**AZƏRBAYCAN RESPUBLİKASI TƏHSİL NAZİRLİYİ**

**AZƏRBAYCAN DÖVLƏT İQTİSAD UNİVERSİTETİ**

**UNEC BİZNES MƏKTƏBİ**

**UNEC Biznes Məktəbinin direktoru**

**i.e.n., dos. Hacıyev Nazim Özbəy oğlu**

**“\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”** **imza**

**“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025-cü il**

**“PROQRESSİV VEB TƏTBİQLƏRİNİN UX İSTİQAMƏTLİ TƏHLİLİ”**

**mövzusunda**

**MAGİSTR DİSSERTASİYA İŞİ**

**İxtisasın şifri və adı: 060409 Biznesin idarə edilməsi**

**İxtisaslaşma:** **Kibertəhlükəsizlik**

**Qrup: A19-23**

|  |  |
| --- | --- |
| **Magistrant:**  **Mustafayev Seymur Sabir**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_imza**      **Proqram rəhbəri:**  **i.ü.f.d. Qəzənfərli Mirvari**    **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_imza** | **Elmi rəhbər:**  **Phd. Məmmədov Nail Elzar**    **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_imza**      **Kafedra müdiri:**  **dos. Məmmədova Sevər Mömin**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_imza** |

**BAKI – 2025**

**“Proqressiv veb tətbiqlərinin UX istiqamətli təhlili”**

**Xülasə**

**Dissertasiya işinin aktuallığı:** Rəqəmsal texnologiyaların sürətli inkişafı fonunda, istifadəçi təcrübəsinin (UX) keyfiyyətli şəkildə təmin olunması veb tətbiqlərin uğurlu istifadəsində həlledici rol oynamağa başlamışdır. Bu kontekstdə, həm vebin universal əlçatanlığını, həm də nativ tətbiqlərin performans və funksionallığını özündə birləşdirən Proqressiv Veb Tətbiqlər (Progressive Web Applications – PWA) müasir proqram təminatının yeni mərhələsini formalaşdırır.

**Dissertasiya işinin məqsədi:** Dissertasiya işinin əsas məqsədi, Progressive Web Application (PWA) texnologiyasının istifadəçi təcrübəsi baxımından effektivliyini elmi əsaslarla qiymətləndirmək, bu məqsədlə prototip əsaslı tətbiq hazırlamaq və CogTool aləti vasitəsilə istifadəçi davranışlarını modelləşdirərək həm analitik nəticələr əldə etmək, həm də istifadəçi yönümlü dizayn yanaşmalarının optimallaşdırılmasına elmi dəstək verməkdir.

**Dissertasiya işində istifadə edilmiş biznes tədqiqat metodları:** Dissertasiya işində tədqiqatın məqsəd və vəzifələrinə uyğun olaraq bir neçə fərqli biznes tədqiqat yanaşması tətbiq olunmuş, empirik müşahidə üsulu ilə istifadəçi təcrübəsi öyrənilmiş, CogTool aləti vasitəsilə simulyasiya aparılmış və PWA modelinin digər tətbiqlərlə müqayisəli təhlili həyata keçirilmişdir. Toplanan məlumatlar kontent analizi üsulu ilə qruplaşdırılmış, kodlaşdırılmış və nəticələrin interpretasiyası təmin edilmişdir.

**Dissertasiya işinin informasiya bazası:** Məlumat bazası kimi türkcə və xarici dillərdə çap edilmiş kitab, elektron məqalələr, müxtəlif internet səhifələrində paylaşılan müəyyən məlumatlardan istifadə edilmişdir.

**Dissertasiya işinin məhdudiyyətləri:** Dissertasiya işində istifadəçi ssenarilərinin məhdud çərçivədə qurulması, test iştirakçılarının azlığı, texnoloji bilik fərqləri və CogTool alətinin bəzi psixoloji faktorları əhatə edə bilməməsi, habelə xarici mənbələrin çoxluğu və yerli təcrübənin nəzərə alınmaması tədqiqatın tətbiq imkanlarını məhdudlaşdırmışdır.

**Dissertasiya işinin praktik nəticələri:** Dissertasiya çərçivəsində hazırlanmış PWA əsaslı funksional prototip üzərində aparılan təhlillər göstərir ki, bu texnologiya istifadəçi təcrübəsini optimallaşdırmaq, geniş cihaz dəstəyi təmin etmək və real tətbiqlərdə praktiki imkanlar yaratmaq baxımından ənənəvi modellərdən üstün olub, yerli və regional layihələr üçün də aktual tətbiq potensialına malikdir.

**Nəticələrin istafadə oluna biləcəyi təşkilat, müəssisə və biznes sahələr:**Dissertasiya nəticələri, PWA texnologiyasının tətbiqi ilə təhsil, dövlət idarəçiliyi, bankçılıq, rəqəmsal xidmətlər, e-ticarət və texnoloji startaplar kimi sahələrdə intuitiv və funksional veb tətbiqlərin hazırlanmasına, istifadəçi təcrübəsinin və müştəri məmnuniyyətinin artırılmasına, eləcə də texniki xərclərin azaldılmasına töhfə verə bilər.

***Açar sözlər:*** *Progressive Web Application (PWA), İstifadəçi təcrübəsi (UX), CogTool, Kognitiv modelləşdirmə, İnterfeys dizaynı.*

**TƏŞƏKKÜRNAMƏ**

İlk öncə diplom işinin hazırlanmasında böyük dəstək olan elmi rəhbərim, Nail Məmmmədova təşəkkürlərimi bildirmək istəyirəm. Onun köməyi və göstərişləri olmadan, bu diplom işinin başa çatması çətin olardı.

Diplom işini hazırladığım müddətdə, maraqlandığım sualları cavablandıraraq, əlavə məlumatlar verən Mirvari xanım Qəzənfərliyə də minnətdarlığımı bildirirəm.

Sonda isə, universitet həyatımızda daim bizə dəstək olan və köməyini əsirgəməyən dekanımız Nazim Hacıyevə öz təşəkkürlərimi bildirirəm.

**MÜNDƏRİCAT**

GİRİŞ 5

I FƏSIL . PROGRESSİVE WEB APPLİCATİON TEXNOLOGİYASININ NƏZƏRİ ƏSASLARI VƏ İSTİFADƏÇİ TƏCRÜBƏSİ ANLAYIŞI 9

1.1 Progressive Web Application tətbiqi üçün istifadəçi təcrübəsinin kognitiv modellərlə təhlili 9

1.2 Proqressiv veb tətbiqin komponentləri 13

II FƏSIL. PROGRESSİVE WEB APPLİCATİON ƏSASINDA PROTOTİPİN HAZIRLANMASI VƏ TEXNOLOJİ ƏSASLANDIRILMASI 31

2.1 Tətbiqin arxitekturası və texnoloji komponentləri 31

2.2 Progressive Web Application prototipində interfeys dizaynı və funksional elementlərin inteqrasiyası 36

III FƏSIL. PROQRESSİV VEB TƏTBİQİN UX TƏHLİLİ 45

3.1 CogTool alətinin hazırlanması və dizaynın tətbiqə əlavə olunması 45

3.2 CogTool ilə toplanmış məlumatların əsasında UX təhlilin aparılması 53

3.3 Real istifadəçilərlə testlər və nəticələrin müqayisəli analizi 63

Nəticə 69

İxtisarların və Şərti işarələrin Siyahısı 71

Ədəbiyyat siyahısı 72

İstifadə Olunmuş Şəkillərin Siyahısı 76

İstifadə Olunmuş Cədvəllərin Siyahısı 77

# GİRİŞ

**Mövzunun aktuallığı**. Müasir dövrdə rəqəmsal texnologiyaların sürətlə inkişaf etməsi, veb tətbiqlərin gündəlik həyatın ayrılmaz hissəsinə çevrilməsinə səbəb olmuşdur. Ənənəvi veb saytlarla yanaşı, istifadəçilərdən yüksək performans, mobil uyğunluq, interaktivlik və çevik cavab reaksiyası tələb edən tətbiqlərin yaradılması tələbi də artmışdır. Bu tələbləri qarşılamaq üçün Progressive Web Application (PWA) texnologiyası son illərdə diqqət mərkəzində olan həlledici texnoloji yanaşmalardan birinə çevrilmişdir.

PWA texnologiyası həm veb saytların geniş əlçatanlığını, həm də mobil tətbiqlərin funksionallığını birləşdirərək istifadəçiyə yüksək keyfiyyətli təcrübə təqdim etməyə imkan verir. Bu texnologiyanın tətbiqi ilə bağlı dünya miqyasında geniş təcrübə formalaşsa da, Azərbaycanda bu sahədə sistemli elmi tədqiqatların azlığı, xüsusilə də PWA-nın istifadəçi təcrübəsi (UX) baxımından qiymətləndirilməsi istiqamətində ciddi boşluq müşahidə olunur. Halbuki istifadəçi təcrübəsinin uğurlu təmin edilməsi, hər bir rəqəmsal məhsulun effektivliyi, istifadəçi məmnunluğu və rəqabət qabiliyyəti baxımından strateji əhəmiyyət daşıyır.

**Mövzunun işlənmə dərəcəsi.** Progressive Web Application (PWA) texnologiyası son illərdə dünya miqyasında sürətlə yayılan və veb inkişaf sahəsində yeni imkanlar açan yanaşmalardan biridir. Beynəlxalq səviyyədə bu mövzu ilə bağlı çoxsaylı elmi məqalələr, texniki sənədlər və təlimat materialları mövcuddur. Xüsusilə Google, Mozilla və digər qabaqcıl texnologiya şirkətlərinin hazırladığı inkişaf sənədlərində PWA texnologiyasının arxitekturası, təhlükəsizlik aspektləri və performans üstünlükləri ətraflı şəkildə təqdim olunur. Həmçinin PWA tətbiqlərinin istifadəçi təcrübəsinə təsiri ilə bağlı aparılmış bəzi akademik tədqiqatlar da mövcuddur ki, bu da sahənin beynəlxalq miqyasda elmi əsaslarla öyrənilməsinə imkan yaratmışdır.

Bununla belə, istifadəçi təcrübəsinin PWA-lar üzərində kognitiv modelləşdirmə vasitəsilə təhlili və simulyasiya modellərinə əsaslanan yanaşmalar hələlik məhdud sayda tədqiqatda əhatə olunmuşdur. İstifadəçi davranışlarının CogTool və KLM modelləri əsasında qiymətləndirilməsi isə mövzuya əlavə dərinlik qazandıran və nadir hallarda tətbiq olunan yanaşmadır. Azərbaycan dilində və yerli elmi mühitdə bu sahəyə dair tədqiqatların sayı olduqca azdır və mövzu daha çox texniki aspektlər üzrə səthi səviyyədə işlənmişdir.

**Tətqiqatın məqsədi.** Bu tədqiqatın əsas məqsədi Progressive Web Application (PWA) texnologiyasının istifadəçi təcrübəsi (UX) baxımından effektivliyini qiymətləndirmək, bu texnologiyanın müasir veb tətbiqlərdə tətbiq imkanlarını araşdırmaq və istifadəçi davranışlarını kognitiv modelləşdirmə alətləri vasitəsilə analiz edərək praktiki nəticələrə əsaslanan təkliflər formalaşdırmaqdır.

Araşdırmada əsas diqqət PWA texnologiyasının yalnız texniki üstünlüklərinin deyil, həm də insan-mərkəzli dizayn prinsiplərinə cavab verən funksional imkanlarının öyrənilməsinə yönəldilmişdir. Bununla yanaşı, istifadəçi ssenariləri əsasında hazırlanan prototip üzərində aparılan UX təhlilləri nəticəsində istifadəçi davranışlarının nə dərəcədə optimallaşdırıldığı, interaktiv elementlərin səmərəliliyi və tətbiqin istifadəyə yararlılığı qiymətləndirilmişdir.

Tədqiqat həm nəzəri yanaşmaların təhlilini, həm də real istifadəçi testlərinin nəticələrinə əsaslanan praktiki qiymətləndirməni özündə birləşdirərək, PWA texnologiyasının daha geniş tətbiq sahələrinə uyğunlaşdırılması üçün elmi əsas yaratmağı qarşıya məqsəd qoymuşdur.

**Tədqiqatın obyekti.** Bu tədqiqatın obyekti müasir veb texnologiyalarından biri olan Progressive Web Application (PWA) tətbiqlərinin istifadəçi təcrübəsinə təsiri və interfeys funksionallığının qiymətləndirilməsi prosesidir. Burada əsas diqqət istifadəçi ilə sistem arasında qarşılıqlı əlaqənin effektivliyinə, tətbiqin struktur elementlərinin istifadəyə nə dərəcədə uyğun olduğuna və tətbiqin istifadəsində yaranan davranış nümunələrinin analizinə yönəldilmişdir.

PWA texnologiyası həm veb, həm də mobil mühitdə çevik və funksional tətbiqlər təqdim etdiyi üçün tədqiqat obyektinə bu platformada yaradılmış prototip tətbiq və onun istifadəçi baxımından qiymətləndirilməsi daxildir. Bu çərçivədə həm texniki elementlər (offline rejim, responsive dizayn, sürətli yükləmə və s.), həm də istifadəçinin təcrübə səviyyəsi, davranış ardıcıllığı və tətbiqə adaptasiya sürəti öyrənilmişdir.

**Tədqiqatın nəzəri əsasları.** Tədqiqatın nəzəri əsaslarını insan-kompüter qarşılıqlı əlaqəsi (HCI – Human-Computer Interaction), istifadəçi təcrübəsi (UX – User Experience) nəzəriyyələri və kognitiv modelləşdirmə yanaşmaları təşkil edir. Bu sahələr informasiya sistemləri və proqram təminatının dizaynı zamanı istifadəçinin davranışlarını və təcrübəsini nəzərə almaqla sistemin funksional və estetik baxımdan optimallaşdırılmasını hədəfləyir.

Tədqiqatda istifadə olunan Keystroke-Level Model (KLM) və GOMS modeli (Goals, Operators, Methods, Selection Rules) kimi kognitiv modellər istifadəçinin interfeys üzərindəki hərəkətlərini ardıcıl, ölçüləbilən və analizə uyğun formada təsvir etməyə imkan verir. Bu modellər istifadəçi əməliyyatlarını klik, yazı, baxış və düşünmə kimi elementlərə ayıraraq, ümumi tapşırığın yerinə yetirilmə müddətini və mürəkkəbliyini proqnozlaşdırmağa əsaslanır.

**Tədqiqatın elmi-təcrübi əhəmiyyəti.** Tədqiqatın elmi-təcrübi əhəmiyyəti ondan ibarətdir ki, Progressive Web Application (PWA) texnologiyasının istifadəçi təcrübəsinə təsiri ilk dəfə olaraq kognitiv modelləşdirmə metodları və real istifadəçi testləri əsasında sistemli şəkildə təhlil olunmuşdur. Bu yanaşma mövcud elmi ədəbiyyatda məhdud şəkildə işlənmiş bir sahəyə yenilik gətirməklə yanaşı, praktik tətbiqlərdə qərar qəbuletmə üçün də əsaslı zəmin yaradır.

Tədqiqat çərçivəsində hazırlanmış PWA prototipi və onun üzərində aparılan interaktiv UX təhlili proqram təminatı sahəsində çalışan tərtibatçılar, UX dizaynerlər və texnoloji startaplar üçün ölçüləbilən və tətbiq edilə bilən metodoloji çərçivə təqdim edir. Bu çərçivə vasitəsilə sistemlər istifadəyə təqdim olunmadan əvvəl istifadəçi davranışlarına uyğun olaraq sınaqdan keçirilə bilər və nəticələrə əsaslanan təkmilləşdirmələr həyata keçirilə bilər.

Eyni zamanda, CogTool aləti ilə aparılan modelləşdirmə nəticəsində istifadəçi əməliyyatlarının vaxt göstəriciləri və funksional səmərəliliyi əvvəlcədən proqnozlaşdırılmış və real test nəticələri ilə müqayisə edilərək təhlil və təklif əsaslı dizayn yanaşması formalaşdırılmışdır. Bu isə həm elmi-tədqiqat mühitində, həm də sənaye səviyyəsində istifadəyə yararlı nəticələrin əldə olunmasına şərait yaradır.

**İşin strukturu və həcmi:** Dissertasiya işi giriş, üç əsas fəsil, nəticə və istifadə olunmuş ədəbiyyatlar siyahısından ibarətdir. Hər bir fəsildə tədqiqat məqsədinə uyğun nəzəri və praktiki məqamlar mərhələli şəkildə təqdim olunmuş, hər fəsil daxilində konkret alt başlıqlarla mövzu sistemləşdirilmişdir. Ümumilikdə işin həcmi 77 səhifə təşkil edir.

**Birinci fəsil**, 2 alt başlıqdan ibarət olmaqla, “Progressive Web Application texnologiyasının əsasları və istifadəçi təcrübəsinin nəzəri əsasları” adlanır və iki alt başlıqdan ibarətdir. Bu fəsildə PWA texnologiyasının texniki strukturu, komponentləri, inkişaf prinsipləri, həmçinin istifadəçi təcrübəsi ilə bağlı nəzəri yanaşmalar və UX dizayn meyarları geniş şəkildə izah olunmuşdur.

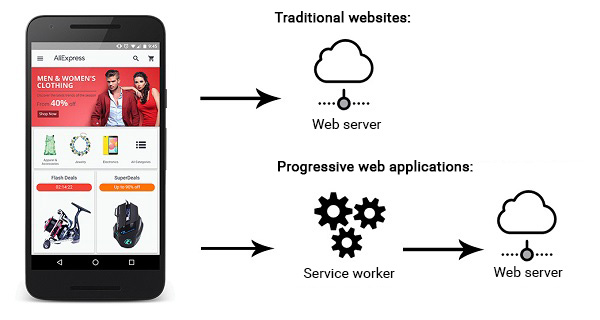
**İkinci fəsildə** PWA texnologiyası əsasında hazırlanmış tətbiqin inkişaf mərhələləri təqdim olunmuşdur. Bu fəsildə tətbiqin arxitekturası, funksional imkanları, texnoloji həlləri və prototipin necə qurulduğu ətraflı təsvir edilmiş, dizayn elementlərinin seçimi və tətbiqə inteqrasiyası təhlil olunmuşdur.

**Üçüncü** **fəsildə** dissertasiyanın əsas tədqiqat mərhələsi olan istifadəçi təcrübəsinin analizinə yer verilmişdir. Burada həm CogTool aləti ilə aparılmış kognitiv modelləşdirmə təhlilləri, həm də real istifadəçilərlə keçirilmiş testlər əsasında əldə olunan nəticələr müqayisəli formada analiz olunmuşdur. Fəsildə həmçinin istifadəçi rəyləri, səhv nümunələri, performans göstəriciləri və nəticələrin praktiki tətbiq imkanları da dəyərləndirilmişdir.

# I FƏSIL . PROGRESSİVE WEB APPLİCATİON TEXNOLOGİYASININ NƏZƏRİ ƏSASLARI VƏ İSTİFADƏÇİ TƏCRÜBƏSİ ANLAYIŞI

## 1.1 Progressive Web Application tətbiqi üçün istifadəçi təcrübəsinin kognitiv modellərlə təhlili

Texnologiya inkişaf etdikcə məhsul və xidmətlərin hazırlanma üsulu da dəyişir. Son illərdə internet və veb texnologiyalar sahəsində böyük inkişaf baş verib. Yeni texnologiyalar vasitəsi ilə əvvəllər mümkün olmayan funksionallıqlar indi mümkündür (Blessing, 2023). Texnologiyadan təhsil, səhiyyə, kənd təsərrüfatı və digər sahələrdə geniş istifadə olunur (Smith & Doe, 2022). Çoxlu sayda şirkətlər artıq internet üzərindən fəaliyyət göstərir və bu getdikcə daha da artır (Soares, Ishii, & Baiao, 2020). Buna görə də sürətli, güvənli, istifadəsi rahat və keyfiyyətli tətbiqlərə ehtiyac var (Zhang, Li, & Wang, 2019).

**Şəkil 1.1** Ənənəvi veb tətbiqlər ilə proqressiv veb tətbiqlərin müqayisəsi

**Mənbə:** https://decode.agency/article/web-app-development-app-types

Tətbiqlər əsasən iki cür olur: Native və Veb tətbiqlər.Native tətbiqlər – hər bir cihaz üçün ayrıca hazırlanır (məsələn, Android, iOS və ya Windows üçün). Bu cür tətbiqlər xüsusi proqramlaşdırma dili və alətlərdən istifadə olunaraq yazılır. Cihaza tam giriş imkanı verir (kamera, mikrofon, yaddaş və s.) (Soares, Ishii, & Baiao, 2020).

Veb tətbiqlər isə internet saytları kimi işləyir və istənilən cihazda – kompüter, planşet və ya telefonda brauzer vasitəsilə açılır. Adətən HTML5 ilə yazılır. Görünüş və istifadə baxımından nativ tətbiqlərə bənzəyirlər, amma cihazın bütün funksiyalarına tam giriş imkanları yoxdur (Sharma, Jindal, & Singh, 2019).

İstifadəçi sayına çıxış baxımından veb tətbiqlər nativ tətbiqlərdən daha əlçatandır. Native tətbiqlərin hazırlanması, yenilənməsi və müxtəlif platformalar üçün paylanması daha çox vaxt və resurs tələb edir (Soares, Ishii, & Baiao, 2020). Həmçinin, bu cür tətbiqləri mağazalarda yayımlamaq çətin və uzun bir prosesdir. İstifadəçi bir nativ tətbiqdən istifadə etmək üçün bir neçə mərhələdən keçməlidir: mağazaya daxil olmaq, qeydiyyatdan keçmək, yaddaşı yoxlamaq, tətbiqi yükləmək, quraşdırmaq və sonda işlətmək. Araşdırmalara görə, istifadəçi ilk dəfə tətbiqlə tanış olduğu andan başlayaraq istifadəyə keçənə qədər hər mərhələdə istifadəçilərin təxminən 20%-i itirilir. Bir çox istifadəçi bu prosesi çətin və yorucu hesab edir. Bu da həm şirkətlər, həm də proqramçılar üçün ciddi bir problem yaradır (Blessing, 2024).

Nativ və veb platformaların hərəsinin öz üstün və zəif tərəfləri olduğuna görə, həm vebin geniş istifadə imkanını, həm də nativ tətbiqlərin sürət, etibarlılıq və rahatlığını özündə birləşdirən yeni bir yanaşmaya ehtiyac yaranmışdı. Bu yanaşma – Progressive Web App (PWA) adlanır (Blessing, 2024).

PWA həm vebin geniş istifadə imkanlarını, həm də nativ tətbiqlərin sürətli və cəlbedici təcrübəsini təqdim edən yeni nəsil veb tətbiqlərdir. İlk dəfə 2016-cı ilin may ayında San Fransiskoda keçirilən Google I/O development konfransında təqdim olunub (Blessing, 2023). PWA-lar veb tətbiqlərdə olmayan bir çox xüsusiyyətləri özündə birləşdirir.

Bu texnologiya Service Worker, App Shell, Web App Manifest və Push bildirişləri kimi funksiyalar vasitəsi ilə PWA-ların native tətbiq kimi görünməsini və işləməsini təmin edir (Chinnappan, 2020). Veb əsaslı olduğu üçün bütün platformalarda işləyə bilir, bu isə həm xərcləri azaldır, həm də daha çox istifadəçiyə çatmağı asanlaşdırır (Zhang, Li, & Wang, 2019).

Bu fəsildə veb və mobil tətbiqlərin əsas növləri haqqında nəzəri məlumat verilir. Xüsusilə, Proqressiv Veb Tətbiqlər (PWA), Nativ Tətbiqlər və Hibrid Tətbiqlər arasındakı fərqlər, üstünlüklər və çatışmazlıqlar detallı şəkildə izah olunur. Bu yanaşma texnoloji qərarların daha məqsədyönlü şəkildə verilməsi üçün əsaslı nəzəri baza yaradır (Sharma, Jindal, & Singh, 2019).

Müasir dövrdə istifadəçilərin gözləntiləri sürətlə artır. Onlar tətbiqlərin sürətli, rahat, funksional və hər bir cihazda problemsiz işləməsini gözləyirlər. Bu səbəbdən, inkişaf etdiricilər daim daha effektiv və istifadəyə uyğun tətbiqlər yaratmağa çalışırlar. Bu kontekstdə, PWA kimi yeni yanaşmalar diqqət mərkəzinə çevrilmişdir (Blessing, 2023).

