

Programi për Shkenca Kompjuterike dhe Inxhinierise

**RENDËSIA E TESTIMIT NË ZHVILLIMIN E APLIKACIONEVE DHE BENEFITET E TESTIMIT AUTOMATIK**

Shkalla Bachelor

Suhejb Morina

Shtator / 2023

Prishtinë



Programi për Shkenca Kompjuterike dhe Inxhinierise

Punim Diplome

Viti akademik 2017 – 2018

Suhejb Morina

**RENDËSIA E TESTIMIT NË ZHVILLIMIN E APLIKACIONEVE DHE BENEFITET E TESTIMIT AUTOMATIK**

Mentori: MSc Lamir Shkurti

Shtator / 2023

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme për Shkallën Bachelor

**ABSTRAKTI**

Ky punim hulumtues shqyrton rëndësinë e testimit në zhvillimin e aplikacioneve dhe paraqet përfitimet e testimit automatik në krahasim me testimin manual. Qëllimi kryesor është të ndërtohet një argument i fortë për rëndësinë e testimit si një komponent e rëndësishme në procesin e zhvillimit të aplikacioneve dhe të paraqiten përfitimet konkrete që testimi automatik sjell në aspektet e shpejtësisë, cilësisë dhe efikasitetit të procesit të zhvillimit.  
 Në këtë studim, u përdorën një kombinim i burimeve të shkencës kompjuterike, literaturës shkencore, rasteve të suksesit dhe bisedimeve me profesionistë të fushës së zhvillimit të aplikacioneve për të marrë një perspektivë të gjerë mbi temën. Analiza e literaturës dhe burimeve të tjera u bë për të identifikuar dhe vlerësuar konceptet, metodologjitë dhe përparësitë e testimit në zhvillimin e aplikacioneve.  
Rezultatet e këtij studimi tregojnë se testimi është një element kyç për garantimin e sigurisë, stabilitetit dhe cilësisë së aplikacioneve. Testimi automatik ka përfitime të dukshme në krahasim me testimin manual, duke përmirësuar shpejtësinë, numrin e përsëritjeve, efikasitetin dhe kostot e procesit të zhvillimit. Gjithashtu, testimi automatik rrit aftësinë e aplikacioneve për të përballuar nevojat e një numri të madh të përdoruesve dhe sjell përfitime të dukshme në aspektet e zbulimit dhe korrigjimit të gabimeve.  
Përfundimet e këtij studimi përmblidhen në faktin se testimi është një pjesë e zhvillimit të aplikacioneve dhe duhet të konsiderohet si një proces thelbësor në çdo fazë të zhvillimit. Testimi automatik ofron një ndihmë të çmuar për zhvilluesit në garantimin e cilësisë së aplikacioneve, rritjen e shpejtësisë së zhvillimit dhe zvogëlimin e kostove. Me këto përfitime, testimi automatik përbën një investim të vlefshëm dhe sjell vlerë të shtuar në procesin e zhvillimit të aplikacioneve.  
Fjalë kyçe: testimi automatik, zhvillimi i aplikacioneve, cilësia e aplikacioneve, rëndësia e testimit, paraqitja e rasteve të suksesit.

**FALËNDERIME**

Fillimisht falënderoj Zotin për shëndetin, familjen dhe shoqërinë për mbështetjen e vazhdueshme si dhe të gjithë kolegët që kisha gjatë këtyre viteve.

Ne këtë faqe të falënderimeve, dëshiroj të shpreh mirënjohjen time më të thellë dhe respektin për departamentin e Shkencave Kompjuterike dhe Inxhinierisë në Universitetin e Biznesit dhe Teknologjisë (UBT), si dhe mirënjohjen time për mentorin tim MSc Lamir Shkurti.

Kjo është një rast i veçantë për mua për të shprehur përshtypjet dhe ndjenjat e mia në lidhje me rrugën time të studimeve dhe përvojën e shkëlqyer që kemi pasur gjatë kohës si student dhe përfundimit të shkalles bachelor në shkenca kompjuterike.

Fillimisht, dëshiroj të falënderoj të gjithë stafin akademik të Fakultetit të Shkencave Kompjuterike dhe Inxhinierisë për ndihmën, përkrahjen dhe udhëzimet e tyre gjatë këtij udhëtimi të ngritjes akademike. Profesorët e kualifikuar kanë sjellë një ndikim të madh në rritjen time profesionale dhe kanë rritur pasionin tim për teknologjinë dhe inxhinierinë kompjuterike.

Në përgjithësi, do të dëshiroja ta përshëndes edhe atmosferën fantastike të Universitetit të Biznesit dhe Teknologjisë (UBT), e cila ka bërë që kjo kohë e ndërlikuar të kthehet në një periudhë të paharrueshme të jetës sime. Ambienti i këtij institucioni ka përbërë një burim frymëzimi dhe motivimi për të nxjerrë më të mirën prej vetes time.

Nga thellësia e zemrës falënderoj sërish departamentin e Shkencave Kompjuterike dhe Inxhinierisë në UBT për këtë mundësi të çmuar për shpërndarjen e njohurive nga profesionistët e fushave të ndryshme për mua si student, kujtime të paharrueshme që do të mbeten në mendjen time përjetë.

**PËRMBAJTJA**

[1. HYRJE 1](#_heading=h.tyjcwt)

[1.1 Parashikimi i temës dhe qëllimi i studimit 1](#_heading=h.2lwamvv)

[1.2 Problemi 2](#_heading=h.111kx3o)

[1.3 Propozimi 2](#_heading=h.3l18frh)

[1.4 Rëndësia e temës 3](#_heading=h.206ipza)

[1.5 Struktura e temës 4](#_heading=h.4k668n3)

[2. LITERATURAT 6](#_heading=h.2zbgiuw)

[3. KONCEPTET THEMELORE TË TESTIMIT 8](#_heading=h.1egqt2p)

[3.1 Çka është testimi 8](#_heading=h.3ygebqi)

[3.1.1 Objektivat tipike të testimit 9](#_heading=h.2dlolyb)

[3.1.2 Testimi dhe korrigjimi 10](#_heading=h.sqyw64)

[3.2 Roli i testimit dhe pse testimi është i domosdoshëm 10](#_heading=h.3cqmetx)

[3.2.1 Gabime, defekte dhe dështime 11](#_heading=h.1rvwp1q)

[3.3 Shtatë parimet e testimit 12](#_heading=h.4bvk7pj)

[3.4 Aktivitetet dhe etapat e testimit 14](#_heading=h.2r0uhxc)

[3.5 Testimi automatik dhe roli i testimit automatik 18](#_heading=h.1664s55)

[4. RENDESIA E TESTIMIT NE ZHVILLIMIN E APLIKACIONEVE 19](#_heading=h.3q5sasy)

[4.1 Identifikimi i sfidave të mungesës së testimit në zhvillimin e aplikacioneve 19](#_heading=h.25b2l0r)

[4.2 Efektet pozitive të testimit në cilësinë dhe përformancën e produktit 20](#_heading=h.kgcv8k)

[5. PËRFITIMET E TESTIMIT MANUAL DHE AUTOMATIK 21](#_heading=h.34g0dwd)

[5.1 Testimi manual 21](#_heading=h.1jlao46)

[5.2 Përshkrimi i testimit automatik 23](#_heading=h.43ky6rz)

[5.3 Shpjegimi i përfitimeve të testimit automatik 25](#_heading=h.2iq8gzs)

[5.3.1 Shpejtësia 25](#_heading=h.xvir7l)

[5.3.2 Përmirësimi i cilësisë 26](#_heading=h.3hv69ve)

[5.3.3 Rishfrytëzimi i burimeve 27](#_heading=h.1x0gk37)

[6. FRAMEWORKS DHE MJETET E TESTIMIT AUTOMATIK 28](#_heading=h.4h042r0)

[6.1 Përshkrimi i disa frameworks dhe mjete të përdorura për testimin automatik 28](#_heading=h.2w5ecyt)

[6.2 Analiza e karakteristikave dhe avantazheve të secilit framework/mjet 31](#_heading=h.1baon6m)

[6.3 Zgjedhja e një frameworku/mjeti për testimin automatik në platformën e UBT-s 33](#_heading=h.3vac5uf)

[7. TESTIMI NË PLATFORMEN MOODLE TË UBT'S 34](#_heading=h.2afmg28)

[7.1 Përshkrimi i procesit të testimit në platformën Moodle 34](#_heading=h.pkwqa1)

[7.2 Identifikimi dhe përcaktimi i skenarëve të testimit në platformë 35](#_heading=h.39kk8xu)

[7.3 Realizimi i testimeve dhe analiza e rezultateve 35](#_heading=h.1opuj5n)

[8. ISO 27001 NË PROCESIN E TESTIMIT 40](#_heading=h.48pi1tg)

[8.1 Përfshirja e ISO 27001 në procesin e testimit 40](#_heading=h.2nusc19)

[9. DISKUTIM DHE PËRFUNDIM 41](#_heading=h.1302m92)

[9.1 Diskutimet me zhvilluesit dhe testuesit e aplikacioneve 41](#_heading=h.3mzq4wv)

[9.2 Sfidat që mund ta presin në të ardhmen testimin 43](#_heading=h.2250f4o)

[10. REFERENCAT 44](#_heading=h.haapch)

# 

LISTA E FIGURAVE

[Figura 1: Krijimi i fjalës kyqe për hapjen e shfletuesit 34](#_Toc151126456)

[Figura 2: Krijimi i fjalëve kyqe për procesin e navigimit 34](#_Toc151126457)

[Figura 3: Krijimi i fjalëve kyqe për procesin e hyrjes në platformë 35](#_Toc151126458)

[Figura 4: Krijimi i variabileve që nevojiten për navigim apo si lokator 35](#_Toc151126459)

[Figura 5: Shkrimi i skenarit me fjalët kyqe të krijuara deri me tani 36](#_Toc151126460)

[Figura 6: Statusi i testimit dhe raportet e gjeneruara 36](#_Toc151126461)

[Figura 7: Përmbledhja e raportit përfundimtar 37](#_Toc151126462)

[Figura 8: Statistikat për raportin e skenarit përfundimtar 37](#_Toc151126463)

[Figura 9: Përgjigjja për pyetjen e parë 40](#_Toc151126464)

[Figura 10: Përgjigjja për pyetjen e dytë 41](#_Toc151126465)

[Figura 11: Përgjigjja për pyetjen e tretë 41](#_Toc151126466)

[Figura 12: Përgjigjja për pyetjen e katërt 41](#_Toc151126467)

**LISTA E TERMAVE**

API Application Programming Interface

CI/CD Continuous Integration/Continuous Deployment

PIP Python Package Manager

IDE Integrated Development Environment

npm Node Package Manager

ISMS Information Security Management System

AI Artificial Intelligence

AR Augmented Reality

# 1. HYRJE

## 1.1 Parashikimi i temës dhe qëllimi i studimit

Në një epokë kur inovacioni teknologjik ka përfshirë çdo aspekt të jetës moderne, zhvillimi i aplikacioneve ka shërbyer si një katalizator i ndryshimeve thelbësore në mënyrën se si ndihemi, veprojmë dhe bashkëveprojmë me botën rreth nesh. Ky zhvillim madhor sjell me vete një sërë sfidash dhe kërkesash të reja, e njëra prej tyre është siguria dhe cilësia e këtyre aplikacioneve.

Në këtë kontekst, testimi i aplikacioneve bëhet një komponent kritik për garantimin e cilësisë dhe performancen e aplikacioneve. Ky proces përfshin verifikimin e funksionalitetit, stabilitetin, sigurisë dhe performancen e aplikacioneve për tu siguruar që ato plotësojnë pritshmërinë e përdoruesve. Për shkak të përdorimit të madhe të aplikacioneve në jetën tonë të përditshme, domosdoshmëria e një testimi të kujdesshëm dhe efikas është bërë më e rëndësishme se kurrë më parë.

Ky studim synon të hulumtojë thelbin e testimit në zhvillimin e aplikacioneve dhe të paraqesë përfitimet që sjellin testimi automatik dhe testimi manual në këtë kontekst. Do të përqendrohemi në përshkrimin e rëndësisë së testimit si pjesë thelbësore e procesit të zhvillimit të aplikacioneve dhe hulumtime, përfitimet e testimit automatik në krahasim me metodat tradicionale të testimit manual.

Në kuadër të këtij hulumtimi, do të eksplorohen metodat, teknikat dhe mjete të testimit manual dhe testimit automatik të cilat kontribuojnë në përmirësimin e efikasitetit, cilësisë dhe sigurisë së aplikacioneve të zhvilluara.

Me anë të këtij studimi, synohet të që të kemi një pamje më të detajuar për testimin dhe rolin e testimit në zhvillimin e aplikacioneve dhe të nxisë përdorimin e metodave të automatizuara për të përmbushur kërkesat e një bote që përdor metoda të avancuara teknologjike.

Në këtë temë ne kemi përmendur konceptet themelore të testimit, historikun e testimit, rendësin e testimit e shumë komponentë tjerë të testimit, kurse për pjesën e automatizimit ne kemi krahasuar disa nga mjetet(framworks) për testimin automatik dhe në bazë te rezultati përfundimtar ne kemi përzgjedhur një mjetë për testimin e platformës moodle ku në baze të atij testimi automatik ne kemi nxjerrë një rezultat se për 47 sekonda përfundon një skenar i caktuar i përzgjedhur nga ne, dhe në fund ne kemi diskutuar edhe për përfshirjen e standardit ISO 27001 në procesin e testimit.

## 1.2 Problemi

Në zhvillimin e aplikacioneve, testimi ka një rëndësi të veçantë sepse përdoret për të arritur siguri, stabilitet, performancen dhe përvojë të mirë për përdoruesin. Megjithatë, testimi manual në zhvillimit e aplikacioneve mund të jetë i ngadalshëm dhe i kushtueshëm, kurse testimi automatik mund të jetë një alternativë për të përmirësuar efikasitetin dhe cilësinë e testimit. Problemi që adresohet këtu është: a mund të përmirësohet cilësia e testimit në zhvillimin e aplikacioneve duke përdorur testimin automatik. Për të zgjidhur këtë problem, duhet të analizohen dobitë e testimit automatik, si dhe të krahasohen me testimin manual për të vlerësuar avantazhet dhe sfidat e këtyre llojeve të testimit.

## 1.3 Propozimi

Qëllimi i kësaj teme është të shqyrtojë dhe të paraqesë rëndësinë e testimit në procesin e zhvillimit të aplikacioneve dhe të argumentojë për përfitimet e testimit automatik në krahasim me testimin manual. Ky studim synon të ofrojë një kuptim të thellë rreth rolit të testimit në cilësisë e aplikacioneve dhe të demonstruarit e avantazheve të testimit automatik në aspektet e shpejtësisë, stabilitetit, sigurisë dhe efikasitetit të procesit të zhvillimit të aplikacioneve.

