste. Ai mbështet shumë gjuhë programimi si Java, Python dhe JavaScript, duke e bërë atë fleksibël dhe të arritshëm për një gamë të gjerë përdoruesish.

WebDriver gjithashtu përfshin karakteristika si lokalizues të elementeve, pritje dhe vërtetime, të cilat e bëjnë të lehtë krijimin e testeve të qëndrueshme dhe të besueshme. Ai gjithashtu mbështet shfletues të pamundur për tu shfaqur në ekran, duke e bërë të mundur ekzekutimin e testeve pa një ndërfaqe përdoruesi.

Në përgjithësi, Selenium WebDriver është një mjet i fuqishëm dhe i fleksueshëm për testimin e aplikacioneve web që përdoret gjerësisht nga ekipe të testim-it në të gjithë botën. Fleksibiliteti, kompatibiliteti dhe lehtësia e përdorimit e bëjnë atë një zgjedhje ideale për të dy fillestarët dhe testuesit me përvojë.

4.2 Avantazhet dhe disavantazhet e perdorimit te Selenium

Avantazhe:

Burim i Hapur: Selenium është një mjet burimor i hapur, që do të thotë se është falas për përdorim dhe përdoruesit kanë akses në kodin burimor. Kjo e bën atë një zgjedhje me kosto efektive për automatizimin e testeve.

Kompatibiliteti në Shfletues të Ndryshëm: Selenium mbështet shumë shfletues si Chrome, Firefox dhe Internet Explorer, duke e bërë të lehtë ekzekutimin e testeve në platforma të ndryshme.

Fleksibiliteti: Selenium mbështet shumë gjuhë programimi si Java, Python dhe JavaScript, duke e bërë atë të arritshëm për një gamë të gjerë përdoruesish.

Kuadër i Fortë: Selenium ofron një kuadër të fortë për testimin e aplikacioneve web, me karakteristika si lokalizues të elementeve, pritje dhe vërtetime.

Mbështetja e Komunitetit: Selenium ka një komunitet të madh dhe aktiv të përdoruesve që ofrojnë mbështetje dhe ndajnë njohuri dhe burime.

Disavantazhe:

Mbështetje e Kufizuar për Testimin e Mobilave: Selenium është projektuar kryesisht për testimin e aplikacioneve web dhe ka mbështetje të kufizuar për testimin e mobilave.

Kërkohet njohuri teknike: Përdorimi i Selenium kërkon një nivel të caktuar njohurish teknike dhe aftësish programimi, që mund të jetë një pengesë për testuesit jo-teknikë.

Konfigurimi dhe Mirëmbajtja: Konfigurimi dhe mirëmbajtja e Selenium mund të kërkojë kohë dhe kërkon ekspertizë në konfigurimin e mjedisit të testim-it.

Përbaltja: Testet e Selenium mund të jenë të prirura ndaj përbaltjes ose moskonsistencës, që mund të jetë sfiduese për të identifikuar dhe ndrequr.

Në përgjithësi, megjithëse ka disa sfida të lidhura me përdorimin e Selenium, fleksibiliteti i tij, kuadri i fortë dhe mbështetja e komunitetit e bëjnë atë një zgjedhje të njohur për testimin e aplikacioneve web.

4. 3 Selenium komandat kryesore:

1. Komandat e navigimit
    a. get(url): Kjo komandë hap një faqe interneti në URL-në e dhënë.
    b. back(): Kjo komandë e kthen shfletuesin një faqe prapa.
    c. forward(): Kjo komandë e drejton shfletuesin përpara një faqe.
    d. refresh(): Kjo komandë rifreskon faqen aktuale të internetit.
    e. maximize_window(): Kjo komandë maksimizon dritaren e shfletuesit.
    f. set_window_size(width, height): Kjo komandë vendos madhësinë e dritares së shfletuesit në gjerësinë dhe lartësinë e specifikuar.
    g. get_current_url(): Kjo komandë merr URL-në aktuale të faqes së internetit.
    h. get_title(): Kjo komandë merr titullin e faqes aktuale të internetit.
2. Komandat e driverit
    a. ChromeDriver()
    b. FirefoxDriver()
    c. close()

d. quit()

3. Komandat e gjetjes se elementeve **TODO – change with right methods**
    a. find_element_by_id(): Kjo metodë gjen një element nga atributi i saj ID.
    b. find_element_by_name(): Kjo metodë gjen një element sipas atributit të emrit të tij.
    c. find_element_by_xpath(): Kjo metodë gjen një element duke përdorur një shprehje XPath.
    d. find_element_by_link_text(): Kjo metodë gjen një element nga teksti i saktë i një lidhjeje.
    e. find_element_by_partial_link_text(): Kjo metodë gjen një element nga një përputhje e pjesshme e tekstit të lidhjes së saj.
    f. find_element_by_tag_name(): Kjo metodë gjen një element me emrin e etiketës HTML.
    g. find_element_by_class_name():Kjo metodë gjen një element me emrin e klasës së saj CSS.
    h. find_element_by_css_selector(): Kjo metodë gjen një element duke përdorur një përzgjedhës CSS.
4. Komandat e nderverprimit me elementet ueb
    a. click(): Kjo metodë klikon mbi elementin.
    b. send_keys(): Kjo metodë dërgon tekst te elementi.
    c. clear(): Kjo metodë pastron tekstin nga një element hyrës.
    d. get_attribute(): Kjo metodë kthen vlerën e atributit të specifikuar të elementit.
    e. is_displayed(): Kjo metodë kthen një vlerë boolean që tregon nëse elementi shfaqet aktualisht në faqe.
    f. is_enabled(): Kjo metodë kthen një vlerë boolean që tregon nëse elementi është aktualisht i aktivizuar.
    g. is_selected(): Kjo metodë kthen një vlerë boolean që tregon nëse elementi është zgjedhur aktualisht.

Këto metoda mund të përdoren për të bashkëvepruar me elementët e uebit në mënyra të ndryshme, si klikimi mbi butona, plotësimi i fushave të formularit ose marrja e informacionit nga faqja. Për të përdorur këto metoda, fillimisht duhet të lokalizoni elementin duke përdorur një nga metodat për gjetjen e elementeve të ueb-it në Selenium, si find_element_by_id(), find_element_by_xpath(), ose find_element_by_css_selector(). Pasi të keni gjetur elementin, mund të përdorni këto metoda për të bashkëvepruar me të.

Në testimin e automatizuar, pohimet përdoren për të kontrolluar nëse rezultatet e pritura të një testi përputhen me rezultatet aktuale. Pohimet ju lejojnë të verifikoni nëse disa kushte janë përmbushur, dhe nëse jo, testi do të dështojë.

Këtu janë disa pohime të përdorura zakonisht në Selenium:

- assertEqual(): Ky pohim kontrollon që dy vlera janë të barabarta.
- assertNotEqual(): Ky pohim kontrollon që dy vlera të mos jenë të barabarta.
- assertTrue(): Ky pohim kontrollon që një vlerë boolean është e vërtetë.
- assertFalse(): Ky pohim kontrollon që një vlerë boolean është false.
- assertIn(): Ky pohim kontrollon që një vlerë është e pranishme në një listë, tuple ose varg.
- assertNotIn(): Ky pohim kontrollon që një vlerë nuk është e pranishme në një listë, tuple ose varg.
- assertIs(): Ky pohim kontrollon që dy objekte janë i njëjti objekt.
- assertIsNot(): Ky pohim kontrollon që dy objekte nuk janë i njëjti objekt.

Pohimet janë një komponent kritik i automatizimit të testit, pasi ato ju lejojnë të siguroheni që testet tuaja po prodhojnë rezultatet e pritura. Duke përdorur pohimet, ju mund të kapni gabimet dhe defektet në fillim të procesit të zhvillimit, gjë që mund t'ju kursejë kohë dhe burime në afat të gjatë.

KAPITULLI 5

5.1 Cypress: Një Teknologji e Re për Testimin e Aplikacioneve të Internetit

Cypress është një teknologji relativisht e re, por mjaft e njohur e testimit automatik me burim të hapur që është projektuar për të testuar aplikacionet e reja të internetit. Ai u lëshua për herë të parë në vitin 2014 nga Brian Mann, Drew Lanham dhe Alan Chang, dhe ka fituar një popullaritet të madh në vite të fundit. Ndryshe nga teknologjite e tjera të testimit, Cypress nuk bazohet në Selenium, por përdor një arkitekturë unike që i lejon të ekzekutojë testet në të njëjtin kontekst me aplikacionin që po testohet.

5.2 Lehtësia e Përdorimit dhe Ekzekutimi i Shpejtë

Një nga përfitimet kryesore të Cypress është lehtësia e përdorimit. Ai ka një API të thjeshtë dhe intuitiv që lejon zhvilluesit dhe testerët të shkruajnë teste duke përdorur gjuhën JavaScript. Cypress gjithashtu përfshin një panel miqësor për përgjigjen në kohë reale të rezultateve të testeve, duke e bërë të lehtë procesin e debug-imin dhe zbulimin e problematikave.

Një përfitim tjetër i Cypress është ekzekutimi i shpejtë. Për shkak se ai ekzekutojë testet në të njëjtin kontekst me aplikacionin që po testohet, ai është në gjendje të ofrojë përgjigje të shpejta dhe të besueshme për rezultatet e testeve. Përveç kësaj, Cypress përfshin mbështetje të brendshme për teknologjitë e reja të front-end si React, Angular dhe Vue.js, duke e bërë atë ideal për testimin e aplikacioneve të internetit të kohës së sotme.

Cypress gjithashtu përfshin një numër veçorish që e bëjnë atë ideal për testimin end-to-end, përfshirë mbështetje për kërkesat dhe përgjigjet e rrjetit, testim vizual dhe kohëzgjatje dhe periudha të personalizuara. Ai gjithashtu mund të integrohet lehtësisht me teknologjite e tjera të popullarë të testimi si Mocha dhe Chai.