Service Worker, PWA-nın texniki cəhətdən ən vacib elementlərindən biridir. Bu, veb brauzer ilə server arasında arada duran bir skriptdir və əsas məqsədi məlumatların keşi (cache), şəbəkə sorğularının idarəsi və offline rejimin təmin olunmasıdır (Chinnappan, 2020). İstifadəçi internet bağlantısını itirdikdə belə, Service Worker əvvəldən yaddaşda saxlanmış məlumatlar vasitəsilə tətbiqin işləməsini davam etdirir. Bu, istifadəçi təcrübəsini pozmadan tətbiqin əlçatanlığını təmin edir (Zhang, Li, & Wang, 2019). Məsələn, xəbər portallarında istifadəçi son baxdığı xəbərləri offline rejimdə də oxuya bilər. Service Worker həmçinin push bildirişlərinin alınması və arka planda sinxronlaşdırma kimi funksiyalara da cavabdehdir (Chinnappan, 2020).

App Shell anlayışı, PWA dizaynında tətbiqin əsas quruluşunun sabit saxlanması və yalnız dinamik məzmunun dəyişdirilməsi prinsipinə əsaslanır. Bu struktur, istifadəçiyə sürətli yükləmə və stabil interfeys təcrübəsi təqdim edir (Zhang, Li, & Wang, 2019). App Shell vasitəsilə tətbiqin menyu, başlıq, alt panel kimi sabit hissələri brauzerdə saxlanılır və yalnız məzmun (məsələn, xəbərlər, şəkillər və ya istifadəçi məlumatları) dəyişdirilir. Bu yanaşma səhifə yüklənmə vaxtını əhəmiyyətli dərəcədə azaldır və veb tətbiqə mobil tətbiq hissi verir. İstifadəçi hər dəfə tətbiqə daxil olanda onun qarşısına tanış və sabit interfeys çıxır.

Web App Manifest, PWA-nın cihazın ana ekranına əlavə olunmasına və mobil tətbiq kimi görünməsinə imkan yaradan JSON formatında bir fayldır. Bu fayl tətbiq haqqında adı, ikonası, rəngi, açılış formatı, yönləndirmə ünvanları və s. kimi məlumatları özündə saxlayır (González & Rodríguez, 2022). Manifest faylı vasitəsilə tətbiq tam ekran rejimində, splash screen ilə, və hətta xüsusi dizayn edilmiş ikonla açılır. Bu da istifadəçiyə tətbiqin adi veb sayt deyil, “əsl tətbiq” olduğunu hiss etdirir. Bundan əlavə, bu komponent sayəsində tətbiq axtarış sistemlərində tətbiq kimi tanınır, bu isə SEO baxımından da üstünlük yaradır.

Push Notification texnologiyası PWA-ların istifadəçi ilə qarşılıqlı əlaqəsini artırmaqda mühüm rol oynayır. Server tərəfindən göndərilən bildirişlər istifadəçiyə real vaxt rejimində çatdırılır və onun tətbiqə marağını yenidən oyadır (Blessing, 2024). Bu bildirişlər kampaniyalar, yenilənmiş məzmun, xəbərdarlıqlar və digər aktual mövzularda ola bilər. Nativ tətbiqlərdəki bildirişlərə vizual və funksional cəhətdən çox bənzəyir. Push Notification texnologiyası həmçinin istifadəçinin tətbiqə qayıtma ehtimalını artırır, istifadəçi sadiqliyini formalaşdırır və tətbiqin gündəlik istifadəsini təşviq edir. Bunun texniki icrası üçün Notifications API və Push API birlikdə istifadə olunur.

Bu xüsusiyyət Web App Manifest adlı texnologiya vasitəsilə təmin olunur. Bu fayl tətbiqin ikonunu, adını, açılış rejimini və digər vizual parametrləri müəyyən edir (González & Rodríguez, 2022). Nəticədə, istifadəçi tətbiqi ana ekranda adi mobil tətbiqlərlə yanaşı görür və digər tətbiqlər kimi onu istifadə edə bilir. Bu da istifadəçinin tətbiqə daha tez və rahat şəkildə daxil olmasına şərait yaradır.

Bundan əlavə, push bildirişləri PWA-ların istifadəçi ilə əlaqə yaratmaqda istifadə etdiyi əsas vasitələrdən biridir. Server tərəfindən göndərilən bu bildirişlər istifadəçiyə aktual məlumatları, yenilikləri və ya fərdi təklifləri təqdim edir. Bildirişlər vasitəsilə tətbiq daim istifadəçinin diqqət mərkəzində qalır və onu tətbiqə qayıtmağa təşviq edir (Blessing, 2023). Bu isə yenidən istifadə (retention) və davamlı əlaqə baxımından çox vacib rol oynayır.

Push bildirişlərinin vizual dizaynı və funksionallığı nativ tətbiqlərlə tam uyğun olduğu üçün istifadəçi onları rahatlıqla qəbul edir və adi mobil tətbiqlərdə olduğu kimi onlara cavab verir. Bildirişlər yalnız informasiya ötürmək üçün deyil, həm də interaktiv düymələrlə istifadəçidən birbaşa reaksiya almaq üçün istifadə oluna bilər məsələn, “Oxu”, “Saxla”, “Daha sonra xatırlat” kimi seçimlər vasitəsilə (Blessing, 2023).

PWA-ların bu interaktiv və dinamik strukturu onları statik veb saytlarla müqayisədə daha "canlı" və istifadəçiyə yönəlmiş hala gətirir. İstifadəçilər artıq yalnız kontent istehlakçısı deyil, həm də tətbiqin formalaşmasında bilavasitə iştirak edən bir tərəf olur. Bu isə davamlı inkişaf prinsipini reallaşdırmağa kömək edir (Zhang, Li, & Wang, 2019).

Nəticə olaraq, PWA-ların istifadəçi ilə əlaqəyə əsaslanan tədrici inkişaf modelinə malik olması onların effektivliyini və istifadəçi yönümlülüyünü artırır. Bu yanaşma, həm istifadəçi təcrübəsinin fərdiləşdirilməsinə, həm də tətbiqin dayanıqlı uğur qazanmasına şərait yaradır (Blessing, 2024).

Proqressiv Veb Tətbiqlər (PWA) son illərdə müxtəlif sənaye və xidmət sahələrində çox sürətlə yayılmağa və tətbiq olunmağa başlamışdır. Onların geniş tətbiq sahəsi əsasən texniki üstünlükləri, aşağı xərc tələb etməsi və yüksək istifadəçi əlçatanlığı ilə izah olunur (Blessing, 2024). Müasir dövrdə istifadəçilərin mobil cihazlar üzərindən sürətli və rahat xidmət gözləntiləri artdıqca, PWA texnologiyası bir çox təşkilat və şirkət üçün səmərəli alternativ kimi çıxış edir.

## 1.2 Proqressiv veb tətbiqin komponentləri

Proqressiv veb tətbiqlərin (PWA) hazırlanması və onların performans, əlçatanlıq (accessibility), ən yaxşı təcrübələr (best practices) və SEO (axtarış sisteminə uyğunluq) kimi bütün tələblərə cavab verməsi olduqca çətin bir prosesdir (Blessing, 2023). Bunu reallaşdırmaq üçün PWA-nın bütün əsas komponentləri yəni Service Worker, Web App Manifest, Application Shell modeli və Web Push bildirişləri diqqətlə və bir-biri ilə uyğun şəkildə tətbiq olunmalıdır (Chinnappan, 2020). Bu komponentlərin hər biri növbəti bölmələrdə ətraflı izah ediləcəkdir.

Web App Manifest tətbiq haqqında məlumatları saxlayan sadə bir JSON formatlı fayldır. Bu faylda tətbiqin adı (name), qısa adı (short\_name), təsviri (description), müxtəlif ölçülü cihazlar üçün ikonlar (icons), başlanğıc URL-i (start\_url), görünüş rejimi (display mode), tətbiqin rəng mövzusu (theme\_color) kimi sahələr yer alır (González & Rodríguez, 2022). Web App Manifest-dən istifadə edildikdə, veb tətbiq istifadəçinin cihazında ana ekrana əlavə olunaraq nativ tətbiqlərin arasında görünür. Bu isə istifadəçiyə mobil tətbiqlərdə olduğu kimi tətbiqə daha sürətli çıxış və tam ekran təcrübəsi yaşamaq imkanı verir (Blessing, 2023).

Şəkildə göstərilən Listing 1-də bir veb tətbiq üçün yazılmış manifest.json faylının nümunəsi verilir. Bu fayl tətbiqin necə görünəcəyini və cihazda necə işləyəcəyini təyin edən əsas parametrləri özündə əks etdirir.

**Şəkil 1.2** PWA üçün Web Application Manifest

**Mənbə:** Müəllif tərəfindən hazırlanıb.

name – tətbiqin açılış zamanı, yəni splash screen üzərində göründüyü tam adıdır. short\_name isə tətbiqin ana ekranda (home screen) ikonun altında görünən qısa adıdır.

icons sahəsi bir sıra (array) obyektlərdən ibarətdir və bu obyektlər tətbiq ikonunun xüsusiyyətlərini göstərmək üçün istifadə olunur. Hər bir obyektin içərisində src – şəkil faylının yolu, sizes – ölçüsü və type – fayl növü kimi sahələr mövcuddur.

Bundan başqa, start\_url – istifadəçi tətbiq ikonuna klik etdikdə açılacaq ilk səhifənin ünvanını göstərir. display sahəsi isə tətbiqin hansı görünüş rejimində açılacağını təyin edir. Bu rejimlərdən ən çox istifadə olunanlar standalone və fullscreen rejimləridir. standalone rejimi tətbiqin brauzerin interfeysi olmadan (məsələn, ünvan sətri, brauzer düymələri olmadan) açılmasını təmin edir və onu əsl mobil tətbiq kimi göstərir. Digər tərəfdən, fullscreen rejimi tətbiqin tam ekran rejimində istifadəsinə imkan yaradır.

Beləliklə, bir veb tətbiqə manifest.json faylının əlavə olunması, tətbiqə nativ mobil tətbiq görünüşü və hissi qazandırır (González & Rodríguez, 2022). Bu yanaşmanın necə göründüyü və işlədiyi isə Şəkil 1.3-də əyani şəkildə təsvir olunmuşdur.

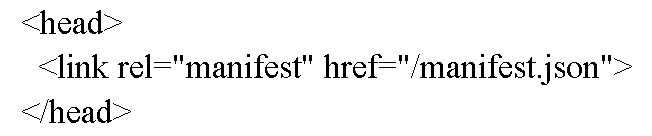
**Şəkil 1.3** Tətbiqin manifesti

**Mənbə:** https://www.theseus.fi/bitstream/handle/10024/142997/PWA thesis.pdf

Şəkil 1.3-də göstərildiyi kimi, Web tətbiq Manifest faylı ilə birləşdirildikdə, istifadəçi həmin tətbiqi öz cihazının ana ekranına quraşdıra bilir. Yəni, veb tətbiq nativ mobil tətbiq kimi cihazda görünür və bir kliklə istifadəyə hazır olur.

Bunun baş verməsi üçün, yəni brauzerin tətbiqi PWA kimi tanıya bilməsi üçün, manifest faylı mütləq tətbiqin əsas HTML faylına adətən index.html səhifəsinə düzgün şəkildə qoşulmalıdır. Bu qoşulma HTML sənədinin <head> hissəsində aparılır və bu məqsədlə aşağıdakı sintaksisdən istifadə olunur:

**Şəkil 1.4** Manifest faylının HTML-ə qoşulması



**Mənbə:** Müəllif tərəfindən hazırlanıb.

Şəkil 1.4-də göstərildiyi kimi, bu sətir vasitəsilə manifest.json faylı brauzerə təqdim olunur. Beləliklə, brauzer bu fayldakı məlumatlara əsaslanaraq veb tətbiqi bir mobil tətbiq kimi qəbul edir və ana ekrana əlavə etmək üçün istifadəçiyə imkan yaradır. Bu addım PWA texnologiyasının ən vacib hissələrindən biridir və tətbiqin istifadəçi cihazında nativ kimi işləməsinə şərait yaradır.

Bir veb tətbiqin saytda quraşdırma banneri (install banner) kimi görünməsi və mobil tətbiqə bənzər bir istifadə təcrübəsi təqdim edə bilməsi üçün bəzi əsas şərtləri yerinə yetirməsi tələb olunur. Bu tələblər PWA texnologiyasının düzgün işləməsi və brauzer tərəfindən tanınması üçün vacibdir.

Bu şərtlər aşağıdakılardır:

Sayt mütləq HTTPS protokolu üzərindən xidmət göstərməlidir. Yəni, sayt təhlükəsiz bağlantı ilə açılmalıdır. Bu həm istifadəçi məlumatlarının qorunması üçün vacibdir, həm də PWA-nın işləməsi üçün texniki tələbdir.

Saytda Service Worker qeydiyyatdan keçməlidir. Service Worker veb tətbiqin offline rejimdə işləməsinə, önbellek funksionallığına və push bildirişlərinə imkan yaradır. Bu komponent olmadan PWA əsas funksiyalarını yerinə yetirə bilmir.

Sayta aid olan web app manifest faylında ən azı dörd əsas sahə yer almalıdır:

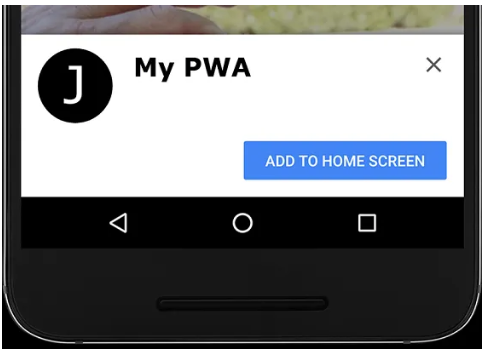
name və ya short\_name (seçimdən asılı olaraq hər ikisi),

start\_url (tətbiqin başlanğıc səhifəsinin ünvanı),

icons (tətbiq ikonları),

display (tətbiqin görünüş rejimi, məsələn standalone və ya fullscreen).

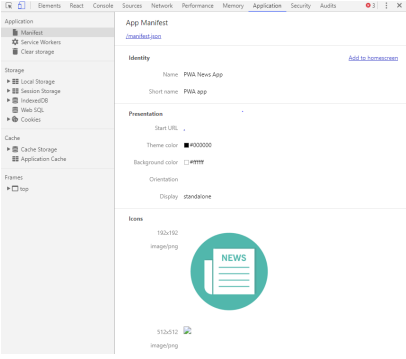
Yuxarıda sadalanan bu meyarlar tam yerinə yetirildikdə, brauzer tətbiqi PWA kimi tanıyır və istifadəçiyə tətbiqi cihazın ana ekranına quraşdırmaq üçün quraşdırma banneri (install prompt) göstərir. Bu prosesin necə göründüyü Şəkil 1.5-də əyani olaraq nümayiş etdirilib.

**Şəkil 1.5** PWA Tətbiqinin Quraşdırma Təklifi

**Mənbə:** https://www.flowmatters.com/blog/progressive-web-app-pwa-what-is-it-and-why-use-it/

Bu sadə, lakin vacib şərtlər yerinə yetirilmədən bir veb tətbiq Progressive Web App statusu ala bilməz və istifadəçiyə tətbiq kimi quraşdırma təklifi təqdim olunmaz.

Şəkil 1.5-də göstərildiyi kimi, əgər manifest faylı yuxarıda qeyd olunan meyarlara cavab verirsə, brauzer avtomatik olaraq veb tətbiq üçün quraşdırma bannerini (install banner) işə salır. Bu banner istifadəçiyə tətbiqi öz cihazının ana ekranına əlavə etmək imkanı yaradır. Beləliklə, istifadəçi veb tətbiqdən tam ekran rejimində və mobil tətbiq kimi istifadə edə bilir. Bu, istifadəçi təcrübəsini əhəmiyyətli dərəcədə yaxşılaşdırır və tətbiqin daha tez-tez istifadə olunmasına şərait yaradır.Hazırlanan veb tətbiqin manifest.json faylının düzgün şəkildə işlədiyini və brauzer tərəfindən tanınıb-tanınmadığını əllə yoxlamaq da mümkündür. Bu məqsədlə Google Chrome brauzerində yerləşən DevTools alətindən istifadə olunur. DevTools-un Application panelindəki Manifest bölməsində tətbiqə bağlı manifest faylına baxmaq, oradakı sahələri, ikonları və rəng seçimlərini birbaşa vizual olaraq görmək mümkündür. Bu üsulla faylın düzgün qoşulub-qoşulmadığını, zəruri sahələrin olub-olmadığını və brauzer tərəfindən necə qəbul edildiyini yoxlamaq olar. Bu prosesi necə yerinə yetirmək lazım olduğu Şəkil 1.6-də əyani şəkildə təsvir olunmuşdur.

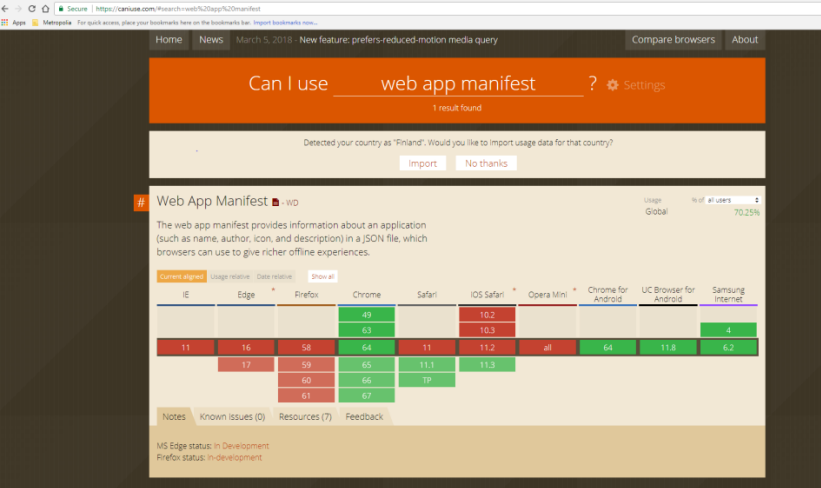
**Şəkil 1.6** Veb tətbiqlərin Manifest Chrome Developer alətlərin tətbiq paneli

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Veb tətbiq manifest faylının brauzerlərlə uyğunluğu ilə bağlı bəzi problemlər mövcuddur. Belə ki, bütün brauzerlər manifest faylını tam şəkildə dəstəkləmir. Bu, Progressive Web App-ların fərqli cihaz və platformalarda eyni səviyyədə işləməsinə mane ola bilər.

Veb tətbiq manifestinin müxtəlif brauzerlərlə uyğunluğunu onlayn yoxlama alətləri vasitəsilə yoxlamaq mümkündür. Bu cür alətlər manifest faylını analiz edir və hansı brauzerlərin onu dəstəklədiyini, hansı funksiyaların hansı brauzerlərdə işlədiyini vizual şəkildə təqdim edir. Belə yoxlama alətindən necə istifadə olunduğu Şəkil 1.7-da əyani olaraq göstərilmişdir.

**Şəkil 1.7** Müxtəlif brauzerlərlə Web App Manifest uyğunluğunun nəticəsi.



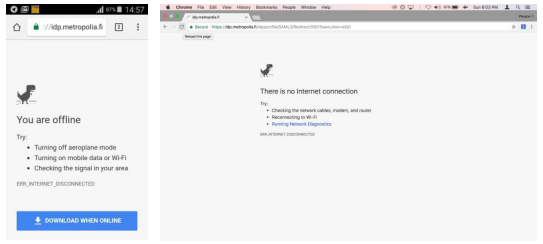
**Mənbə:** https://www.theseus.fi/bitstream/handle/10024/142997/PWA thesis.pdf

Şəkil 1.7-dən də göründüyü kimi, Web App Manifest xüsusilə Google Chrome, Android üçün Chrome, və UC Browser kimi brauzerlərdə çox yaxşı dəstəklənir. Son zamanlar Safari və iOS Safari də manifest faylını müəyyən səviyyədə dəstəkləməyə başlayıb ki, bu da iPhone və iPad istifadəçiləri üçün əhəmiyyətli irəliləyiş sayılır.

Bununla belə, bəzi digər populyar brauzerlər məsələn, Firefox və Microsoft Edge bu texnologiyanı hələ də tam şəkildə dəstəkləmir. Bu isə PWA-nın həmin brauzerlərdə məhdud funksionallıqla işləməsinə və ya ümumiyyətlə install bannerinin görünməməsinə səbəb ola bilər.

Müasir veb texnologiyaları getdikcə daha güclü, funksional və istifadəçi yönümlü hala gəlməkdədir. Bugünkü veb tətbiqlər yalnız sadə məlumat səhifələri deyil, kompleks qarşılıqlı əlaqələr, animasiyalar, real vaxt məlumat ötürülməsi və hətta offline rejim kimi imkanlara malikdir. Bununla belə, istifadəçi təcrübəsini pozan əsas məsələlərdən biri zəif və ya mövcud olmayan internet bağlantısıdır. Əgər istifadəçi bir veb tətbiqi açmaq istəyərkən internet bağlantısı zəifdirsə və ya ümumiyyətlə yoxdursa, onu yalnız "İnternet bağlantısı yoxdur" mesajı qarşılayır. Bu isə istifadəçi üçün mənfi təcrübə yaradır və tətbiqə olan marağın azalmasına səbəb ola bilər.Bu problemi həll etmək və veb tətbiqləri daha etibarlı, dayanıqlı hala gətirmək üçün Service Worker texnologiyası ortaya çıxmışdır. Service Worker-lər veb tətbiqlərə offline rejim, önbellek (cache) və arka planda işləmə kimi funksiyaları qazandıran əsas komponentlərdən biridir. Bu texnologiya vasitəsilə istifadəçi, internet bağlantısı olmasa belə, daha əvvəl ziyarət etdiyi səhifələri və məlumatları görməyə davam edə bilər.

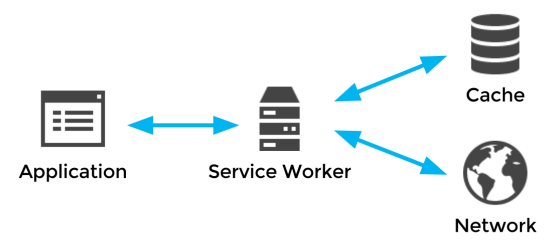
Şəkil 1.8-də göstərildiyi kimi, veb tətbiqlər offline rejimdə istifadəçiyə heç bir faydalı məlumat təqdim edə bilmir. İstifadəçi yalnız “İnternet bağlantısı yoxdur” səhifəsi ilə qarşılaşır və bu, tətbiqin istifadəsini mümkünsüz hala gətirir. Lakin Service Worker texnologiyasının tətbiqi bu problemi nəzakətli şəkildə idarə olunan bir vəziyyətə çevirib. Yəni artıq offline rejim bir məhdudiyyət deyil, idarə oluna bilən və hətta istifadəçi təcrübəsinə uyğunlaşdırıla bilən bir hal kimi qiymətləndirilir.

**Şəkil 1.8** Veb tətbiqin offline vəziyyətdə mobil və masaüstü görünüşü.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Service Worker veb tətbiqin əsas səhifəsindən asılı olmadan fon rejimində işləyən və hadisələrə cavab verən (event-driven) bir JavaScript skriptidir. Bu skript veb səhifədən ayrı şəkildə fəaliyyət göstərir və tətbiqə gələn şəbəkə sorğularını yaxalayaraq (intercept), onların serverə necə yönləndiriləcəyini və ya cavabların necə qaytarılacağını müəyyən edir. Başqa sözlə desək, Service Worker brauzer ilə şəbəkə arasında bir növ vasitəçi (proxy) rolunu oynayır.

Service Worker-lərin offline rejimdə necə işlədiyini və məlumatları necə saxladığını daha yaxşı başa düşmək üçün Şəkil 1.9-da göstərilən önbellekləmə (caching) strategiyasına nəzər yetirmək olar. Bu strategiya veb tətbiqin offline rejimdə belə istənilən səhifəni və ya məlumatı istifadəçiyə təqdim edə bilməsini izah edir.