Për të arritur qëllimin e kësaj teme, do të përdoren burime dhe metoda të ndryshme hulumtuese:

1. Do të shqyrtohen dhe analizohen studime shkencore, libra dhe artikuj që lidhen me testimin e aplikacioneve dhe testimin automatik. Kjo analizë do të ofrojë një bazë teorike për të kuptuar rëndësinë e testimit dhe përfitimet e testimit automatik.
2. Do të shqyrtohen raste të suksesshme të zhvillimit të aplikacioneve që kanë përdorur teste automatike. Nëpërmjet kësaj analize, do të vlerësohen përfitimet konkrete që këto aplikacione kanë arritur përmes përdorimit të testeve automatike në krahasim me teste manuale.
3. Do të zhvillohen anketime dhe bisedime me profesionistë të fushës së zhvillimit të aplikacioneve, të cilët kanë përvojë në përdorimin e testeve automatike. Këto bisedime do të sigurojnë një perspektivë praktike mbi rëndësinë e testimit dhe përfitimet e testimit automatik në industrinë e zhvillimit të aplikacioneve.
4. Do të kryhet një analizë krahasuese mes testimit manual dhe testimi automatik në aspektet e shpejtësisë, cilësisë, stabilitetit dhe efikasitetit të procesit të zhvillimit të aplikacioneve. Kjo analizë do të ndihmojë në vlerësimin e përfitimeve të testimi automatik në krahasim me testimin manual.
5. Të dhënat e fituara nga analiza e literaturës, studimet e rasteve dhe bisedimet me profesionistë do të analizohen me qëllim që të dalin në pah tendencat dhe përfitimet kryesore të testimit automatik në zhvillimin e aplikacioneve.

## 1.4 Rëndësia e temës

Përshkrimi i nevojës për të kuptuar dhe për të ndërgjegjësuar komunitetin për rëndësinë e testimit dhe rëndësinë e testimit si një pjesë kritike e zhvillimit të aplikacioneve me poshtë janë të listuar edhe disa arsye të rëndësishme:

1. Testimi i aplikacioneve është kritik për sigurinë dhe stabilitetin e tyre. Përdoruesit presin që aplikacionet të jenë të sigurta dhe të punojnë siç duhet. Testimi lejon zbulimin dhe korrigjimin e gabimeve, defekteve dhe dobësitë e aplikacionit përpara se ai të publikohet në treg. Një testim i mirë i ndihmon zhvilluesit të sigurojnë që aplikacioni të jetë i qëndrueshëm dhe të shmangë rrezikun e mos funksionimit te aplikacionit gjatë përdorimit.
2. Testimi i mirë ndikon në cilësinë e produktit të zhvilluar. Një testim i plotë i lejon ekipin e zhvillimit të identifikojë dhe të rregullojë probleme të ndryshme, duke siguruar që aplikacioni të ofrojë një përvojë të shkëlqyeshme për përdoruesit. Përdorimi i testeve automatike shton vlerën e këtij procesi, duke i mundësuar zhvilluesve të identifikojnë probleme dhe të zhvillojnë aplikacione cilësore në një mënyrë më efikase dhe të shpejtë.
3. Testimi automatik mund të përmirësojë kufijtë dhe qëndrueshmërinë e procesit të zhvillimit. Duke përdorur teste automatike, aplikacionet mund të testohen në mënyrë konsistence dhe të përsëritshme në çdo fazë të zhvillimit.

## 1.5 Struktura e temës

HYRJE: Në këtë kapitull ne kemi biseduar për qëllimin, problemin dhe rendësin e temës që ka të bëjë me testimin manual dhe atë automatik.

LITERATURAT: Këtu janë përmendur disa literatura ose citate të cilat i kemi shfletuar dhe kisha dëshirë që ti ndaja me ju.

KONCEPTET THEMELORE TË TESTIMIT: Në këtë kapitull ne kemi shtjelluar tema si: Çka është testimi?, Objektivat tipike të testimit?, Roli i testimit?, Pse testimi është i domosdoshëm?, Parimet e testimit? Për të gjitha këto pyetje dhe pyetje të tjera jam përgjigjur ne këtë kapitull.

RENDESIA E TESTIMIT NE ZHVILLIMIN E APLIKACIONEVE: Këtu ne kemi folur për identifikimin e sfidave dhe mungesa e testimit në zhvillimin e aplikacioneve dhe për efektet pozitive të testimit në cilësinë dhe përformancen e produktit.

PËRFITIMET E TESTIMIT AUTOMATIK: Në këtë kapitull ne kemi tregu për historinë e testimit automatik, për përparësitë e testimit automatik si dhe për ri shfrytëzimi e burimeve dhe përmirësimi e cilësisë.

FRAMEWORKS DHE MJETET E TESTIMIT AUTOMATIK: Këtu ne kemi diskutuar për disa frameworks dhe mjete punë të ndryshme me teknologji të ndryshme për ato më të përdorurat në tregun e pune aktualisht.

TESTIMI NË PLATFORMEN MOODLE TË UBT'S: Këtu ne kemi përzgjedh një mjet dhe framework pune dhe ne kemi tregu një skenar në baze të atij skenari ne kemi bërë testimin e platformës dhe pastaj kemi nxjerr një rezultat prej 47 sekonda të një skenari.

ISO STANDARDET: Në këtë kapitull kemi diskutuar edhe për përfshirje e ISO standardit 27001 në procesin e testimit të aplikacioneve.

DISKUTIM DHE PËRFUNDIM: Këtu ne kemi diskutuar për hapat dhe sfidat që mund ta presin në të ardhmen testimin.

REFERENCAT: Këtu ne kemi përmendur referencat e burimeve të ndonjë teme apo punimi shkencorë po ashtu edhe librave të ndryshëm në lidhje me testimin manual dhe automatik.

# 2. LITERATURAT

Arsyeja pse ne përdorim testim të automatizuar është për shkak se gabimet zbulohen më lehtë, më shpejt, me sigurt, pra edhe kostoja e zhvillimit të aplikacioneve dhe testimit të tyre është më e lirë dhe më e përballueshme për bizneset dhe organizatat tjera.

Në një studim kërkimor ka përcaktuar se roli i testimit automatik kontribuon nga 40% deri në 80% të kostove të zhvillimit të përgjithshëm. [1]

Automatizimin i një softueri mbështete ne një mjet për testim siç është Selenium(ka edhe shume mjete te tjera), këto mjete shërbejnë për testimin e funksionalitetit ne mënyrën e duhur, pra këto mjete janë disa aplikacione programueses te cilat mbështetën në të gjitha shfletuesit e mundshëm ne treg për momentin. [1]

Testimi manual përkufizohet si procedura e testimit te një produkti ne mënyre manuale, për shkak te zbulimit te defekteve, Shumica e inxhinieri ose eksperteve pajtohen se testimi manual është procedura fizike e testimit te softuerit dhe kërkon qe testuesi ta luan rolin e blerësit te produktit si përdorues fundor ku ata përdorin kryesisht veçoritë e aplikacionit nëse ato funksionojnë siç duhet. [1]

Software testing është procesi i cili gjendet gjate ciklit jetësor, ky testim ka te beje me planifikim, përgatitje dhe vlerësim për produktet qofshin ato softuerike ose produkte te punës përkatëse se nëse ato i plotësojnë kërkesat e specifikuara, dhe demontimi i atyre nëse janë te përshtatshme për qëllimin e caktuar. [2]

Qëllimet i testit te penetrimit është te vërtetoj se masat e marruar nga një organizatë ne lidhej me testin e penetrimit a janë efektive, e testi i penetrimit e vërteton ketë duke e simuluar një sulm nga kundërshtaret për te zbuluar dobësitë e asaj organizate ose softuer-it të tyre. [3]

Ka shume procese te ndryshme ne lidhje me testin e penetrimit. Ne varësi te nevojës zgjidhe një proces specifik, Sipas Thorsen, Nufryk, & Taylor një proces i tillë ka 8 faza tradicionale te testimit te penetrimit ato janë: Planifikimi, Zbulimi, Skanimi, Fitimi i qasjes/aksesit, Mirëmbajtja e aksesit, Mbulimi i gjurmëve, Analiza, Raporti. [3]

Kosto mesatare për testimin e penetrimit ne vitin 2020 ka qene diku ne mes: $4000 to $100,000. këto janë disa nga gjerat qe duhet marr parasysh: Madhësia e kompanisë(sa me e madhe kompania aq me shume rritet kosto), Kompleksiteti, Fushëveprimi, Metodologjia, Eksperienca, Testimi i brendshëm dhe i jashtëm, Rehabilitimi(Kjo përfshin ata qe nëse gjate testimit kanë gjete edhe diçka tjetër ekstra pra paguhet ekstra, një gabim qe është gjete jashtë kontratës). [3]

Ka lloje te ndryshme te testimit qe janë shpikur. Çdo teknike testimi shërben për një qellim te testimit te produkteve te ndryshme si: struktura e dizajnit, kodimi dhe planifikimi dhe specifikimeve të kërkesave të softuerit etj. Teknika e testimit ndahen kryesisht ne tri kategori: Testimi i kutisë se zeze(Black Box), testimit i kutisë se bardhe(White Box), dhe testimi i kutisë gri(Grey Box). Teknika qe teston sjelljen e jashtme te sistemit hyjnë ne kutin e zeze te testimit. Teknika qe lejojnë te kemi qasje te brendshme te sistemit bëjnë pjese ne kutin e bardh te testimit Kurse kombinimi i tyre njihet si kutia gri e testimit. [4]

# 3. KONCEPTET THEMELORE TË TESTIMIT

## 3.1 Çka është testimi

Sistemet softuerike janë një pjesë integrale e jetës, bizneset si (p.sh., bankat) deri te produktet për konsumatorin si (p.sh., maketet). Shumica e njerëzve kanë pasur përvojë me softuer që nuk ka funksionuar siç pritej. Softueri që nuk funksionon në mënyrë të saktë mund të çojë në shumë probleme, duke përfshirë humbjen e parave, kohës, reputacionin e biznesit dhe madje edhe dëmtimet ose vdekjen. Testimi i softuerit është një mënyrë për të vlerësuar cilësinë e softuerit dhe për të ulur riskun e dështimit të softuerit gjatë përdorimit.

Procesi i testimit përfshin veprimtari si planifikimi i testeve, analiza, projektimi dhe zbatimi i testeve, raporti i progresit dhe rezultateve të testeve, dhe vlerësimi i cilësisë së një objekti. Një ide e gabuar për testimin është se ai përbëhet vetëm nga ekzekutimi i testeve, pra ekzekutimi i softuerit dhe verifikimi i rezultateve.

Disa teste përfshijnë ekzekutimin e një komponenti ose sistemin që po testohet, testet e tilla quhen teste dinamike. Teste të tjera që nuk përfshijnë ekzekutimin e një komponenti ose sistemin që po testohet njihen si teste statike. Kështu testimi përfshin gjithashtu shqyrtimin e produktit final si kërkesat funksionale, veçorit e një softueri dhe kodi i aplikacionit testohet.

Një ide tjetër e gabuar për testimin është se ai përqendrohet plotësisht në verifikimin e kërkesave, veçorive të softuerit ose specifikimeve të tjera. Përderisa testimi përfshin verifikimin se nëse sistemi plotëson kërkesat e specifikuara, ai gjithashtu përfshin edhe verifikimin, që është verifikimi nëse sistemi ose softueri do të plotësojë nevojat e përdoruesve dhe palëve të tjera në një mjedise pune ose në tregun e punës.

### 3.1.1 Objektivat tipike të testimit

Për çdo projekt të dhënë, objektivat e testimit mund të përfshijnë:

* Parandalimin e defekteve duke vlerësuar produktin në kërkesat funksionale, veçorit e një softueri si dhe në kodin e shkruar.
* Verifikimin nëse janë plotësuar të gjitha kërkesat e specifikuara.
* Kontrollimi nëse objekti i testimit është i plotë dhe nëse verifikimin punon ashtu siç e presin përdoruesit dhe palët e tjera të interesuara.
* Ngritja e besimit në nivelin e cilësisë së objektit të testimit.
* Zbulimi i defekteve dhe dështimeve duke reduktuar nivelin e rrezikut të cilësisë të dobët të softuerit.
* Furnizim i mjaftueshëm me informacion për palët e interesuara për t'u lejuar atyre të marrin vendime të informuara, veçanërisht lidhur me nivelin e cilësisë së objektit të testuar.
* Përputhshmëria në mes kërkesave ose standardet kontraktuale, ligjore dhe krahasimi me përputhshmërinë e objektit të testimit me këto kërkesa ose standarde.

Qëllimet e testimit mund të ndryshojnë, në varësi të kontekstit të komponentit ose sistemit që po testohet dhe nivelit të testimit dhe ciklit të zhvillimit të softuerit. Këto dallime mund të përfshijnë, për shembull:

* Gjatë testimit të komponentit, një qëllim mund të jetë zbulimi i sa më shumë gabimeve të mundshme në mënyrë që defektet bazë të identifikohen dhe riparohen në mënyrë të hershme. Një qëllim tjetër mund të jetë rritja e mbulimit të kodit nga testet e komponentit.
* Gjatë testimit të miratimit, një qëllim mund të jetë konfirmimi që sistemi funksionon ashtu siç pritej dhe plotëson kërkesat. Një qëllim tjetër i këtij testimi mund të jetë të ofrojë informacion palëve të interesuara lidhur me rrezikun e lëshimit të sistemit për një kohë të caktuar.

### 3.1.2 Testimi dhe korrigjimi

Testimi dhe korrigjimi i gabimeve janë të ndryshme. Ekzekutimi i testeve mund të tregojë dështime që shkaktohen nga defektet në softuer. Korrigjimi i gabimeve është aktivitet i zhvillimit që gjen, analizon dhe rregullon këto defekte. Testimi pas rregullimit të mëvonshëm verifikon nëse përpunimet rregulluan defektet. Në disa raste, testuesit janë përgjegjës për testimin fillestar dhe testimin e rregullimit final, ndërsa zhvilluesit rregullojnë gjetjen e gabimeve, testimin e komponentëve dhe testimi i lidhjeve të komponentëve. Megjithatë, në metodologjinë Agile dhe në disa cikle të tjerë të zhvillimit të softuerit, testuesit mund të jenë të përfshirë në gjetjen e gabimeve dhe testimin e komponentëve.

## 3.2 Roli i testimit dhe pse testimi është i domosdoshëm

Testimi kontribuon në suksese e sidomos kur aty janë të përfshirë edhe testuesit, me përfshirjen e testueseve dhe gjetjen e gabimeve në faza të hershme kursen shumë kohë dhe para por më e rëndësishmja sistemi sigurohet që është i aftë të punoj në treg. Duke i pasur testuesit e përfshirë në shqyrtimin e kërkesave ose përmirësimin e veçorive të një softueri mund të zbulohen më lehtë defektet në këto produkte që do të përdoren më vonë për punë.

Identifikimi dhe përmisimi i kërkesave që mendohet që kane defekte zvogëlon rrezikun për të pasur funksionalitetit të gabuar ose defekt të pakapshëm gjatë zhvillimit.

Duke pasur testuesit që punojnë ngushtë me dizajnerët e sistemit përderisa sistemi është duke u projektuar mund të rrisë njohurit e dizajnerëve dhe të testuesve si është mënyra e saktë për ta dizajnuar dhe testuar atë produkt.  
Ky kuptim nga të dy palët mund të zvogëlojë rrezikun e defekteve themelore të dizajnit dhe mundëson që gabimet të identifikohen në një fazë të hershme nga ana e testeve.  
Ky bashkëpunim dhe kuptim vlen edhe për zhvilluesit e sistemi që duhet të punojnë ngushtë ndërmjet vete.

Testimi i ashpër i komponentëve dhe sistemeve, mund të ndihmoj që së bashku me dokumentacioni e tyre, të punohet në zvogëlimin e rrezikut të dështimeve gjatë operacionit në punë. Kur defekte zbulohen dhe më pas riparohen nga zhvilluesit, kjo kontribuon në cilësinë e komponentëve ose sistemeve. Përveç kësaj, testimi i softuerit gjithashtu mund të kërkohet për të përmbushur kërkesat kontraktuale ose ligjore, ose standardet e veçanta të industrisë.