5.3 Një Alternativë Efiçente dhe Modernizuar

Në përgjithësi, Cypress është një teknologji e fuqishëm dhe e përdorshme për testimin e aplikacioneve të internetit të reja. Arkitektura e tij unike dhe ekzekutimi i shpejtë e bëjnë atë një mjet të çmuar për çdo ekip të testimi. Përveç kësaj, Cypress ofron një API më modern dhe më fleksibël për ndërveprimin me elementet e internetit, duke e lejuar zhvilluesit të testojnë një gamë më të gjerë të funksionaliteteve me më pak kod.

5.4 Komandat kryesore te Cypress

cy.get(): Ky është komanda kryesor për zgjedhjen e elementeve DOM në Cypress, ngjashëm me findElement() në Selenium. Megjithatë, Cypress automatikisht rifillon zgjedhjen derisa elementi të jetë i disponueshëm ose koha e pritjes të përfundojë.

cy.click(): Kjo komandë është ngjashme me click() në Selenium, por ajo prit automatikisht që elementi të bëhet i klikueshëm.

cy.type(): Kjo komandë është ngjashme me sendKeys() në Selenium, por ju lejon të shkruani tekst në një element me sjellje më natyrale të tastierës dhe automatikisht pret që elementi të bëhet i disponueshëm.

cy.visit(): Kjo komandë është ngjashme me get() në Selenium, por ju lejon të vizitoni një URL dhe automatikisht pret që faqja të përfundojë ngarkimin.

cy.contains(): Kjo komandë ju lejon të zgjidhni një element sipas përmbajtjes së tekstit të tij, ngjashëm me findElement(By.xpath("//*[contains(text(),'teksti')]")) në Selenium.

cy.url(): Kjo komandë ju lejon të merrni URL aktual të faqes, ngjashëm me driver.getCurrentUrl() në Selenium.

cy.wait(): Kjo komandë ju lejon të ndaloni testin për një periudhë të caktuar kohore, ngjashëm me time.sleep() në Selenium.

Në përgjithësi, Cypress ofron një API më modern dhe më të lehtë për ndërveprimin me elementet e internetit krahasuar me Selenium. Ai gjithashtu ka mbështetje të brendshme për pritjen e elementeve për të bërë atë që të bëhen të disponueshëm dhe për të menaxhuar sjelljet asinkrone, gjë që mund të jetë më e vështirë në Selenium.

5.5 Komandat e Cypress që Nuk Janë Në Selenium:

cy.intercept(): Kjo komandë ju lejon të kapëni dhe modifikoni kërkesat dhe përgjigjet HTTP të bëra nga aplikacioni juaj, gjë që mund të jetë e dobishme për simulimin e API-ve dhe testimin e funksionit të lidhur me rrjetin.

cy.clock(): Kjo komandë ju lejon të kontrolloni orën JavaScript të brendshme në shfletues, një gjë e dobishme për testimin e funksionit që është i varur nga koha.

cy.wrap(): Kjo komandë ju lejon të mbështesni një objekt ose vlerë jo-Cypress dhe të trajtohet si një objekt Cypress, një gjë e dobishme për integrimin e bibliotekave ose utiliteteve të jashtme në teste tuaja Cypress.

cy.fixture(): Kjo komandë ju lejon të ngarkoni dhe përdorni të dhënat e testit nga skedarë të jashtëm JSON ose YAML, një gjë e dobishme për parametrizimin e testeve dhe ndarjen e të dhënave nga logjika e testit.

cy.task(): Ky komandë ju lejon të ekzekutoni kod të ndërveprimit të përshtatur me Node.js si një detyrë brenda testit tuaj Cypress. Kjo është e dobishme për integrimin me sisteme të jashtme ose për të kryer logjikë të kompleksuar të konfigurimit dhe shkatërrimit të mjedisit.

Në përgjithësi, Cypress ofron një API më të pasur dhe më fleksibël për testimin e aplikacioneve të internetit krahasuar me Selenium, duke lejuar zhvilluesit të testojnë një gamë më të gjerë të funksionalitetit me më pak kod.

KAPITULLI 6

6.1 Test Complete – Mjet komercial

TestComplete është një mjet i komercializuar i plotë për testimin automatik i zhvilluar nga SmartBear Software. Ai njihet në industrinë e softuerëve për ndërfaqen e tij të larmishme, setin e karakteristikave të fortë dhe mbështetjen e gjerë për lloje të ndryshme të aplikacioneve, përfshirë aplikacione web, desktop, mobile dhe aplikacione të ndryshme platformash. Në këtë seksion, do të eksplorojmë karakteristikat kyçe, aftësitë dhe avantazhet e përdorimit të TestComplete si një zgjidhje për testimin automatik.

6.2 Karakteristikat dhe funksionalitetet kryesore

TestComplete ofron një varg të gjerë të karakteristikave dhe aftësive që e bëjnë atë një zgjedhje të preferuar për shumë organizata në përpjekjet e tyre për testimin automatik. Disa nga karakteristikat e tij të dallueshme përfshijnë:

a) Mbështetje për Shumë Platforma:
   TestComplete mbështet testimin në shumë platforma, përfshirë Windows, macOS dhe Linux, duke e bërë të përshtatshëm për ekosistemet e ndryshme të aplikacioneve.

b) Mbështetje për Shumë Gjuhë:
   Ajo lejon zhvillimin e skripteve të testeve në gjuhë programimi të njohura si JavaScript, Python dhe VBScript, duke ofruar fleksibilitet ekipit të zhvillimit me preferenca të ndryshme të gjuhës.

c) Njohje e Elementeve:
   TestComplete përdor teknika të zhvilluara të njohjes së elementeve për të identifikuar dhe ndërvepruar me elementet në ndërfaqen e përdoruesit të aplikacionit. Kjo siguron stabilitet dhe rezistencë në skriptet e testeve, edhe kur ndryshon ndërfaqja e përdoruesit.

d) Visualizues i Testit:
   Mjeti ofron një karakteristikë "Test Visualizer", që lejon testuesit të kapin screenshot-e dhe video gjatë ekzekutimit të testeve për debugim dhe raportim efikas.

e) Raportim i Detajuar i Testit:
   TestComplete gjeneron raporte të detajuara të testeve me një pasuri të informacionit, përfshirë regjistra, metrika dhe dëshmi vizuale, për të ndihmuar në identifikimin dhe zgjidhjen e problemave.

6.3 Avantazhet e TestComplete

1. Zhvillim i Shpejtë i Automatizimit të Testit:
   Funksionaliteti i regjistrimit dhe ekzekutimit të TestComplete simplifikon krijimin e skripteve të testeve, duke lejuar testuesit të krijojnë shpejt teste automatike me minimal kodim.
2. Mbështetje e Gjerë e Testit:
   Ajo mbështet testimin në shumë shfletues, sisteme operativë dhe pajisje të ndryshme, duke lehtësuar mbulimin e gjerë të testeve për të siguruar kompatibilitetin e aplikacionit.
3. Aftësi për Integrim:
   TestComplete mund të integrohet pa problem me shumë mjetet e CI/CD, sistemet e menaxhimit të testeve dhe platformat e ndjekjes së çështjeve, duke promovuar bashkëpunimin dhe automatizimin brenda rrjedhës së zhvillimit.
4. Mbështetje e Fortë e Komunitetit:
   Komuniteti dhe baza e përdoruesve e fortë e SmartBear kontribuon në disponueshmërinë e burimeve, tutorialëve dhe mbështetjes, duke e bërë më të lehtë për ekipet të adoptojnë dhe të mësojnë TestComplete.

6.4 Licencimi dhe Kostot

Është e rëndësishme të theksohet se TestComplete është një mjet komercial, dhe kostot e licencimit mund të ndryshojnë në varësi të faktorëve si numri i përdoruesve, kompleksiteti i projektit dhe karakteristikat e kërkuara. Organizatat duhet të konsiderojnë kufizimet buxhetore kur e vlerësojnë TestComplete si zgjidhje për testimin automatik.

6.5 Aplikimi në Botën Reale

Për të treguar aplikueshmërinë në botën reale të TestComplete, kjo tezë do të përfshijë studime të rasteve dhe shembuj praktikë të mënyrave se si organizatat kanë përdorur me sukses TestComplete për të përmirësuar proceset e tyre të testimi dhe për të përmirësuar cilësinë e produkteve të tyre të softuerit.

Në përmbledhje, TestComplete ofron një zgjidhje të fortë, të larmishme dhe të mbështetur mirë për testimin automatik për organizatat që duan të sigurojnë cilësinë dhe besueshmërinë e aplikacioneve të tyre të softuerit. Seti i karakteristikave të tij, mbështetja për shumë platforma dhe aftësitë e integrimi e bëjnë atë një pasuri të vlerësuar në trupat e testuesve të automatizimit dhe profesionistëve të sigurisë së cilësisë.