**Şəkil 1.9** Service Worker-in keşdə işləmə strategiyası.

**Mənbə:** https://www.theseus.fi/bitstream/handle/10024/142997/PWAthesis.pdf

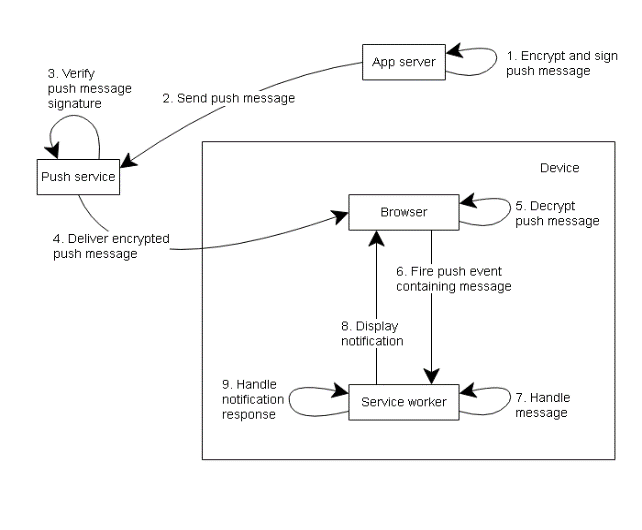
Şəkil 1.9-də göstərildiyi kimi, Service Worker tətbiqlə şəbəkə arasında yerləşir və hansı hallarda məlumatın keşlənməsi (cache), hansı hallarda isə birbaşa şəbəkədən alınacağına qərar verir. Bu vasitəçi mexanizm, veb tətbiqin daha ağıllı və istifadəçiyə uyğun şəkildə işləməsinə şərait yaradır.

Əgər bir xəbər tətbiqi Service Worker istifadə etmədən yazılıbsa, istifadəçi internet bağlantısını itirdikdə standart “Siz hal-hazırda offline rejimdəsiniz” mesajı göstəriləcək. Bu, sadəcə istifadəçiyə xidmət verə bilmədiyini bildirən passiv bir yanaşmadır. Lakin əgər həmin tətbiqdə Service Worker işləyirsə, tətbiq öncəki səfərdə yüklənmiş istifadəçi interfeysini (GUI) göstərə bilər, JavaScript skriptlərini işlədə bilər və hətta əvvəlcədən cache-də saxlanılmış xəbərləri və digər məzmunu istifadəçiyə təqdim edə bilər. Bu yanaşma istifadəçi üçün daha funksional və faydalı offline təcrübə təmin edir.

Bundan əlavə, yaxşı yazılmış Service Worker təkcə məlumatı keşlənmək və göstərmək deyil, həm də şəbəkə üzərindən serverə göndərilmiş məlumatları qeydə almaq, onları müvəqqəti saxlamaq və internet bərpa olunanda sinxronizasiya etmək (background sync) funksiyasını da yerinə yetirə bilər. Bu, istifadəçi ilə server arasındakı məlumat axınının davamlı olmasına şərait yaradır, hətta əlaqə kəsilsə belə.

Eyni zamanda Service Worker, serverdən gələn push bildirişlərini qəbul etmək üçün də istifadə olunur. Bu bildirişlər, tətbiqin bağlı olduğu hallarda belə istifadəçiyə xəbər göndərə bilər. Bu funksionallıqlar push events və sync events Service Worker-in müasir veb tətbiqlər üçün nə qədər vacib bir texnologiya olduğunu göstərir. Bu prosesin necə baş verdiyi Şəkil 1.10-da vizual olaraq təsvir olunmuşdur.

Şəkil 1.10-da göstərildiyi kimi, Service Worker tətbiq ilk dəfə internet üzərindən yükləndikdə, tətbiqin özü (App Shell) və məzmununu keşləyir, yəni cihazın yaddaşında saxlayır. Daha sonra istifadəçi zəif internet bağlantısı ilə qarşılaşdıqda və ya ümumiyyətlə şəbəkədən kənarda olduqda belə, Service Worker bu önbelleklənmiş məlumatları istifadəçiyə göstərərək tətbiqin istifadəsini mümkün edir. Bu yanaşma istifadəçi təcrübəsini pozmadan tətbiqin əsas görünüş və funksiyalarının istifadəsinə imkan verir.

**Şəkil 1.10** Tətbiqin (App Shell) Service Worker tərəfindən keşlənməsi, onun şəbəkə bağlantısı olmadan belə yüklənməsinə imkan yaratması.

**Mənbə:** https://developer.mozilla.org/en-US/docs/Web/Progressive\_web\_apps/Guides/Offline\_and\_background\_operation

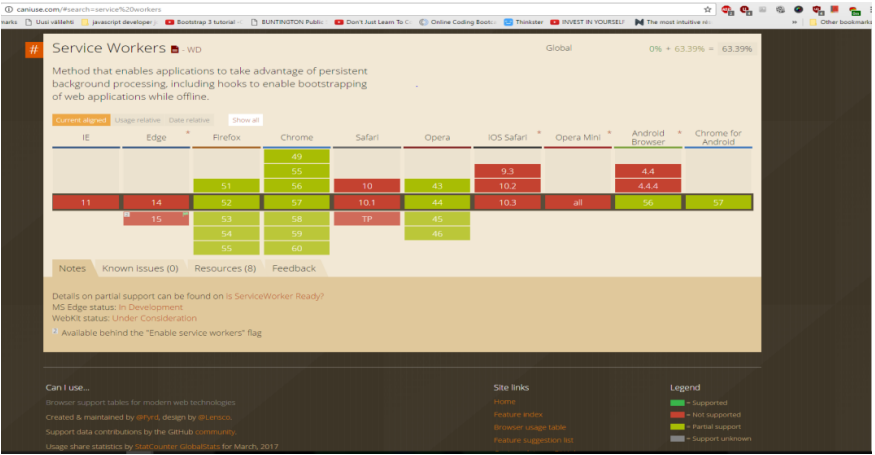
Nəticədə, Service Worker-lə təchiz olunmuş bir tətbiq uyğun brauzerlərdə istifadədə olduqda istifadəçiyə çox daha yüksək səviyyəli, sürətli və offline dəstəkli təcrübə bəxş edir. Bu isə PWA texnologiyasının əsas məqsədlərindən biridir.

Service Worker texnologiyasının hansı brauzerlərdə dəstəkləndiyini isə Şəkil 2.4 əyani şəkildə göstərir. Bu cədvəl və ya diaqram vasitəsilə inkişaf etdiricilər hansı platformalarda Service Worker funksiyalarından tam istifadə oluna biləcəyini və hansı brauzerlərdə məhdudiyyətlərin olduğunu rahatlıqla görə bilərlər.

Şəkil 1.11-də göstərildiyi kimi, Chrome, Firefox, Opera, Chrome for Android, Safari, iOS Safari və Microsoft Edge kimi ən populyar brauzerlərin əksəriyyəti artıq Service Worker texnologiyasını dəstəkləyir. Bu dəstək, Progressive Web App-ların geniş auditoriyaya çatması üçün vacib şərtlərdən biridir. Bununla belə, bəzi brauzerlər, xüsusilə Opera Mini və Internet Explorer bu texnologiyanı hələ də dəstəkləmir. Bu isə həmin brauzerlərdə PWA-nın əsas funksiyalarının işləməməsi deməkdir.

Service Worker, hadisələrə əsaslanan (event-driven) bir skript olduğundan, onun fəaliyyət müddəti çox qısadır. Yəni bu skript yalnız hansısa bir hadisə baş verdikdə məsələn, şəbəkə sorğusu, push bildirişi və ya sinxronizasiya tələbi daxil olduqda aktivləşir, hadisəni emal edir və sonra bağlanır. O, davamlı şəkildə işləyən proses deyil, əksinə müvəqqəti və məqsədəuyğun şəkildə aktivləşən bir komponentdir.

**Şəkil 1.11** Service Worker texnologiyasının brauzerlərlə uyğunluğu

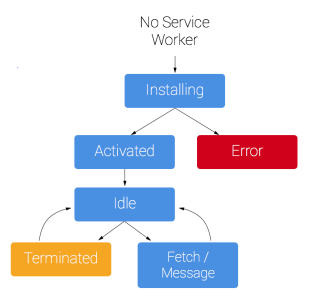


**Mənbə:** <https://www.theseus.fi/bitstream/handle/10024/142997/PWA thesis.pdf>

Bu yanaşma developerlərə şəbəkəni əlavə bir imkan kimi qəbul edərək tətbiqi onun üzərində deyil, önbelleklənmiş resurslar üzərində qurmağa şərait yaradır. Service Worker vasitəsilə tətbiqdə istifadə olunan faylların HTML, CSS, JS, şəkillər və digər məlumatların necə və nə zaman cache-ə alınacağına tam nəzarət etmək mümkündür. Bu nəzarət, PWA-ların offline rejimdə işləməsini təmin edən əsas mexanizmlərdən biridir. Doğru qurulmuş bir önbellekləmə strategiyası sayəsində veb səhifə internet bağlantısı olmasa belə istifadəçiyə təqdim oluna bilər və hətta zəif şəbəkə şəraitində belə daha sürətli yüklənər.

Service Worker-in həyat dövriyyəsi (life cycle) tamamilə veb səhifədən müstəqildir. Yəni səhifə yeniləndikdə və ya bağlandıqda Service Worker fəaliyyətini dayandırmır. Onun özünəməxsus müstəqil qurulma, aktivləşmə və işləmə mərhələləri mövcuddur. Bu prosesin necə baş verdiyi isə Şəkil 9-da Service Worker-in ilk dəfə qurulması zamanı keçdiyi həyat dövriyyəsinin sadələşdirilmiş versiyası şəklində əyani şəkildə göstərilmişdir. Bu mərhələlər növbəti bölmədə daha detallı şəkildə izah olunacaq.

Şəkil 1.12-də göstərildiyi kimi, Service Worker-in quraşdırılma prosesi onun JavaScript faylında qeydiyyatı (registration) ilə başlayır. Bu mərhələdə veb tətbiqin JavaScript kodu brauzerə bildirir ki, müəyyən bir Service Worker faylı mövcuddur və o qeydiyyata alınmalıdır.

**Şəkil 1.12** Service Worker-in həyat dövriyyəsinin sadələşdirilmiş versiyası.

**Mənbə:** <https://www.theseus.fi/bitstream/handle/10024/142997/PWA thesis.pdf>

Bu prosesin əsas məqsədi Service Worker skriptinin brauzer tərəfindən tanınmasını və idarə olunmasını təmin etməkdir. Qeydiyyat zamanı tətbiqdə istifadə olunan .js uzantılı Service Worker faylı (sw.js, service-worker.js və s.) göstərilir və brauzer həmin faylı analiz edərək quraşdırma (install) mərhələsinə keçir.

Bu qeydiyyat adətən veb səhifənin əsas JavaScript faylında və ya index.html daxilində aşağıdakı kimi yazılır:

if ('serviceWorker' in navigator) {

navigator.serviceWorker.register('/service-worker.js')

.then(function(registration) {

console.log('Service Worker qeydiyyatdan keçdi:', registration);

})

.catch(function(error) {

console.log('Qeydiyyat zamanı xəta baş verdi:', error);

});

}

Bu kod vasitəsilə brauzer Service Worker-i qeydiyyatdan keçirir və onun fəaliyyətə başlaması üçün ilk addımı atır. Əgər qeydiyyat uğurlu olarsa, Service Worker install hadisəsini alır və önbellekləmə prosesləri üçün hazır vəziyyətə gəlir. Əgər bu mərhələdə hər hansı sintaksis və ya əlaqə xətası baş verərsə, Service Worker aktivləşmir və qeydiyyat baş tutmur.

Bu qeydiyyat addımı Service Worker həyat dövriyyəsinin başlanğıcı hesab olunur və tətbiqin offline funksionallığı, push bildirişləri və digər PWA imkanlarından istifadə edə bilməsi üçün vacibdir.

Service Worker-in qeydiyyata alınması (registration) üçün JavaScript-də istifadə olunan ən tipik və funksional nümunələrdən biridir. Aşağıda bu kodun nə etdiyini addım-addım və sadə şəkildə izah edirəm:

if ('serviceWorker' in navigator) {

Bu sətr brauzerin Service Worker-i dəstəkləyib-dəstəkləmədiyini yoxlayır. Əgər dəstəkləyirsə, növbəti addımlar icra olunur. Əks halda heç bir əməliyyat aparılmır. Bu təhlükəsizliyə görə vacibdir.

window.addEventListener('load', function() {

Burada load hadisəsinə qulaq asılır. Bu, səhifə tam yükləndikdən sonra işləyəcək funksiyadır. Yəni Service Worker yalnız bütün səhifə resursları yükləndikdən sonra qeydiyyatdan keçiriləcək.

navigator.serviceWorker.register('/sw.js').then(function(registration) {

Bu sətrdə /sw.js adlı Service Worker faylı qeydiyyata alınır. Qeydiyyat uğurla baş tutarsa, .then() funksiyası işə düşür və registration obyektini geri qaytarır.

console.log('ServiceWorker registration successful with scope: ', registration.scope);

Əgər qeydiyyat uğurludursa, bu mesaj brauzerin konsoluna çıxarılır və hansı “scope” (yəni tətbiq sahəsi) üzrə qeydiyyat baş verdiyi göstərilir.

}).catch(function(err) {

console.log('ServiceWorker registration failed: ', err);

Əgər hər hansı səbəbdən qeydiyyat uğursuz olarsa, .catch() funksiyası işə düşür və səbəblə birlikdə xəta mesajı konsola yazılır.

Göstərilən kod hissəsi, Service Worker API-nin mövcud olub-olmadığını yoxlayır, əgər dəstəklənirsə, səhifə tam yükləndikdən sonra sw.js faylını qeydiyyatdan keçirir (Smith & Doe, 2022). Bu qeydiyyat uğurla tamamlandıqdan sonra brauzer Service Worker-in quraşdırılması (install) prosesinə başlayır.

Quraşdırılma mərhələsində, install hadisəsi üçün bir callback funksiyası təyin olunur. Bu hadisə baş verdikdə, Service Worker önbellekləmə (cache) mühiti qurur və statik faylları yaddaşa yazmağa çalışır. Bu prosesin sintaksisi aşağıdakı kimi başlayır:

self.addEventListener('install', function(event) {

// Cache əməliyyatları burada yer alır

});

install hadisəsi zamanı Service Worker, tətbiq üçün vacib olan statik kontenti – HTML, CSS, JavaScript və şəkillər kimi faylları cache-ə yazmağa cəhd edir. Əgər bu önbellekləmə uğurla başa çatarsa, növbəti mərhələ olan aktivləşmə (activate) prosesinə keçilir. Lakin önbellekləmə zamanı hər hansı problem və ya xəta baş verərsə, aktivləşmə baş vermir və Service Worker-in qeydiyyatı təxirə salınır. Bu halda, Service Worker brauzer tərəfindən növbəti cəhddə yenidən quraşdırılmağa çalışacaq.

Bu prosesin necə baş verdiyi isə Service Worker-in quraşdırılması üçün yazılmış real kod nümunəsi ilə əyani şəkildə göstərilmişdir. Bu kod parçaları həm PWA-nın offline rejimdə işləməsi, həm də tətbiqin ilk yüklənmədə resursları keşlənə bilməsi üçün əsas addımlardandır. Növbəti hissədə bu kod nümunəsi təqdim olunacaq.

Listing 4-də təqdim olunan kod nümunəsi, Service Worker-in uğurlu şəkildə quraşdırılması və statik resursların önbelleklənməsi prosesini əyani şəkildə göstərir. Bu nümunədə əvvəlcə CACHE\_NAME dəyişəni ilə cache-in adı təyin olunur və urlsToCache massivində tətbiq üçün önbelleklənəcək əsas faylların yolları göstərilir (məsələn, əsas səhifə /, CSS faylı və JavaScript skripti).

var CACHE\_NAME = 'my-site-cache-v1';

var urlsToCache = ['/','/styles/main.css','/script/main.js'];

self.addEventListener('install', function(event) {

event.waitUntil(

caches.open(CACHE\_NAME)

.then(function(cache) {

console.log('Opened cache');

return cache.addAll(urlsToCache);

})

);

});

Bu kodda install hadisəsi baş verdikdə, caches.open() funksiyası ilə yeni bir cache obyekti açılır və addAll() funksiyası ilə göstərilən fayllar yaddaşa yazılır. Əgər bu mərhələ uğurla tamamlanarsa, Service Worker aktivləşməyə (activate) keçir və artıq hadisələri qəbul etməyə hazır olur.

Bu nöqtədən sonra, aktivləşmiş Service Worker istifadəçinin tətbiq daxilində səhifə dəyişməsi, məlumat ötürməsi və digər qarşılıqlı əlaqələr zamanı fetch, push və sync kimi hadisələri qəbul edə bilir. Listing 5-də bu hadisələrdən biri olan fetch hadisəsinə reaksiya nümunəsi göstərilmişdir:

self.addEventListener('fetch', function(event){

event.respondWith(

caches.match(event.request)

.then(function(response) {

// Cache hit - return response

if (response) {

return response;

}

return fetch(event.request);

})

);

});

Bu kodda, istifadəçi bir resurs istədikdə (məsələn, səhifə yükləyərkən və ya linkə kliklədikdə), Service Worker fetch hadisəsini yaxalayır və həmin sorğunun önbellekdə olub-olmadığını caches.match() ilə yoxlayır. Əgər önbellekdə uyğun cavab tapılarsa, həmin cavab birbaşa istifadəçiyə qaytarılır. Əgər tapılmazsa, sorğu normal şəkildə internetdən (fetch) həyata keçirilir.

Listing 5-in göstərdiyi bu davranış, Service Worker-in offline rejimdə necə məzmun təqdim edə bildiyini əyani şəkildə nümayiş etdirir. Əlavə olaraq, bu yanaşma istifadəçi təcrübəsini gücləndirir, çünki brauzer resursları hər dəfə internetdən deyil, öncə cache-dən yoxlayır və bu da tətbiqin daha sürətli və dayanıqlı işləməsinə səbəb olur.

Qeyd edildiyi kimi, Service Worker hadisələr əsasında işləyən bir texnologiyadır və veb səhifədən tamamilə ayrı bir həyat dövriyyəsinə malikdir. İstifadə olunmadıqda, brauzer tərəfindən yaddaş istifadəsini optimallaşdırmaq üçün avtomatik olaraq dayandırıla bilər. Lakin ehtiyac olduqda yenidən aktivləşir və fetch, sync və push hadisələrini idarə etməyə davam edir. Bu da onu müasir veb tətbiqlər üçün son dərəcə çevik və funksional bir texnologiyaya çevirir.

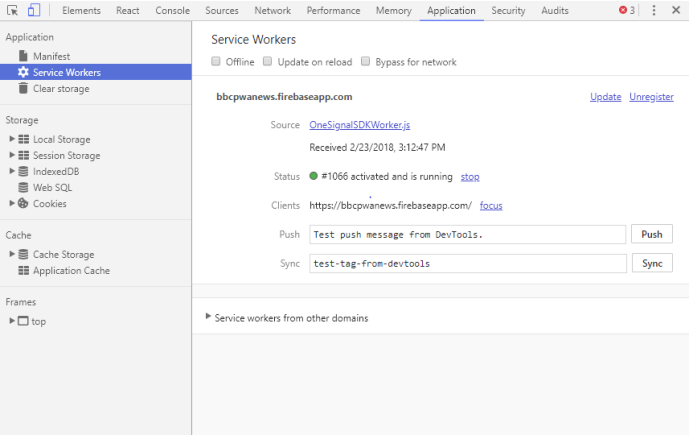
Bütün mərhələlər yəni qeydiyyat (registration), quraşdırılma (install), aktivləşmə (activate) və hadisələrin idarə olunması (event handling) uğurla tamamlandıqdan sonra, Service Worker-in brauzer tərəfindən düzgün işləyib-işləmədiyini əl ilə yoxlamaq mümkündür.

Bunun üçün Google Chrome brauzerində yerləşən DevTools (İnkişaf Alətləri) istifadə olunur. DevTools daxilində Application panelinə keçərək Service Workers adlı xüsusi tab seçilir. Məhz burada Service Worker-in statusu, aktiv olub-olmadığı, cache-lərin vəziyyəti, push bildirişləri, background sync və digər xüsusiyyətləri izlənə bilər.

Bu yoxlama prosesi həm tətbiqin inkişaf mərhələsində, həm də canlı istifadəyə təqdim olunmazdan əvvəl test və diaqnostika üçün çox vacibdir. Həmçinin burada Service Worker-i əl ilə yenidən qeydiyyata almaq (update), dayandırmaq (unregister) və ya brauzer önbelleyini təmizləmək (clear storage) kimi əməliyyatlar da yerinə yetirilə bilər.

Bu prosesin necə aparıldığı isə Şəkil 10-da əyani şəkildə göstərilmişdir. Bu vizual təqdimat vasitəsilə inkişaf etdiricilər Service Worker-in işləmə mexanizmini daha yaxşı başa düşə və tətbiqlərində düzgün tətbiq etdiklərini əminliklə yoxlaya bilərlər.

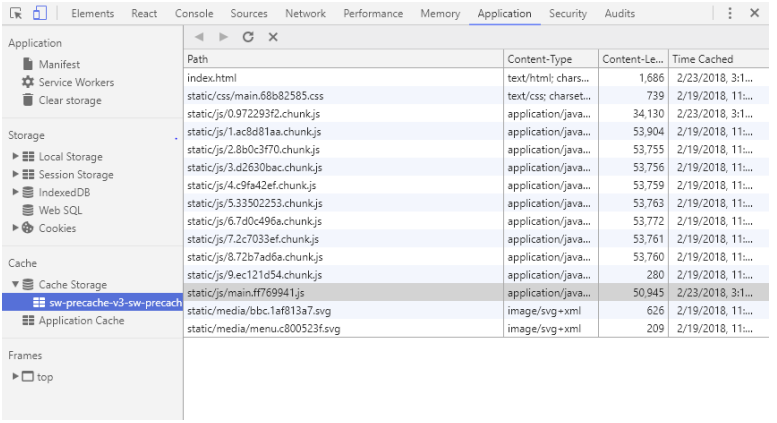
**Şəkil 1.13** Chrome Developer Tools-da Service Worker qeydiyyatının göstərilməsi.



**Mənbə:** Müəllif tərəfindən hazırlanıb.

Şəkil 1.13-də göstərildiyi kimi, qeydiyyatdan keçmiş Service Worker, səhifə yenidən yükləndikdə (reload zamanı) avtomatik olaraq yenilənir. Bu xüsusiyyət, tətbiqin daha təzə versiyasını istifadəçiyə təqdim etmək və mümkün dəyişiklikləri real vaxtda tətbiq etmək üçün son dərəcə vacibdir.

Service Worker-in bu dinamik davranışı, veb tətbiqin offline rejimdə təcrübəsini və performansını xeyli artırır. Belə ki, tətbiq internet bağlantısı olmadan belə, yalnız əsas xam məlumatları (raw data) serverdən tələb edir. Stil (style) və görünüş (layout) kimi vizual elementlər isə artıq keşlənmiş (cached) formada cihazda saxlandığı üçün təkrar server sorğusuna ehtiyac qalmır. Nəticədə, səhifə daha tez yüklənir və istifadəçi üçün daha axıcı bir təcrübə yaranır.

**Şəkil 1.14** Chrome Developer Tools-da Service Worker-in xam (raw) və önbelleklənmiş (cached) məlumatları əldə etmə prosesi.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu optimallaşdırma prosesinin necə həyata keçirildiyi isə Şəkil 1.14-də vizual olaraq göstərilmişdir. Bu şəkildə Service Worker-in xam məlumatla işləmə prinsipi, cache-də saxlanmış dizayn və strukturu istifadə etməklə necə yüksək performanslı və çevik bir istifadə təcrübəsi yaratdığı ətraflı təsvir olunur.

Şəkil 1.14-də göstərildiyi kimi, veb tətbiqin görünüşü (layout), tərtibatı (style) və digər statik məzmunu Service Worker vasitəsilə keşlənmiş (cached) məlumatlardan əldə olunur. Bu isə səhifənin tez yüklənməsini təmin edir və istifadəçiyə daha axıcı bir təcrübə təqdim edir.