Gjatë historisë së zhvillimit të teknologjisë, është shumë e zakonshme që softueri dhe sistemet t'i dorëzohen testimit, për shkak të pranisë së defekteve, më pas ato shkaktojnë dështime ose ndryshe nuk i plotësojnë nevojat e palëve të interesuara. Megjithatë, përdorimi i teknikave të duhura të testimit mund të zvogëlojë numrin e dorëzimeve të tilla problematike, kur ato teknika zbatohen me nivelin e duhur të ekspertizës së testimit, në nivele të duhura të testimit dhe në pikat e duhura gjatë ciklit së zhvillimit të softuerit.

Përveç këtyre shembujve, arritja e objektivave të përcaktuara të testit kontribuon për suksesin e përgjithshëm të zhvillimit dhe mirëmbajtjes së softuerit.

### 3.2.1 Gabime, defekte dhe dështime

Një person mund të bëjë një gabim, i cili mund të çojë në gjetjen e një defekti (gabim ose mungesë) në kodin e softuerit ose në ndonjë produkt/vegël të punës. Një gabim që çon në gjetjen e një defekti në një produkt të punës mund të shkaktojë një gabim tjetër që çon në gjetjen e një defekti tjetër në një produkt që është i lidhur me produktin e punës. Për shembull, një gabim në identifikimin e kërkesave mund të çojë në një defekt të kërkesave, i cili më pas rezulton në një gabim programimi ose zhvillimi e që çon në një defekt në kod.

Nëse një defekt në kod ekzekutohet, kjo mund të shkaktojë një dështim, por jo domosdoshmërish në të gjitha rastet. Për shembull, disa defekte kërkojnë të dhëna shumë specifike ose parakushte për të shkaktuar një dështim, i cili mund të ndodhë rrallë ose kurrë.

Gabimet mund të ndodhin për shumë arsye, si:

* Presioni i kohës.
* Padituria e njeriut.
* Pjesëmarrësit e projektit të pa përvojë ose pa eksperiencë.
* Komunikimi i gabuar në mesës të pjesëmarrësve në projekt, duke përfshirë komunikimin e gabuar për kërkesat dhe projektimin e projektit.
* Kompleksiteti i kodit, projektimit, arkitekturës, problemi bazë që duhet të zgjidhet dhe/ose teknologjitë e përdorura.
* Teknologjitë e reja, të panjohura për epikën e projektit.

Përveç dështimeve të shkaktuara nga defektet në kod, dështimet mund të shkaktohen gjithashtu nga kushtet mjedisore. Për shembull, valët elektromagnetike, fushat elektromagnetike dhe ndotja mund të shkaktojnë defekte në softuer ose të ndikojnë në ekzekutimin e softuerit duke ndryshuar kushtet e harduerit.

Jo të gjitha rezultatet e papritura të testeve janë dështime. Rezultatet pozitive të rreme mund të ndodhin për shkak të gabimeve në mënyrën se si janë ekzekutuar testet, ose për shkak të defekteve në të dhënat e testit, mjedisin e testimit, ose për shkaqe të tjera. Situata e kundërt mund të ndodhë gjithashtu, ku gabimet ose defektet e ngjashme çojnë në rezultate negative të rreme. Rezultatet negative të rreme janë testet që nuk zbulojnë defekte që duhet të kishin zbuluar, rezultatet pozitive të rreme raportohen si defekte, por në të vërtetë nuk janë defekte.

## 3.3 Shtatë parimet e testimit

Në të kaluarën për 50 vjet janë sugjeruar një numër parimesh të testimit që ofrojnë udhëzime të përgjithshme të përbashkëta për të gjitha llojet e testimit.

1. **Testimi tregon praninë e defekteve, jo mungesën e tyre.**

Testimi mund të tregojë se defekte janë të pranishme, por nuk mund të provojë që nuk ka defekte. Testimi zvogëlon mundësinë që defekte të paidentifikuar të mbeten në softuer por, edhe nëse nuk gjenden defekte, testimi nuk është një dëshmi e saktësisë.

1. **Testimi i plotë/gjithëpërfshirës është i pamundur.**

Testimi i gjithëpërfshirës (të gjitha kombinimet, inputet dhe parakushtet) nuk është i realizueshëm përveç rasteve trivial (të thjeshta). Më mirë se sa të përpiqemi për një testim të plotë, prioritete duhet te jetë analiza e rrezikut, teknikat e testimi për të rregulluar përpjekjet e testimit.

1. **Testimi i hershëm e kursen kohë dhe para.**

Për të zbuluar defektet në mënyrë të hershme, të dy llojet e testimi ajo statike dhe dinamike duhet të fillojnë sa më herët të jetë e mundur në ciklin/metodologjinë e punës së zhvillimit të softuerit. Testimi i hershëm zakonisht quhet "shift left" (zhvendosje majtas). Testimi i hershëm në ciklin/metodologjinë e punës së zhvillimit të softuerit ndihmon në zvogëlimin ose eliminimin e ndryshimeve të kushtueshme.

1. **Defektet grumbullohen së bashku.**

Një numër i vogël modulesh zakonisht përmban shumicën e defekteve të gjetura para se të bëhet testimit final, ose është përgjegjës për shumicën e dështimeve funksional. Në atë grumbull të defekteve duhet të fokusohet analiza e rrezikut.

1. **Kujdes me teste të përsëritura(paradoksi pesticid).**

Nëse të njëjtat teste përsëriten herë pas here, në fund këto teste nuk gjejnë më defekte të reja. Për të zbuluar defekte të reja, mund të duhet të përditësohen testet dhe të dhënat e testimit ekzistuese, ose mund të duhet të shkruhen teste të reja. (Testet nuk janë më efektive në zbulimin e defekteve, pikërisht ashtu si pesticidet nuk janë më efektive në vrasjen e insekteve pas një kohe të caktuar.) Në disa raste, siç është testimi automatik përsëritës, paradoksi pesticid ka një rezultat të favorshëm, që është numri relativisht i ulët i defekteve përsëritëse.

1. **Testimi është i varur nga konteksti.**

Testimi kryhet në mënyra të ndryshme në kontekste të ndryshme. Për shembull, softueri i qendrës së kontrollit në ndonjë industri kritike për siguri testohet ndryshe nga një aplikacion i thjeshtë mobil. Si një shembull tjetër, testimi në një projekt Agile bëhet ndryshe se sa testimi në një projekt pa metodologjinë e punës së një produkti.

1. **Mungesa e gabimeve është një iluzion.**

Disa organizata presin që testuesit të mund të kryejnë të gjitha testet e mundshme dhe të gjejnë të gjitha defektet e mundshme, por parimet 1 dhe 2, respektivisht na tregojnë që kjo është e pamundur. Për më tepër, është një iluzion (pra, një mendim dhe besim i gabuar) të presësh që thjesht gjetja dhe rregullimi i një numri të madh të defekteve do të siguronte suksesin e një sistemi. Për shembull, testimi i plotë i të gjitha kërkesave të specifikuara dhe rregullimi i të gjitha defekteve të gjetura ende mund të prodhojë një sistem që është i vështirë për t'u përdorur, që nuk plotëson nevojat dhe pritshmërinë e përdoruesve, ose që është në një nivel më të ulët krahasuar me sistemet të tjera konkurruese në treg.

## 3.4 Aktivitetet dhe etapat e testimit

Procesi i testimi përbëhet nga këto grupe kryesore si më mëposhtëm:

* Planifikimi i testimit
* Monitorimi dhe kontrollimi i testimit
* Analiza e testimit
* Dizajnimi i testimit
* Implementimi i testimit
* Ekzekutimi i testimit
* Finalizimi i testimit

**Planifikimi i testimit**

Planifikimi i testimit përfshin aktivitete që përcaktojnë objektivat e testimit dhe qasjen për të përmbushur ato objektiva të testimit brenda kufizimeve të imponuara nga konteksti (p.sh., përcaktimi i teknikave dhe detyrave të përshtatshme të testimit dhe përfundimi i testimit në një afat kohor). Planifikimet e testimit mund të rishikohen edhe më vonë.

**Monitorimi dhe kontrolli i testimit**

Monitorimi i testimit përfshin krahasimin e vazhdueshëm të suksesit aktual me suksesin e planifikuar duke përdorur metrika(grafe) për monitorimin e testimit të përcaktuara në planin e testimit.

Suksesi i testimit në përputhje me planin ju komunikohet(tregohet) palëve të interesuara në raportin e përparimit të testimit, duke përfshirë devijimet nga plani dhe informacionin për të mbështetur çdo vendim ose edhe për të ndaluar testimin(si proces i përfunduar ose kompletuar).

**Analiza e testimit**

Gjatë analizës së testimit, faza fillestare e testimit është që të analizohen dhe të identifikohen karakteristikat e testimit dhe për të përcaktuar kushtet e testimit. Me fjalë të tjera, analiza e testimit përcakton "çfarë të testohet" si kriterë për mbulimi të caktuar.

Aplikimi i teknikave të testimit black-box (kuti e zeze), white-box (kuti e bardhë), dhe teknika e bazuar në përvojë mund të jetë e dobishëm në procesin e analizës së testimit, për të reduktuar mundësinë e harrimit të kushteve të rëndësishme të testimit dhe për të përcaktuar kushte më precize dhe më të sakta në testim.

Identifikimi i defekteve gjatë analizës së testimit është një potencial i rëndësishëm, veçanërisht kur nuk përdoret ndonjë proces tjetër i rishikimit ose proces tjetër i testimit. Aktivitetet e analizës së testimit jo vetëm verifikojnë nëse kërkesat janë konsistente por duhet të jenë të shprehura ne mënyrë të duhur dhe të kuptueshme, por edhe kontrollojnë nëse kërkesat plotësojnë në mënyrë të duhur nevojat e klientit, dhe palë tjera të interesuara.

**Dizajnimi i testimit**

Gjatë dizajnimit të testimit, kushtet e testimit elaborohen në raste të komplikuara të testimit, Prandaj, analiza e testimit përgjigjet pyetjes "çfarë të testohet?" ndërsa dizajnimi i testimit përgjigjet pyetjes "si të testohet?"

Dizajnimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Dizajnimi dhe prioritetizim i skenarëve të testimit.
* Identifikimi i të dhënave të nevojshme të testimit për të mbështetur kushtet e testimit dhe skenarëve e testimit.
* Dizajnimi i mjedisit dhe identifikimi i infrastrukturës dhe mjeteve të nevojshme.

**Implementimi i testimit**

Gjatë implementimit të testimit, krijohet dhe/ose përfundohet pajisja e nevojshme për ekzekutimin e testimi, përfshirë vendosjen e skenarëve të testimit në procedura të testimit. Prandaj, dizajni i testimit përgjigjet pyetjes "si të testohet?" ndërsa implementimi i testimit përgjigjet pyetjes "a kemi tani të gjitha gjërat në vend për të ekzekutuar testet?" Implementimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Zhvillimi dhe prioritetizimi i procedurave të testimit dhe, potencialisht, krijimi i skripteve të automatizuara të testimit.
* Vendosja e koleksione të testimit në një orar ekzekutimi për testet në mënyrë që rezulton në një ekzekutim efikas.
* Ndërtimi i mjedisit të testimit (përfshirë potencialisht pajisjet e testimit, virtualizimin e shërbimit, simulatorët dhe të tjera elementë të infrastrukturës) dhe verifikimi që çdo gjë e nevojshme është vendosur në mënyrë të duhur.
* Verifikimi dhe përditësimi i të dhënave dy drejtimore midis bazës së testimit, kushteve të testimit, skenarëve të testimit, procedurave të testimit dhe koleksione të testimit.

**Ekzekutimi i testimit**

Gjatë ekzekutimit të testimit, koleksionet për testimi ekzekutohen në përputhje me orarin e caktuar për ekzekutim të testimit.

Ekzekutimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Krahasimi i rezultateve aktuale me rezultatet e pritura.
* Ekzekutimi i testeve, ose manualish ose testet automatike.
* Regjistrimi i rezultateve të ekzekutimit të testimit (kaloj, dështoj, bllokoj).

**Finalizimi i testimit**

Veprimtaritë e finalizimit të testimit mbledhin të dhëna nga veprimtaritë e përfunduara të testimit dhe çdo informacion të rëndësishëm tjetër.

Përfundimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Kontrollimi nëse të gjitha raportet e defekteve janë mbyllur.
* Krijimi i një raporti të përgjithshëm që do të komunikohet me palët e interesit.
* Përfundimi dhe arkivimi i mjedisit të testimit, të të dhënave të testimit, infrastrukturës së testimit dhe testimet e tjera.
* Dorëzimi i testimeve tek ekipet e mirëmbajtjes, ekipet e projektit, ose palët e interesit dhe të tjerë që mund të përfitojnë nga përdorimi i tyre.
* Analiza e mësimeve të nxjerra nga veprimtaritë e përfunduara të testimit për të përcaktuar ndryshimet e nevojshme për ciklet e ardhshme, publikime dhe projekte.
* Përdorimi i informacionit të mbledhur për të përmirësuar kohen e procesit të testimit.

## 3.5 Testimi automatik dhe roli i testimit automatik

Testimi automatik është një praktikë e rëndësishme në fushën e zhvillimit të softuerit, e cila përdoret për automatizimin e procesit të testimit për aplikacione kompjuterike, celulare etj. Kjo praktikë përdor skripte dhe mjete tjera te automatizuara për të ekzekutuar teste, krahason rezultatet e tyre me pritshmërinë dhe identifikon çdo defekt apo problem në softuer.

Testimi automatik është i rëndësishëm për shkak të shumë arsyeve të ndryshme.

Një nga arsyet kryesore përse testimi automatik është i rëndësishëm është që ai mund të përmirësojë efikasitetin dhe saktësinë e procesit të testimit. Përdorimi i skripteve automatike lejon ekzekutimin e shpejtë dhe të përsëritshëm të testeve, duke kursyer kohën dhe përpjekjet e ndryshme nga persona të cilët do të kryenin testet manualisht. Kjo gjithashtu zvogëlon rrezikun e gabimeve nga faktori njeri në procesin e testimit, pasi skriptet janë të programuara për të ekzekutuar të njëjtin test në mënyrë konsistence çdo herë.

Testimi automatik gjithashtu ndihmon në zbulimin e defekteve në fillimet e zhvillimit e softuerit. Duke ekzekutuar teste automatike në çdo fazë të zhvillimit, problemet mund të identifikohen dhe zgjidhen më herët, duke zvogëluar kostot dhe kohën e riparimit të tyre. Kjo gjithashtu ndihmon në sigurimin e cilësisë së përgjithshme të produktit final.

Një avantazh tjetër i testimit automatik është aftësia për të kryer teste të shumta dhe komplekse që nuk do të ishin të mundshme për t'u kryer ne mënyre manuale. Kjo përfshin teste me të dhëna, teste të përformancës, dhe teste të ndërveprimit të aplikacioneve në mjedis të ndryshëm. Testimi automatik siguron që të gjitha këto aspekte të aplikacionit të testohen në mënyrë efikase dhe të përsëritshme.

Në përgjithësi, testimi automatik është shumë i rëndësishëm për të siguruar cilësinë dhe përformancën e lartë të aplikacioneve te zhvilluara. Ai ndihmon në zbulimin e defekteve, përmirëson efikasitetin e procesit të testimit, dhe u mundëson zhvilluesëve dhe ekipeve të testimit të jenë më produktive në projektet e tyre.