6.6 Komandat kryesore

Komandat dhe Veçoritë Kryesore të TestComplete:

a) Rregjistro dhe Luaj: TestComplete lejon përdoruesit të regjistrojnë veprimet e tyre në një aplikacion dhe pastaj t'i luajnë ato veprime si teste të automatizuara. Ky është një mënyrë e thjeshtë për të filluar me testimin automatik.
b) Gjuhet skriptive: TestComplete mbështet shumë gjuhë programimi të tilla si JavaScript, Python, VBScript, dhe të tjera. Ky aspekt i jep fleksibilitet ekipit të zhvillimit në zgjedhjen e gjuhës së tyre të preferuar.
c) Identifikimi i elementeve: TestComplete ka aftësi për të njohur dhe identifikuar elementet në ndërfaqen e përdoruesit të aplikacionit. Kjo siguron që testet të jenë stabile edhe kur ndryshojnë elementet në ndërfaqe.
d) Ekzekutimi i Testeve: Mjeti lejon ekzekutimin e testeve automatike në mënyrë të ngjarjeve. Mund të zgjidhet të ekzekutohen në një numër të shfletuesve dhe sistemeve operativë.
e) **Test Debugging**: TestComplete ofron mjete për të **debuguar** testet automatike, duke përfshirë ruajtjen e ekzekutimit, raportet e detajuara të testeve, dhe mundësinë për të shfaqur elementët e identifikuar në ndërfaqen e përdoruesit.
f) Integrimi: Ky mjet mund të integrohet lehtë me mjete të tjera të zhvillimit dhe testimi, përfshirë mjete të CI/CD (Continuous Integration/Continuous Delivery), sisteme të menaxhimit të testeve, dhe më shumë.
g) Raporti i Testimit: TestComplete gjeneron raporte të detajuara të testeve, duke përfshirë logjet e ekzekutimit, grafikët e performancës, dhe të dhënat e detajuara për të ndihmuar në identifikimin dhe zgjidhjen e problemeve.
h) Testimi mobile: TestComplete ofron mbështetje për testimin e aplikacioneve mobile në platformat Android dhe iOS.
i) **Cross-Browser Testing:** Mundëson testimin e aplikacioneve në shumë shfletues të ndryshëm përfshirë Chrome, Firefox, dhe Internet Explorer.
j) Testimi Web: TestComplete ofron aftësi për të testuar shërbimet e internetit (web services) duke komunikuar me API-të e tyre.

Këto janë vetëm disa prej komandave dhe karakteristikave kryesore të TestComplete. Mjeti vazhdon të zhvillohet dhe shtohen veçori të tjera për të përmirësuar aftësitë e testimin automatik në mjediset e ndryshme të zhvillimit.

KAPITULLI 7

7.1 Metodologjia

Ne kete kapitull do te shyrtojme te gjitha metodat e perdorura per te kryer kete studim. Qellimi kryesor i kesaj analize eshte krahasimi i mjeteve te perdorura per automatizimin e testimit te aplikacioneve web, ne varesi te karakteristikave te seciles prej tyre.

7.2 Skenaret e testimit

Skenaret e testimit janë një pjesë kritike e procesit të testimin të softuerit dhe luajnë një rol të rëndësishëm në sigurimin e cilësisë së aplikacioneve të uebit. Ata janë skenarë të dizajnuar për të verifikuar funksionalitetin e një aplikacioni dhe për të identifikuar çdo problem potencial. Në këtë kapitull, do të përdorim të njëjtët skenare testimit për të parë qasjen e tre teknologjive të automatizimit të uebit: Selenium, Cypress dhe TestComplete. Për rezultatet e secilës teknologji, do të bëjmë një krahasim të hollësishëm dhe do të analizojmë pikët e forta dhe të dobëta të secilës. Ky studim synon të ofrojë një vlerësim të thellë të aftësive të secilës teknologji dhe të ndihmojë organizatat të marrin vendime të informuara rreth zgjedhjes së teknologjisë më të përshtatshme për nevojat e tyre të specifikuara të testimi të uebit.

Funksonaliteti 1: Identifikimi dhe Hyrja e Përdoruesit ne Llogari

ID e testit	Emri i Testit	Përshkrimi	Parakushtet	Hapat e Testit	Rezultatet e Pritura	Statusi
TC001	Log in - Wrong credentials (Selenium)	Verifikoni që një përdorues nuk mund të kyçet me kredenciale të gabuara.	Përdoruesi duhet të jetë në faqen e hyrjes.	1. Shko në faqen e hyrjes. 2. Vendosni 'test' si emrin e përdoruesit dhe 'password' si fjalëkalimin. 3. Kliko	Shfaqet mesazhi i gabimit. Përdoruesi nuk kyçet.	Kaluar

				butonin e hyrjes.		
TC002	Log in - Successful Login (Selenium)	Verifikoni që një përdorues mund të kyçet me kredenciale të vlefshme.	Përdoruesi duhet të ketë kredenciale të vlefshme për hyrje.	1. Shko në faqen e hyrjes. 2. Vendosni 'standard_user' si emrin e përdoruesit dhe 'secret_sauce' si fjalëkalimin. 3. Kliko butonin e hyrjes.	Përdoruesi kyçet me sukses dhe drejtohet në faqen kryesore.	Kaluar
TC003	Log out (Selenium)	Verifikoni që një përdorues mund të dilni nga llogaria e tyre.	Përdoruesi duhet të jetë kyçur.	1. Shko në faqen kryesore. 2. Hapni menunë. 3. Kliko butonin "Log Out" në fund të menysë.	Përdoruesi del nga llogaria dhe kthehet në faqen e hyrjes.	Kaluar.

Tabela 2: Identifikimi dhe Hyrja e Përdoruesit ne Llogari

Funksionaliteti 2: Listimi dhe filtrimi i produkteve

ID e testit	Emri i Testit	Përshkrimi	Parakushtet	Hapat e Testit	Rezultatet e Pritura	Statusi
TC001	Pamja e produkteve dhe Detajet	Verifikoni që një përdorues mund të	Përdoruesi duhet të jetë kyçur dhe të	1. Shko në faqen kryesore. 2.	Detajet e produktit janë të sakta	Kaluar.

	e Produktit (Selenium)	shohë detajet e një produkti dhe kthehet pas nga faqja e detajeve.	ketë produktet në faqen kryesore.	Shfaqet lista e produkteve. 3. Kliko në produktin e parë. 4. Shfaqet faqja e detajeve të produktit. 5. Verifikoni se detajet janë të sakta. 6. Kliko butonin "Kthehu te produktet". 7. Shfaqet përsëri faqja e produktit.	dhe përdoruesi kthehet me sukses te lista e produkteve.	
TC002	Filtrimi i Produktit sipas Çmimit (Selenium)	Verifikoni që një përdorues mund të filtrojë produktet sipas çmimit dhe ti shikojë ato në rendin e përzgjedhur.	Përdoruesi duhet të jetë kyçur dhe të ketë produktet në faqen kryesore.	1. Shko në faqen kryesore. 2. Shfaqet lista e produkteve. 3. Kliko në dropdownin e filtrimit. 4. Përzgjidhni opcionin "Çmimi në	Produktet janë të radhitura sipas çmimit të zbritur dhe përdoruesi i sheh ato në rendin e përzgjedhur.	Kaluar

				zbritje". 5. Produktet renditen sipas çmimit të përzgjedhur.		

Tabela 3: Listimi dhe filtrimi i produkteve

Funksionaliteti 3: Veprime ne shporten e blerjeve

ID e testit	Emri i Testit	Përshkrimi	Parakushtet	Hapat e Testit	Rezultatet e Pritura	Statusi
TC001	Shtimi në Shportën e Blerjeve dhe Shikimi i Detajeve (Selenium)	Verifikoni që një përdorues mund të shtojë një produkt në shportën e blerjeve, ta shikojë shportën dhe të verifikojë detajet e produktit.	Përdoruesi duhet të jetë kyçur dhe të ketë produktet në faqen kryesore.	1. Shko në faqen kryesore. 2. Shfaqet lista e produkteve. 3. Kliko në butonin "Shto në shportë" për produktin "Sauce Labs Backpack". 4. Shporta e blerjeve shfaq numrin "1" në ikonën e shportës në këndin e djathtë të	Produkti është i pranishëm në shportën e blerjeve dhe detajet janë të sakta.	Kaluar

				sipërme. 5. Kliko në ikonën e shportës së blerjeve. 6. Produkti i shtuar është i listuar në shportë dhe detajet janë të sakta.		
TC002	Hiqja e Produktit nga Shporta e Blerjeve	Verifikoni që një përdorues mund të hiqë një produkt nga shporta e blerjeve.	Përdoruesi duhet të jetë kyçur dhe të ketë një produkt në shportën e blerjeve.	1. Kliko në butonin "Hiq" për produktin në shportën e blerjeve. 2. Produkti është larguar nga shporta e blerjeve.	Produkti është larguar me sukses nga shporta e blerjeve.	Kaluar
TC003	Shtimi i Produktit të Dytë në Shportën e Blerjeve	Verifikoni që një përdorues mund të shtojë një produkt të dytë në shportën e blerjeve dhe ta shikojë atë.	Përdoruesi duhet të jetë kyçur dhe të ketë produktet në faqen kryesore.	1. Kliko në butonin "Vazhdo të bësh blerje" pasi të kesh hequr produktin e parë nga shporta e blerjeve. 2. Shfaqet faqja e	Produkti i dytë është i listuar me sukses në shportën e blerjeve.	Kaluar

				përmbledhjes së produkteve. 3. Kliko në butonin "Shto në shportë" për produktin "Sauce Labs Bike Light". 4. Kliko në ikonën e shportës së blerjeve. 5. Produkti i dytë është i listuar në shportë dhe është i pranishëm.		

Tabela 4: Veprime ne shporten e blerjeve

Funksionaliteti 4: Perfundimi me sukses i blerjes

ID e testit	Emri i Testit	Përshkrimi	Parakushtet	Hapat e Testit	Rezultatet e Pritura	Statusi
TC001	Procesi i Blerjes dhe Anulimi	Verifikoni që një përdorues mund të bëjë blerjen e një produkti, ta anulojë atë	Përdoruesi duhet të jetë kyçur dhe të ketë produktin në	1. Përdoruesi navigon në shportën e blerjes. 2. Produkti është në	Blerja dhe anulimi u kryen me sukses, dhe përdoruesi u ridrejtua në	Kaluar

		dhe pastaj të përfundojë blerjen me sukses.	shportën e blerjes.	shportë. 3. Përdoruesi klikon butonin "Kontrollo". 4. Faqja e kontrollit hapet. 5. Përdoruesi klikon butonin "Anulo". 6. Përdoruesi ridrejtohet përsëri në faqen e shportës së blerjes. 7. Përdoruesi klikon përsëri butonin "Kontrollo". 8. Faqja e kontrollit hapet përsëri. 9. Përdoruesi plotëson informacionin e kërkuar dhe klikon "Vazhdo". 10. Detajet e blerjes shfaqen. 11.	faqen e porosisë së suksesshme.	