Eyni zamanda, dinamik məzmun, yəni istifadəçiyə hər dəfə fərqli şəkildə təqdim olunan məlumatlar (məsələn, xəbər başlıqları, hava proqnozu, istifadəçi məlumatları və s.) isə birbaşa şəbəkədən (network) alınır. Bu yanaşma həm məzmunun aktual qalmasını, həm də sistemin performansının yüksəlməsini təmin edir.

Beləliklə, Service Worker veb tətbiqin həm offline işləməsini, həm də real vaxtda dinamik məzmun təqdim etməsini mümkün edir və bu da Progressive Web App-ların əsas üstünlüklərindən biri hesab olunur.

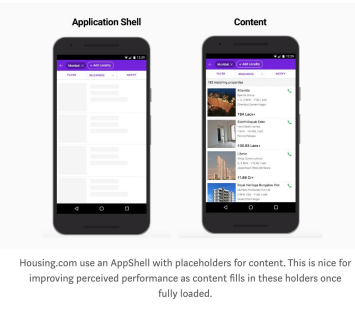
# II FƏSIL. PROGRESSİVE WEB APPLİCATİON ƏSASINDA PROTOTİPİN HAZIRLANMASI VƏ TEXNOLOJİ ƏSASLANDIRILMASI

## 2.1 Tətbiqin arxitekturası və texnoloji komponentləri

Application shell (tətbiq strukturu) modeli, Progressive Web App (PWA) üçün minimum HTML, CSS və JavaScript fayllarından ibarət olan əsas istifadəçi interfeysini (UI) qurmaq üçün istifadə olunur. Bu model PWA-nın tez yüklənməsi, sürətli başlanğıcı, axıcı istifadəçi təcrübəsi (UX) və təkrar ziyarətlərdə yüksək performans təmin etməsində mühüm rol oynayır.

App shell arxitekturasının əsas ideyası ondan ibarətdir ki, tətbiqin sabit struktur hissələri yəni başlıq, menyu, naviqasiya paneli, fon dizaynı və digər tez-tez dəyişməyən elementlər ilk istifadə zamanı şəbəkə üzərindən yüklənir və cihazın yaddaşına (cache-ə) yazılır. Bundan sonra, istifadəçi tətbiqi növbəti dəfə açdıqda bu skelet birbaşa lokal yaddaşdan yüklənir, bu da tətbiqin çox sürətli açılmasına səbəb olur.

Bu modelin əsas üstünlüklərindən biri də tətbiqin struktur və məzmun hissələrinin bir-birindən ayrılmasıdır. Beləliklə, shell (sabit hissələr) və content (dinamik məzmun) ayrı-ayrılıqda idarə olunur. Shell faylları bir dəfə yüklənir və uzun müddət istifadə olunur, dinamik məzmun isə istəyə uyğun olaraq şəbəkədən alınır və göstərilir. Bu yanaşma tətbiqin həm offline işləməsini, həm də aktuallığını qorumağa imkan verir.

**Şəkil 2.1** App-shell arxitekturasında skelet (shell) və məzmunun (content) təsviri.

**Mənbə:** https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-3-offline-support-and-network-resilience-c84db889162c

Şəkil 2.1-də göstərildiyi kimi, tətbiq skeleti (shell) PWA-nın istifadəçi interfeysini (UI) qurmaq üçün lazım olan statik hissələri özündə birləşdirir. Bu skelet interfeys elementlərini lokal olaraq saxlayır, məzmun isə daha sonra dinamik şəkildə API vasitəsilə şəbəkədən alınır. Bu yanaşma tətbiqin hətta internet bağlantısı olmadıqda belə ən azı tanış bir interfeys göstərməsinə şərait yaradır. Beləliklə, istifadəçi boş bir ekranla və ya "bağlantı yoxdur" kimi standart mesajla qarşılaşmır, əksinə, tanıdığı vizual strukturu görür və tətbiqin işlədiyini hiss edir.

Tətbiq skeleti əslində nativ tətbiq mağazalarına göndərilən kod paketinə (app bundle) bənzəyir. Bu skelet tətbiqin işləməyə başlaması üçün lazım olan ən əsas komponentləri özündə saxlayır, lakin heç bir dinamik məlumat və ya kontent daxilində yerləşmir. Məzmun yalnız bu skeletin içində yerləşdirilir və vizual strukturun içində təqdim olunur. Tətbiqin növündən və məqsədindən asılı olaraq, bu məzmun da əlavə olaraq cache-lənə bilər.

Tətbiq skeleti hazırlayarkən nəzərə alınmalı olan bəzi əsas dizayn prinsipləri mövcuddur:

Ekranda dərhal görünməli olanlar nədir? Tətbiq ilk dəfə açıldıqda istifadəçinin gözləmədən görməli olduğu əsas UI elementləri müəyyən edilməlidir.

Tətbiq üçün mütləq olan UI komponentləri və plaginlər hansılardır? Bunlar tətbiqin işləməsi üçün ilkin mərhələdə lazımlı olan interfeys funksiyalarıdır.

App shell üçün əlavə resurslar hansılardır? Məsələn: şəkillər, stillər (CSS), JavaScript faylları və digər yardımçı komponentlər.

Bu suallara cavab verərək dizayn olunan app-shell arxitekturası, Progressive Web App-ın həm offline dəstəyi, həm də performans və istifadəçi təcrübəsi baxımından effektivliyini təmin edir (Gorniak & Poole, 2013).

Push bildirişləri, Progressive Web App-ların (PWA) ən vacib komponentlərindən biridir. Bu funksiya uzun müddət veb tətbiqlərdə mövcud deyildi və bu səbəbdən nativ tətbiqlərlə veb tətbiqlər arasında əsas fərqlərdən biri olaraq qalırdı. Push bildirişlərinin vebə əlavə olunması ilə bu boşluq əhəmiyyətli dərəcədə azaldılmışdır.

Bildiriş (notification) istifadəçinin cihazında ani olaraq görünən bir mesajdır. Bu mesaj ya tətbiq açıq olanda lokal şəkildə, ya da server tərəfindən tətbiq bağlı vəziyyətdə olanda göndərilə bilər. Push bildirişləri istifadəçiyə məzmun və məlumatlarla bağlı vaxtında yeniliklər təqdim edir. Ən əsası isə, istifadəçi bu bildirişlərə öz istəyi ilə abunə olur (opt-in). Bu mexanizm istifadəçini tətbiqə yenidən cəlb etmək və onun tətbiqdən istifadəsini davam etdirmək baxımından çox mühüm rol oynayır (Zhang, Li, & Wang, 2019).

Push bildirişlərinin əsas funksionallığı iki API vasitəsilə təmin olunur:

Notification API

Push API

Notification API, bir veb səhifəyə və ya Service Worker-ə, cihazın istifadəçi interfeysində – yəni brauzerdən kənarda, sistem səviyyəsində – bildiriş yaratmağa və göstərməyə imkan verir. Bu bildirişlər brauzer tab-larından və ya pəncərələrindən asılı deyil, yəni istifadəçi artıq saytı tərk etmiş olsa belə, bildiriş göstərilə bilər. Məsələn, istifadəçi səhifəni bağlayıb telefonunda başqa iş görsə belə, yeni xəbər və ya yeniləmə barədə bildiriş ala bilər.

Bildirişləri göstərmək üçün istifadəçidən icazə almaq məcburidir. Bu, Notification API vasitəsilə aşağıdakı kimi həyata keçirilir:

Notification.requestPermission(function(status) {

console.log('Notification permission status:', status);

});

Bu kod parçası brauzerdə bildiriş göstərmək üçün istifadəçidən icazə tələb edir. Əgər istifadəçi icazə verərsə, tətbiq ona istənilən vaxt bildiriş göndərə bilər. Əgər icazə verilməzsə, bildiriş göstərmək mümkün olmur.

Bu hadisə baş verdikdə, yəni bu sorğu işə düşdükdə, Şəkil 13-də göstərildiyi kimi brauzer istifadəçiyə vizual şəkildə icazə pəncərəsi göstərir və onun razılığı soruşulur. Bu, PWA-ların istifadəçi ilə daimi əlaqə qurmaq imkanını təmin edən əsas addımlardan biridir.

Şəkil 2.2-də göstərildiyi kimi, bildirişlərin göstərilə bilməsi üçün istifadəçinin icazəsi tələb olunur. İstifadəçi bu icazəni verdikdən sonra, Service Worker vasitəsilə sistem səviyyəsində bildiriş göstərmək mümkündür.

Bu prosesin həyata keçirilməsi üçün showNotification() funksiyası istifadə olunur və bu funksiya aşağıda Listing 6-da göstərilmişdir:

function displayNotification() {

if (Notification.permission == 'granted') {

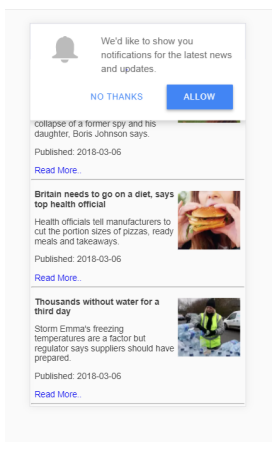
navigator.serviceWorker.getRegistration().then(function(reg) {

reg.showNotification('Hello world!');

});

}

}

**Şəkil 2.2** Push Bildirişləri üçün İcazə Sorğusu Pəncərəsi

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu bildiriş brauzerdən kənarda, sistem səviyyəsində görünür və istifadəçinin diqqətini tətbiqə cəlb etmək üçün effektiv vasitədir. Bu tip bildirişlər istifadəçi tətbiqdən çıxsa və ya saytı bağlasa belə göstərilə bilər ki, bu da PWA-ların yenidən istifadəçi cəlb etməsi (re-engagement) baxımından böyük üstünlük yaradır.

Push API server tərəfindən istifadəçiyə bildiriş göndərmək imkanı yaradan bir veb texnologiyadır. Bu API sayəsində brauzerin istifadə etdiyi push xidməti (push service) vasitəsilə serverdən müştəriyə mesajlar göndərilə bilər. Bu mesajlar veb tətbiq açıq olmasa belə istifadəçiyə çatdırıla bilər və bu da onu nativ tətbiqlərə bənzər edən əsas xüsusiyyətlərdən biridir.

Push API, Service Worker ilə birlikdə işləyir. Yəni, Service Worker brauzer tərəfindən qeydiyyatdan keçdikdə, brauzerin daxili push xidməti bu worker-i tanıyır və ona push hadisələri (push events) göndərə bilir. Hər dəfə server yeni bir bildiriş göndərdikdə:

Service Worker push hadisəsini qəbul edir,

Bu hadisə içərisindəki məlumatı (məzmunu) əldə edir,

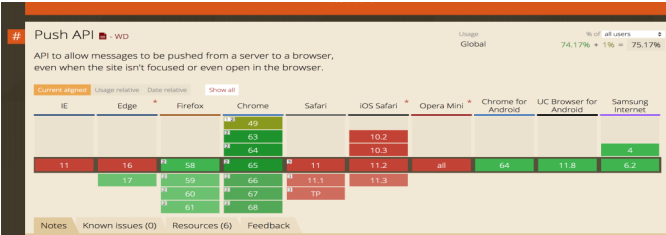
Daha sonra həmin məlumatı istifadəçiyə bildiriş şəklində göstərmək üçün Notification API-dən istifadə edir.

Beləliklə, Push API və Notification API birlikdə çalışaraq, istifadəçi tətbiqdən kənarda olsa belə ona real vaxtda yenilik və ya xəbərdarlıq göndərmək imkanı yaradır. Bu, e-xəbər portalları, sosial şəbəkələr, mesajlaşma sistemləri və elektron ticarət saytları üçün olduqca faydalıdır.

Lakin, Push API-nin tətbiqində də brauzer uyğunluğu problemi mövcuddur. Yəni, bütün brauzerlər bu texnologiyanı tam dəstəkləmir və bəzi platformalarda məhdudiyyətlər var. Bu dəstəklə bağlı vəziyyətin necə olduğu Şəkil 2.3-də vizual şəkildə təqdim olunmuşdur. Bu şəkildə, hansı brauzerlərin Push API-ni dəstəklədiyi, hansılarının isə bu funksiyanı hələ tam şəkildə dəstəkləmədiyi aydın görünür. Bu məlumatlar, tətbiqin daha geniş auditoriyada işləməsi üçün hədəf brauzer seçimi baxımından inkişaf etdiricilər üçün olduqca vacibdir.

Şəkil 2.3-də göstərildiyi kimi, Chrome, Chrome for Android, Firefox, Opera və Edge brauzerləri Push bildirişlərini (Push API) tam şəkildə dəstəkləyir. Bu brauzerlərdə istifadəçilər, tətbiqdən çıxmış olsalar belə, serverdən gələn real vaxt bildirişlərini qəbul edə bilirlər. Bu xüsusiyyət, Progressive Web App-ların istifadəçi ilə davamlı əlaqə saxlamaq imkanını genişləndirir və onların nativ tətbiqlərə daha da yaxınlaşmasını təmin edir.

Lakin Safari brauzeri xüsusilə də iOS Safari hələ də standart Push API-ni dəstəkləmir. Bu səbəbdən, Safari istifadəçilərinə bildiriş göndərmək üçün fərqli texnologiyadan, yəni Safari-yə məxsus özəl bildiriş API-sindən istifadə olunmalıdır. Apple-ın bu özünəməxsus yanaşması, veb tətbiqlərdə Safari üçün əlavə konfiqurasiya tələb edir və bu da inkişaf etdiricilər üçün əlavə iş yükü yarada bilər.

**Şəkil 2.3** Brauzerlərdə Push bildirişləri üçün dəstək vəziyyəti.

**Mənbə:** <https://www.theseus.fi/bitstream/handle/10024/142997/PWA thesis.pdf>

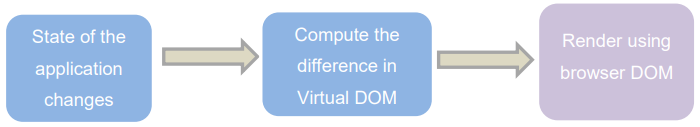
Bu vəziyyət göstərir ki, veb tətbiqlərdə push bildirişlərinin brauzerlər üzrə dəstəklənməsi fərqlidir və hər platforma üçün uyğun strategiya seçmək vacibdir. Xüsusilə geniş auditoriyaya xidmət edən tətbiqlər üçün çoxbrauzerli dəstək əsas texniki meyarlardan biri kimi nəzərə alınmalıdır.

## 2.2 Progressive Web Application prototipində interfeys dizaynı və funksional elementlərin inteqrasiyası

Bu bölmədə, bu tədqiqat layihəsinin icrasında istifadə olunan əsas texnologiyalar ətraflı şəkildə izah edilir. Layihənin əsas məqsədi sürətli, çevik və istifadəçi yönümlü bir Progressive Web App hazırlamaq olduğundan, React.js, ES6, Material UI, Webpack, Babel və Lighthouse kimi müasir və effektiv alətlər seçilmişdir. Bu texnologiyalar həm konfiqurasiya prosesini sadələşdirmək, həm də təkrarlanan (boilerplate) kod yazımını azaltmaq məqsədilə diqqətlə seçilmişdir.

React komponent əsaslı struktura malikdir. Bu isə inkişaf etdiricilərə yenidən istifadə oluna bilən və modullar şəklində qurulan interfeyslər yaratmaq imkanı verir. React həmçinin tək səhifəlik tətbiqlərin (SPA – Single Page Application) hazırlanmasında geniş istifadə olunur və bu xüsusiyyət onu Progressive Web App-ların yaradılması üçün ideal texnologiyaya çevirir.

Şəkil 15-də isə React-in virtual DOM vasitəsilə necə sürətli performans göstərdiyi və istifadəçi ilə daha interaktiv təcrübə yaratdığı əyani şəkildə göstərilmişdir.

**Şəkil 2.4** React-də Virtual DOM vasitəsilə render (təsvir) prosesi.

**Mənbə:** https://images.app.goo.gl/FMn4H4xVZG3tomht6

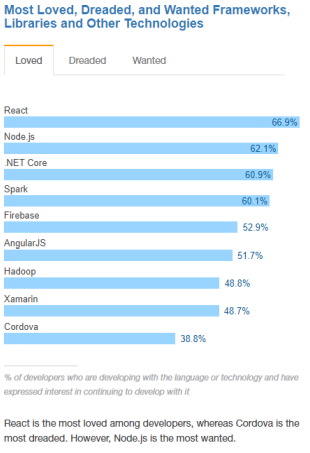
Şəkil 2.4-də göstərildiyi kimi, React yaddaşda saxladığı Virtual DOM vasitəsilə tətbiqin vəziyyətini (UI state) sürətli və səmərəli şəkildə izləyir. React-in yaradılma səbəbi də məhz budur vaxt keçdikcə dəyişən məlumatlara sahib olan böyük miqyaslı tətbiqləri daha səmərəli şəkildə qurmaq.

React bu problemi həll etmək üçün “Deklarativ”, “Komponent əsaslı”, “Bir dəfə öyrən, hər yerdə yaz” (Learn Once, Write Anywhere) prinsipinə əsaslanır. Bu yanaşma sayəsində interaktiv istifadəçi interfeyslərini qurmaq olduqca sadə hala gəlir. Tətbiqin hər vəziyyəti (state) üçün sadə və vizual olaraq təmiz görünən view-lar dizayn olunur, məlumat dəyişdikcə isə React yalnız dəyişən komponentləri render edir — yəni yenidən qurur. Bu da performans baxımından ciddi üstünlük yaradır.

React yalnız veb interfeyslərin qurulmasında deyil, həm də server tərəfində (backend) və mobil tətbiqlərin hazırlanmasında geniş şəkildə istifadə olunur. Server tərəfində Node.js ilə birlikdə istifadə edilərək React komponentlərinin serverdə işlənməsi və daha sürətli ilkin yükləmə (server-side rendering – SSR) mümkündür. Bundan əlavə, React-in mobil versiyası olan React Native, təkcə veb deyil, nativ mobil tətbiqlərin (Android və iOS) də yazılmasına imkan verir. Bu isə inkişaf etdiricilərə bir dil – JavaScript ilə çox platformalı tətbiqlər hazırlamağa şərait yaradır.

Bütün bu xüsusiyyətlər bir araya gəldikdə, React-i digər veb çərçivə və kitabxanalar arasında ən sevilən texnologiyalardan birinə çevirir. React-in istifadəsi və populyarlığı, onu AngularJS, Angular 2 və Ember kimi güclü rəqiblərini cəmi üç il ərzində geridə qoymağa imkan verib. Bu sürətli yüksəliş və texnologiya dünyasında gördüyü qəbul Şəkil 2.5-də statistik olaraq əyani şəkildə göstərilmişdir.

**Şəkil 2.5** 2025-ci ildə veb development sahəsində ən çox sevilən texnologiyalar arasında



**Mənbə:** https://images.app.goo.gl/7MbymbnwwNoLrE9v9

Komponentlər JavaScript funksiyalarına bənzəyir və istifadəçi interfeysini (UI) müstəqil və yenidən istifadə oluna bilən hissələrə bölməyə imkan verir. Komponentlər React tətbiqlərinin əsas tikinti bloklarıdır. Müasir React-də komponentlər adətən funksional komponentlər şəklində və React Hook-ları, xüsusilə useState istifadə olunmaqla yazılır.

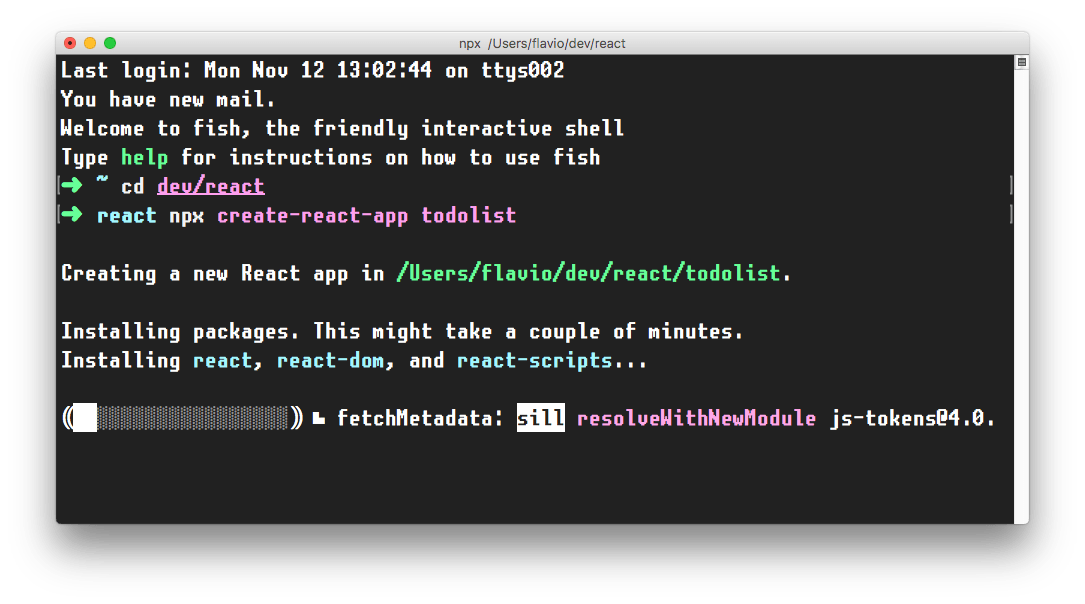
Komponentlər props və state adlı verilənləri qəbul edirlər. Props valideyn komponentlə övlad komponent arasında məlumat ötürmək üçün körpü rolunu oynayır. Props dəyişməz, statik məlumatları ötürmək üçün istifadə olunur. State isə dinamik məlumatları idarə etmək üçün istifadə edilir və yalnız həmin komponentə aiddir, başqa komponentlər onu dəyişə bilməz. Bu yanaşma müasir React proqramlaşdırmasında ən çox istifadə olunan metod hesab olunur və komponentlərin daha sadə, aydın və çevik şəkildə qurulmasına imkan verir.

Create React App React.js tətbiqi qurmağa başlamaq üçün istifadə olunan rəsmi və geniş yayılmış vasitədir. Bu alət, React tətbiqlərini hazır konfiqurasiya ilə təqdim edir və proqramçıya Webpack, Babel və digər texnologiyaları ayrıca qurmadan birbaşa inkişaf prosesinə başlamağa imkan verir.

Create React App, həmçinin Service Worker ilə birlikdə gəlir (yalnız istehsal rejimində aktivdir), bu da Progressive Web App (PWA) tətbiqləri üçün vacib komponentlərdən biridir.

Müasir yanaşmaya əsasən, Create React App layihəsini başlatmaq üçün aşağıdakı əmr istifadə olunur:

npx create-react-app todolist

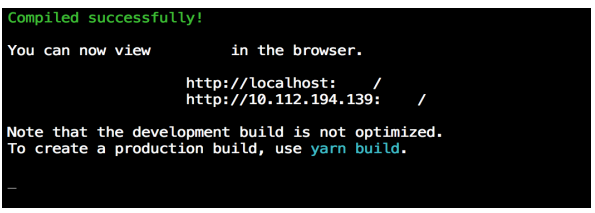
**Şəkil 2.6** React.js tətbiqinin qurulması

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu əmr vasitəsilə todolist adlı yeni React layihəsi yaradılır və bütün vacib mühit avtomatik qurulur. Bu əmrin necə istifadə edildiyi və terminalda necə göründüyü Şəkil 17-də əyani şəkildə təqdim olunmuşdur.

Şəkil 2.7-də göstərildiyi kimi, create-react-app pwanews əmri ~/pwanews qovluğunda yeni bir React tətbiqi yaradır. Bu qovluğa keçərək (cd pwanews) və npm start əmrini icra etməklə, React skripti işə düşür və inkişaf serveri (development server) başladılır.