# 

# 4. RENDESIA E TESTIMIT NE ZHVILLIMIN E APLIKACIONEVE

## 4.1 Identifikimi i sfidave të mungesës së testimit në zhvillimin e aplikacioneve

Edhe testimi ka sfida ku do ti shqyrtojmë me hollësi sfidat që shfaqen kur procesi i testimit mungon ose nuk është i përsosur në zhvillimin e aplikacioneve. Është e rëndësishme të kuptojmë që testimi është një komponent kritik i sigurisë dhe cilësisë së produkteve digjitale, dhe mungesa ose përdorimi i pakët i testimit mund të sjellë pasojë të rënda për cilësinë, përformancën dhe besueshmërinë e aplikacioneve, disa nga ato janë:

**Rreziku i zbulimit të defekteve në fazat e vona të zhvillimit**

Një nga sfidat kryesore që vjen me mungesën e testimit të duhur është rreziku i zbulimit të defekteve në fazat e vona të zhvillimit. Kur testimi fillohet vonë si proces, defektet mund të jenë të fshehta dhe të vështira për t'u zbuluar. Kjo shpesh çon në kosto të lartë riparimesh dhe vonesa të mëdha në projekt. Në raste ekstreme, aplikacionet me defekte të paidentifikuara mund të dalin në treg, duke sjellë pasojat më të rënda për reputacionin e biznesit dhe besimin e klientëve**.**

**Mundësia e humbjes së reputacionit të biznesit dhe besimit të klientëve**

Në një mjedis të tregut të egër ose të vështirë, reputacioni është thelbësor për suksesin e një biznesi. Kur aplikacionet dështojnë ose shfaqin defekte të dukshme, klientët humbin besimin dhe ndikimi negativ ndaj reputacionit të biznesit është i paevitueshëm. Nëse një konkurrent ofron një alternativë më të besueshme, klientët mund të largohen dhe të mos kthehen më.

**Mungesa e sigurisë dhe privatësisë së të dhënave**

Testimi është thelbësor për garantimin e sigurisë dhe privatësisë së të dhënave. Aplikacionet që nuk janë të testuar mjaftueshëm për sigurinë mund të jenë të prekura nga shkelësit dhe rreziqet e tjera të sigurisë. Kjo mund të çojë në shfrytëzimin e të dhënave personale dhe në rrezikun e ndërhyrjeve të paligjshme ne te dhënat personale.

**Kostot e ndërhyrjeve pas publikimit të aplikacioneve në treg**

Nëse defektet zbulohen pasi aplikacionet janë publikohen në treg, kostoja e riparimeve është shumë më e lartë se sa kur ato zbulohen në fazat e hershme të zhvillimit. Këtu kostoja përfshijnë kohën dhe burimet e nevojshme për të adresuar defektet, përditësimin e aplikacioneve dhe ndonjë dëm të mundshëm të reputacionit.

Nëpërmjet analizës së këtyre sfidave, bëhet e qartë se testimi është thelbësor për të arritur një cilësi të lartë të aplikacioneve dhe për të minimizuar rreziqet dhe koston e riparimeve të vonshme.

## 4.2 Efektet pozitive të testimit në cilësinë dhe përformancën e produktit

Testimi ka një ndikim të rëndësishëm në përformancën, cilësinë dhe optimizimin e aplikacioneve. Ky ndikim vjen nëpërmjet identifikimit dhe korrigjimit të problemeve potenciale në fazën e zhvillimit të aplikacioneve. Këtu janë disa nga mënyrat se si testimi ndikon në këto aspekte:

1. **Përformance e përmirësuar:** Testimi i përformancës identifikon ngadalësime dhe problemet e përformancës në një aplikacion. Me identifikimin e këtyre problemeve, zhvilluesit kanë mundësi të kryejnë optimizime dhe përmirësime për të siguruar që aplikacioni të punojë më shpejt dhe të jete me efikas.
2. **Cilësi të përmirësuar:** Testimi në cilësi zbulon defekte, publikime të gabuara dhe probleme të tjera që ndikojnë në cilësinë e aplikacionit. Duke korrigjuar këto probleme, aplikacioni bëhet më i qëndrueshëm dhe më i besueshëm për përdoruesit.
3. **Identifikimi i rreziqeve në siguri:** Testimi i sigurisë identifikon rreziqet potenciale të shkeljes së sigurisë në aplikacion. Ky proces ndihmon në përmirësimin e sigurisë së aplikacionit dhe në parandalimin e incidenteve të mundshme të sigurisë ne pjesën e te dhënave ose gabime te shkaktuara nga njeriu.
4. **Optimizimi i resurseve:** Testimi mund të ndihmojë/ndihmon në identifikimin e përdorimit te panevojshëm të burimeve të sistemit si procesori, memoria, dhe komponentët e tjera. Kjo ndihmon në optimizimin e resurseve dhe në uljen e koston e infrastrukturës, ose ne raste te tjera nëse kemi te bëjmë me aplikacione te mëdha dhe komplekse atëherë duhet ti shtojmë resurset për një optimizim sa më të mirë të aplikacionit që ndodhet nën proces te testimit.
5. **Identifikimi i problemeve para publikimit:** Testimi zbulon probleme dhe defekte para se aplikacioni të publikohet në treg. Kjo na lejon korrigjimin e problemeve në fazat e hershme të zhvillimit dhe redukton kostot dhe vonesat e mundshme.

Përmbushja e testeve të cilësisë dhe përformancës është një hap kyç në zhvillimin e çdo aplikacioni. Kjo ndihmon në sigurimin e një produkti të aftë për nevojat e përdoruesve dhe të konkurrojë në një treg gjithnjë me produkte sa më cilësore.

# 5. PËRFITIMET E TESTIMIT MANUAL DHE AUTOMATIK

## 5.1 Testimi manual

Identifikimi dhe përcaktimi i skenarëve të testimit manual në platformën moodle.

Ky është një shembull i një skenari të testimit manual të thjeshtë për platformën Moodle në kontekstin e Universitetit të Biznesit dhe Teknologjisë (UBT).

**Emri i skenarit të testimit:** Verifikimi i procesit të hyrjes dhe kërkimit në Moodle.

**Përshkrimi i skenarit:**

**Hapi 1:** Shko në faqen kryesore të UBT.

**Hapi 2:** Kliko menu “E-Shërbimet”.

**Hapi 3:** Kliko në opsionin “Moodle”.

**Hapi 4:** Zgjidh opsionin për hyrje me llogarinë Google.

**Hapi 5:** Shëno email-in dhe fjalëkalimin për llogarinë tënde.

**Hapi 6:** Kliko butonin “Login” për të hyrë në platformën Moodle.

**Hapi 7:** Kliko tek “Orari i provimeve”.

**Hapi 8:** Navigoni tek menu “Dashboard”

**Hapi 9:** Kliko në opsionin “Algorimet dhe Strukturat e te Dhenave CSE 16/17”.

**Hapi 10:** Verifiko nëse dokumenti “Syllabus” është i qasshëm.

**Hapi 11:** Verifiko nëse tab kontenti përmban tekstin me titullin “Course: Algortimet dhe Strukturat e te Dhënave”.

**Hapi 12:** Kliko në profilin e përdoruesit.

**Hapi 13:** Kliko në butonin "Logout" për të dalë nga llogaria.

**Përshkrimi i pritshëm i rezultateve të testimit:**

* Pas hapit 5, pritet që të hysh me sukses në llogarinë tënde në Moodle.
* Pas hapit 7, pritet që të dhënat mbi algoritmet të jenë të sakta dhe të plotësuar më informacion relevant.
* Pas hapit 9, pritet që të dalësh me sukses nga llogaria në Moodle.

Ky është një skenar i thjeshtë i testimit manual ne platformën moodle që mund të përdoret për të verifikuar funksionalitetin e hyrjes, kërkimit të materialeve specifike dhe dalja nga Moodle. Mund të shtohen hapa të tjerë të testimit në varësi të kompleksitetit dhe nevojave specifike të platformës Moodle në UBT, por ne këtë shembull nuk janë shtuar për arsyeje se i njëjti skenar do te përdoret edhe për testimin automatik.

Ne këtë skenar ne varësi te shpejtësisë se përdoruesit dallon edhe koha me te cilën mund ta përfundosh këtë skenar por kjo procedurë edhe shume e shkurtet dhe nëse përsëritet disa here atëherë do te bëhet edhe e mundimshme, pas disa tentimeve ky skenar mund te themi qe merr kohe me shume se një minutë çdo here. Ky lloj testimi nuk ka një stabilitet si testimi automatik, është shumë i pa qëndrueshëm dhe i paparashikueshëm.

## 5.2 Përshkrimi i testimit automatik

Testimi automatik ka një histori të gjatë, dhe ka filluar të përdoret që në vitet e hershme të kompjuterit. Kjo histori ka ndjekur zhvillimin e teknologjisë dhe nevojat e fushës së shfrytëzimit të mjeteve digjitale. Këtu janë disa momente të rëndësishme në historinë e testimit automatik:

1. **Vitet e 1940 dhe 1950:** Fillimet e testimit automatik lidhen me zhvillimin e kompjuterëve të hershëm. Gjuhët programuese si Fortran dhe Assembly u përdorën për të zhvilluar mjete për testimin automatik të softuerit. Këto ishin periudha kur kompjuterët filluan të përdoren në mënyrë më të gjerë në shkencë dhe industri për aplikacione të ndryshme.
2. **Viti 1979:** Përparimi i testimit automatik filloi të ndikojë më shumë në fushën e inxhinierisë së softuerit. "Mercury Interactive" mundësoj krijimin e një mjedisin të testimit automatik të njohur si "WinRunner."
3. **Vitet e 1990:** Gjuhët programuese si C++ dhe Java u përdorën për të zhvilluar mjete të avancuara të testimit automatik. Këto mjete lejuan organizatat të zhvillojnë dhe të ekzekutojnë teste komplekse automatike.
4. **Vitet e 2000 dhe më pas:** Teknologjitë e reja, si Selenium dhe Appium, janë zhvilluar për të lejuar testimin automatik të aplikacioneve web dhe mobile. Këto teknologji janë zhvilluar duke përdorur gjuhë programuese të ndryshme si Python dhe JavaScript.
5. **Automatizimi i proceseve të DevOps:** Në dekadat e fundit, testimi automatik ka fituar një rol të rëndësishëm në kuadër të praktikave të zhvillimit të shpejtë si DevOps.

Kjo është bërë e mundur përmes skripteve të testeve automatike dhe integrimit të tyre në “CI/CD”.

Testimi automatik është një metodë e cila përdor skripte të programuara ne gjuhe te ndryshme për ekzekutimin e testeve automatike për një aplikacion ose softuer. Këto skripte janë të shkruara paraprakisht dhe janë të cilësuara ose përgatitura për të verifikuar funksionalitetin e aplikacionit nëpërmjet një game të hollësishëm të veprimeve.

Përparësitë e testimit automatik përfshijnë:

1. **Shpejtësia dhe efikasiteti:** Testet automatike ekzekutohen shpejt dhe pa ndërhyrje njerëzore, duke e bërë procesin më efikas.
2. **Riprodhimi i testeve:** Testet automatike mund të riprodhohen me konsistencë dhe mund të ekzekutohen në çdo moment.
3. **Gama e gjerë e testimit:** Testet automatike lejojnë testimin në game të gjerë dhe mund të përfshijnë shumë teste dhe skenarë te ndryshëm.

Gjuhët programuese e kanë bërë të mundur testimin automatik dhe kanë ndikuar në zhvillimin e mjeteve të ndryshme për testimin automatik të softuerit. Përdorimi i gjuhëve të programimit për të zhvilluar skripte të testimit ka lejuar kompani të ndryshme të automatizojnë shumicën e testeve të tyre dhe të sigurojnë cilësi më të lartë dhe efikasitet në zhvillimin e softuerit.

## 5.3 Shpjegimi i përfitimeve të testimit automatik

Testimi automatik është një praktikë e rëndësishme në zhvillimin e aplikacioneve, dhe ka shumë përfitime që i ofron kompanive te cilat e shfrytëzojnë atë. Këtu do të shpjegojmë disa nga këto përfitime kryesore:

### 5.3.1 Shpejtësia

Shpejtësia është një nga përfitimet kyçe të testimit automatik në kontekstin e zhvillimit të aplikacioneve. Ky përfitim ka ndikim të madh në mënyrën se si kompanitë zhvillojnë dhe publikojne produkte të tyre. Disa aspekte të shpejtësisë në testimin automatik janë të rëndësishme:

1. **Ekzekutimi i shpejtë i testeve**: Skriptet e testimi automatik mund të ekzekutohen shpejt dhe pa ndërprerje. Në krahasim me testimin manual, ku çdo skenar testimi kërkon kohë dhe përpjekje të manuale, testimi automatik është jashtëzakonisht efikas dhe i shpejtë. Kjo do të thotë që ndryshimet në kodin e aplikacionit mund të testohen dhe verifikohen shpejt para publikimit.
2. **Testimet e përsëritur:** Testimi automatik lejon testet të ekzekutohen ne mënyre te përsëritur, ndërsa ndryshimet bëhen në kodin e aplikacionit. Kjo është e rëndësishme në mjediset e zhvillimit të shpejtë, të tilla si zhvillimi i softuerit me metodën e zhvillimit Agile. Përmes testimit automatik, ndryshimet mund të testohen në çdo cikël të zhvillimit, duke përmirësuar cilësinë dhe stabilitetin e produktit.
3. **Identifikimi i shpejtë i problemeve:** Testimi automatik lejon identifikimin e shpejtë të problemeve dhe defekteve. Pas ekzekutimit të një testi, raportimi automatik i rezultateve identifikon problemet në mënyrë automatike. Kjo lejon ekipet e testimit të adresojnë problemet menjëherë, duke reduktuar koston dhe vonesën që lidhet me zbulimin e problemeve në fazat më të vona të zhvillimit.

Në përfundim, shpejtësia është një avantazh thelbësor i testimit automatik. Ajo ndihmon organizatat të ofrojnë aplikacione më cilësore në një kohë më të shpejtë, duke i lejuar ato të reagojnë më efikasitet në ndryshimet e kërkesave dhe zbulimin e defekteve.

### 5.3.2 Përmirësimi i cilësisë

Përmirësimi i cilësisë është një ndër përfitimet kryesore të testimit automatik në procesin e zhvillimit të aplikacioneve. Testimi automatik kontribuon në përmirësimin e cilësisë në mënyra të ndryshme:

1. **Konsistencë dhe përsëritshmëri:** Testimi automatik siguron që testet të ekzekutohen në mënyrë konsistencë dhe përsëriten në mënyrë të vazhdueshme. Përdoruesit e nje softueri mund të kenë eksperienca të ndryshme rreth përformancës dhe funksionalitetit të tij. Përmes testimit automatik, mund të sigurohet që aplikacioni është konsistentë dhe funksionon pa ndonjë ndryshim të papritur në cilësi.
2. **Identifikimi i defekteve:** Testimi automatik ndihmon në identifikimin e shpejtë të defekteve dhe problemeve. Përmes skripteve të testimi automatik, aplikacioni testohet me dhëna të ndryshme dhe në këtë mënyrë mund te shpërblehet me defektet që mund të shfaqen në kushte të ndryshme. Ky proces përmirëson cilësinë e aplikacionit duke identifikuar dhe adresuar problemet më herët në zhvillim.
3. **Kohëzgjatja e testeve:** Testimi automatik lejon ekzekutimin e testeve në kohëzgjatje të gjatë dhe të përsëritura pa humbur cilësi. Në krahasim me testimin manual, ku testet e gjata mund të jenë të lodhshme dhe të supozuara për gabime njerëzore, skriptet automatike mund të kryejnë këto teste pa humbur kujdesin dhe cilësinë.

Në përfundim, përmirësimi i cilësisë është një pasuri e rëndësishme që sjell testimi automatik në procesin e zhvillimit të aplikacioneve. Duke përdorur skripte të testimi automatik, kompanitë mund të sigurojnë që aplikacionet e tyre janë konsistence, të përsëritura ose vazhdueshme, dhe cilësore, duke adresuar defektet herët dhe duke përmirësuar përvojën e përdoruesit.