				Përdoruesi klikon butonin "Përfundo". 12. Faqja e porosisë së suksesshme shfaqet.		

Tabela 5: Perfundimi me sukses i blerjes

7.3 Gjetja e elementeve ne web

Para se të implementojme testet me Cypress dhe Selenium, është e rëndësishme të kuptojme si të gjejme elementet në web. Identifikimi i elementeve ne web eshte i rendesishem per implementimin e skenareve te testimit.

Per gjetjen e elementeve ne web, mund te hapim nje nga shfletuesit dhe te elementi i deshiruar klikojme me te djathten e mouse-it. Me pas zgjedhim opsion inspect dhe pamja e inspektorit do te hapet, si ne figuren me poshte:

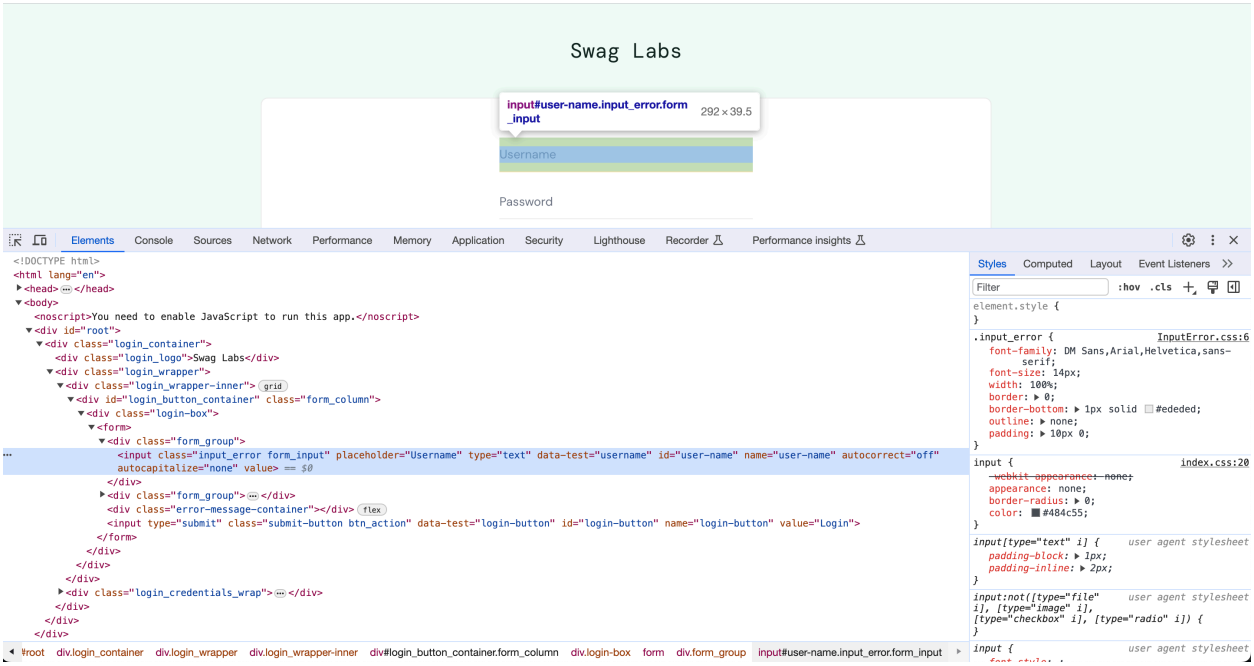

Figura 1: Identifikimi e elementeve ne web

Lehtesisht veme re, se per inputin ‘Username’ kemi tag-un ‘input’, atributin ‘class’ dhe gjithashtu ‘id’. Secila nga keto opsione mund te perdoret per kapjen e ketij elementi web. Ne rastet kur atributi ‘id’ eshte prezent, keshillohet kapja e elementit me kete atribut, pasi eshte unik. Me pas keshillohet perdorimi i atributit ‘class’ ose ne rast se kemi nje element web te tipit buton ose link, mund te perdoret dhe gjetja e elementit me ate te tekstit. Kjo pasi nuk eshte shume e zakonshme, qe dy butona me te njejtin emer te jene prezent ne nje faqe.

Per raste me te vecanta, ku kapja e elementit nuk behet me metodat e mesiperme, klikojme me te djathten e mouse-it ne tag-un e elemenit dhe me pas klikojme opsionin ‘Copy’. Nje dritare e re do te hapet dhe disa opsione per kapjen e elementit, si ne figuren me poshte:

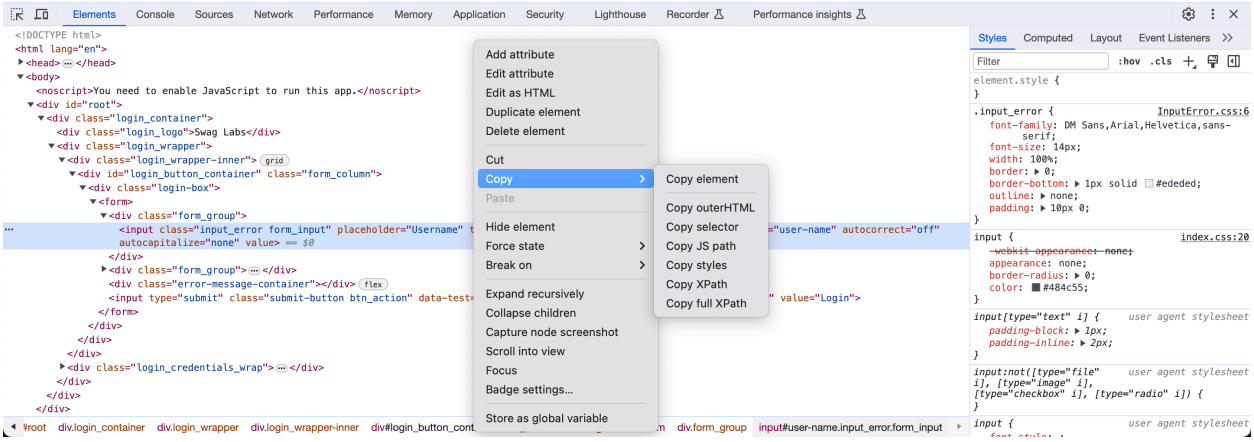

Figura 2: Kopjimi i selektoreve te sugjeruar

Sugjerohet perdorimi i ‘selector’ ose ‘XPath’. Sigurohuni qe selektori i zgjedhur te jete unik, ne kete menyre nuk do te kemi anomali ne momentin qe testet ekzekutohen. Ne momentin qe selektori identifikon me teper se nje element web, Selenium dhe Cypress do te perdorin elementin e pare ne liste.

7.4 Aplikacioni Sauce Demo

Aplikacioni Sauce Demo është një aplikacion web i disponueshëm për testim që përdoret shpesh për demonstrimin e veprimit të testimit automati.

Përmes aplikacionit Sauce Demo, mund të bëhen shembuj të testimit për çështje të ndryshme të një aplikacioni web, përfshirë veprimet e kyçjes, kërkimin e produkteve, shtimin në shportë, dhe procesin e

blerjes. Ky aplikacion është një mjedisi i pershtatshem për të demonstruar aftësitë dhe veçoritë e secilës nga këto teknologji të automatizimit të uebit.

Gjatë këtij studimi krahasues, aplikacioni web Sauce Demo do të përdoret për të ekzekutuar dhe shqyrtuar testet me Selenium, Cypress dhe TestComplete. Rezultatet dhe performanca e secilës teknologji do të vlerësohen në kontekstin e këtij aplikacioni. Ky aplikacion është një ambient i rëndësishëm për t'u siguruar që testimet janë të përshtatshme dhe që secila teknologji ka përputhshmëri me sfidat reale të një aplikacioni web të zakonshëm.

7.5 Projekti Selenium

Projekti është i organizuar në mënyrë të qartë dhe të strukturuar. Ka disa elemente kryesore:

Testet e Cucumber: Këto janë skenarët e testimit të shkruar në gjuhën e Cucumber. Ata përshkruajnë hapat e testeve në një mënyrë të lexueshme dhe janë të organizuar në dosje të ndara përkatëse.

Kod Selenium: Ky është kod Python i përdorur për të ekzekutuar veprimet e testeve në Selenium. Përdorimi i Selenium lejon automatin e faqes për të kryer veprimet e përshtatura në shkrimin e testeve.

Konfigurimi i Shfletuesit: Projekti përdor Selenium WebDriver për të kontrolluar një shfletues web. Konfigurimi i shfletuesit është përfshirë për të siguruar që teste ekzekutohen në një mjedis të përshtatshëm.

Skedarët e Konfigurimit: Skedarët e konfigurimit janë përdorur për të vendosur parametrat e projektin, përfshirë konfigurimin e WebDriver dhe shfletuesit.

Pasqyrat e Testimit dhe Rezultatet: Pasqyrat e testimit dhe rezultatet e testit janë dokumentuar në mënyrë të qartë për të ndjekur progresin e testit dhe për të vlerësuar nëse testi është kaluar ose jo.