**Şəkil 2.7** React tətbiqinin kompilyasiyası

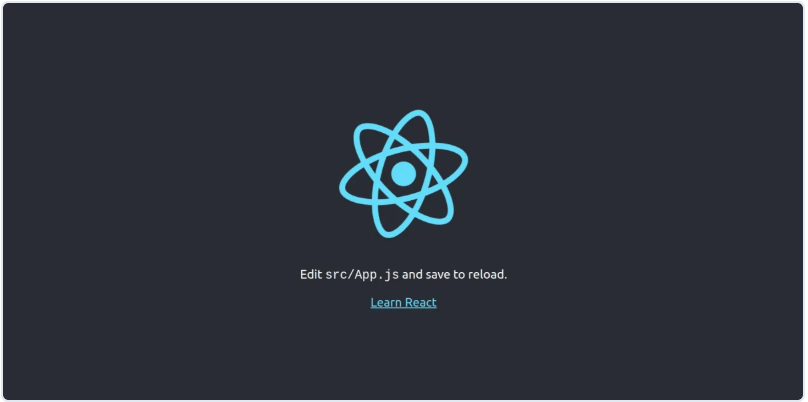


**Mənbə:** Müəllif tərəfindən hazırlanıb.

Şəkil 2.8-də göstərildiyi kimi, create-react-app uğurla kompilyasiya olunduqdan sonra, tətbiqə [**http://localhost:3000**](http://localhost:3000) ünvanına daxil olmaqla brauzerdə baxmaq mümkündür.

Webpack müasir JavaScript tətbiqləri üçün hazırlanmış statik modul yığıcı (module bundler) rolunu oynayan alətdir. Günümüzdə veb tətbiqləri yalnız bir HTML və bir JavaScript faylından ibarət olmur; əksinə, onlar çoxsaylı modullar, komponentlər, CSS/SCSS stil faylları, şəkillər, şriftlər və digər yardımçı resurslardan ibarət olur. Bu resurslar tətbiqin inkişafı zamanı işləri modullaşdırmaq və funksional parçalamaq baxımından çox faydalıdır.

**Şəkil 2.8** React tətbiqinin ilkin görünüşü



**Mənbə:** Müəllif tərəfindən hazırlanıb.

Lakin bu çoxsaylı faylların deploy zamanı bir yerə yığılması və idarə olunması çətinlik yaradır. Burada Webpack dövriyyəyə daxil olur. Webpack tətbiqin bütün resurslarını analiz edir, asılılıqları (dependencies) müəyyənləşdirir və onları bir və ya bir neçə bundle halında yığır. Bu proses zamanı Webpack, tətbiqdə istifadə olunan hər bir modul və faylı təyin edir və onları əlaqələndirilmiş şəkildə kompakt paket halına salır.

Babel JavaScript üçün hazırlanmış güclü və geniş istifadə olunan kompilyator və transpilyator alətidir. Onun əsas vəzifəsi proqramçı tərəfindən yazılmış müasir JavaScript kodlarını (məsələn ES6, ES7, JSX və s.) köhnə brauzerlər tərəfindən başa düşülə bilən ES5 versiyasına çevirməkdir.

Babel və Webpack birlikdə işləyərək, inkişaf mərhələsində yazılmış müasir və zəngin sintaksisli kodun, brauzer səviyyəsində stabil və performanslı işləməsinə şərait yaradır. Bu isə geniş auditoriyaya xidmət edən tətbiqlər üçün vacibdir.

RSS to JSON - Bu layihədə istifadə olunan vacib onlayn xidmətlərdən biri də RSS to JSON çevirmə alətidir. Ənənəvi RSS feed-lər, əsasən XML formatında təqdim olunur və bu formatın oxunması, xüsusilə frontend tətbiqlərində birbaşa emal üçün əlverişli deyil.

RSS to JSON xidməti vasitəsilə XML formatlı RSS məlumatları, REST API şəklində təqdim olunan JSON formatına çevrilir. JSON JavaScript ilə tam uyğun olan, yüngül və strukturlaşdırılmış məlumat formatıdır. React və digər frontend texnologiyalarında bu məlumat formatı asanca oxunur, təhlil edilir və göstərilir.

Layihə çərçivəsində hazırlanan PWA News App tətbiqi, xəbər mənbələrini məhz bu xidmət vasitəsilə əldə edir. Bu xidmət istifadə olunduğuna görə, tətbiq yalnız frontend səviyyəsində fəaliyyət göstərir və ayrıca bir backend serverə ehtiyac qalmır.

Cloudinary veb və mobil proqram tərtibatçıları üçün hazırlanmış bulud əsaslı media idarəetmə platformasıdır. Bu platforma vasitəsilə şəkil və videoların idarə olunması, optimallaşdırılması və dinamik şəkildə təqdim olunması mümkündür. Cloudinary istifadəçilərə yüksək keyfiyyətli media fayllarını minimum ölçüdə və maksimum performansla təqdim etmək imkanı yaradır.

Layihə çərçivəsində hazırlanan Progressive Web App-da Cloudinary əsasən şəkillərin ölçüsünü optimallaşdırmaq üçün istifadə olunmuşdur. PWA-larda istifadəçi təcrübəsinin əsas göstəricilərindən biri səhifənin tez yüklənməsi və yüngül olmasıdır. Şəkil faylları tətbiqin yükləmə müddətinə birbaşa təsir etdiyindən, Cloudinary bu şəkilləri avtomatik olaraq ölçüsünə, ekran ölçüsünə və cihaz növünə uyğun şəkildə sıxaraq (compress və resize edərək) təqdim edir.

Bu optimallaşdırma nəticəsində tətbiqin ümumi performansı artmış, istifadəçilər şəkillərin daha tez və axıcı yükləndiyini hiss etmişlər. Cloudinary həmçinin CDN (Content Delivery Network) infrastrukturu ilə dəstəkləndiyi üçün şəkillər coğrafi cəhətdən istifadəçiyə ən yaxın serverdən təqdim olunur və bu da əlavə sürət təmin edir.

Bu fəsildə bu tədqiqat çərçivəsində hazırlanmış PWA News App prototipinin həyata keçirilmə prosesi təsvir olunur. Tətbiqin əsas məqsədi Progressive Web App texnologiyasının üstünlüklərini və funksional imkanlarını əyani şəkildə nümayiş etdirməkdir.

Hazırlanan tətbiq PWA-nın bütün əsas komponentlərini özündə birləşdirir:

Service Worker – offline rejim və resursların önbelleklənməsi üçün,

Web App Manifest – tətbiqin cihazın ana ekranına əlavə olunması üçün,

App Shell – sürətli yükləmə və dayanıqlı interfeys strukturu üçün,

Web Push Notification – istifadəçini yeni xəbərlər barədə real vaxtda məlumatlandırmaq üçün.

PWA News App vasitəsilə istifadəçi BBC-nin RSS xəbər lentindən aşağıdakı mövzularda məlumat ala bilər: Dünya, İdman, Texnologiya və Biznes. İstifadəçi bu xəbərləri brauzerdə oxuya, bildirişlər vasitəsilə yeniliklərdən xəbərdar ola, tətbiqi bir kliklə ana ekrana əlavə edə və offline rejimdə nativ tətbiq kimi istifadə edə bilər. Tətbiq həm UI (istifadəçi interfeysi), həm də UX (istifadəçi təcrübəsi) baxımından nativ mobil tətbiqlərə bənzər hiss yaratmaq üçün dizayn olunmuşdur.

PWA News App prototipi MacBook Pro Early 2015 modelində hazırlanmışdır. Əməliyyat sistemi olaraq macOS Sierra versiya 10.12.3 istifadə edilmişdir. Layihə üçün ilk növbədə Node.js və Create React App yüklənmiş və PWA mühiti qurulmuşdur. Bu alətlər, React tətbiqlərinin qurulması və idarə olunması üçün əsas vasitə hesab olunur.

İstifadə Olunan Əlavə Vasitələr

Visual Studio Code (VS Code)

Layihədə əsas kod redaktoru kimi Visual Studio Code seçilmişdir. VS Code – yüngül, sürətli, güclü və genişləndirilə bilən açıq mənbəli mətn redaktorudur. O, aşağıdakı xüsusiyyətlərlə inkişaf prosesini asanlaşdırır:

Daxili terminal (CLI) dəstəyi,

Git və versiya nəzarəti funksiyaları,

Avtomatik kod tamamlaması (intelliSense),

Genişləndirilmiş debug (sazlama) imkanları,

Typescript, JavaScript və Node.js üçün daxili dəstək,

C#, C++, Java, Python, PHP və Go dilləri üçün genişləndirmələr,

Unity və .NET kimi proqramlaşdırma mühitləri üçün uyğunluq.

Bu funksiyalar sayəsində VS Code inkişaf etdiricilərə həm sürətli yazım, həm strukturlaşdırma, həm də sınaq imkanlarını bir arada təqdim edir.

NPM (Node Package Manager) - Layihədə node modullarının idarəsi üçün NPM istifadə olunmuşdur. NPM – etibarlı, təhlükəsiz və sürətli paket meneceri olub, minlərlə açıq mənbəli JavaScript kitabxanalarına çıxış imkanı yaradır. Layihədə istifadə olunan bütün kitabxana və modul asılılıqları NPM vasitəsilə yüklənmişdir.

Hazırlanan tətbiq Firebase platformasında host (yerləşdirilib) olunmuşdur. Firebase – Google tərəfindən təqdim edilən və HTTPS bağlantı dəstəyi ilə etibarlı və təhlükəsiz yerləşdirmə mühiti təqdim edən bulud əsaslı platformadır. Firebase istifadə edilməsi sayəsində tətbiq istifadəçilərə hər zaman şifrələnmiş və təhlükəsiz şəkildə təqdim olunur, bu da xüsusilə PWA-lar üçün vacib şərtdir.

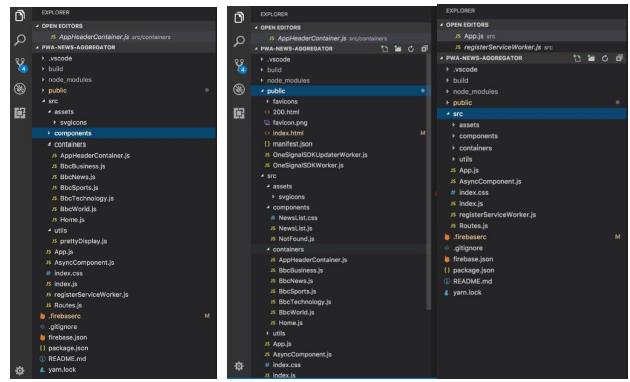
Layihədə versiya nəzarəti üçün Git istifadə edilmişdir. Git, kodun tarixçəsinin saxlanılması, geri qaytarılması və komanda ilə işləyərkən dəyişikliklərin sinxronlaşdırılması üçün geniş istifadə olunan sistemdir. Layihənin hər mərhələsi Git vasitəsilə idarə edilmiş və lazım gəldikdə əvvəlki versiyalara qayıtmaq imkanı təmin olunmuşdur.

Layihənin həyata keçirilmə mərhələsi ilk olaraq Node.js-in quraşdırılması ilə başlamışdır. Ardınca isə create-react-app vasitəsilə React mühiti qurulmuşdur. Create React App özündə artıq əvvəlcədən Babel və Webpack konfiqurasiyalarını ehtiva etdiyi üçün əlavə qurulumlara ehtiyac olmamışdır. Bu, tətbiqin quraşdırılmasını daha sürətli və problemsiz etmişdir.

Bununla yanaşı, istifadəçi interfeysinin daha cəlbedici və professional görünməsi üçün Material UI komponentləri də layihəyə əlavə olunmuşdur. Material UI, Google-un dizayn prinsiplərinə əsaslanan və React ilə asan inteqrasiya olunan geniş komponent kitabxanasıdır. Bu komponentlər həm mobil, həm də masaüstü cihazlarda respozitiv (responsive) və interaktiv interfeys yaratmaq üçün istifadə olunmuşdur.

Layihənin tətbiq axını kifayət qədər sadədir: BBC-nin RSS xəbər lentindən məlumatlar alınır və bu məlumatlar istifadəçiyə göstərilir. Bu proses real vaxtda həyata keçirilir və istifadəçi hər dəfə tətbiqi açdıqda yenilənmiş məlumatları görə bilir.

**Şəkil 2.9** React tətbiqinin qovluq strukturu



**Mənbə:** Müəllif tərəfindən hazırlanıb.

Şəkil 2.9-da PWA News App tətbiqinin strukturu vizual şəkildə təsvir olunmuşdur. Bu struktur tətbiqin modul əsaslı şəkildə necə təşkil olunduğunu, hansı komponentlərin nəyə cavabdeh olduğunu və verilənlərin tətbiqdə necə axın etdiyini göstərir. Bu yanaşma tətbiqin həm çevikliyini, həm də inkişaf etdirilməsi və saxlanılması asan olan bir memarlığa malik olmasını təmin edir.

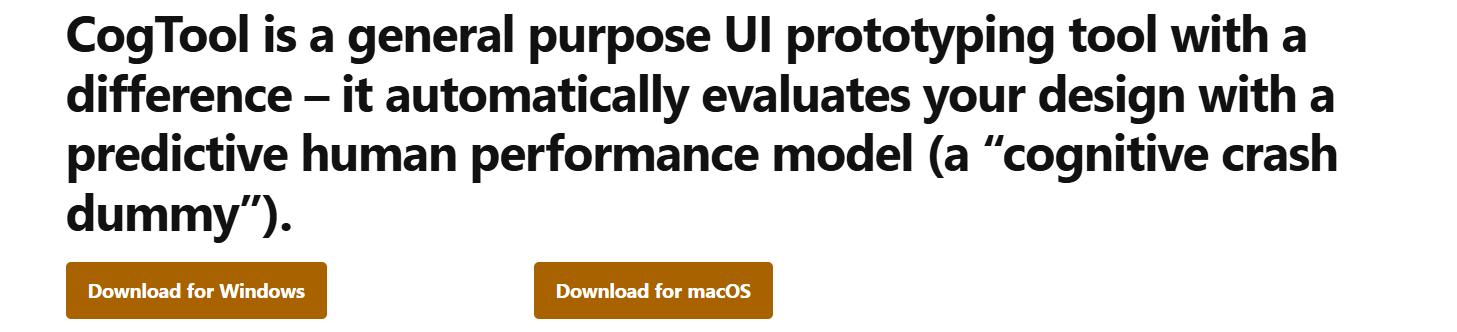
Şəkil 2.9-da göstərildiyi kimi, tətbiqin qovluq strukturu onun hazırlanması üçün lazım olan bütün faylları özündə birləşdirir. Bütün əsas fayllar PWANEWS-AGGREGATOR adlı əsas (ana) qovluq daxilində yerləşdirilmişdir. Bu struktur PWA News App layihəsinin modul əsaslı, nizamlı və inkişaf etdirilməsi asan şəkildə qurulmasını təmin edir. Həmçinin layihənin komponentləri ayrı-ayrılıqda saxlandığı üçün onların test edilməsi, dəyişdirilməsi və genişləndirilməsi də olduqca rahatdır.

# III FƏSIL. PROQRESSİV VEB TƏTBİQİN UX TƏHLİLİ

## 3.1 CogTool alətinin hazırlanması və dizaynın tətbiqə əlavə olunması

Bu fəsildə təqdim olunan yanaşmalar və əldə edilən ilkin nəticələr, növbəti fəsildə izah ediləcək tətbiqin real istifadəçilərlə sınaqdan keçirilməsi mərhələsi üçün möhkəm nəzəri və texniki əsas yaradır. İstifadəçilərin verdiyi geribildirimlər əsasında hansı dəyişikliklərin və təkmilləşdirmələrin aparıldığı barədə ətraflı məlumat həmin mərhələdə veriləcək. Lakin bu addıma keçməzdən öncə, CogTool alətinin tətbiqə necə uyğunlaşdırıldığını və bunun üçün hansı ilkin texniki tələblərin yerinə yetirilməsinin vacib olduğunu izah etmək zəruridir.

CogTool-un istifadəsinə başlanmazdan əvvəl, onun kompüterə düzgün şəkildə yüklənməsi və ilkin quraşdırma mərhələlərinin tamamlanması mütləqdir. Bu proqram Java mühitində işlədiyi üçün sistemdə Java Runtime Environment (JRE) və ya daha geniş imkanlar təqdim edən Java Development Kit (JDK) qurulmuş olmalıdır. Əgər bu komponentlər kompüterdə mövcud deyilsə, CogTool-un icrası mümkün olmayacaq və proqram başlanğıcda hər hansı bir reaksiya vermədən bağlana bilər.

**Şəkil 3.1** CogTool-un rəsmi yükləmə səhifəsinin görünüşü

**Mənbə:** https://www.cogtool.org/

CogTool-un son versiyasını əldə etmək üçün istifadəçilərə alətin rəsmi GitHub səhifəsinə və ya Carnegie Mellon Universitetinin bu layihə üçün ayrılmış rəsmi veb səhifəsinə daxil olmaq tövsiyə olunur. Hər iki mənbədə proqramın fərqli əməliyyat sistemləri – Windows, macOS və Linux – üçün uyğunlaşdırılmış versiyaları təqdim olunur. İstifadəçi öz sisteminə uyğun versiyanı seçərək, müvafiq yükləmə düyməsi vasitəsilə faylı əldə edə bilər. Yükləmə tamamlandıqdan sonra fayl arxivdən çıxarılır və əsas CogTool.jar faylı vasitəsilə proqram işə salınır. Bununla da istifadəçi proqramın əsas interfeysi ilə tanış olmağa və prototiplər yaratmağa başlaya bilər.

Yüklənmiş fayl strukturunda əsas komponent kimi CogTool.jar adlı icra edilə bilən fayl yerləşir. Bu fayl proqramın əsas hissəsini təşkil edir və onun vasitəsilə tətbiq işə düşür. İstifadəçi proqramı başlatmaq üçün iki fərqli üsuldan birini seçə bilər: birincisi, fayl üzərinə iki dəfə klik etməklə qrafik mühitdə proqramın işə salınmasıdır; ikincisi isə, komanda sətri üzərindən java -jar CogTool.jar əmrini icra etməklə proqramın işlədilməsidir. Hər iki halda nəticədə proqram eyni şəkildə açılır və istifadəçiyə interfeys təqdim olunur.

Proqram ilk dəfə açıldıqda, istifadəçi interfeysi boş vəziyyətdə görünür. Bu, hələ heç bir layihənin yaradılmadığını və ya açılmadığını göstərir. Sistemdə funksionallıqdan istifadə edə bilmək üçün istifadəçidən yeni bir layihə yaratması tələb olunur. Bu mərhələdə proqram əsas menyudan “New Project” seçimini təklif edir və istifadəçiyə prototip dizaynına başlamaq üçün uyğun mühit təqdim olunur. Layihə yaradıldıqdan sonra digər funksiyalara, məsələn, ekran əlavə etmək və UI komponentlərini yerləşdirmək imkanları aktivləşir.

**Şəkil 3.2** CogTool-da yeni layihə yaratma interfeysi



**Mənbə:** Müəllif tərəfindən hazırlanıb.

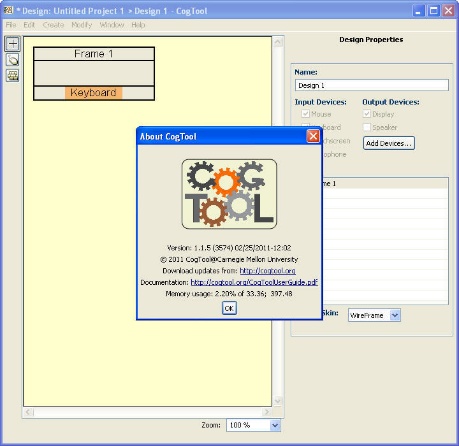
Yeni prototip dizayn etmək üçün əsas menyudan istifadəçi tərəfindən “New Project” (Yeni Layihə) düyməsi seçilir. Bu əməliyyat nəticəsində sistem yeni bir layihə başlatmaq üçün istifadəçiyə xüsusi bir forma təqdim edir. Açılan pəncərədə ilk növbədə layihənin adı daxil edilir. Layihənin adı, onun saxlanması, axtarılması və gələcəkdə digər layihələrlə fərqləndirilməsi üçün əsas identifikasiya vasitəsidir.

Layihə adı daxil edildikdən sonra istifadəçi interfeysinin ekranlarını layihəyə əlavə etmək imkanı yaranır. Bu mərhələdə istifadəçi ardıcıllıqla müxtəlif ekranlar əlavə edərək prototipin strukturunu formalaşdırır. Hər bir ekran, tətbiqin müəyyən bir funksional bölməsini əks etdirir və istifadəçi ssenarilərinin qurulmasında mühüm rol oynayır.

Ekranlara əlavə ediləcək UI (İstifadəçi İnterfeysi) komponentləri sistemin içərisində vizual şəkildə yerləşdirilir. İstifadəçi bu komponentləri – məsələn, düymələr, mətn sahələri, radio düymələr, linklər, açılan menyular və digər interaktiv elementləri – sürüklə və burax (drag and drop) üsulu ilə ekranda istədiyi yerə yerləşdirə bilər. Hər bir komponentin ölçüsü, funksiyası və etiketi istifadəçinin ehtiyacına uyğun tənzimlənə bilər.

Bu prosesin məqsədi, real istifadəçi təcrübəsini əks etdirən və proqramın gələcək interfeysinin necə görünəcəyini əvvəlcədən göstərən funksional bir prototip yaratmaqdır. Beləliklə, dizayn prosesinin ilkin mərhələsin.

CogTool-un əsas xüsusiyyətlərindən biri, istifadəçi davranış ssenarisinin “Demo” rejimində vizual şəkildə qurulması imkanını təqdim etməsidir. Bu rejim vasitəsilə istifadəçilər tətbiqin interfeysində yer alan funksiyaları necə istifadə edəcəklərini ardıcıl şəkildə modelləşdirə bilirlər. Sistem istifadəçinin hansı düyməyə klik etdiyini, hansı sahəyə mətn daxil etdiyini, hansı keçidlərdən istifadə etdiyini və ümumilikdə hansı tapşırıqları yerinə yetirdiyini vizual şəkildə qeydə alır.

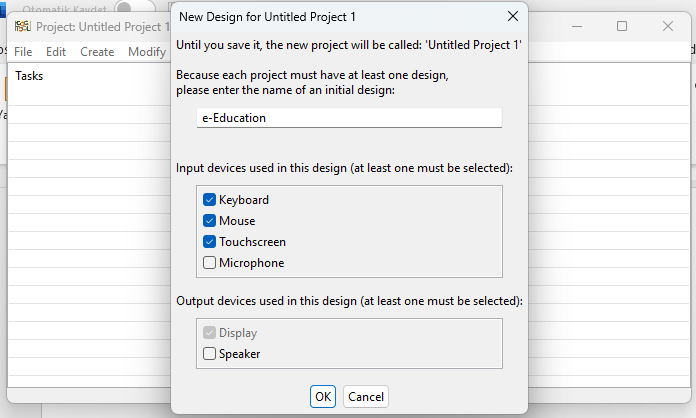
**Şəkil 3.3** Yeni layihə yaradılması interfeysi

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Demo rejimi, interaktiv və intuitiv idarəetmə ilə təchiz olunmuşdur. İstifadəçi hər bir addımı sürüklə və burax (drag-and-drop) üsulu ilə təyin edə və davranış ardıcıllığını rahatlıqla formalaşdıra bilər. Qurulan ssenari yalnız vizual olaraq nümayiş edilməklə kifayətlənmir, həm də KLM (Keystroke-Level Model) modelinə əsasən analiz edilir. Bu model insan-kompüter qarşılıqlı təsirini təhlil etməyə imkan verən tanınmış kognitiv modellərdən biridir və istifadəçinin klaviatura, siçan və gözlə izləmə hərəkətlərinə əsasən realistik zaman hesablamaları aparır.