### 5.3.3 Rishfrytëzimi i burimeve

Rishfrytëzimi i burimeve është një avantazh tjetër i testimit automatik që ndikon pozitivisht në zhvillimin e aplikacioneve. Ky koncept lidhet me përdorimin e burimeve të disponueshme, përfshirë kohën, shkathtësitë e ekipit, dhe infrastrukturën teknike, në një mënyrë të zgjuar dhe efikase. Këtu janë disa mënyra se si testimi automatik rishfrytëzon burimet:

1. **Rishfrytëzimi i kohës:** Testimi automatik ndihmon në rritjen e efikasitetit të kohës së përdorur për testim. Pasi skriptet e testimit automatik mund të ekzekutohen në mënyrë automatike dhe të përsëriten në kohë shumë më të shpejtë, zhvilluesit dhe ekipet e testimi kanë më shumë kohë për të fokusuar në përmirësimin e cilësisë së produktit. Kjo rrit produktivitetin dhe lejon që burimet e kohës të përdoren më efikasitet.
2. **Rishfrytëzimi i shkathtësive:** Me testimin automatik, shkathtësitë e ekipit të testimi janë të ri shfrytëzueshme në mënyrë të përsëritur. Pas zhvillimit të skripteve të testimi automatik, këto skripte mund të përdoren për shumë teste të ndryshme dhe për shumë versione të produktit. Kjo lejon që eksperienca dhe shkathtësitë e fituara nga ekipi të rishfrytëzohen në projekte të tjera dhe në fazat e tjera të zhvillimit.
3. **Rishfrytëzimi i infrastrukturës:** Për testimin automatik, është e nevojshme të krijohet një infrastrukturë e testimit që përfshin pajisje dhe mjete të automatizuara. Kjo infrastrukturë mund të përdoret në mënyrë të përsëritur për teste të ndryshme dhe aplikacione të ndryshme. Nëse një herë infrastruktura është e krijuar mire, mund të rishfrytëzohet për shumë projekte të tjera pa shpenzime të mëdha shtesë.

Si përfundim, rishfrytëzimi i burimeve është një avantazh kyç i testimit automatik. Duke i përdorur me efikasitet burimet e disponueshme, kompanitë mund të zhvillojnë produkte cilësore, të përmirësojnë produktivitetin dhe të jenë të gatshme për sfidat e të ardhmes.

# 6. FRAMEWORKS DHE MJETET E TESTIMIT AUTOMATIK

## 6.1 Përshkrimi i disa frameworks dhe mjete të përdorura për testimin automatik

Me poshtë janë te listuara disa gjuhe programuese te cilat ofroj struktura te llojllojshme për testimin automatik disa prej tyre do ti përmendim me poshtë duke filluar nga me te njohurat dhe te përdorurat aktualisht ne treg.

Python dhe Robot Framework janë një kombinim i fuqishëm për testimin automatik. Python është një gjuhë programimi e njohur për sintaksën e thjeshtë dhe fuqinë e saj të gjerë, ndërsa Robot Framework është një framework i lehtë dhe fleksibil për menaxhimin e testeve automatike. Kombinimi i këtyre dy mjeteve ofron disa përfitime të rëndësishme në testimin automatik.

Robot Framework është një framework i fuqishëm për menaxhimin e testeve automatike. Disa nga përfitimet kryesore të përdorimit të Robot Framework janë:

1. **Lehtësi gjatë përdorimit:** Robot Framework është i njohur për sintaksën e thjeshtë që e bëjnë të lehtë shkrimin dhe leximin e testeve.
2. **Ekzekutimi i testeve në gamë të gjerë:** Mundëson ekzekutimin e testeve në mënyrë paralele në gamë të madhe, duke përmirësuar efikasitetin.
3. **Integrimi i përmirësuar:** Robot Framework mund të integrohet me shumë mjete të tjerë dhe platforma të zhvillimit, përfshirë Jenkins, Docker dhe shërbime tjerat të testimit.

Ky kombinim i Python dhe Robot Framework është i fuqishëm dhe fleksibil dhe ofron një mjedis të përshtatshëm për zhvilluesit dhe inxhinierët e testimit për të krijuar dhe ekzekutuar teste automatike në mënyrë efikase.

Tjetër kombinim fantastik është edhe Selenium dhe Java.

Selenium dhe Java janë një tandem i fuqishëm për testimin automatik të aplikacioneve web. Selenium siguron mjetet dhe bibliotekat e nevojshme për të automatizuar veprimet në faqet e internetit, ndërsa Java është një gjuhë programimi e fuqishme dhe e njohur që mund të përdoret për të zhvilluar skripte të testeve automatike në Selenium. Kombinimi i këtyre dy mjeteve sjell disa përfitime të rëndësishme në testimin automatik.

Përdorimi i Selenium dhe Java në testimin automatik:

1. **Stabiliteti dhe efikasiteti:** Përdorimi i Java për të zhvilluar teste automatike në Selenium sjell një nivel të lartë të stabilitetit dhe efikasitetit. Java është një gjuhë e kompiluar/përkthyes dhe ofron përformancë të shkëlqyer në ekzekutimin e testeve.
2. **Bibliotekat dhe komuniteti:** Java ka një gamë të pasur të librave dhe bibliotekave që mund të përdoren në testimin automatik. Përveç Selenium, mund të përdorni biblioteka të tjera të njohura si TestNG dhe JUnit për të menaxhuar dhe ekzekutuar testet në Java.
3. **Përputhshmëria:** Java është një gjuhë e gjerë që përdoret gjithandej në zhvillimin e aplikacioneve, duke lehtësuar integrimin e testeve automatike në projektin tuaj të zhvillimit.

Selenium dhe Java janë një kombinim i fuqishëm për testimin automatik të aplikacioneve web. Ky tandem ofron aftësinë për të automatizuar veprimet në faqet e internetit në mënyrë efikase dhe të sigurojë cilësinë dhe përformancën e aplikacioneve tuaja.

Tjetër kombinim fantastik është edhe Cypress dhe Javascript.

Cypress dhe Javascript janë një kombinim i fuqishëm për testimin automatik të aplikacioneve web. Cypress është një framework modern i testimin automatik për aplikacionet e internetit, ndërsa Javascript është gjuhe programuese e përdorur për të zhvilluar skriptet e testeve automatike në Cypress. Ky kombinim sjell disa përfitime të rëndësishme në testimin automatik të faqeve të internetit.

Përdorimi i Cypress dhe JavaScript në Testimin Automatik:

1. **Ekzekutimi i testeve brenda ne shfletues:** Cypress integron një shfletues të brendshëm për ekzekutimin e testeve, duke lejuar teste të shpejta dhe të efikase. Javascript përdoret për të zhvilluar skriptet e testeve në këtë mjedis pune.
2. **Lehtësi gjatë përdorimit:** Cypress është i njohur për sintaksën e thjeshtë dhe të kuptueshme që e bën të lehtë shkrimin dhe leximin e testeve. Javascript është gjithashtu një gjuhë me sintaksë të qartë dhe të lexueshme.
3. **Integrimi i leht:** Cypress lejon integrimin me shumë mjete dhe platforma të tjera të zhvillimit, përfshirë Jenkins, Git dhe shërbimet e tjera të testimit.
4. **Gjurmimi i defekteve:** Cypress vjen me mjetet e fuqishme të gjurmimit te defekteve, dhe Javascript është një gjuhë e njohur për mjetet e saj të zgjidhjes së problemeve.

Ky tandem i Cypress dhe Javascript ofron një mënyrë efikase dhe të përshtatshme për të automatizuar testimin e aplikacioneve web. Kjo e bën atë një zgjedhje të preferuar për zhvilluesit dhe inxhinierët e testimit që punojnë në faqet e internetit dhe dëshirojnë të sigurojnë cilësi dhe përformancë të lartë për produktet e tyre.

## 6.2 Analiza e karakteristikave dhe avantazheve të secilit framework/mjet

Karakteristikat dhe Avantazhet e Robot Framework.

Karakteristikat:

1. **Sintaksa e thjeshtë:** Robot Framework ka një sintaksë të thjeshtë dhe lexueshme që e bën të lehtë shkrimin e testeve.
2. **Lehtësim gjatë përdorimit:** Është i njohur për lehtësinë e përdorimit dhe mundëson krijimit të testeve të shpejta dhe të thjeshta.
3. **Mundësi për integrim të gjerë:** Ka aftësi të mira për integrim me mjete të tjera të testimit dhe burime të hapura për komunitetin.

Avantazhet:

1. **Ndryshime të shpejta:** Lejon ndryshimin e testeve pa shumë vështirësi.
2. **Ekzekutim i testeve në masa të mëdha:** Mundëson ekzekutimin e testeve në mënyrë paralele, përmirëson efikasitetin dhe shpejtësinë e testimit.
3. **Komuniteti i madh:** Ka një komunitet të gjerë të zhvilluesve dhe mbështetësve që kontribuojnë në zhvillimin dhe përmirësimin e framework.

Karakteristikat dhe Avantazhet e Seleniumit.

Karakteristikat:

1. **Shumëllojshmëria e shfletuesve:** Mund të përdoret në mënyrë efektive për të testuar në shumë shfletues të ndryshëm, përfshirë Chrome, Firefox, dhe Safari.
2. **Përshtatshmëria e gjuhëve të programimit:** Mund të përdoret me gjuhë programimi të njohura si Java, Python, C#, etj.
3. **Librari të pasura:** Ka një gamë të pasur të librarive dhe metodave që mund të përdoren për të kontrolluar veprimet në web faqet.

Avantazhet:

1. **Kompatibiliteti i shfletuesve:** Mund të sigurojë që aplikacionet janë kompatibile me shumë shfletues të ndryshëm.
2. **Proceset:** Mund të përdoret me integrimin e proceseve të zhvillimit, siç është CI/CD.
3. **Sintaksa e njohur:** Përdor sintaksën dhe metoda të njohura të programimit në gjuhët si Java dhe Python.

Karakteristikat dhe Avantazhet e Cypress.

Karakteristikat:

1. **Shpejtësia:** Cypress është një framework i shpejtë që ekzekuton teste në kohë reale.
2. **Shprehje të drejtpërdrejta:** Lejon përdorimin e shprehjeve të drejtpërdrejta për të identifikuar elementet në faqen web.
3. **Nëndërtimi i testeve në përgjithësi i lehtë:** Cypress është i dizajnuar që të jetë i lehtë për përdorim, me një sintaksë të thjeshtë dhe të kuptueshme.

Avantazhet:

1. **Testim i shpejtë dhe i efektshëm:** Cypress ofron ekzekutim të shpejtë të testeve dhe ndihmon në zbulimin e problemeve në kohë reale.
2. **Integrim me devTools:** Mund të integrohet me DevTools për të analizuar dhe menaxhuar veprimet në web faqe.
3. **Përdorim i këndshëm për zhvilluesit:** Përmban mjetet e nevojshme për zhvilluesit për të shkruar teste cilësore.

## 6.3 Zgjedhja e një frameworku/mjeti për testimin automatik në platformën e UBT-s

Një zgjedhje e shkëlqyer për testimin e platformës UBT dhe sistemit Moodle eshte robot framework i cili siguron mjete dhe burime të nevojshme për të zhvilluar dhe ekzekutuar teste të suksesshme automatike dhe për të siguruar cilësi dhe përformancë të lartë gjatë testimit të platformës të UBT-s, disa arsyeje për zgjedhjen e ketij mjeti jane:.

1. **Popullariteti i madh i komunitetit të robot framework:** Robot Framework është një nga framework-ët më të njohur dhe të përdorur për testimin automatik. Ky fakt është shumë i rëndësishëm për një ambient si UBT, pasi ka një komunitet të madh të zhvilluesve dhe inxhinierëve të testimi që përdorin dhe kontribuojnë në këtë framework. Ky komunitet ofron një burim të pasur të dokumentacionit, librave, dhe përgjigjeve të pyetjeve që mund të përdoren për të zhvilluar teste të sofistikuara dhe për të zgjidhur problemet e mundshme gjatë testimit.
2. **Përqindja e madhe e testuesve që përdorin robot framework:** Robot Framework është një zgjedhje popullore në botën e testimit automatik. Shumë kompani dhe organizata e përdorin për testimin e aplikacioneve të tyre. Kjo do të thotë që ka një bazë të madhe të inxhinierëve të testimit dhe profesionistëve të testimit që kanë njohuri ne robot framework.
3. **Përditësime të reja dhe të shpeshta nga ekipi i zhvilluesve të robot framework:** Robot Framework është një projekt i hapur me një ekip të përkushtuar të zhvilluesve. Përditësimet dhe përmirësimet janë të rregullta dhe janë në përputhje me zhvillimet më të fundit në botën e testimit automatik. Kjo do të thotë se robot framework është i gatshëm për të përballuar sfidat dhe ndryshimet e reja në industrinë e testimit.

# 7. TESTIMI NË PLATFORMEN MOODLE TË UBT'S

## 7.1 Përshkrimi i procesit të testimit në platformën Moodle

Përshkrimi i procesit të testimit në platformën Moodle është një hap i rëndësishëm që duhet të përfshijë disa elemente kyç. Këtu janë disa prej tyre:

1. **Ambienti i zhvillimit:** Për të filluar me testimin në platformën Moodle, është e rëndësishme të keni një ambient të zhvillimit të para përgatitur. Kjo përfshin një sistem operativ të përshtatshëm, dhe një IDE (Integrated Development Environment) për shkrimin e skripteve të testeve. IDE të njohura si Visual Studio Code, PyCharm, ose Eclipse janë zgjedhje të mira, ne kemi përzgjedhur Visual Studio Code.
2. **Gjuhët programuese:** Për të shkruar teste automatike në moodle me mjetin që ne kemi përzgjedhur, ne do të përdorim python për të automatizuar veprimet në faqe.
3. **Mjetet e testimit:** Pasi ne kemi zgjedhur një mjet për testimin automatik që do të përdorim për të shkruar dhe ekzekutuar testet. Duke pasur parasysh që kemi zgjedhur Robot Framework, do të duhet ta instaloni robot framework dhe ndonjë librari të nevojshme për të automatizuar veprimet në moodle. Në këtë rast, mund të jetë e nevojshme të instalojmë edhe Selenium Library për Robot Framework.
4. **Node.js dhe npm (Node Package Manager):** Pasi kemi të ngjarë të përdorim ndonjë mjet ose skripte shtesë për testimin automatik në Robot Framework, siç është "robotframework-seleniumlibrary," këto mund të bazohen në Node.js. Prandaj, është e rekomandueshme të keni Node.js dhe npm të instaluar në kompjuterin tuaj.
5. **Pip (Python Package Manager):** Nëse përdorni Python për të shkruar testet, ju mund të keni nevojë të instaloni librari të ndryshme ne Python. Pip është menaxheri i paketave i Python-it.

Pas instalimit të këtyre komponentëve dhe mjeteve të nevojshme, ju do të keni një ambient të gatshëm për të filluar me procesin e shkrimit, ekzekutimit dhe monitorimit të testeve në platformën Moodle. Kjo do të përfshijë krijimin e skripteve të testeve, konfigurimin e mjetit të testimi (Robot Framework në këtë rast), dhe ekzekutimin e testeve automatike për të vlerësuar përformancën, cilësinë dhe funksionalitetin e platformës Moodle.

## 7.2 Identifikimi dhe përcaktimi i skenarëve të testimit në platformë

Ky është një shembull i një skenari të testimit të thjeshtë për platformën Moodle në kontekstin e Universitetit të Biznesit dhe Teknologjisë (UBT).