Cucumber:

1. Cucumber është një mjet për zhvillimin e bazuar në sjellje (BDD) që përdor një gjuhë të thjeshtë dhe të kuptueshme për të shkruar specifikime të testimi.
2. Shpërndahet në disa versione për shumicën e gjuhëve programuese.
3. Cucumber lejon shkrimin e specifikimeve të testimi në një format të lexueshëm nga njerëzit duke përdorur sintaksën e "scenario-ve" dhe "steps" të legjendës së testimi.
4. Përmban një përkthyes që ekzekuton këto specifikime dhe i lidh ato me kodin e testimi, në mënyrë që specifikimet të ekzekutohen automatikisht nëpërmjet kodit të testit.

Gherkin:

1. Gherkin është gjuha ose sintaksa e përdorur për të shkruar specifikimet e testimi në Cucumber.
2. Është një gjuhë me tekst të thjeshtë dhe një strukturë të rregullt që lejon për shkrimin e hapat e testimi në mënyrë të lexueshme nga njerëzit dhe më pas ekzekutimi i tyre automatikisht.
3. Specifikimet e testimi të shkruara në Gherkin janë të përkthyer në kod të ekzekutueshëm nga Cucumber për të kryer testimin automatik.

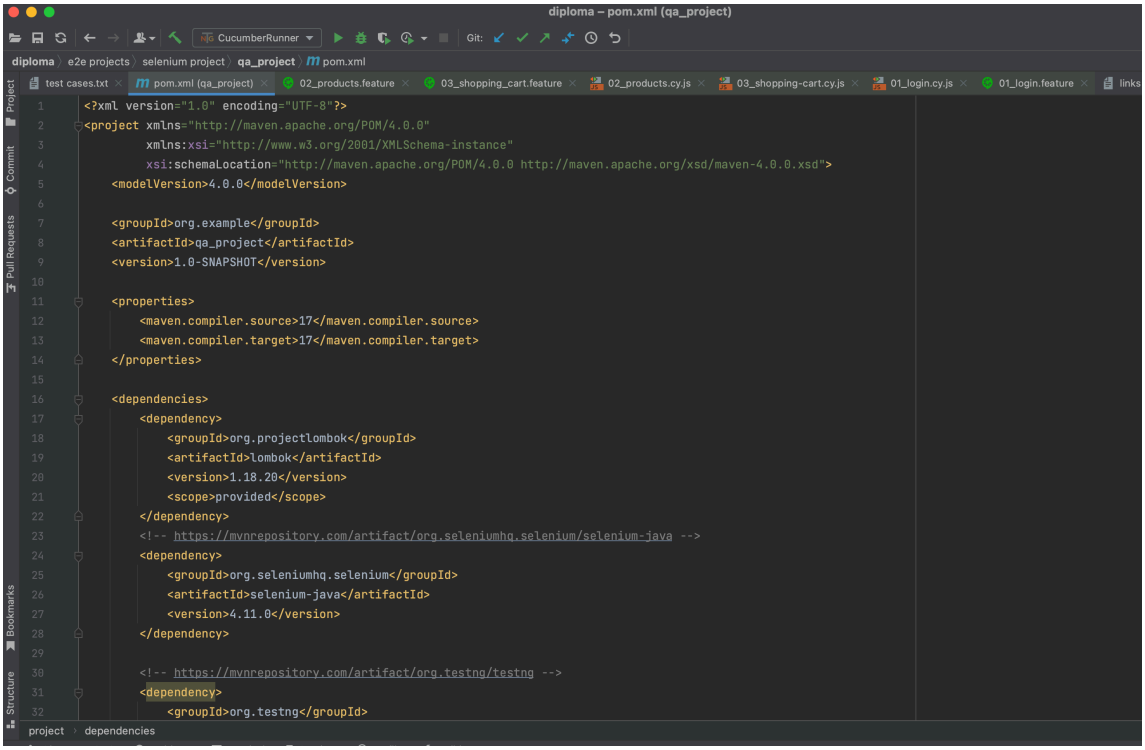

Figura 3: Maven

```java
package test;

import ...

@CucumberOptions(
        features = "src/test/java/feature",
        glue = "qa.steps",
        tags = "@Demo"
)
public class CucumberRunner extends AbstractTestNGCucumberTests {

    @BeforeClass(alwaysRun = true)
    @Override
    public void setUpClass(ITestContext context) {
        Driver.start(DriverOption.FIREFOX);
        super.setUpClass(context);
    }

    @AfterClass(alwaysRun = true)
    @Override
    public void tearDownClass() {
        Driver.quit();
        super.tearDownClass();
    }
}
```

Figura 4: Klasa baze ku fillon ekzekutimi i testeve

```
Feature:01 - Identifikimi dhe Hyrja e Përdoruesit

  #@Demo
  @01 @01.1
  Scenario: 01.1 - Unsuccessful Login - Wrong credentials
    Given the homepage is opened
    Then we type 'test' in the 'Username' input field
    And we type 'password' in the 'Password' input field
    And click 'Login' button
    Then make sure an error message with the following text is shown
      | Epic sadface: Username and password do not match any user in this service |

  #@Demo
  @01 @01.2
  Scenario: 01.2 - Successful Login
    Given the homepage is opened
    Then we type 'standard_user' in the 'Username' input field
    And we type 'secret_sauce' in the 'Password' input field
    And click 'Login' button
    Then make sure user navigates to products page
```

Figura 5: Shembull i nje Cucumber feature

```java
import ...

@NoArgsConstructor(access = AccessLevel.PRIVATE)
public class DriverFactory {

    private static final String CHROME_DRIVER = "webdriver.chrome.driver";
    private static final String GECKO_DRIVER = "webdriver.gecko.driver";
    private static final String CHROME = "src/main/resources/driver/macos/chromedriver";
    private static final String FIREFOX = "src/main/resources/driver/macos/geckodriver";

    public static WebDriver getDriver(final DriverOption driverOption){
        switch (driverOption){
            case CHROME:
                initChrome();
                ChromeDriverService service = new ChromeDriverService.Builder()
                    .withLogFile(new File( pathname: "src/target/"))
                    .build();
                return new ChromeDriver(service);
            case FIREFOX:
                initFirefox();
                return new FirefoxDriver();
            case IE:
                return new InternetExplorerDriver();
            default:
                return new ChromeDriver();
        }
    }

    private static void initChrome() { System.setProperty(CHROME_DRIVER, CHROME); }

    private static void initFirefox() { System.setProperty(GECKO_DRIVER, FIREFOX); }
}
```

Figura 6: Driver Factory – Klasa baze ku behet incializimi i WebDriver

Shpjegimi i kodit:

Fragment i meposhtem i kodit perdoret për te krijuar dhe konfiguruar një instancë të WebDriver për testimin automatik të një aplikacioni të internetit. WebDriver është një mjet i përdorur për të kontrolluar dhe ekzekutuar teste mbi shfletuesit web si Chrome, Firefox, dhe Internet Explorer.

Disa pika kyçe në kodin e dhënë janë:

1. Importimi i paketave dhe klasave: Kjo pjesë e kodit përdor disa paketa dhe klasa të importuara për të punuar me WebDriver dhe për të konfiguruar shfletuesin e dëshiruar.

2. Klasa `DriverFactory`: Kjo është një klasë që përmban metoda për krijimin e instancave të WebDriver, bazuar në opsionin e zgjedhur të shfletuesit (si Chrome, Firefox, IE).

3. Metoda `getDriver`: Ky është metoda kryesore e kësaj klase. Ajo pranon një opsion për shfletuesin (p.sh., CHROME ose FIREFOX) dhe bazuar në këtë opsion, krijon dhe kthen një instancë të përshtatur të WebDriver për shfletuesin e zgjedhur.

4. Metodat `initChrome` dhe `initFirefox`: Këto metoda përdoren për të vendosur sistem variablën për drejtuesin e Chrome dhe Firefox përpara se WebDriver të inicializohet. Ky është një hap i rëndësishëm për të siguruar që WebDriver do të gjejë dhe përdorë driver-in e duhur të shfletuesit.

```
@NoArgsConstructor(access = AccessLevel.PRIVATE)
public class DriverFactory {
    private static final String CHROME_DRIVER = "webdriver.chrome.driver";
    private static final String GECKO_DRIVER = "webdriver.gecko.driver";
    private static final String CHROME = "src/main/resources/driver/macos/chromedriver";
    private static final String FIREFOX = "src/main/resources/driver/macos/geckodriver";

    public static WebDriver getDriver(final DriverOption driverOption){
        switch (driverOption){
            case CHROME:
                initChrome();
                ChromeDriverService service = new ChromeDriverService.Builder()
                    .withLogFile(new File("src/target/"))
                    .build();
                return new ChromeDriver(service);
            case FIREFOX:
                initFirefox();
                return new FirefoxDriver();
            case IE:
                return new InternetExplorerDriver();
            default:
```

```
            return new ChromeDriver();
        }
    }

    private static void initChrome(){
        System.setProperty(CHROME_DRIVER, CHROME);
    }

    private static void initFirefox(){
        System.setProperty(GECKO_DRIVER, FIREFOX);
    }
}
```

Ne fragmentin e radhes do te shofim klasen CheckoutPage. Kjo klase perdoret per te deklaruar elementet e web-it qe gjenden ne faqen e hapit te pare dhe te dyte, kur behet procesimi i blerjes se nje produkti. Gjithashtu perdoret per te deklaruar metodat qe do te perdoren perkundrejt ketyre elementeve duke krijuar kod te riperdorshem dhe te thjeshtesuar.

```
/**
 * Checkout page (step one and step two)
 */
public class CheckoutPage extends Page {
    @FindBy(id = "cancel")
    public static WebElement cancelButton;
    @FindBy(id = "continue")
    public static WebElement continueButton;
```

```
    @FindBy(id = "finish")
    public static WebElement finishButton;


    public void clickButton(String buttonName) {
        WebElement webElement = switch (buttonName) {
            case "Cancel" -> cancelButton;
            case "Continue" -> continueButton;
            case "Finish" -> finishButton;
            default -> throw new NotFoundException(buttonName + " button not found!");
        };

        Driver.getWait().until(ExpectedConditions.elementToBeClickable(webElement));
        webElement.click();
    }
}
```