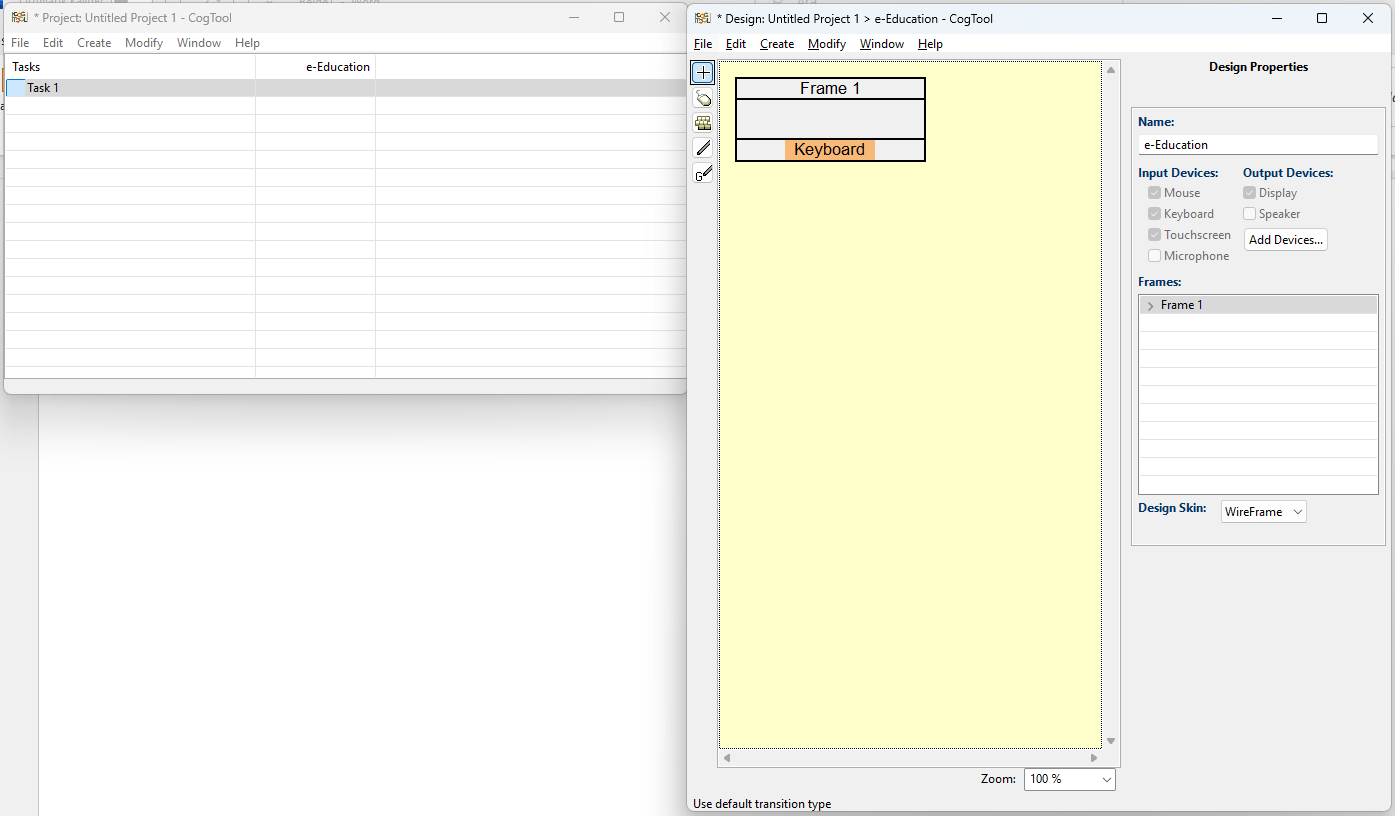
Nəticədə, istifadəçinin müəyyən bir tapşırığı yerinə yetirməsi üçün tələb olunan təxmini vaxt avtomatik olaraq hesablanır. Bu xüsusiyyət dizaynerlərə və tədqiqatçılara müxtəlif dizayn variantlarının effektivliyini müqayisə etmək, istifadəyə yararlılıq baxımından hansı interfeys elementlərinin daha optimal olduğunu müəyyənləşdirmək imkanı verir. Beləliklə, CogTool istifadəçi təcrübəsinin modelləşdirilməsi və təkmilləşdirilməsi prosesində güclü alət kimi çıxış edir.

Layihə uğurla yaradıldıqdan sonra CogTool istifadəçiyə tətbiqin interfeysini qurmaq üçün bir sıra imkanlar təqdim edir. Bu mərhələdə istifadəçi yeni ekranlar əlavə edə, bu ekranlarda müxtəlif UI (istifadəçi interfeysi) komponentlərini yerləşdirə və nəticədə funksional prototip hazırlaya bilər. Aşağıda bu prosesin detalları təqdim olunur və hər bir mərhələni vizual olaraq dəstəkləyən şəkillərlə izah edilmişdir.

**Şəkil 3.4** New Design for Untitled Project 1 pəncərəsi

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Layihə yaradıldıqdan sonra istifadəçidən ilkin dizayn üçün bir ad daxil etməsi tələb olunur (məsələn, e-Education). Eyni zamanda, bu dizaynda istifadə olunacaq giriş və çıxış qurğuları da müəyyən edilməlidir.

**Şəkil 3.5** Frame 1 və “Keyboard” komponenti ilə dizayn pəncərəsi

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu mərhələdə:

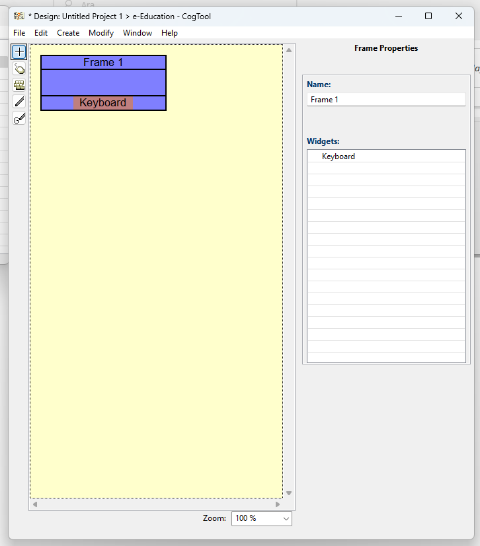
* Keyboard, Mouse, və Touchscreen kimi giriş qurğuları işarələnmişdir.
* Çıxış qurğularından isə yalnız Display istifadə edilir.

Bu seçimlər, sonrakı mərhələdə tapşırıqların və istifadəçi fəaliyyətlərinin düzgün modelləşdirilməsi üçün əsas təşkil edir.

Layihə interfeysində artıq bir əsas ekran – Frame 1 yaradılmışdır. Bu ekran üzərində “Keyboard” adlı bir UI komponenti yerləşdirilmişdir. Bu komponent vizual şəkildə ekran sahəsinə sürüklənərək əlavə olunur.

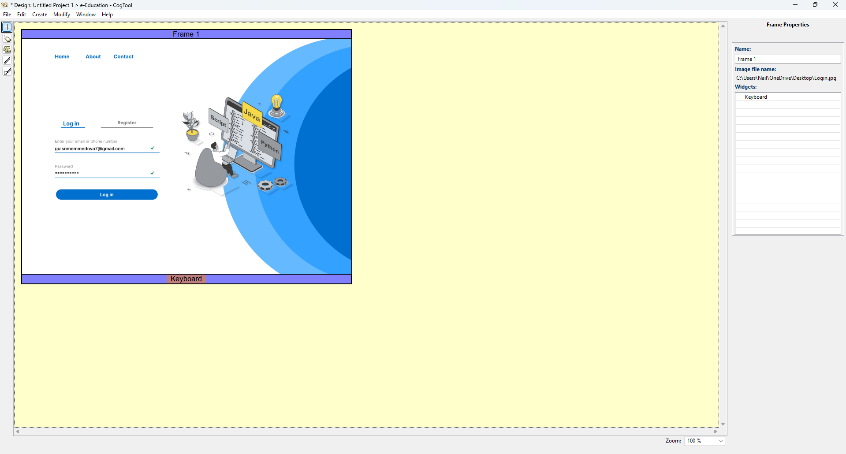
Sol tərəfdə “Tasks” panelində yeni yaradılmış tapşırıq görünür. Bu tapşırıq üzərində edilən hər bir vizual dəyişiklik istifadəçi ssenarilərinə daxil edilir və sonda simulyasiya üçün istifadə olunur.

Şəkil 3.6-da artıq əlavə olunan komponentlərin dizayna necə daxil olduğu daha aydın görünür. Sağ tərəfdə Frame Properties paneli yerləşir, burada “Frame 1” adı altında yerləşdirilmiş komponentlərin siyahısı – burada yalnız “Keyboard” göstərilir. İstifadəçi bu panel vasitəsilə komponentləri redaktə edə, adlarını dəyişdirə və ya silə bilər. Aşağı hissədə Zoom funksiyası yerləşir ki, bu da dizayn sahəsinin istənilən miqyasda baxışını təmin edir və rahat redaktəyə imkan yaradır.

**Şəkil 3.6** Frame Properties paneli ilə tam dizayn görünüşü

**Mənbə:** Müəllif tərəfindən hazırlanıb.

CogTool-un əsas üstünlüklərindən biri, istifadəçi interfeysi (UI) komponentlərinin vizual şəkildə yəni sürüklə-burax (drag-and-drop) üsulu ilə yerləşdirilməsinə imkan verməsidir. İstifadəçi ekran üzərində komponentləri intuitiv şəkildə əlavə edir, daha sonra isə bu komponentlər üzərində istifadəçi davranış ssenarilərini quraraq onların funksional ardıcıllığını təyin edir. Aşağıdakı şəkillər bu mərhələləri əyani olaraq göstərir:

**Şəkil 3.7** Login interfeysinin əlavə olunması və komponentlərin tanıdılması

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu mərhələdə, Frame 1 adlı ekrana vizual login səhifəsi fon şəkli olaraq əlavə edilmişdir (sağ paneldə “Image File name” sahəsində bu aydın görünür). Ekranda istifadəçi adı və şifrə sahələri, eyni zamanda "Log in" düyməsi yer alır.

Bu elementlər UI komponentləri kimi tanıdıldıqda, sistem onlara qarşı ediləcək hər bir istifadəçi hərəkətini simulyasiya edə bilir. Məsələn, "Keyboard" widget-ı vasitəsilə istifadəçinin məlumat daxil etməsi əməliyyatı modelləşdirilə bilər.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

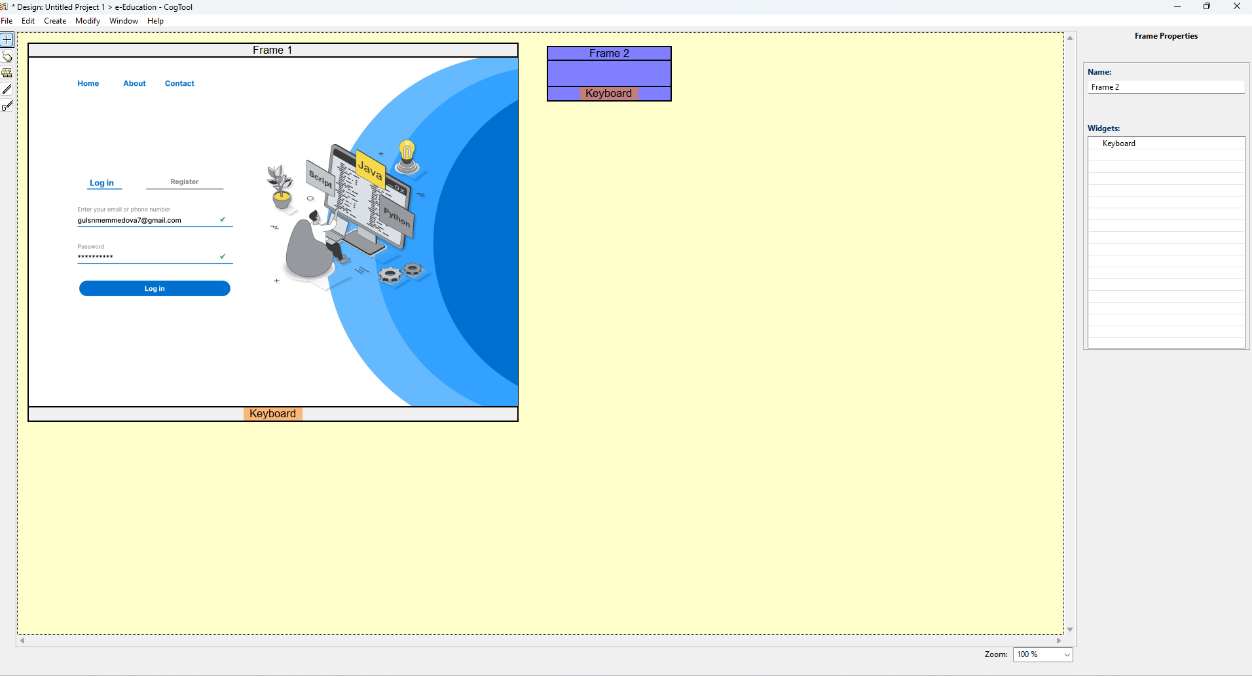
AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.8** Kontekst menyusu vasitəsilə yeni ekran əlavə olunması

**Mənbə:** Müəllif tərəfindən hazırlanıb

Şəkil 3.8-də istifadəçi ekranın üzərində sağ klik edərək kontekst menyusunu açır. Bu menyuda müxtəlif əmrlər mövcuddur:

New Frame (Ctrl+Shift+F) – yeni ekran (frame) yaratmaq

Duplicate, Zoom, Delete və digər interfeys əməliyyatları

**Şəkil 3.9** Çoxekranlı ssenarinin qurulması

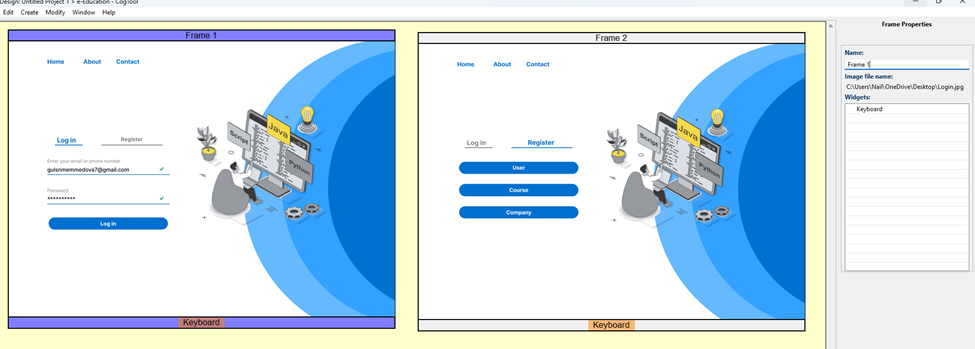
**Mənbə:** Müəllif tərəfindən hazırlanıb.

Bu imkanlar istifadəçiyə bir neçə ekran arasında keçid ssenariləri qurmaq və kompleks tətbiq axını modelləşdirmək üçün şərait yaradır. Ekranın sağ tərəfində yer alan Frame Properties panelində cari ekranda yerləşdirilmiş komponentlərin siyahısı göstərilir. Burada, əlavə edilmiş “Keyboard” komponenti görünür. Bu panel vasitəsilə komponentlərin adları dəyişdirilə, əlavə xüsusiyyətlər təyin edilə və ya yeni komponentlər əlavə edilə bilər.

Bu mərhələdə layihəyə bir neçə Frame (ekran) əlavə olunmuşdur: Frame 1, Frame 2 və RegisterChoose. Bu ekranlar istifadəçi axışına uyğun tərtib edilib və onların hər biri müəyyən addımı və interfeys halını təmsil edir.

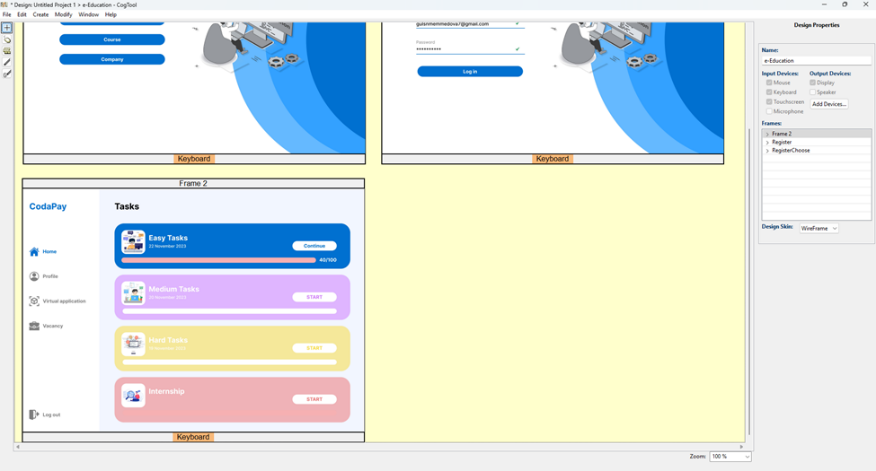
Sağ paneldə bu ekranların siyahısı görünür və hər biri üçün müxtəlif komponentlər (məsələn, "Keyboard", "User", "Course") əlavə olunub.

**Şəkil 3.10** Qeydiyyat (Register) səhifəsinə keçid



**Mənbə:** Müəllif tərəfindən hazırlanıb.

İstifadəçi əvvəlki giriş ekranından (Login) qeydiyyat səhifəsinə keçid etdikdə "Register" bölməsi aktivləşir. Bu yeni səhifədə istifadəçiyə “User”, “Course” və “Company” seçimləri təqdim olunur. Bu seçimlər interaktiv komponentlər şəklində təyin olunmuşdur və istifadəçi bu düymələrdən birinə klik etdikdə, növbəti ekrana keçid baş verir.

**Şəkil 3.11** Tapşırıqlar səhifəsi (Tasks Panel)

**Mənbə:** Müəllif tərəfindən hazırlanıb.

İstifadəçi uğurla qeydiyyatdan keçdikdən sonra sistem onu tapşırıqlar səhifəsinə yönləndirir. Burada “Easy Tasks”, “Medium Tasks”, “Hard Tasks” və “Internship” kimi bölmələr mövcuddur. Bu ekran, real bir öyrənmə platformasında istifadəçinin qarşılaşacağı interfeysin prototipidir.

Bu mərhələdə Demo rejimində komponentlər üzərində səhnələşdirilmiş istifadəçi addımları təyin edilir:

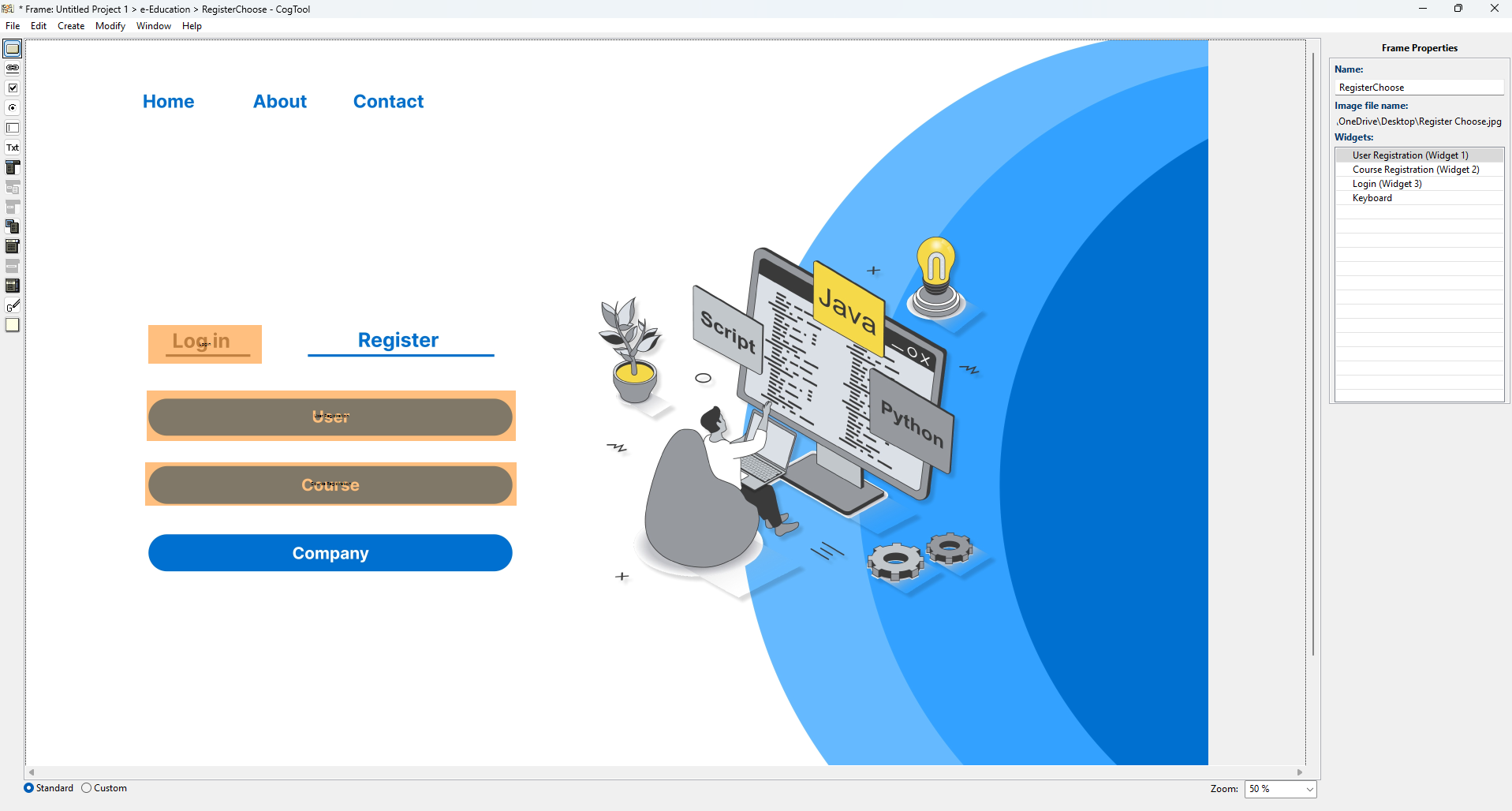
**Şəkil 3.12** a. – "User" düyməsi klik üçün işarələnmişdir.

**Şəkil 3.12** b. – "Log in", "User" və "Course" düymələri ssenariyə daxil edilib.

**Şəkil 3.12** c. – Sağ paneldə bu düymələrin Widget olaraq sistemə əlavə olunduğu görünür.

**Şəkil 3.12** Demo rejimində komponent aktivləşdirmə

|  |  |
| --- | --- |
| **3.12 a** | **3.12 b** |
|  |  |



**3.12 c**

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Sistemdə bu addımlar təyin olunduqdan sonra, Demo rejimi aktivləşdirilir və proqram ardıcıl olaraq hər bir addımı KLM modelinə əsasən simulyasiya edir. Bu zaman:

• M (mental preparation),

• K (keyboard typing),

• P (pointing with mouse),

• H (homing between devices) kimi hərəkətlər üçün standart vaxt intervalları tətbiq edilir.

## 3.2 CogTool ilə toplanmış məlumatların əsasında UX təhlilin aparılması

İstifadəçi təcrübəsinin ölçüləbilən və proqnozlaşdırıla bilən formada təhlili, istifadəçi yönümlü dizaynın əsas tərkib hissələrindən biridir. Bu kontekstdə CogTool aləti, interfeys dizaynlarının erkən mərhələlərində istifadəçi davranışlarının simulyasiyası və səmərəliliyin qiymətləndirilməsi üçün geniş istifadə olunan kognitiv modelləşdirmə vasitəsidir. Bu bölmədə əsas məqsəd, tətbiq üçün yaradılmış prototip üzərində istifadəçi ssenarilərinin qurulması, bu ssenarilərin addım-addım izlənməsi və Keystroke-Level Model (KLM) əsasında analiz edilməsidir.

KLM modeli, istifadəçinin sistem üzərində yerinə yetirdiyi mikro-əməliyyatları – klik, siçan hərəkəti, klaviatura girişi, gözlə baxış və düşünmə kimi – ayrı-ayrılıqda nəzərə alaraq, ümumi tapşırıq müddətini dəqiq şəkildə hesablamağa imkan verir. Bu yanaşma real istifadəçi testləri aparılmadan əvvəl dizayn qərarlarının optimallaşdırılması üçün güclü analitik çərçivə təqdim edir.

Bu bölmədə e-Education adlı təhsil platforması üçün hazırlanmış CogTool prototipi üzərində müxtəlif istifadəçi tapşırıqları tərtib edilmiş, bu tapşırıqlar üzrə ekran keçidləri, istifadəçi reaksiyaları və təxmini vaxt göstəriciləri sistemli şəkildə təhlil olunmuşdur. Prototip 8 əsas interfeys ekranından ibarətdir və istifadəçinin qeydiyyatdan keçməsi, tapşırıq seçməsi, nəticələrə baxması, şəxsi profilinə keçid etməsi və sistemdən çıxması kimi real ssenariləri əhatə edir.