**Emri i skenarit të testimit:** Verifikimi i procesit të hyrjes dhe kërkimit në Moodle.

**Përshkrimi i skenarit:**

**Hapi 1:** Shko në faqen kryesore të UBT.

**Hapi 2:** Kliko menu “E-Shërbimet”.

**Hapi 3:** Kliko në opsionin “Moodle”.

**Hapi 4:** Zgjidh opsionin për hyrje me llogarinë Google.

**Hapi 5:** Shëno email-in dhe fjalëkalimin për llogarinë tënde.

**Hapi 6:** Kliko butonin “Login” për të hyrë në platformën Moodle.

**Hapi 7:** Kliko tek “Orari i provimeve”.

**Hapi 8:** Navigoni tek menu “Dashboard”

**Hapi 9:** Kliko në opsionin “Algorimet dhe Strukturat e te Dhenave CSE 16/17”.

**Hapi 10:** Verifiko nëse dokumenti “Syllabus” është i qasshëm.

**Hapi 11:** Verifiko nëse tab kontenti përmban tekstin me titullin “Course: Algortimet dhe Strukturat e te Dhënave”.

**Hapi 12:** Kliko në profilin e përdoruesit.

**Hapi 13:** Kliko në butonin "Logout" për të dalë nga llogaria.

**Përshkrimi i pritshëm i rezultateve të testimit:**

* Pas hapit 5, pritet që të hysh me sukses në llogarinë tënde në Moodle.
* Pas hapit 7, pritet që të dhënat mbi algoritmet të jenë të sakta dhe të plotësuar më informacion relevant.
* Pas hapit 9, pritet që të dalësh me sukses nga llogaria në Moodle.

Ky është një skenar i thjeshtë i testimit që mund të përdoret për të verifikuar funksionalitetin bazë të hyrjes, kërkimit të materialeve specifike dhe dalja nga Moodle. Mund të shtohen hapa të tjerë të testimit në varësi të kompleksitetit dhe nevojave specifike të platformës Moodle në UBT.

## 7.3 Realizimi i testimeve dhe analiza e rezultateve

Me poshtë do të paraqes disa fotografi te kodit të realizuar për skenarin e shkruar më lart.