Ne dy fragmentet pasardhese do te shofim sesi duket nje test ne Cucumber dhe implementimi i tij ne prapaskene.

```
  #@Demo
  @01 @01.1
  Scenario: 01.1 - Unsuccessful Login - Wrong credentials
    Given the homepage is opened
    Then we type 'test' in the 'Username' input field
    And we type 'password' in the 'Password' input field
```

```
    And click 'Login' button
    Then make sure an error message with the following text is shown
      | Epic sadface: Username and password do not match any user in this service |


public class LoginSteps {
    private final LoginPage loginPage;

    public LoginSteps() {
        loginPage = new LoginPage();
    }

    @Then("^we type '(.+?)' in the '(.+?)' input field$")
    public void navigateToHomePage(String text, String inputField) {
        loginPage.typeInTheInputField(text, inputField);
    }

    @And("^we click '(.+?)' button from modal$")
    public void clickModalButton(String buttonName) {
        loginPage.clickModalButton(buttonName);
    }

    @Then("^order is successfully placed and the following message is shown$")
    public void checkOrderMessage(DataTable dataTable) {
```

```
        Driver.getWait().until(ExpectedConditions.textToBe(By.cssSelector(".showSweetAlert h2"),
dataTable.cells().get(0).get(0)));
    }
}
```

Lidhja nepermjet nje hapi ne skenarin e cucumber dhe kodit te java behet nepermjet text qe gjendet brenda @Then, @And, @When. E rendesishme eshte qe pjesa e tagut mund te ndryshoje gjate riperdorimit te hapit te skenarit. Keto lidheza perdoren qe fjalite e skenarit te kene nje lidhje midis njera tjeter dhe te jene sa me te kuptueshme per personat qe nuk dine te lexojne kod.

Gjithashtu karakteret (.+?) simbolizojne nje parameter qe kalon nga skenari i cucumber ne kodin e implementuar ne prapaskene. Tipi i parametrit eshte 'String'. Gjithashtu behet e mundur dhe kalimi i te dhenave tabelare si parameter. Kto te dhena kalojne si nje objekt DataTable dhe iterimi i ketyre te dhenave eshte mjaft i ngjashem me bredhjen e te dhenave ne nje tabele dy dimensionale.

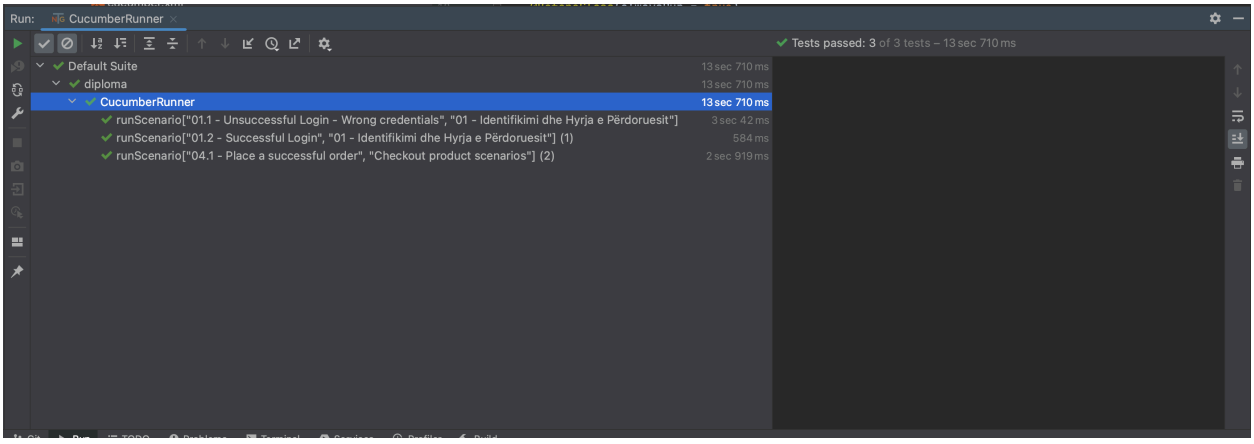

Figura 7: Ekzekutimi i funksionalitetit nr. 1 - Identifikimi dhe Hyrja e Përdoruesit

7.6 Projekti Cypress

Insatimi i Cypress behet duke perdorur komanden npm install 'cypress --save-dev'. Parakusht eshte qe ne mjedisin e punes te jete i instaluar Node.js.

Pas instalimit ne mund te inicializojme Cypress duke ekzekutuar komanden 'npx cypress open'.

Me pas do te hapet nderfaqja e Test Runner si me poshte:

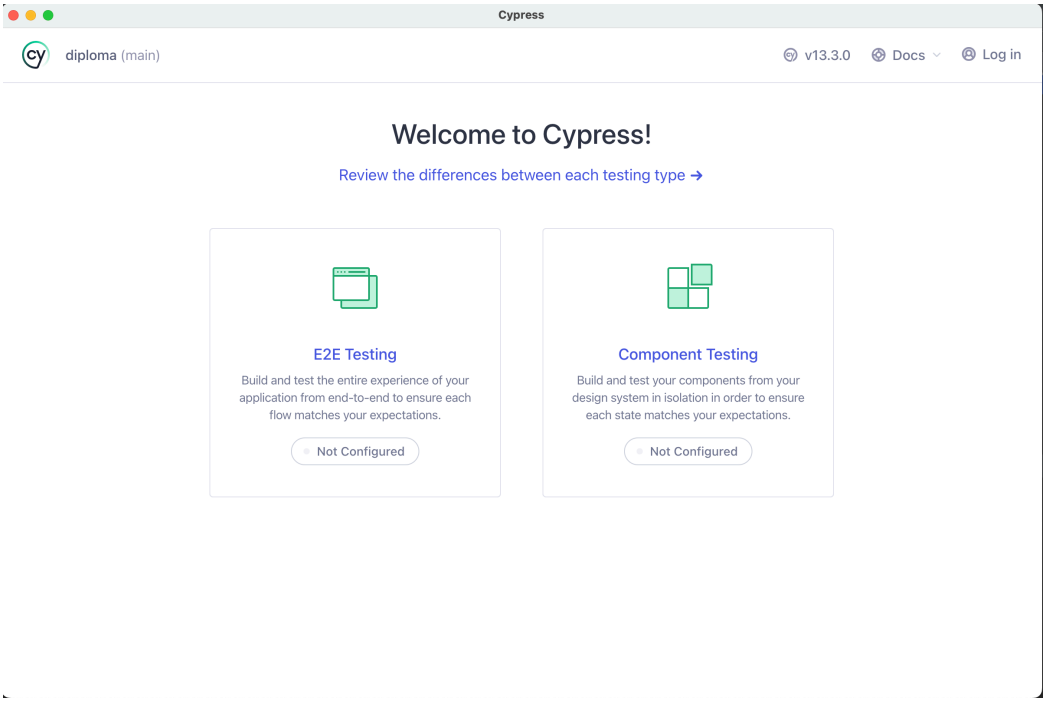

Figura 8: Nderfaqja Test Runner Cypress

Me pas zgjedhim opsionin E2E Testing dhe me pas ne nderfaqe do te kemi mundesine te zgjedhim nje shfletues per ekzekutimin e skenareve te testimit.

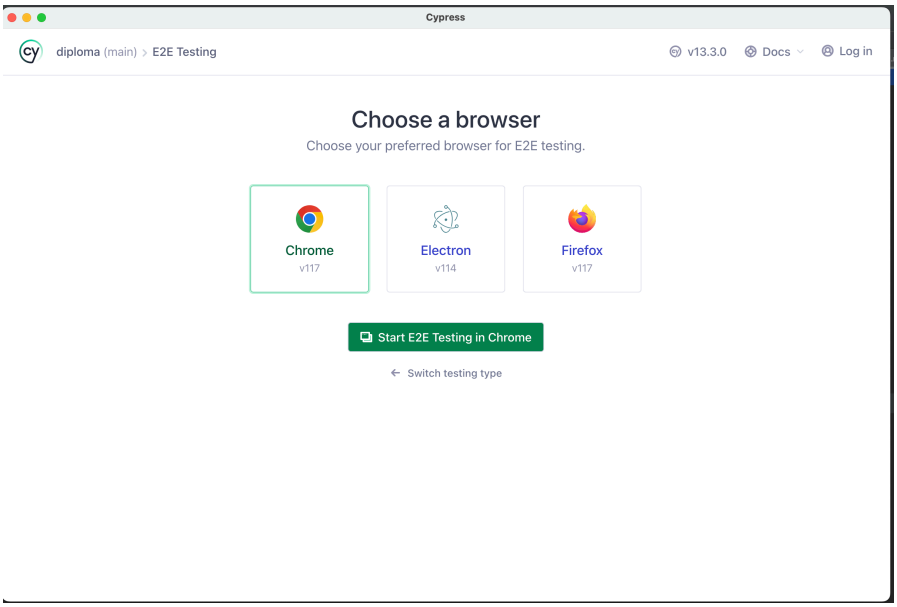

Figura 9: Perzgjedhja e shfletuesit

Me pas klikojme ne butonin ‘Start E2E Testing in Chrome’ dhe ne nderfaqe do te shofim si me poshte listen e funksionaliteve kryesore qe do te testohen.