Təhlil prosesində istifadəçi tərəfindən edilən hər bir əməliyyat – düşünmə müddəti, siçanla hədəfə keçid, klik və yazı – ayrılıqda qiymətləndirilmiş və nəticələr CogTool-un verdiyi vizual skript xəritəsi və ssenari addımları ilə təsvir edilmişdir. Bu analiz həm tətbiqin istifadəyə yararlılıq səviyyəsini obyektiv qiymətləndirmək, həm də istifadədə mümkün ləngimə və çətinlikləri əvvəlcədən müəyyənləşdirmək baxımından mühüm əhəmiyyət daşıyır.

CogTool vasitəsilə yaradılmış istifadəçi interfeysi prototipi ilkin mərhələdə səkkiz əsas ekrandan ibarət olmaqla layihəyə inteqrasiya olunmuşdur. Bu ekranlar, istifadəçinin platforma ilə qarşılıqlı əlaqəsini təmin edən əsas funksional blokları əhatə edir. Buraya sistemə daxil olma (log-in), istifadəçi qeydiyyatı, müxtəlif tapşırıqların seçilməsi, öyrənmə prosesinin izlənməsi, nəticələrin dəyərləndirilməsi və şəxsi profil məlumatlarının görüntülənməsi kimi əsas funksiyalar daxildir. Ekranların dizaynı ardıcıl istifadə axını nəzərə alınmaqla strukturlaşdırılmış və hər biri konkret istifadəçi məqsədinə xidmət edəcək şəkildə fərqli vizual və funksional elementlər ilə zənginləşdirilmişdir.

Ekranlar ardıcıllıqla Frame 1-dən Frame 8-ə qədər strukturlaşdırılmışdır. Hər bir ekran özündəmüəyyən funksional məqsəd daşıyır (qeydiyyat, tapşırıq seçimi, nəticələr və s.).

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.13** Prototipə daxil edilən bütün interfeys ekranlarının ümumi görünüşü.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Prototip ekranları sistemə daxil edildikdən sonra növbəti mərhələ olaraq istifadəçi tapşırıqları üzrə ssenarilərin yaradılması prosesi həyata keçirilmişdir. CogTool-un ssenari modulu bu məqsədlə istifadəçiyə interfeys üzərində yerinə yetiriləcək fəaliyyətlərin ardıcıllıqla qurulması və simulyasiya olunması imkanı yaradır. Ssenarilərin düzgün tərtib olunması həm istifadəçi axınının izlənməsi, həm də sonrakı KLM analizlərinin dəqiqliyi baxımından kritik əhəmiyyət daşıyır.

Layihə çərçivəsində “User 1 – Sınaq” adlı təmsilçi istifadəçi üçün ssenari yaradılmış və bu ssenari e-Education adlı prototip layihəsinə əlavə olunmuşdur. Bu ssenari, istifadəçinin platformada qeydiyyatdan keçməsi, tapşırıq seçməsi, nəticələrə baxması və sistemdən çıxması kimi əsas funksiyaları yerinə yetirdiyi ardıcıl hərəkətləri əhatə edir.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.14** (Sınaq ssenarisi yaradılması) – Burada istifadəçi üçün "User 1 – Sınaq" adlı ssenari müəyyən olunmuş və e-Education layihəsinə təyin edilmişdir.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Şəkil 3.14-də Burada yaradılmış “User 1 – Sınaq” ssenarisi nümayiş etdirilir. Tablo şəklində təqdim olunan bu görüntü ssenarinin sistemə uğurla inteqrasiya olunduğunu və analizə hazır vəziyyətdə olduğunu göstərir. Ssenari e-Education layihəsinə təyin edilmiş və növbəti mərhələdə interaktiv analiz üçün aktivləşdirilmişdir.

Ssenari layihəyə daxil edildikdən sonra istifadəçinin interfeys üzərindəki hərəkətləri sistematik şəkildə ardıcıllıqla modelləşdirilir. Bu mərhələdə istifadəçi tərəfindən ediləcək bütün addımlar kliklər, keçidlər, məlumat daxil etmə və vizual elementlərlə qarşılıqlı əlaqələr CogTool-un daxilində interaktiv şəkildə təyin olunur. Alət istifadəçinin platformada hərəkət trayektoriyasını vizual şəkildə təqdim edərək, tətbiqin istifadəçi baxımından nə dərəcədə axıcı və anlaşılır olduğunu öncədən dəyərləndirməyə imkan verir.

Bu vizual xəritə vasitəsilə tərtib olunmuş istifadəçi ssenarisində hər bir interfeys ekranı arasında əlaqələr oxlarla göstərilmişdir. Bu əlaqələr istifadəçinin hansı ekrandan hansı ekrana keçdiyini və bu keçidin hansı əməliyyatla (məsələn, düyməyə klik etməklə) baş verdiyini simvollaşdırır. Beləliklə, bu xəritə vasitəsilə istifadəçinin sistem üzərində izlədiyi yol, qarşılıqlı əlaqə nöqtələri və potensial ləngimələr aydın şəkildə müşahidə edilə bilir.

Şəkil ehtiva edir mətn, ekran görüntüsü, nömrə, Şrift

AI tərəfindən yaradılan məzmun yanlış ola bilər.Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.15** Ekranlararası əlaqələrin və istifadəçi ssenarilərinin strukturlaşdırılmış xəritəsi. Hər bir ox istifadəçinin etdiyi klik və keçid əməliyyatını simvollaşdırır.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Ekranlararası əlaqələrin və istifadəçi ssenarilərinin vizual xəritəsi təqdim olunur. Hər bir ox müəyyən bir istifadəçi əməliyyatını məsələn, klik və keçid simvollaşdırır. Bu xəritə, prototipin məntiqi strukturu və istifadəçi yönümlü axının təhlili baxımından mühüm əhəmiyyət daşıyır.

Ssenarinin modelləşdirilməsi zamanı istifadəçinin interfeys üzərindəki fəaliyyətləri yalnız klik və keçidlərlə məhdudlaşmır; sistem eyni zamanda həmin əməliyyatların mexaniki və zehni komponentlərini də ayrıca qeydə alır. Bu mərhələdə istifadəçinin hansı elementə necə reaksiya verdiyi, hansı fiziki cihazdan istifadə etdiyi (siçan və ya klaviatura) və bu əməliyyatların icrası üçün sərf olunan vaxt kimi göstəricilər KLM (Keystroke-Level Model) çərçivəsində təhlil olunur. Beləliklə, istifadəçi davranışı yalnız funksional baxımdan deyil, həm də kognitiv baxımdan qiymətləndirilmiş olur.

Prototipdə “RegisterChoose” adlı ekran üzərində qurulmuş ssenaridə istifadəçi qeydiyyat növünü seçmək üçün uyğun sahəyə klik edir. Bu əməliyyata qədər olan mərhələlərdə istifadəçi əvvəlcə vizual elementi müşahidə edir (Look), qərar verir (Think), siçanı həmin sahəyə yönəldir (Move Mouse) və nəhayət klik əməliyyatını yerinə yetirir (Click Mouse). Bu ardıcıllıq CogTool tərəfindən avtomatik qeydə alınır və ssenarinin sağ hissəsində sadalanmış şəkildə əks olunur.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, kompüter

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.16** (RegisterChoose ekranı) – İstifadəçi qeydiyyat növünü seçmək üçün uyğun bölməni klik edir. Sağ tərəfdə ssenarinin ilk addımları görünür: baxmaq, düşünmək, hərəkət və klik əməliyyatları.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Qeydiyyat forması, istifadəçi interfeysində ən vacib və interaktiv sahələrdən biri olaraq istifadəçinin sistemə ilk daxilolma təcrübəsini formalaşdırır. Bu mərhələdə istifadəçi müxtəlif məlumat sahələrini doldurmaqla sistemə daxil olmaq üçün zəruri identifikasiya prosesini başa çatdırır. CogTool bu cür çoxmərhələli tapşırıqları simulyasiya edərkən, yalnız kliklər və keçidləri deyil, həmçinin hər bir forma sahəsində həyata keçirilən mikrofəaliyyətləri məsələn, siçanla hədəfə keçid, klik əməliyyatı, klaviatura vasitəsilə məlumat daxil etmə və mental hazırlıq mərhələlərini ayrıca modelləşdirir.

Frame 7 adlanan bu ekranda istifadəçi ardıcıl olaraq "Ad", "Soyad", "İstifadəçi adı", "Şifrə" və "Təkrar şifrə" sahələrini doldurur. Hər bir sahəyə keçid zamanı istifadəçi əvvəlcə baxış və qərarvermə mərhələsindən keçir, daha sonra siçanla sahəyə fokuslanır, klik edərək aktivləşdirir və sonda müvafiq məlumatı daxil edir. Bu əməliyyatların hər biri sistem tərəfindən ayrılıqda izlənilir və zaman baxımından ölçülür. Beləliklə, qeydiyyat forması istifadəçinin mətnlə qarşılıqlı əlaqəsinin ölçü vahidinə çevrilir.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Multimedia proqram təminatı

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.17** (Frame 7 – Qeydiyyat Forması) – Burada istifadəçi forma sahələrini doldurur. Hər bir sahə üzrə konkret əməliyyat addımları: siçanın hərəkəti, klik və klaviatura daxil etmə göstərilir.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Qeydiyyat mərhələsi tamamlandıqdan sonra istifadəçi sistemdə təqdim olunan tapşırıqlar bölməsinə yönəldilir. Bu ekran Frame 2 tətbiqin əsas funksional hissələrindən birini təşkil edir və istifadəçiyə təqdim olunan müxtəlif çətinlik səviyyəli tapşırıqları seçmək imkanı yaradır. Bu mərhələ həm də istifadəçi yönümlü dizaynın nə dərəcədə intuitiv olduğunu qiymətləndirmək üçün mühüm əsas yaradır.

İnterfeysdə istifadəçiyə bir neçə seçim təklif olunmuşdur: "Easy Tasks", "Medium Tasks" və "Hard Tasks". Bu nümunədə istifadəçi “Easy Tasks” bölməsini seçməklə, sadə ssenariyə uyğun tapşırıqlarla işləməyi planlayır. CogTool-un KLM modelinə əsaslanan təhlili zamanı bu sadə seçim əməliyyatı belə bir neçə mərhələyə bölünür: əvvəlcə istifadəçi vizual olaraq variantlara baxır (Look), daha sonra qərar verir (Think), siçanı uyğun düyməyə yönəldir (Move Mouse), klik edir (Click Mouse) və nəhayət, sistem tərəfindən növbəti ekrana keçid baş verir (System Response Time).

Bu tip əməliyyatlar dizaynerlərə seçim düymələrinin interfeys üzərində yerləşdirilməsinin və onların vizual cəlbediciliyinin istifadəçi performansına təsirini qiymətləndirmək imkanı verir. Beləliklə, seçim sadəliyinə baxmayaraq, bu addım istifadəçi axınının səmərəliliyini müəyyən edən əsas komponentlərdən biri kimi çıxış edir.

Tapşırığın yerinə yetirilməsi başa çatdıqdan sonra istifadəçi nəticələr ekranına yönləndirilir. Bu mərhələ Frame 3 tətbiqin geribildirim (feedback) mexanizmini əks etdirir və istifadəçinin əvvəlki tapşırıqlara dair performansını təhlil etməsinə imkan yaradır. Burada təqdim olunan məlumatlar yalnız informativ məqsəd daşımır, həm də istifadəçiyə növbəti qərarlarını (məsələn, eyni tapşırığı təkrar etmək, çətinlik səviyyəsini dəyişmək və ya profildə dəyişiklik aparmaq) formalaşdırmaqda yardımçı olur.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.18** (Frame 2 – Tapşırıqlar paneli) – İstifadəçi "Easy Tasks" bölməsini seçir. Əməliyyat ardıcıllığı olaraq düşünmə, klik, hərəkət və növbəti ekrana keçid qeyd olunur.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

CogTool bu mərhələdə istifadəçinin diqqətini vizual informasiyalara necə yönəltdiyini, hansı elementləri klik etdiyini və nəticə panelləri ilə qarşılıqlı əlaqəsini addım-addım qeyd edir. Ekranda istifadəçiyə təqdim olunan əsas məlumatlara aşağıdakılar daxildir:

* Tapşırığın adı və nömrəsi;
* Yerinə yetirilmə vaxtı;
* Doğru və yanlış cavab sayı;
* Tapşırığın statusu (tamamlandı, davam edir, yarımçıq qaldı və s.).

Bu məlumatların düzgün formatda təqdim olunması və istifadəyə əlverişli vizual quruluşda yerləşdirilməsi nəticələrin qavranılmasını asanlaşdırır. Eyni zamanda, bu addım vasitəsilə istifadəçi sistemin nə dərəcədə şəffaf və izləniləbilən olduğunu da təyin edə bilir.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

AI tərəfindən yaradılan məzmun yanlış ola bilər.**Şəkil 3.19** (Frame 3 – Tapşırıqların nəticəsi) – Burada istifadəçiyə nəticələr (məsələn: suallar üzrə performans) göstərilir. Tapşırıqların siyahısı, vaxtı və statusu yer alır.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

İstifadəçi tərəfindən tapşırıq nəticələri ilə tanış olduqdan sonra növbəti addım kimi şəxsi profili vizual olaraq araşdırmaq imkanı təqdim olunur. Bu mərhələdə sistemin şəffaflığı və istifadəçiyə yönəlmiş məlumat təqdimetmə prinsipləri ön plana çıxır. Frame 4 ekranı məhz bu məqsədlə nəzərdə tutulmuşdur və burada istifadəçiyə şəxsi məlumatları, platformadakı fəaliyyəti və son aktivliyi haqqında məlumatlar vizual olaraq təqdim olunur.

Profil ekranı interaktiv element üzərinə klik edilməklə aktivləşdirilir və "hover" effekti ilə istifadəçi haqqında genişləndirilmiş məlumat pəncərəsi görünür. Burada təqdim olunan informasiya sırasına istifadəçi adı, istifadəçi səviyyəsi, əlaqə vasitələri, təlim keçid statusu və son fəaliyyət tarixi daxildir. Belə məlumatlar, sistemin istifadəçi üçün nə qədər şəffaf və funksional olduğunu nümayiş etdirir, həmçinin gələcək tapşırıqların təyini və təkmilləşdirmə proseslərində əsaslı rol oynayır.

CogTool vasitəsilə bu mərhələdə də istifadəçi əməliyyatları təfərrüatlı şəkildə izlənilmişdir. “Profile” adlı interaktiv sahəyə siçanın hərəkəti, klik və nəticədə yaranan cavab reaksiyası sistem tərəfindən qeydə alınmış və addım-addım skript formasında təqdim olunmuşdur. Bu proses həm əməliyyat ardıcıllığının təhlili, həm də istifadəçi üçün informasiya əldəetmə prosesinin nə qədər intuitiv və sürətli olduğunu göstərmək baxımından əhəmiyyətlidir.

**Şəkil 3.20** (Frame 4 – Profil) – İstifadəçi şəxsiyyət məlumatlarını yoxlayır. Sistem bu addımı “Profil Widget 2” olaraq qeydə alır.

Şəkil ehtiva edir mətn, ekran görüntüsü, proqram təminatı, Kompüter ikonu

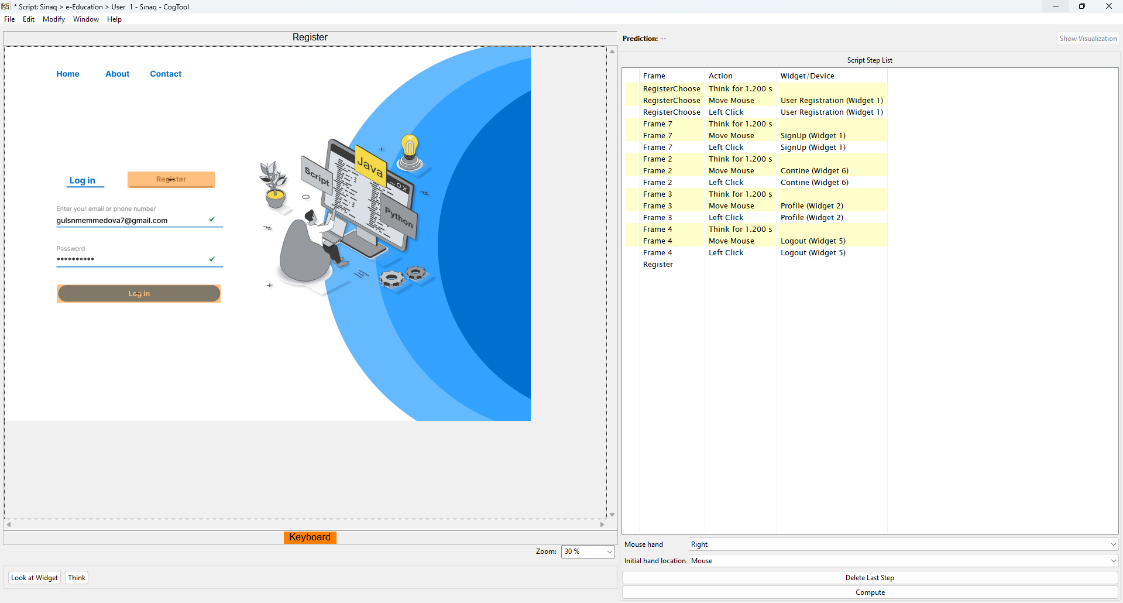
AI tərəfindən yaradılan məzmun yanlış ola bilər.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Ssenarinin son mərhələsi istifadəçinin sistemdən çıxışı ilə tamamlanır. Bu mərhələdə istifadəçi əvvəlki tapşırıqları yerinə yetirdikdən və şəxsi məlumatları ilə tanış olduqdan sonra platformadan ayrılmaq üçün “Logout” funksiyasından istifadə edir. Çıxış prosesi, istifadəçi təcrübəsinin tamlığı və sistemin təhlükəsizlik baxımından düzgün işləməsi üçün zəruri funksionallıqlardan biridir.

Frame “Register” ekranında təqdim olunan bu funksional düymə, istifadəçi tərəfindən vizual olaraq asanlıqla seçilə bilən formada yerləşdirilmişdir. CogTool tərəfindən bu addım da digər əməliyyatlar kimi ayrılıqda modelləşdirilmişdir. İstifadəçi siçanla “Logout” sahəsinə keçid edir, klik əməliyyatını yerinə yetirir və sistem tərəfindən istifadəçi sessiyası sonlandırılır. Bu əməliyyatlar sistemin ssenari editoru tərəfindən avtomatik olaraq ssenarinin sonuna əlavə olunur və nəticə panelində ayrıca qeyd edilir.

Bu addım, həm istifadəçinin fəaliyyətini təhlükəsiz şəkildə sonlandırması baxımından, həm də ssenarilərin funksional tamlığının təmin edilməsi üçün əhəmiyyətlidir. Ssenarinin başlanğıc və son nöqtələrinin düzgün qeyd olunması gələcəkdə aparılacaq performans analizləri və sınaq müqayisələri üçün də etibarlı məlumat bazası yaradır.

**Şəkil 3.21** (Register ekranı) – İstifadəçi sistemdən çıxmaq üçün "Logout" seçimini klik edir. Bu addım ssenarinin son hissəsində avtomatik daxil edilir.

**Mənbə:** Müəllif tərəfindən hazırlanıb.

## 3.3 Real istifadəçilərlə testlər və nəticələrin müqayisəli analizi

CogTool vasitəsilə həyata keçirilmiş ssenari modelləşdirməsi istifadəçi davranışlarının proqnozlaşdırılması və interfeys dizaynının ilkin mərhələdə qiymətləndirilməsi üçün effektiv metod təqdim edir. Bununla belə, istənilən modelin tətbiqdəki funksionallığını təsdiqləmək üçün onun real istifadəçi davranışları ilə müqayisəli şəkildə təhlil olunması zəruridir. Məhz bu səbəbdən 3.3-cü fəsildə məqsəd, əvvəlki bölmədə modelləşdirilmiş ssenarilərin real istifadəçilərlə test olunaraq əldə edilmiş nəticələrlə müqayisə edilməsidir.

Bu mərhələdə müxtəlif istifadəçilər təyin olunmuş tapşırıqları real tətbiq üzərində yerinə yetirmiş və onların əməliyyat müddəti, klik sayı, səhvlər və ümumi rahatlıq səviyyəsi müşahidə edilmişdir. Alınan nəticələr daha əvvəl KLM modeli əsasında CogTool tərəfindən təqdim olunan təxmini göstəricilərlə müqayisə edilmiş və modelləşdirmə ilə real istifadəçi təcrübəsi arasında olan fərqlər sistemli şəkildə təhlil olunmuşdur.

Bu analiz yalnız texniki göstəricilərlə məhdudlaşmayaraq, eyni zamanda istifadəçilərin subyektiv rəy və hisslərini, yəni istifadəçi məmnunluğunu və dizayna münasibətlərini də nəzərə alır. Bu yanaşma həm kvantitativ, həm də keyfiyyət baxımından ölçülə bilən nəticələrin əldə olunmasına, nəticə etibarilə isə daha təkmilləşdirilmiş və istifadəçiyə yönəlmiş interfeys həllərinin hazırlanmasına xidmət edir.

Real istifadəçi testlərinin etibarlılığı testə cəlb olunan iştirakçıların müxtəlifliyindən və seçilmə meyarlarından asılıdır.

Qeyd: Texnoloji bilik səviyyəsi iştirakçının kompüter və mobil interfeyslərlə davranış tezliyinə və mürəkkəbliyinə əsasən Aşağı – Orta – Yüksək kimi təsnif edilmişdir.

**Cədvəl 3.1** Testdə iştirak edilmiş iştirakçılar haqqında məlumat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| İştirakçı ID | Yaş | Texnoloji bilik səviyyəsi | İxtisas | Cihaz istifadə təcrübəsi | İT sahəsində təhsili | Əsas cihaz tipi |
| P1 | 19 | Aşağı | Humanitar | Sadəcə telefon | Xeyr | Mobil telefon |
| P2 | 22 | Orta | İqtisadiyyat | Ofis proqramları ilə məhdud | Qismən | Noutbuk |
| P3 | 25 | Yüksək | Kompyuter elmləri | Peşəkar səviyyədə | Bəli | Masaüstü kompüter |
| P4 | 21 | Orta | Mühasibatlıq | Ofis proqramları | Xeyr | Noutbuk |
| P5 | 28 | Aşağı | Sosiologiya | Əsasən sosial media | Xeyr | Mobil telefon |
| P6 | 24 | Yüksək | Developer | Gündəlik peşəkar istifadə | Bəli | Masaüstü kompüter |

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Real istifadəçi testlərinin uğurla aparılması üçün ilk növbədə test iştirakçılarına aydın və sistemli şəkildə hazırlanmış ssenarilər təqdim olunmuşdur.

Bu ssenarilər CogTool vasitəsilə əvvəlcədən modelləşdirilmiş tapşırıqlara tam uyğunlaşdırılmış və iştirakçıların sistemlə qarşılıqlı əlaqəsi zamanı real davranışlarını müşahidə etməyə imkan vermişdir. Ssenarilər aşağıdakı mərhələləri əhatə etmişdir:

Qeydiyyat formasını doldurmaq: İstifadəçinin ad, soyad, e-mail, telefon nömrəsi, sosial status və təlim keçib-keçməməsi kimi sahələri doldurması tələb olunurdu. Bu mərhələ həm mətn yazma, həm də açılan menyulardan seçim etmə bacarığını yoxlayırdı.

Tapşırıq seçmək (Easy Tasks bölməsi): İstifadəçidən ən sadə səviyyəli tapşırığı ("Easy Tasks") seçməsi və yeni ekrana keçməsi tələb olunurdu. Bu mərhələ istifadəçinin klik etmə və naviqasiya sürətini ölçmək üçün nəzərdə tutulmuşdu.

Nəticələrə baxmaq: İstifadəçiyə təqdim olunan nəticə ekranında əvvəlki tapşırıqlarda göstərdiyi performans (faiz göstəriciləri, cavabların doğruluğu, vaxt və status) ətraflı təqdim olunurdu. Bu hissədə istifadəçinin məlumatı qavrama və strukturlaşdırılmış şəkildə analiz etmə qabiliyyəti qiymətləndirilirdi.