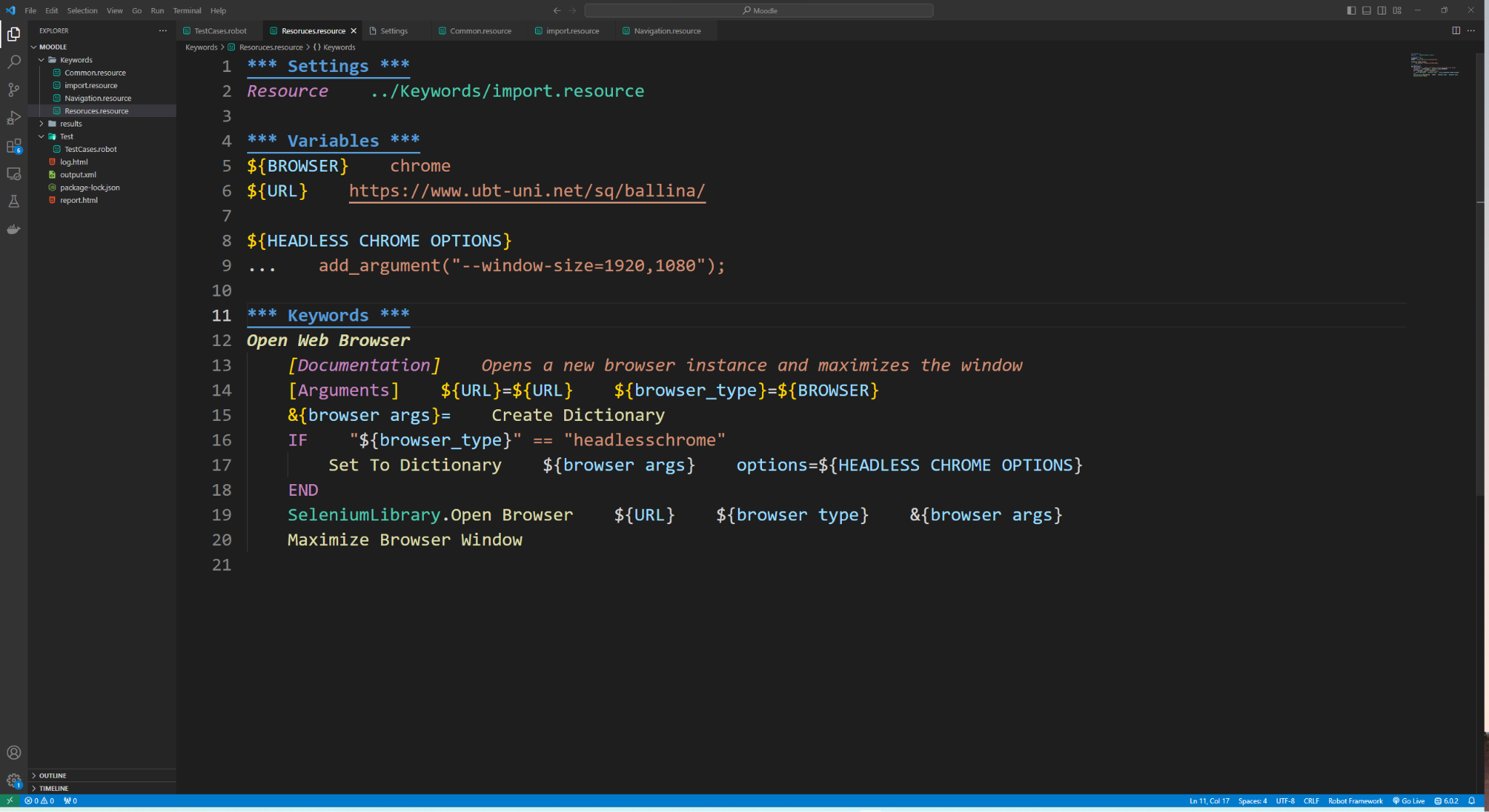


Figura 1: Krijimi i fjalës kyqe për hapjen e shfletuesit

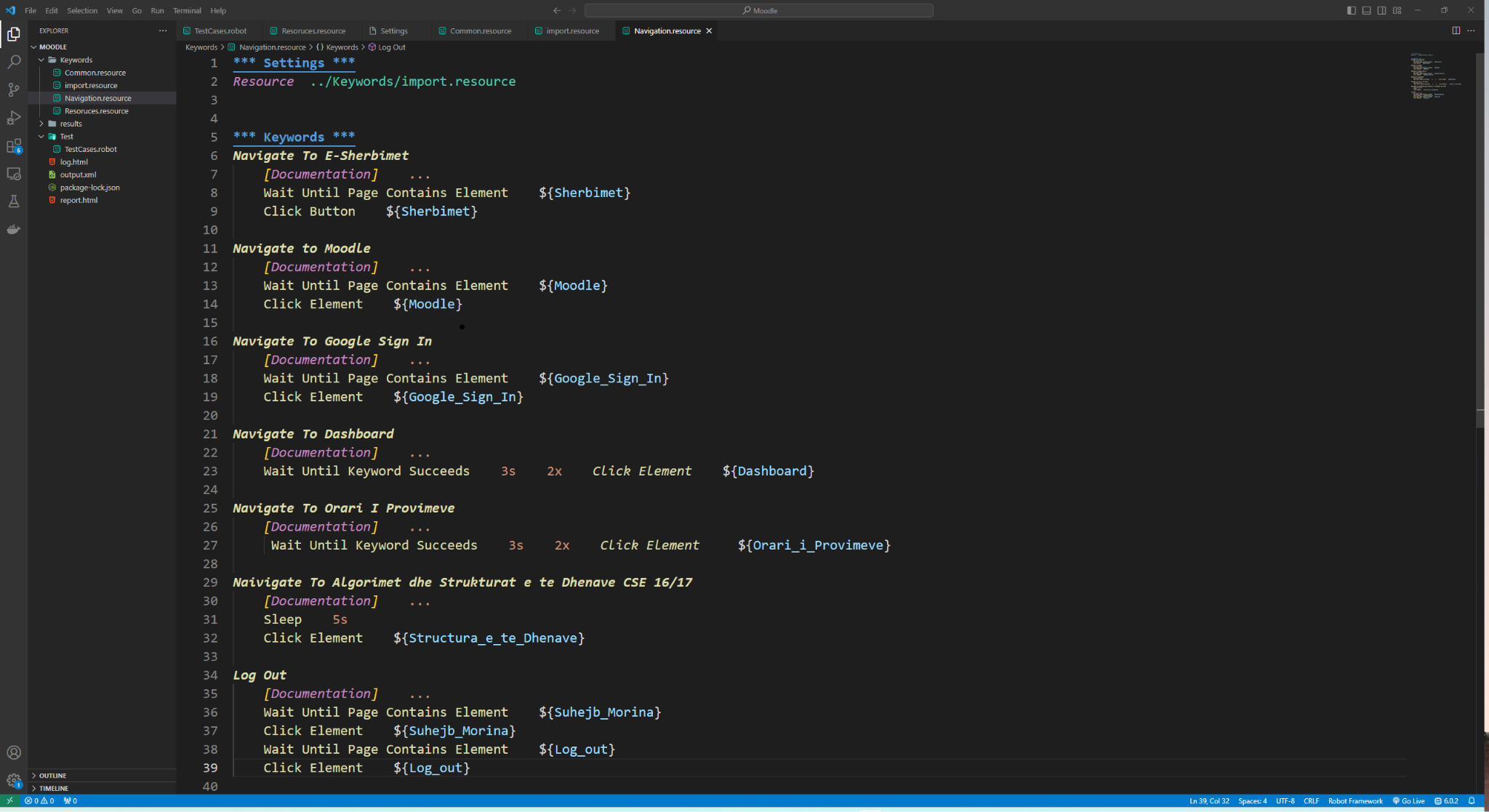


Figura 2: Krijimi i fjalëve kyqe për procesin e navigimit

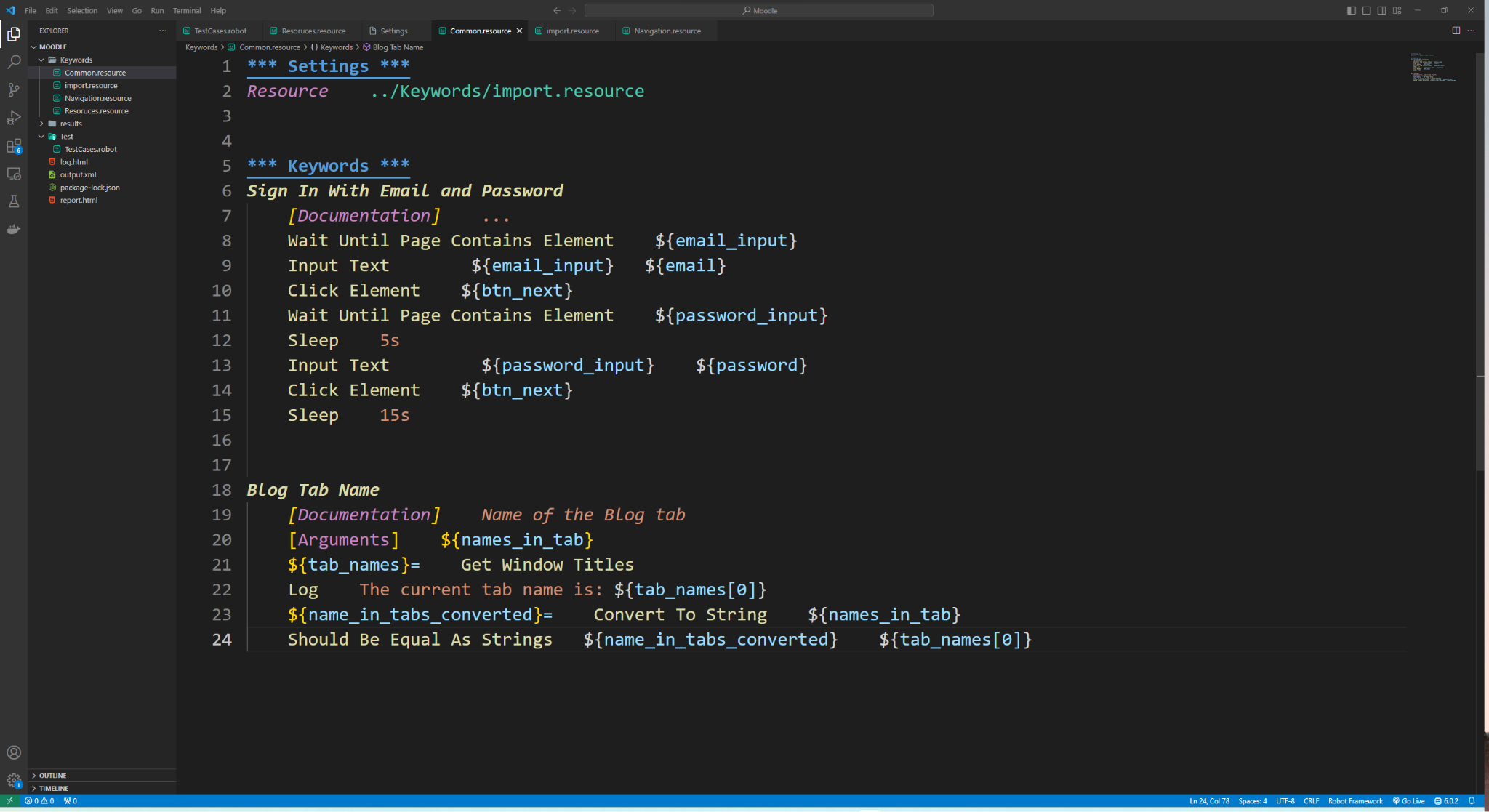


Figura 3: Krijimi i fjalëve kyqe për procesin e hyrjes në platformë

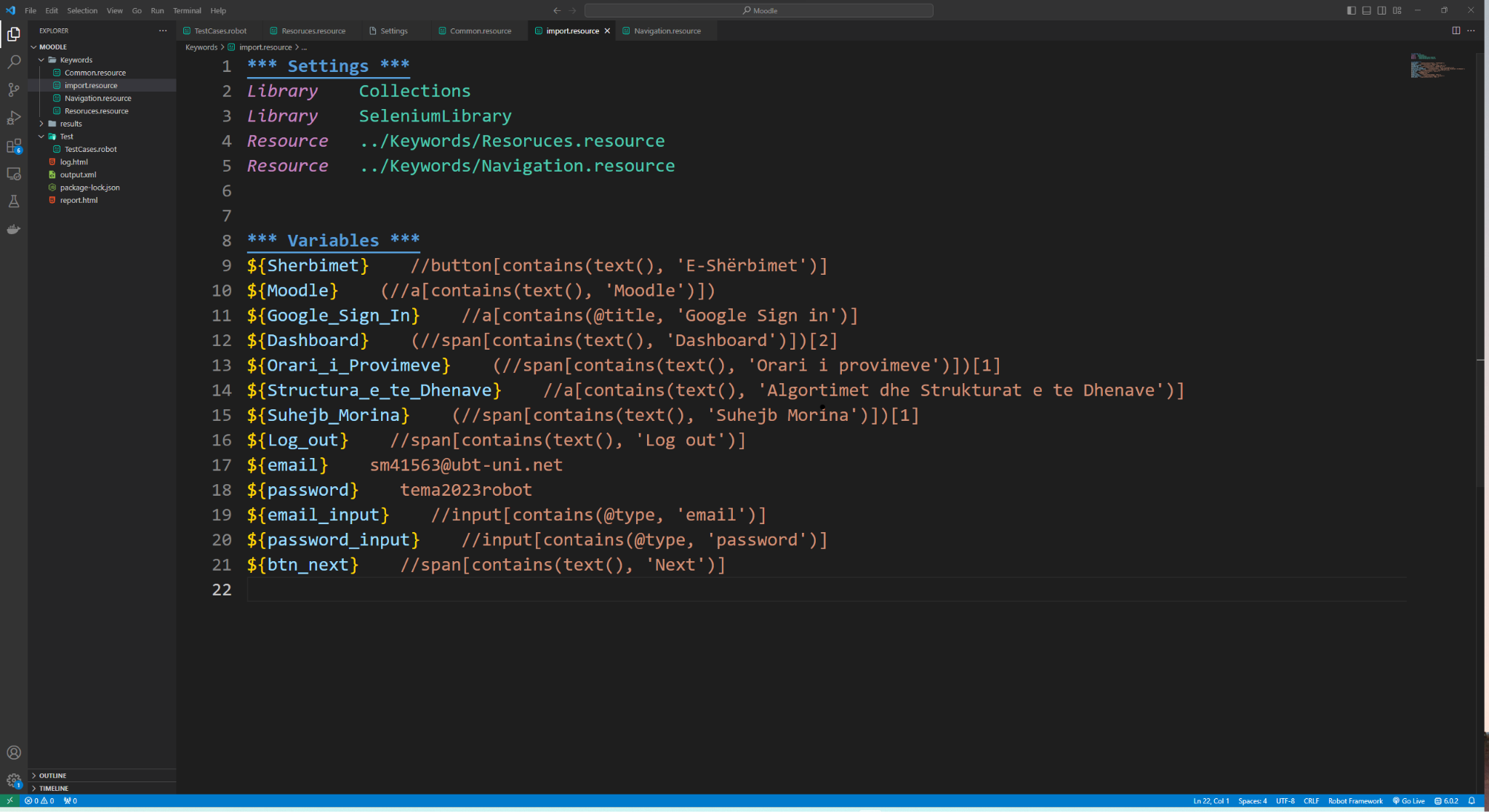


Figura 4: Krijimi i variabileve që nevojiten për navigim apo si lokator

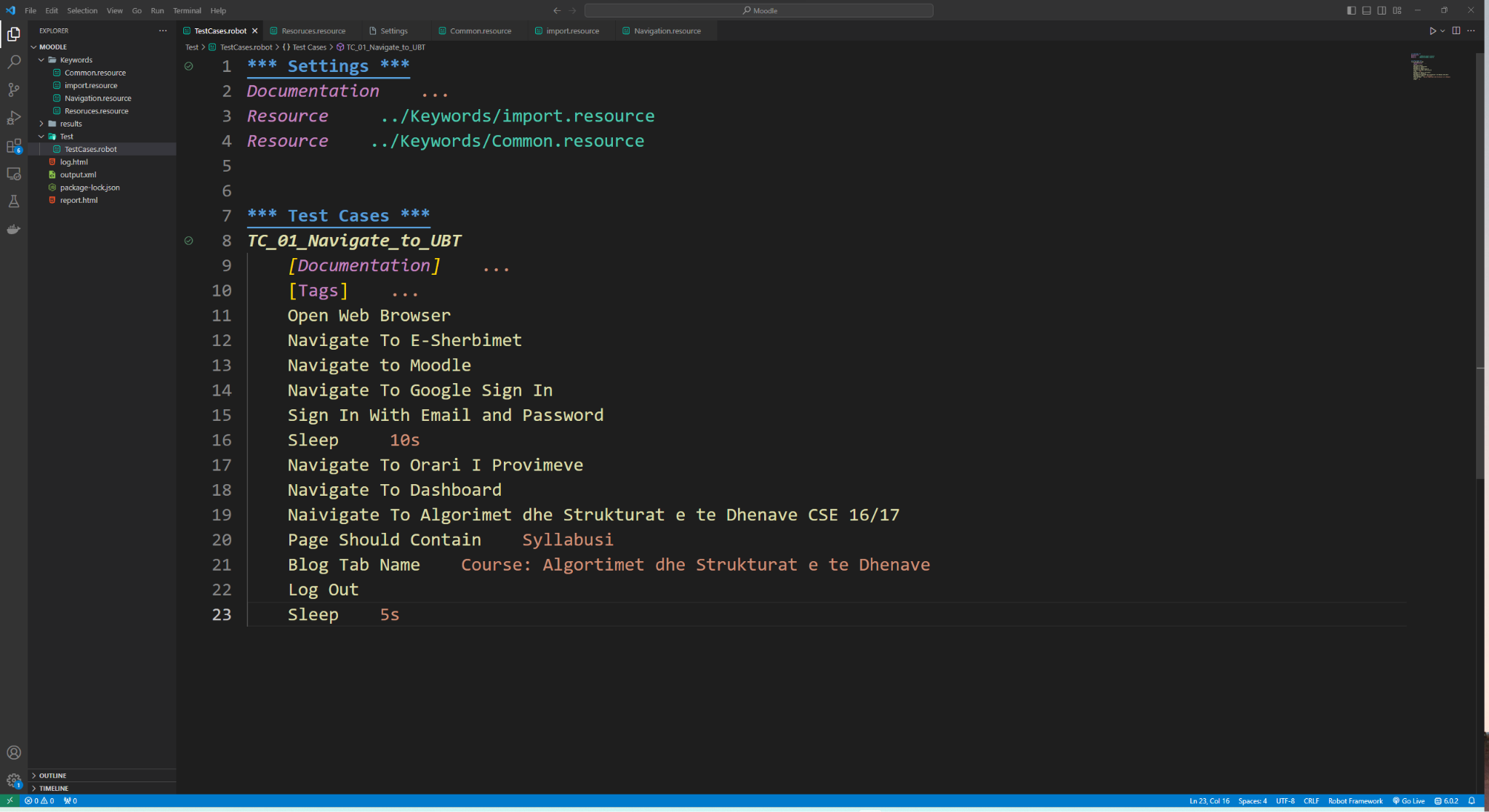


Figura 5: Shkrimi i skenarit me fjalët kyqe të krijuara deri me tani

Pasi që testimi përfundon gjëndërohet një raporte dhe një regjistër në folderin e projektit, aty mundeni me pa raportin dhe log-at(regjistrat) dhe disa fotografi të cilat behën gjatë ekzekutimit të kodit, raportin dhe regjistrat mundeni mi hap dhe me i pa në njërin prej shfletuesve të cilin ju dëshironi mi pa ose edhe ne IDE tuaj, dhe duken si me poshtë.