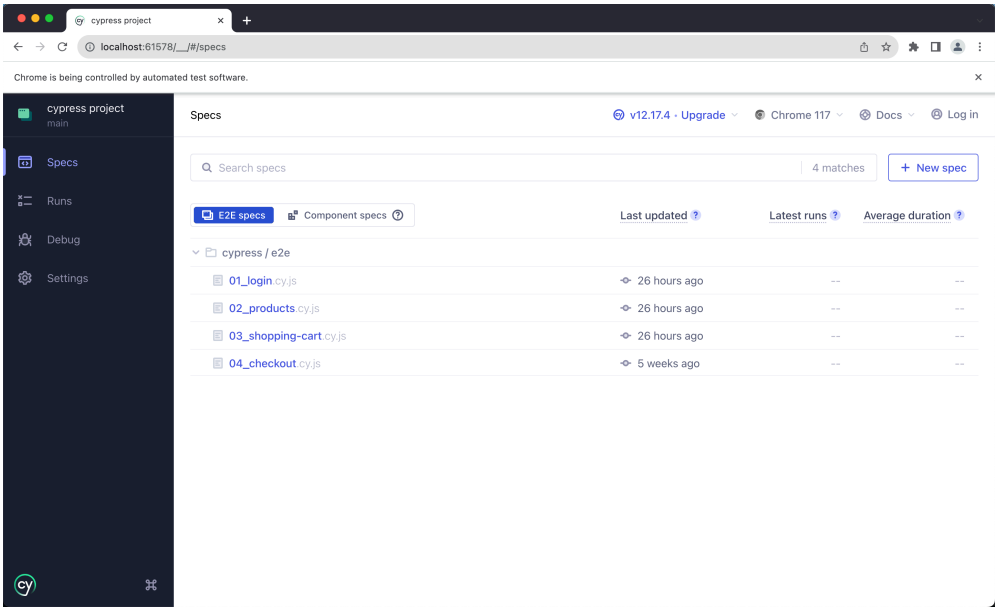

Figura 10: Cypress Specifikimet

Klikojme ne funksionalitetin e pare dhe do te shofim qe ekzekutimi i testeve do te filloje. Me poshte do te shofim dhe rezultatin e tij.

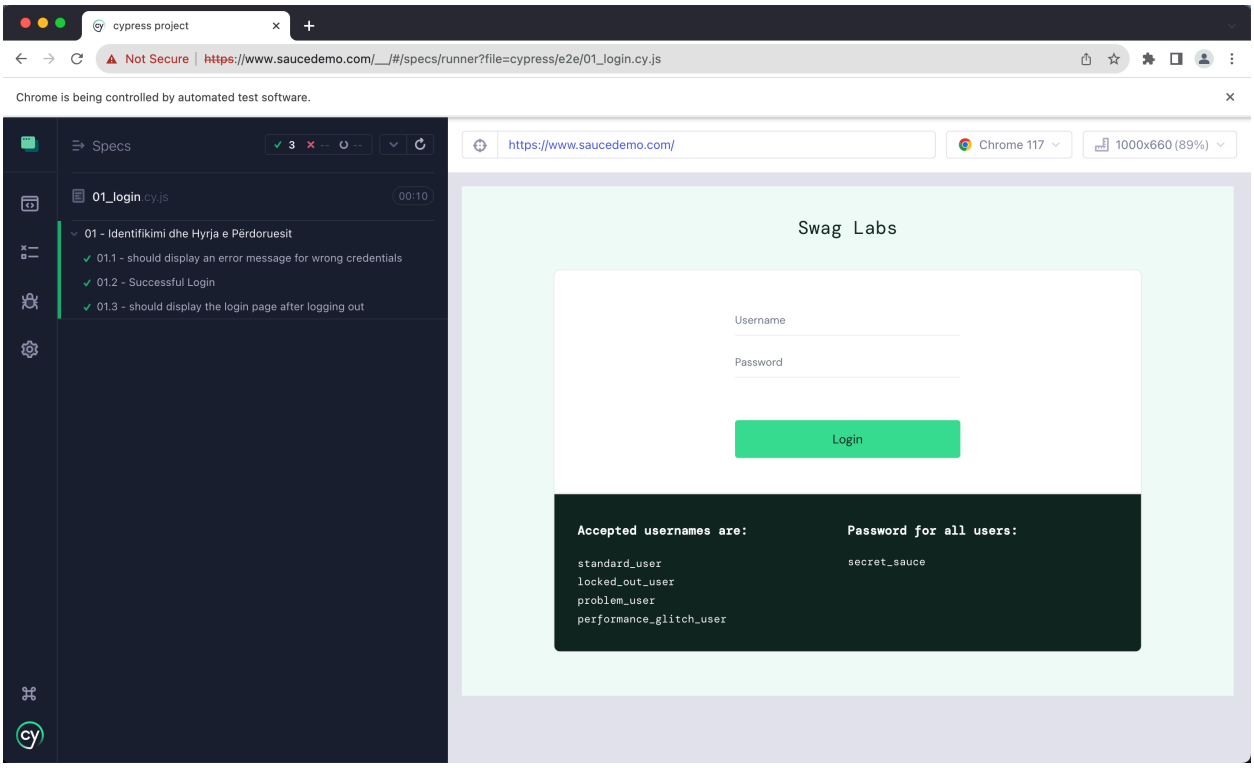

Figura 11: Ekzekutimi i funksionalitetit Identifikimi dhe Hyrja e Perdoruesit

Me poshte do te gjeni dhe disa figura te tjera, ku jane disa nga komponentet baze te krijimit te mjedisit te pershtatshem per te ekzekutuar testet dhe konfiguruar Cypress.

```
{
  "name": "cypress-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "cypress": "^12.17.4"
  }
}
```

Figura 12: Konfigurimet baze te Cypress

Figura 13: Skenar testimi Cypress

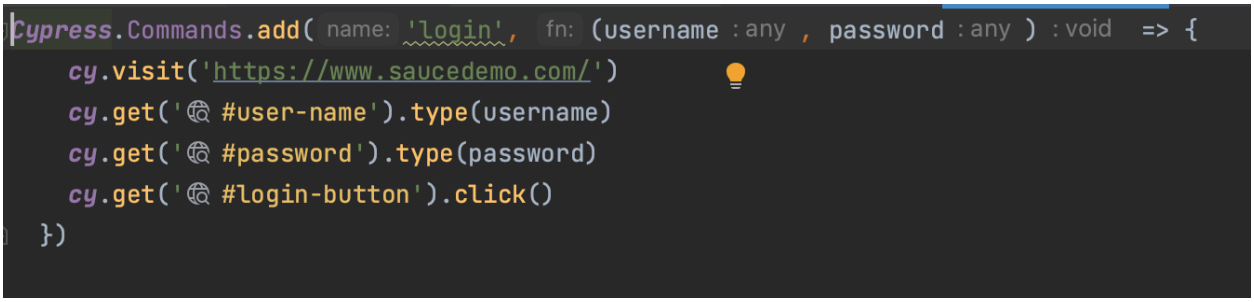

Figura 14: Krijimi i nje komande te riperdorshme per kycjen e perdoruesit

7.7 Projekti Test Complete

Fillimisht, duhet të sigurohemi që kemi instaluar TestComplete në sistemin tone. Shkarkojme versionin e fundit të TestComplete nga faqja zyrtare e TestComplete dhe ndjekim udhëzimet e instalimit.

Hapim TestComplete dhe krijojme një projekt të ri, dhe me pas do te shfaqet nderfaqja e Test Complete si me poshte:

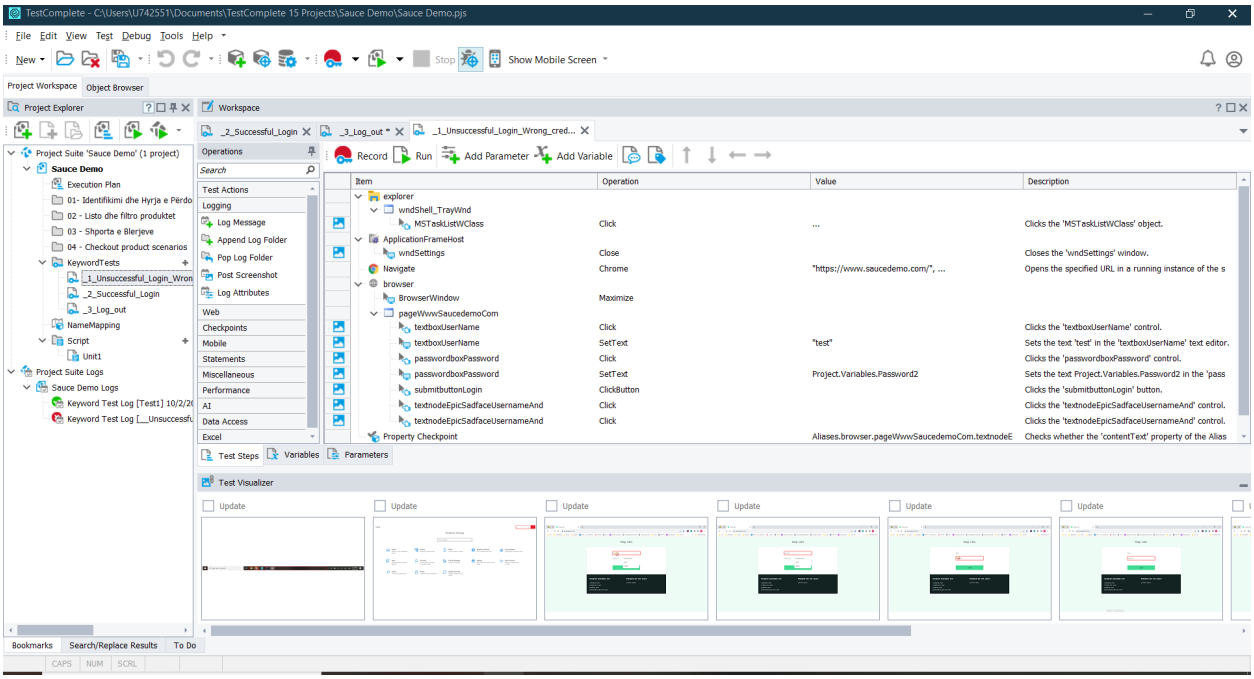

Figura 15: Test Complete – Hapsira e projekteve

Me pas, do te klikojme ne butonin ‘Record’ qe gjendet ne panelin kryesor, dhe hapim shfletuesin Google Chrome. Ne kete momentin fillon dhe regjistrimi i skenarit tone. Cdo veprim qe do te kryejme ne nderfaqen e shfletuesit tone, do te regjistrohet ne Test Complete. Gjate kohes qe ekzekutojme te gjitha hapat e skenareve qe jane percaktuar ne seksion 7.2, mund te vendosim dhe kontrolle te ndryshme si psh. teksti qe njofton perdoruesin, qe fjalekalimi i vendosur eshte I gabuar. Kjo behet duke klikuar opsionin Add Check ne dritaren dialoguese qe shfaqet ne shfletuesin tone.

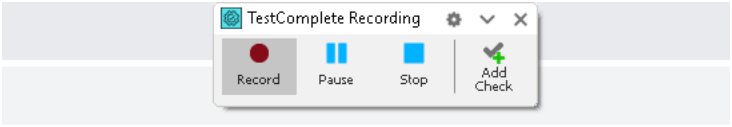

Figura 16: Dritarja dialoguese kur regjistrojme nje skenar testimi

Me pas mund te zgjedhim nje nga shume opsionet e shfaqura, ku ne rastin tone mund te zgjedhim opsionin tekst. Pasi kemi perfunduar me skenarin tone klikojme ne butonin ‘Stop’. Ne kete momentin kemi mundesine dhe qe te bejme ndryshimin e emrit te skenarit tone, si dhe direktorine ku do te ruhet.

Pasi kemi perfunduar dhe me ruajtjen e testit do te shofim qe te gjitha hapat qe ne ekzekutuam jane ruajtur si me poshte:

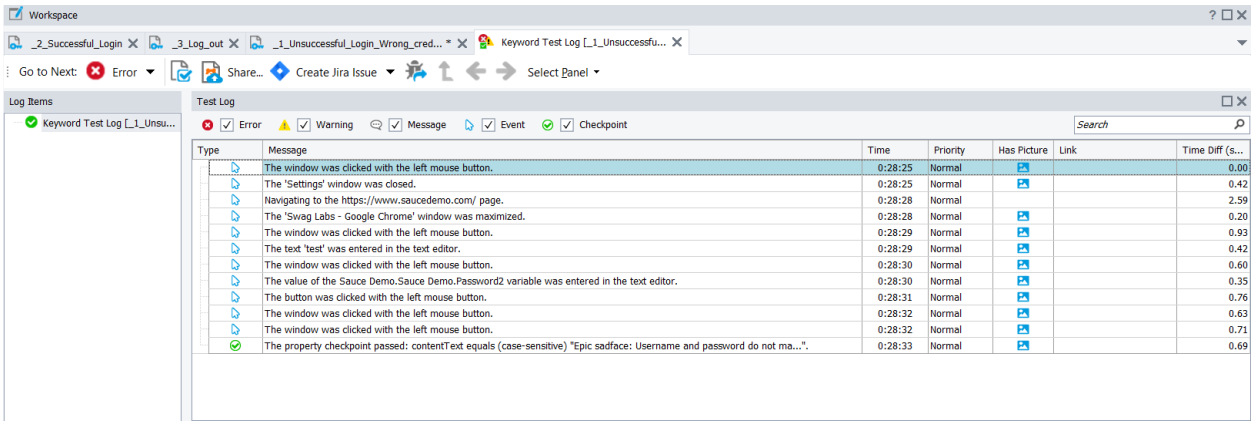

Figura 17: Detajet e skenarit te testimit

Bashkangjitur per secilin hap do te gjejme dhe nga nje foto. Fotot shfaqen te gjitha dhe ne panelin poshte skenarit.

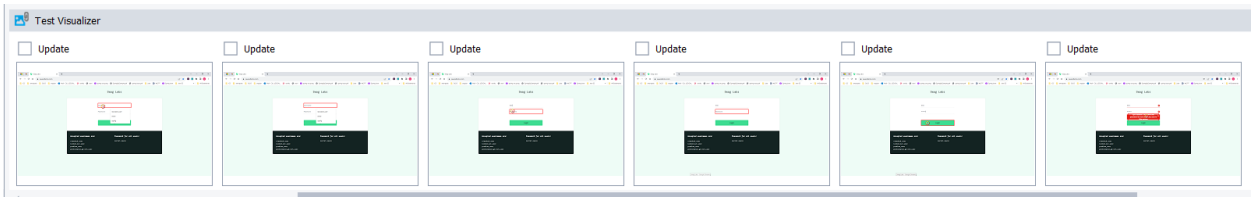

Figura 18: Vizualizimi i skenarit te testimit

Ne te njejten menyre mund te regjistrojme dhe skenare te tjere. Pasi kemi perfunduar me regjistrimin e tyre mund te bejme ekzekutimin e ketyre testeve ne shfletues te ndryshem dhe gjithashtu duke perdorur parametra me vlera te reja.

KAPITULLI 8

8.1 Rezultati i Studimit

Pas analizës së përfshirë dhe ekzekutimit të testeve në Selenium, Cypress dhe TestComplete, mund të nxjerrim disa përfundime të rëndësishme. Secila teknologji ka avantazhet dhe kufizimet e saj, dhe rekomandimet për përdorimin e tyre variojnë sipas karakteristikave të projektin. Këtu janë përfundimet kryesore:

Selenium është një platformë e njohur për automatizimin e uebit dhe është ideale për projekte të mëdha dhe komplekse. Për zhvilluesit me përvojë në kodim dhe nevoja të avancuara të testimit, Selenium është një zgjedhje e fuqishme. Megjithatë, ka disa vështirësi ne konfigurimin fillestar dhe mund të jetë më i vështirë për përdoruesit e rinj.

Cypress është një teknologji e ri i testimit me një fokus të veçantë në thjeshtësi dhe lehtësi të përdorimit. Për projekte të vogla dhe të mesme, Cypress ofron një eksperiencë të mbarë me shpërndarjen e saj të ndërfaqes së përdoruesit dhe mjetet e integruara të testimit. Por, për projektet e mëdha dhe komplekse, mund të ketë sfida me performancën. Gjithashtu mbeshtetja nga komuniteti nuk eshte akoma ne nivelet e Selenium.

TestComplete është një mjet automatizimi komercial me një spektër të gjerë të karakteristikave dhe mbështetje të ndryshme. Është i përshtatshëm për projektet e mëdha dhe të kompleksuara që kërkojnë veçori të përparuara të testimit dhe raportim. Për organizatat që janë të gatshme të investojnë në një mjet komercial dhe kanë nevojë për ndihmë të specializuar në automatizim, TestComplete është një zgjedhje e mirë.

Kriteret	**Selenium**	**Cypress**	**TestComplete**
Burimi i hapur	Po	Po	Jo
Lloji i Licencës	Licenca Apache 2.0	Licenca MIT	Komerciale
Lehtësia e Ndarjes	Mesatare	E lehtë	E lehtë
Gjuhët e Programimit	Java, Python, C#, etj.	JavaScript	JavaScript, Python, VBScript, etj.

Kompatibiliteti me Shfletuesin	Shumë i mirë	Shumë i mirë	Shumë i mirë
Testimi i GUI-s	Po	Po	Po
Testimi i shfletuesit të ndryshëm	Po	Po	Po
Testimi i aplikacioneve mobile	Po (me Appium ose mjete të tjera)	I kufizuar (i fokusuar tek aplikacionet uebi)	Po
Regjistruesimi i Testit	I kufizuar (mund të përdoren mjete të palëve të treta)	Po	Po
Mbështetja e Komunitetit	E fortë	Rritet, por më e vogël se Selenium	E fortë
Mbështetja komerciale	E kufizuar (përmes furnizuesve të palëve të treta)	E kufizuar (përmes furnizuesve të palëve të treta)	Po
Integrimi i Vazhdueshëm	Po (integrohet me mjete CI/CD)	I kufizuar (kërkon konfigurim shtesë)	Po (integrim i pashëm me CI/CD)
Njohja e Objekteve	Ndryshon në varësi të zbatimit të WebDriver	Njohje të avancuara të objekteve	Njohje të avancuara të objekteve
Raportimi i Testit	Ndryshon në varësi të kuadratit të testimi	I kufizuar (kërkon shtesa të shtuara)	Raportim i gjerë me ndihmë vizuale
Kostot e Licencës	Falas (burimi i hapur)	Falas (burimi i hapur)	Komerciale (varion në varësi të përdorimit)
Maturia	E zhvilluar dhe e përdorur gjerësisht	Ndryshon dhe përmirësohet shpejt	E zhvilluar dhe me karakteristika të pasura

Tabela 6: Krahasim Selenium, Cypress dhe Test Complete

Referenca

Certified Tester Foundation Level (CTFL) v4.0 Syllabus by International Software Testing Qualifications Board.

Selenium Documentation (https://www.selenium.dev/documentation/)

Cypress Documentation (https://docs.cypress.io/guides/overview/why-cypress)

Cypress Official Website (https://www.cypress.io)

Test Complete by SmartBear (https://smartbear.com/product/?product=TestComplete)

Node.js Documentation (https://nodejs.org/en/docs)

CT-TAE Syllabus by International Software Testing Qualifications Board

Automation Testing with Selenium and WebDriver by T. T. Nguyen and 'End-to-End Testing with Cypress' by J. Boduch