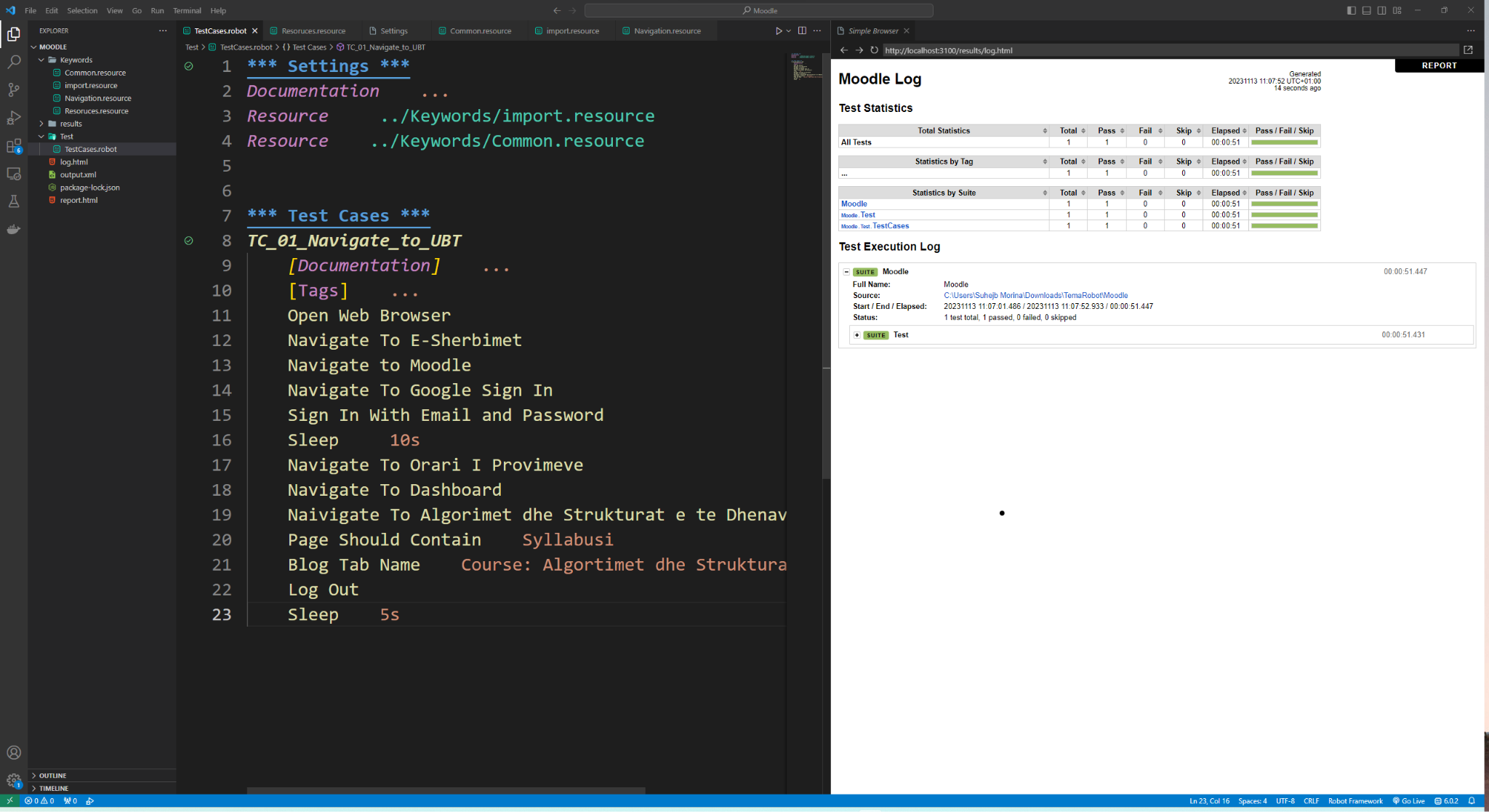


Figura 6: Statusi i testimit dhe raportet e gjeneruara

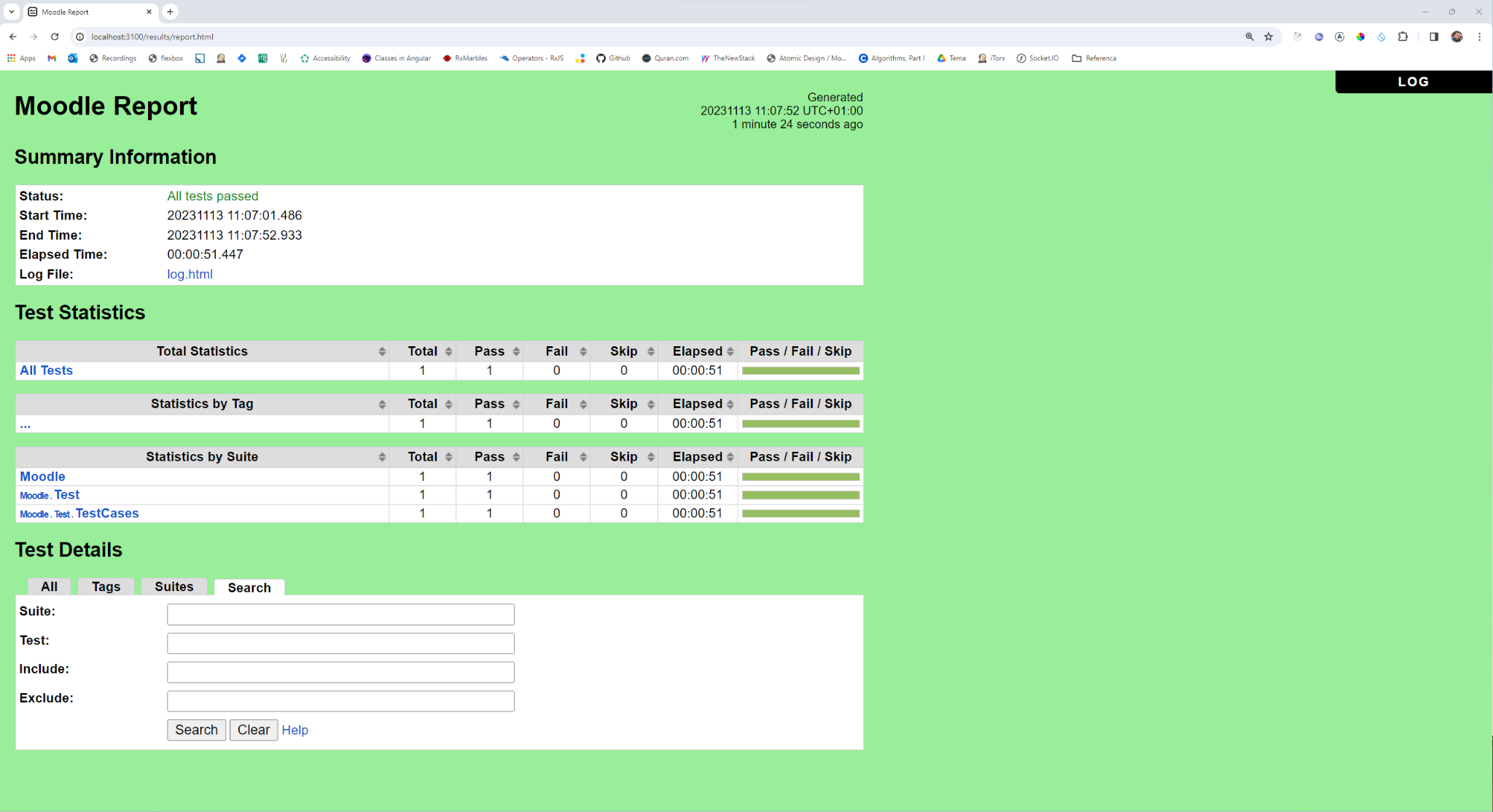


Figura 7: Përmbledhja e raportit përfundimtar

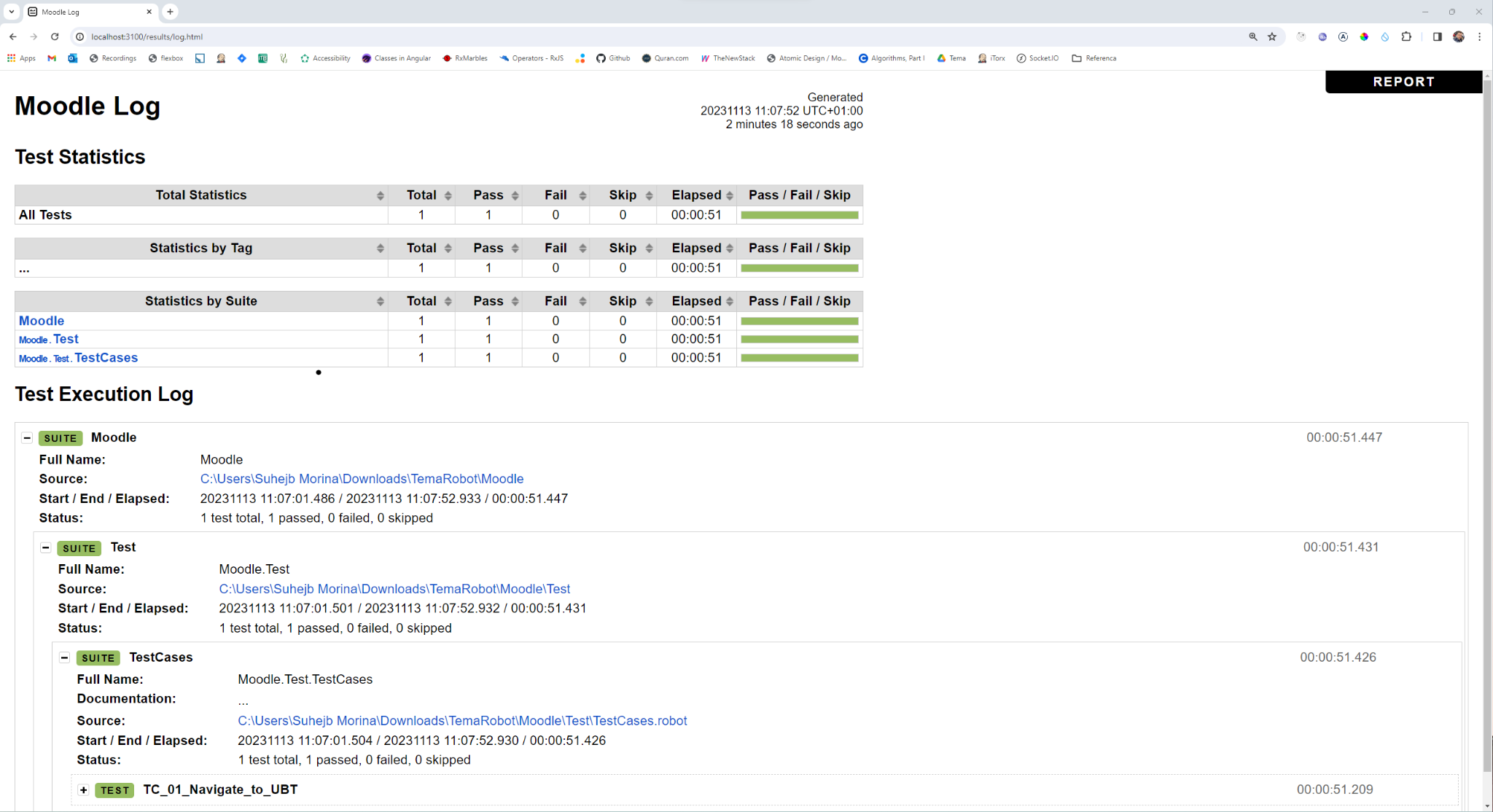


Figura 8: Statistikat për raportin e skenarit përfundimtar

Ne raportin e gjeneruar e shohim se testimi përfundoi me ndihmën e testeve automatike të gjitha funksionet kanë kaluar dhe i gjithë procesi ka përfunduar për kohen 47 sekonda.

Nëse dëshironi që të hyni ne detaje e shihni edhe kohen për secilin funksion dhe ku ka ngecur me se shumti koha gjate ekzekutimit të komplet procedurës.

# 8. ISO 27001 NË PROCESIN E TESTIMIT

## 8.1 Përfshirja e ISO 27001 në procesin e testimit

Përfshirja e standardit ISO 27001 në procesin e testimit është një nga hapat e rëndësishëm për sigurinë e informacionit dhe për mbrojtjen e të dhënave në aplikacione dhe sisteme të ndryshme. ISO 27001 është një standard ndërkombëtar i sigurisë së informacionit që përcakton kërkesat për një sistemi të menaxhimit të sigurisë së informacionit (ISMS).

Në këtë kapitull do të flasim për testimin e penetrimit(Penetration testing).

Testimi i penetrimit, i njohur gjithashtu si pirateria etike ose testimi i sigurisë, është një komponent themelor i procesit të menaxhimit të sigurisë të standardit ISO 27001. Testimi i penetrimin është një aktivitet kritik brenda kësaj strukture për të identifikuar pika e dobëta dhe për të vlerësuar efikasitetin e kontrolleve të sigurisë. Këtu janë disa pika kyçe në lidhje me ISO 27001 dhe testimin e penetrimin:

1. **Kërkesat e ISO 27001:** ISO 27001 përfshin kërkesa specifike për testin e penetrimit në paragrafët A.12.6.1 dhe A.12.6.2. Këto paragrafë theksojnë rëndësinë e testimit periodik të penetrimit për të siguruar që kontrollet e sigurisë që janë në vend janë efektive dhe për të identifikuar pika të dobëta dhe dobësi.
2. **Testues të kualifikuar:** ISO 27001 detyron që testet e penetrimit të kryhen nga testues të kualifikuar dhe kompetent. Testuesit duhet të kenë shkathtësitë dhe njohuritë e nevojshme për të kryer testet në mënyrë efektive.
3. **Përcaktimi i vëmendjes:** ISO 27001 kërkon që organizatat të përcaktojnë qëllimin dhe objektivat e testeve të penetrimit. Kjo përfshin specifikimin e sistemeve, rrjeteve, ose aplikacioneve që do të testohen.
4. **Vlerësimi dhe përmirësimi:** Pas testimit të penetrimit, ISO 27001 inkurajon organizatat të vlerësojnë dhe përmirësojnë sigurinë e tyre në përputhje me rezultatet e testimit. Kjo përfshin zbatimin e veprimeve korrektuese për të adresuar çdo dobësi të identifikuar.

Testimi i penetrimit është një mjet i rëndësishëm për sigurinë e informacionit dhe mbrojtjen aktive të një organizate dhe përfshihet në procesin e menaxhimit të sigurisë sipas ISO 27001.

# 9. DISKUTIM DHE PËRFUNDIM

## 9.1 Diskutimet me zhvilluesit dhe testuesit e aplikacioneve

Pasi qe kemi biseduar me disa zhvillues te cilët kodin i tyre i nënshtrohet një testimi manual dhe automatik, dhe me disa testues te cilët i testojnë dhe krijojnë ato test ne i kemi parashtruar disa pyetje te njëjta atyre me poshtë janë te listuara këto pyetjet.

1. Sa është i rendësishem për ju testimi manual?
2. Sa është i rendësishem për ju testimi automatik?
3. A është e mundur që këto testime të funksionojnë njëra pa tjetren?
4. Çfarë janë pritshmërit e juaj nëse nuk kishte pasur fare testim?

Një përmbledhje se si janë përgjigjur zhvilluesit është:

1. Ne na ndihmon ne gjetjen e gabimeve te cilat na kanë ikur nga syri ose e kemi keqkuptuar funksionalitetin e kësaj risie.
2. Testimi automatik për ne është shume me i rëndësishëm sepse janë skripte te krijuara dhe ne baze te atyre skripteve testohet kodi jone, ku mund ta shohim përmes mjeteve te ndryshime dhe nuk na nevojitet fare diskutimi me testuesit e aplikacionit.
3. Po është e mundur qe te funksionoj, por me njohurit tona nuk është preferuese.
4. Ne besojmë qe do te kishte gabime shume te mëdha te funksionalitetit, përdoruesi i fundit nuk do te ishte i pakënaqur dhe do te kishim probleme te llojllojshme jo të mirëpritura.

Një përmbledhje se si janë përgjigjur testuesit është:

1. Testimi manual është shume i rëndësishëm për arsyeje se na ndihmon ne gjetjen e gabimeve, këtu ne e luajmë rolin e përdoruesit te fundit për këtë arsyeje është shume i rëndësishëm ky lloj i testimi për neve si testues.
2. Edhe testimi automatik është shume i rëndësishëm, kjo na ndihmon qe te kemi shume teste te përsëritura ku e shohim edhe se si aplikacioni do te funksiononte me një mbi ngarkese me te madhe te përdoruesve me kërkesa te veçanta dhe te shumta.
3. Teknikisht funksionon, por ne nuk e preferojmë.
4. Gabime shume te mëdha dhe serioze, mund te këtë aplikacione qe nëse nuk testohen mund te jene edhe vdekjeprurëse për përdoruesit, kështu qe me mire te mos mendohet se çfarë mund te kishte ndodhur po te mos kishte testim.

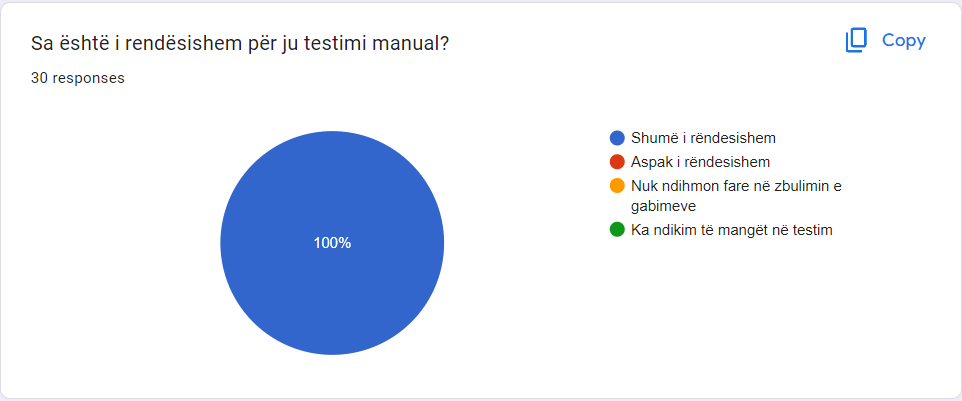
Me poshtë e kemi zhvilluar edhe një anketim me tridhjetë persona me këto pyetje, kemi pyetur edhe person te cilat nuk janë profesionit te zhvillimit ose testimit dhe ata janë përgjigjur kështu:  
  


Figura 9: Përgjigjja për pyetjen e parë

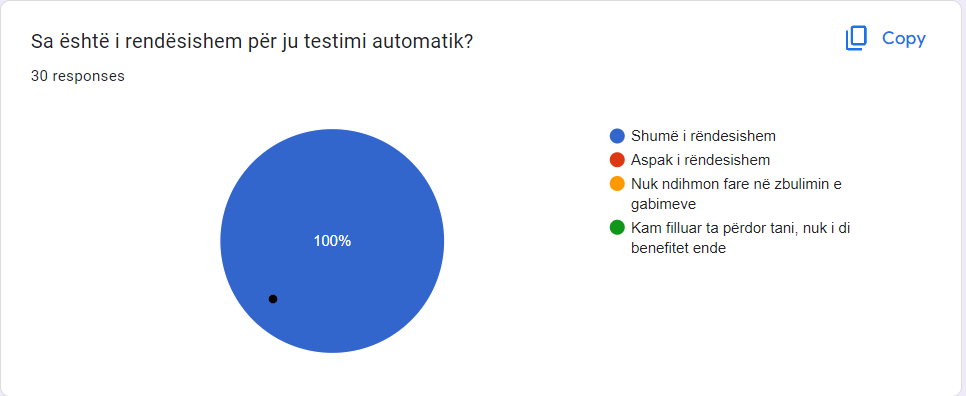


Figura 10: Përgjigjja për pyetjen e dytë

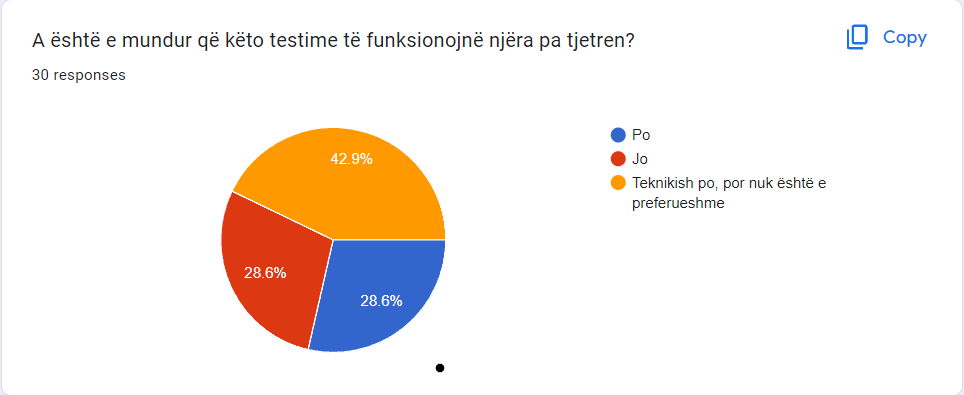


Figura 11: Përgjigjja për pyetjen e tretë



Figura 12: Përgjigjja për pyetjen e katërt

## 9.2 Sfidat që mund ta presin në të ardhmen testimin

Sfidat që mund të na presin në të ardhmen për testimin automatik dhe atë manual në një mjedis ku bota e inteligjencës artificiale (AI), dhe AR dhe web 3.0 janë në zhvillim janë të mëdha dhe të komplekse. Disa prej këtyre sfidave përfshijnë:

1. **Zhvillimi i aplikacioneve të inteligjencës artificiale:** AI është në rritje, dhe testimi i aplikacioneve që përdorin mënyra të reja të inteligjencës artificiale është sfidues. Testimi automatik i aplikacioneve AI është ende në zhvillim dhe ka nevojë për zhvillimin e metodave dhe mjeteve të reja për të kapërcyer këtë sfidë.
2. **Testimi i sigurisë në ambientin Web 3.0:** Në një mjedis të webit 3.0, të dhënat janë të decentralizuara dhe të shpërndara në mënyrë të ndryshme. Testimi i sigurisë duhet të konsiderojë këtë sfidë dhe të sigurojë që të dhënat jenë të mbrojtura nëpërmjet protokolleve dhe teknologjive të reja të sigurisë.
3. **Testimi i ekuacioneve, algoritmeve**: AI përdor ekuacione komplekse dhe algoritme të thella të mësimi automatik. Testimi manual nuk është i efektshëm për të zbuluar probleme të tilla.
4. **Ndryshimet e shpejta të teknologjisë:** Teknologjia zhvillohet me shpejtësi, dhe testuesit duhet të jenë në gjendje të përballen me ndryshimet e shpejta dhe të adaptohen për t'u siguruar që testimi është aktual dhe efektiv.

Përgjithësisht, me zhvillimin e teknologjisë dhe përparimin e aplikacioneve të reja, testimi është një sfidë e vazhdueshme. Përdorimi i metodave të automatizuara, përditësimi i shpeshtë i njohurive dhe shkollimi i vazhdueshëm janë thelbësore për të përballuar këto sfida në të ardhmen.

# 10. REFERENCAT

|  |  |
| --- | --- |
| [1] | A. Contan, C. Dehelean dhe L. Miclea, Test automation pyramid from theory to practice., IEEE, 2018. |
| [2] | M. Daigl dhe R. R. , Keyword-Driven Testing., dpunkt.verlag GmbH, 2022. |
| [3] | ISTQB, ISTQB Study Materials, ISTQB, 2018. |
| [4] | I. 27001, Standard. ISO/IEC 27001:2022 – Information security, cybersecurity and privacy protection – Information security management systems - Requirements., ISO/IEC. I, 2022. |
| [5] | S. Bisht, Robot Framework Test Automation, Packt Publishing, 2013. |
| [6] | G. Blokdyk, Manual Testing A Complete Guide, 5STARCooks, 2021 . |
| [7] | S. R. Mamilla, A Study of P A Study of Penetration T ation Testing Pr esting Processes and T ocesses and Tools, California State University, San Bernardino, 2021. |
| [8] | M. Tuteja dhe G. Dubey, A Research Study on importance of Testing and Quality Assurance in Software DevelopmentLife Cycle SDLC Models, International Journal of Soft Computing and Engineering (IJSCE), 2012. |
| [9] | M. G. V, Software Testing: An Analysis of the Impacts of Test, University of Huddersfield Repository, 2020. |
| [10] | C. O’Broin, The Impact of Test Automation on Software, University of Dublin, 2015. |