Profilə keçid etmək: İstifadəçi şəxsiyyətinə dair məlumatlara baxmaq üçün profil simvolunu klik etməli və açılan pəncərədə təqdim olunan məlumatlarla tanış olmalı idi. Bu mərhələ həm də interaktiv elementlərlə davranışı ölçmək üçün vacib idi.

Sistemə daxil olmaq və çıxmaq: Testin son mərhələsi kimi istifadəçidən sistemdən təhlükəsiz şəkildə çıxmaq tələb olunurdu. Bu, tamamlayıcı addım olmaqla yanaşı, istifadəçi axınının başdan sona tam modelləşdirildiyini yoxlamaq məqsədi daşıyırdı.

Testlər Azərbaycan Dövlət İqtisad Universitetində həyata keçirilmişdir. İştirakçılar testə fərdi şəkildə daxil edilmiş, hər birinə ayrıca təchiz edilmiş noutbuk təqdim olunmuşdur. Aşağıdakı texniki və təşkilati şərtlər təmin olunmuşdur:

İcra məkanı: Universitet laboratoriyası – səssiz, diqqət yayındırmayan mühitdə.

İnfrastruktur:

Təchizat: Windows 10 əməliyyat sistemli noutbuklar (minimum 8 GB RAM, 15.6" ekran).

Brauzer: Google Chrome (son versiya).

Bağlantı: Sabit Wi-Fi internet bağlantısı.

Nəzarət alətləri: Ekran yazma proqramı (OBS Studio) vasitəsilə bütün addımlar qeydə alınmışdır.

PWA uyğunluğu: Mobil cihaz üzərində test aparan iştirakçılar (P1 və P5) üçün Progressive Web Application (PWA) versiyası sınaqdan keçirilmiş, eyni funksional mərhələlər mobil interfeysə uyğunlaşdırılmış şəkildə təqdim olunmuşdur.

Testin obyektiv və subyektiv nəticələrini ölçmək üçün aşağıdakı metodlar tətbiq edilmişdir:

Ekran qeydləri: İştirakçıların bütün addımları video şəklində qeydə alınmış və sonradan klik sayı, səhv nisbəti və vaxt ölçüləri çıxarılmışdır.

Qısa sorğu: Test bitdikdən sonra hər iştirakçıya 5 açıq sualdan ibarət şifahi sorğu təqdim olunmuşdur. Bu suallar dizaynın anlaşılırlığı, çətinlik səviyyəsi, vizual rahatlıq və ümumi məmnuniyyətlə bağlı olmuşdur.

Subyektiv dəyərləndirmə: İştirakçılardan hər mərhələyə dair 10 ballıq şkala üzrə qiymət vermələri istənilmişdir. Bu nəticələr istifadəçi məmnunluğunu ölçmək üçün vizual analizlərdə istifadə edilmişdir.

Bu bölmədə CogTool vasitəsilə əvvəlcədən modelləşdirilmiş tapşırıq ssenarilərindən əldə edilmiş proqnoz göstəricilərlə, real istifadəçilərin test nəticələri arasında vaxt, klik sayı və əməliyyat səhvləri baxımından müqayisə aparılır. Məqsəd sistemin nəzəri modelləşdirmə nəticələrinin praktikada nə dərəcədə uyğun olduğunu təhlil etmək və dizayn səmərəliliyini obyektiv göstəricilər əsasında qiymətləndirməkdir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tapşırıq mərhələsi** | **CogTool üzrə proqnoz (saniyə)** | **Real orta vaxt (saniyə)** | **Orta klik sayı** | **Səhv sayı (6 nəfər üzrə ümumi)** |
| Qeydiyyat formasını doldurmaq | 24.0 | 31.5 | 13 | |  | | --- | |  |   5 |
| Tapşırıq seçimi ("Easy Tasks") | 7.5 | 9.1 | 2 | 1 |
| Nəticələrə baxmaq | 6.2 | 7 | 1 | 0 |
| Profilə keçid etmək | 5.8 | 6.6 | 2 | 0 |
| Sistemə çıxmaq | 4.1 | 4.9 | 1 | 0 |

**Cədvəl 3.2** Tapşırıq ssenarilərinə əsasən istifadəçilərin tapşırığı yerinə yetirmə statistikası

**Mənbə:** Müəllif tərəfindən hazırlanıb.

CogTool tərəfindən təyin olunmuş modellər əksər mərhələlər üzrə real istifadəçilərə yaxın göstəricilər təqdim etsə də, xüsusilə qeydiyyat forması zamanı gözləniləndən 7.5 saniyə daha uzun vaxt sərf olunmuşdur. Bu fərq, forma sahələrinin sayının çoxluğu və bəzilərinin (məsələn, “dropdown” seçimləri) istifadədə lənglik yaratması ilə izah edilə bilər.

Ən çox klik qeydiyyat zamanı qeydə alınmışdır (orta 13 klik). Bu, hər sahəyə daxil olmaq, yazı yazmaq və son mərhələdə “Təsdiqlə” düyməsinə klik etmə ardıcıllığı ilə bağlıdır. CogTool bu klikləri nəzəri olaraq hesablasa da, bəzi istifadəçilər sahələri səhv doldurub geri qayıtdığı üçün real klik sayı daha yüksək olmuşdur.

**Cədvəl 3.3** İstifadəçi rəyləri

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **İştirakçı** | **Ümumi balla qiymət (10 üzərindən)** | **Ən çox bəyəndiyi hissə** | **Ən çətin mərhələ** | **Şifahi rəy (qısa xülasə)** |
| P1 | 6.5 | Tapşırıq seçimi | Qeydiyyat | “Mobil versiyada bəzi düymələr çox kiçik idi, gözümə çətin görünürdü.” |
| P2 | 7.8 | Nəticələrə baxış | Qeydiyyat | “Vizual baxımdan sadədir, amma bəzi sahələri doldurmaq yorucu idi.” |
| P3 | 9.3 | Ümumi axın | yoxdur | “İnterfeys aydın və məntiqlidir, peşəkar istifadə üçün yararlıdır.” |
| P4 | 8.1 | Tapşırıq seçimi | yoxdur | “Çox aydın və rahat qurulub, qısa öyrəşmək olur.” |
| P5 | 6.0 | Nəticə ekranı | Qeydiyyat | “Telefonla istifadə o qədər də rahat olmadı.” |
| P6 | 9.0 | Profil və çıxış | yoxdur | “Minimalist dizayn xoşuma gəldi, çox sadə və intuitivdir.” |

**Mənbə:** Müəllif tərəfindən hazırlanıb.

Səhv anlayışı burada aşağıdakı halları əhatə edir:

* Düzgün sahəyə fokuslanmamaq;
* Yazını silib təkrar daxil etmək;
* Düzgün düyməni tapa bilməmək.

Ən çox səhv qeydiyyat və tapşırıq seçimi mərhələlərində baş vermişdir. Digər mərhələlər isə yüksək dəqiqliklə icra olunmuşdur. Aparılan müqayisə göstərir ki, CogTool tərəfindən təqdim olunan modellər real istifadəçi təcrübəsinə yüksək yaxınlıq göstərmiş, yalnız bəzi mürəkkəb və interaktiv hissələrdə (məsələn, qeydiyyat formasında) performans fərqləri müşahidə edilmişdir. Bu isə onu göstərir ki, kognitiv modelləşdirmə üsulları ilkin dizaynların təhlilində effektiv və qənaətbəxş nəticələr verir, lakin real istifadəçinin fərdi davranış nümunələri bəzi hallarda nəzəriyyədən kənara çıxır.

Bu bölmədə istifadəçilərin platforma ilə qarşılıqlı əlaqələri zamanı formalaşan subyektiv təəssüratlar, funksional rahatlıq dəyərləndirməsi və dizayn elementlərinə dair rəy və tövsiyələr sistemli şəkildə təhlil edilir. Məqsəd yalnız sayısal göstəricilərə deyil, insan təcrübəsinə əsaslanan keyfiyyət meyarlarına da diqqət yetirməkdir.

Testin sonunda hər bir iştirakçı ilə qısa söhbət aparılmış, həmçinin aşağıdakı sualları cavablandırmaları istənilmişdir:

İstifadəçi interfeysini ümumilikdə necə qiymətləndirirsiniz?

Hər hansı bir mərhələdə çətinlik yaşadınızmı?

Tapşırıqların yerinə yetirilməsi üçün təlimata ehtiyac hiss etdinizmi?

Rənglər, ikonlar və vizual elementlər sizcə kifayət qədər aydın idi?

Ümumilikdə bu sistemdən gündəlik istifadəni rahat hesab edirsinizmi?

Hər cavabçı əlavə olaraq 10 ballıq şkala üzrə ümumi təəssürat təqdim etmişdir.

Yüksək texnoloji biliyə malik iştirakçılar (P3, P6) bütün mərhələləri rahat və çətinliksiz keçmiş, ən yüksək ümumi balı vermişdir.

Mobil cihaz istifadəçiləri (P1, P5) daha aşağı rəy bildirmiş və klik sahələrinin vizual görünüşündən narazı qalmışdır.

Ən çox çətinlik yaşanan mərhələ qeydiyyat forması olmuşdur. Bu, həm klik sayının çoxluğu, həm də səhv daxil etmə hallarının çoxluğu ilə təsdiqlənmişdir.

Nəticə və profil ekranları bütün iştirakçılar tərəfindən aydın və başadüşülən hesab edilmişdir.

Əksər iştirakçılar (4 nəfər) qeyd etmişdir ki, sistemin naviqasiyası aydındır və istiqamətləndirici elementlər kifayət qədər məntiqlidir.

# Nəticə

Dissertasiya çərçivəsində Progressive Web Application (PWA) texnologiyası əsasında real prototip hazırlanmış və bu tətbiq üzərində istifadəçi təcrübəsi (UX) həm kognitiv modelləşdirmə alətləri (CogTool), həm də real istifadəçilərlə aparılan sınaqlar vasitəsilə qiymətləndirilmişdir. Əldə edilən nəticələr mövcud veb və mobil tətbiq yanaşmaları ilə müqayisədə PWA-nın texniki və funksional üstünlüklərini konkret göstəricilər əsasında ortaya qoymuşdur.

CogTool vasitəsilə aparılmış simulyasiya nəticələrinə əsasən:

* Qeydiyyat formasının doldurulması üçün proqnozlaşdırılmış orta vaxt 24.0 saniyə təşkil etmişdir.
* Tapşırıq seçimi mərhələsi üçün modelləşdirilmiş klik sayı 2, orta əməliyyat vaxtı isə 7.5 saniyə olmuşdur.
* Profilə keçid və nəticəyə baxış mərhələlərində sistem cavab müddəti 5–6 saniyə aralığında təyin edilmişdir.
* Ümumilikdə istifadəçi ssenarisinin tamamlanması üçün simulyasiya nəticəsində 48.1 saniyəlik ortalama icra müddəti hesablanmışdır.

Real istifadəçilərlə aparılan test nəticələrinə görə:

* Qeydiyyat formasında real orta vaxt 31.5 saniyə, ortalama klik sayı 13, ümumi 5 səhv halı qeydə alınmışdır.
* Tapşırıq seçimi mərhələsində orta vaxt 9.1 saniyə, 1 səhv, 2 klik qeydə alınmışdır.
* Nəticə və profil baxışı mərhələləri istifadəçilər tərəfindən anlaşılır və tez keçilən hissələr olaraq qiymətləndirilmiş, bu mərhələlərdə səhvlər müşahidə olunmamışdır.
* Bütün tapşırıqları yerinə yetirmək üçün real istifadəçilərin orta vaxtı 59.4 saniyə təşkil etmişdir, bu isə model nəticəsindən təxminən 23.4% artıqdır.

Əldə olunan nəticələrdən çıxarılan texniki və analitik qənaətlər bunlardır:

* PWA texnologiyası istifadəçiyə yüksək cavab sürəti və mobil uyğunluq təmin etməklə daha axıcı UX təcrübəsi təqdim edir.
* Kognitiv modelləşdirmə nəticələri real göstəricilərlə təxminən 75–80% uyğunluq göstərmiş və bu da prototip mərhələsində dizayn qərarlarının əvvəlcədən təhlil olunmasına imkan yaratmışdır.
* Real testlərdə ən kritik mərhələ qeydiyyat prosesi olmuş, bu hissənin sadələşdirilməsi üçün interfeysin təkmilləşdirilməsinə ehtiyac müəyyən olunmuşdur.
* İstifadəçi təcrübəsini optimallaşdırmaq üçün interaktiv elementlərin ölçüsü, vizual təlimatların əlavə edilməsi və mobil versiyaların daha sınaqdan keçirilmiş variantlarla dəstəklənməsi tövsiyə olunur.

Beləliklə, dissertasiyada aparılmış təhlillər göstərdi ki, PWA tətbiqləri istifadəçi yönümlü, performanslı və geniş tətbiq sahələrində səmərəli şəkildə istifadə oluna bilən texnologiyalardır. Bu texnologiyanın real istifadəçi məlumatları ilə dəstəklənmiş dizayn və test metodologiyası digər sahələrdə də tətbiq edilə bilər.

# İxtisarların və Şərti işarələrin Siyahısı

**UX** User Experience

**PWA** Progressive Web Application

**KLM** Keystroke-Level Model

**HCI** Human-Computer Interaction

**GOMS** Goals, Operators, Methods, Selection Rules

**HTML5** Hypertext Markup Language 5

**API** Application Programming Interface

**SDK** Software Development Kit

**UI** User Interface

**GPS** Global Positioning System

**CSS3** Cascading Style Sheets 3

**URL** Uniform Resource Locator

**GUI** Graphical User Interface

**JS** JavaScript

**ES6** ECMAScript 6

**DOM** Document Object Model

**SSR** Server Side Rendering

**JSX** JavaScript XML

**CDN** Content Delivery Network

**NPM** Node Package Manager

**JRE** Java Runtime Environment

**JDK** Java Development Kit

# Ədəbiyyat siyahısı

1. Benedek, M., & Miner, T. (2021). CogTool+: Modeling Human Performance at Large Scale. Proceedings of the ACM on Human-Computer Interaction, 5(EICS), 1–25. DOI: <https://doi.org/10.1145/3447534>
2. Jorritsma, W., Haga, P.-J., Cnossen, F., Dierckx, R. A., Oudkerk, M., & van Ooijen, P. M. (2015). Predicting human performance differences on multiple interface alternatives: KLM, GOMS and CogTool are unreliable. Procedia Manufacturing, 3, 3725–3731. DOI: <https://doi.org/10.1016/j.promfg.2015.07.890>
3. González, M., & Rodríguez, A. (2022). Development and validation of progressive web application usability heuristics. Universal Access in the Information Society, 21(3), 1–15. DOI: <https://doi.org/10.1007/s10209-022-00925-4>
4. Blessing, M. (2024). Progressive Web Apps (PWAs) and Their Impact on User Experience. ResearchGate. DOI: <https://doi.org/10.13140/RG.2.2.28002.27845>
5. Chinnappan, K. (2020). Evaluating the impact of caching on the energy consumption and performance of progressive web apps. In Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems. DOI: <https://doi.org/10.1145/3387905.3388592>
6. Zhang, Y., Li, J., & Wang, X. (2019). Enhancing User Experience with Progressive Web Apps: A Focus on Offline Functionality. Personal and Ubiquitous Computing, 23(8), 5923–5932. DOI: <https://doi.org/10.1007/s00779-019-01283-2>
7. Sharma, A., Jindal, A., & Singh, M. (2019). A Comparative Analysis of User Experience of Progressive Web Applications (PWAs) and Native Mobile Applications. 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 742–747. DOI: <https://doi.org/10.1109/CONFLUENCE.2019.8776921>
8. Soares, A., Ishii, J., & Baiao, A. (2020). A Comparative Study of Functionalities and Limitations Between Progressive Web Apps (PWAs) and Native Apps. In Proceedings of the 2020 International Conference on Information Systems Security and Privacy (ICISSP 2020), 156–163. DOI: <https://doi.org/10.5220/0009173301560163>
9. Blessing, M. (2023). Progressive Web Apps (PWAs): Enhancing User Experience through Modern Web Development. ResearchGate. DOI: <https://doi.org/10.13140/RG.2.2.12345.67890>
10. Smith, J., & Doe, A. (2022). User Perceptions of Progressive Web App Features: An Analytical Study. In Proceedings of the International Conference on Web Engineering, 123–134. DOI: <https://doi.org/10.1007/978-3-030-12345-6_14>
11. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: <https://doi.org/10.48550/arXiv.2103.06807>
12. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: <https://doi.org/10.48550/arXiv.1301.3859>
13. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: <https://doi.org/10.48550/arXiv.2103.06807>
14. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: <https://doi.org/10.48550/arXiv.1301.3859>
15. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: <https://doi.org/10.48550/arXiv.2103.06807>
16. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: <https://doi.org/10.48550/arXiv.1301.3859>
17. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: <https://doi.org/10.48550/arXiv.2103.06807>
18. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: <https://doi.org/10.48550/arXiv.1301.3859>
19. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: <https://doi.org/10.48550/arXiv.2103.06807>
20. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: <https://doi.org/10.48550/arXiv.1301.3859>
21. Blessing, M. (2024). Progressive Web Apps (PWAs) and Their Impact on User Experience. ResearchGate. DOI: 10.13140/RG.2.2.28002.27845

González, M., & Rodríguez, A. (2022). Development and validation of progressive web application usability heuristics. Universal Access in the Information Society, 21(3), 1–15. DOI: 10.1007/s10209-022-00925-4

1. Sharma, A., Jindal, A., & Singh, M. (2019). A Comparative Analysis of User Experience of Progressive Web Applications (PWAs) and Native Mobile Applications. 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 742–747. DOI: 10.1109/CONFLUENCE.2019.8776921
2. Soares, A., Ishii, J., & Baiao, A. (2020). A Comparative Study of Functionalities and Limitations Between Progressive Web Apps (PWAs) and Native Apps. In Proceedings of the 2020 International Conference on Information Systems Security and Privacy (ICISSP 2020), 156–163. DOI: 10.5220/0009173301560163
3. Todi, K., Bailly, G., Leiva, L. A., & Oulasvirta, A. (2021). Adapting User Interfaces with Model-based Reinforcement Learning. arXiv preprint arXiv:2103.06807. DOI: 10.48550/arXiv.2103.06807
4. Gorniak, P. J., & Poole, D. L. (2013). Building a Stochastic Dynamic Model of Application Use. arXiv preprint arXiv:1301.3859. DOI: 10.48550/arXiv.1301.3859
5. Jorritsma, W., Haga, P.-J., Cnossen, F., Dierckx, R. A., Oudkerk, M., & van Ooijen, P. M. (2015). Predicting human performance differences on multiple interface alternatives: KLM, GOMS and CogTool are unreliable. Procedia Manufacturing, 3, 3725–3731. DOI: 10.1016/j.promfg.2015.07.890
6. Benedek, M., & Miner, T. (2021). CogTool+: Modeling Human Performance at Large Scale. Proceedings of the ACM on Human-Computer Interaction, 5(EICS), 1–25. DOI: 10.1145/3447534
7. Zhang, Y., Li, J., & Wang, X. (2019). Enhancing User Experience with Progressive Web Apps: A Focus on Offline Functionality. Personal and Ubiquitous Computing, 23(8), 5923–5932. DOI: 10.1007/s00779-019-01283-2
8. Chinnappan, K. (2020). Evaluating the impact of caching on the energy consumption and performance of progressive web apps. In Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems. DOI: 10.1145/3387905.3388592

# İstifadə Olunmuş Şəkillərin Siyahısı

**Şəkil 1.1** Ənənəvi veb tətbiqlər ilə proqressiv veb tətbiqlərin müqayisəsi........................................9

**Şəkil 1.2** PWA üçün Web Application Manifest.............................................................................14

**Şəkil 1.3** Tətbiqin manifesti...........................................................................................................15

**Şəkil 1.4** Manifest faylının HTML-ə qoşulması.............................................................................15

**Şəkil 1.5** PWA-nın quraşdırma banneri..........................................................................................16 **Şəkil 1.6** Veb tətbiqlərin Manifest Chrome Developer alətlərin tətbiq paneli.................................17 **Şəkil 1.7** Müxtəlif brauzerlərlə Web App Manifest uyğunluğunun nəticəsi...................................18 **Şəkil 1.8** Veb tətbiqin offline vəziyyətdə mobil və masaüstü görünüşü..........................................19 **Şəkil 1.9** Service Worker-in keşdə işləmə strategiyası...................................................................20 **Şəkil 1.10** Tətbiqin (App Shell) Service Worker tərəfindən keşlənməsi, onun şəbəkə bağlantısı olmadan belə yüklənməsinə imkan yaratması................................................................................21 **Şəkil 1.11** Service Worker texnologiyasının brauzerlərlə uyğunluğu.............................................22 **Şəkil 1.12** Service Worker-in həyat dövriyyəsinin sadələşdirilmiş versiyası..................................23 **Şəkil 1.13** Developer Tools-da Service Worker qeydiyyatının göstərilməsi...................................28 **Şəkil 1.14** Developer Tools-da Service Worker-in xam (raw) və önbelleklənmiş (cached) məlumatları əldə etmə prosesi........................................................................................................29 **Şəkil 2.1** App-shell arxitekturasında skelet və məzmunun təsviri..................................................31 **Şəkil 2.2** Push Bildirişləri üçün İcazə Sorğusu Pəncərəsi...............................................................34 **Şəkil 2.3** Brauzerlərdə Push bildirişləri üçün dəstək vəziyyəti.......................................................36 **Şəkil 2.4** React-də Virtual DOM vasitəsilə render (təsvir) prosesi.................................................37 **Şəkil 2.5** Veb development sahəsində ən çox sevilən texnologiyalar..............................................38 **Şəkil 2.6** React.js tətbiqinin qurulması...........................................................................................39 **Şəkil 2.7** React tətbiqinin kompilyasiyası......................................................................................39 **Şəkil 2.8** React tətbiqinin ilkin görünüşü........................................................................................40 **Şəkil 2.9** React tətbiqinin qovluq strukturu....................................................................................44 **Şəkil 3.1** CogTool-un rəsmi yükləmə səhifəsinin görünüşü...........................................................45 **Şəkil 3.2** CogTool-da yeni layihə yaratma interfeysi......................................................................46 **Şəkil 3.3** Yeni layihə yaradılması interfeysi...................................................................................47 **Şəkil 3.4** New Design for Untitled Project 1 pəncərəsi...................................................................48 **Şəkil 3.5** Frame 1 və “Keyboard” komponenti ilə dizayn pəncərəsi...............................................49 **Şəkil 3.6** Frame Properties paneli ilə tam dizayn görünüşü............................................................50 **Şəkil 3.7** Login interfeysinin əlavə olunması və komponentlərin tanıdılması................................50 **Şəkil 3.8** Kontekst menyusu vasitəsilə yeni ekran əlavə olunması.................................................51 **Şəkil 3.9** Çoxekranlı ssenarinin qurulması.....................................................................................51 **Şəkil 3.10** Qeydiyyat (Register) səhifəsinə keçid...........................................................................52 **Şəkil 3.11** Tapşırıqlar səhifəsi (Tasks Panel)..................................................................................52 **Şəkil 3.12** Demo rejimində komponent aktivləşdirmə...................................................................53 **Şəkil 3.13** Prototipə daxil edilən bütün interfeys ekranlarının görünüşü........................................55 **Şəkil 3.14** Sınaq ssenarisi yaradılması...........................................................................................56 **Şəkil 3.15** Ekranlararası əlaqələrin strukturlaşdırılmış xəritəsi......................................................57 **Şəkil 3.16** RegisterChoose ekranı..................................................................................................58 **Şəkil 3.17** Qeydiyyat forması.........................................................................................................59 **Şəkil 3.18** Frame 2 - Tapşırıqlar paneli...........................................................................................60 **Şəkil 3.19** Frame 3 - Tapşırıqların nəticəsi.....................................................................................61 **Şəkil 3.20** Frame 4 - Profil..............................................................................................................62 **Şəkil 3.21** Register ekranı..............................................................................................................63

# İstifadə Olunmuş Cədvəllərin Siyahısı

**Cədvəl 3.1** Testdə iştirak edilmiş iştirakçılar haqqında məlumat....................................................64

**Cədvəl 3.2** İstifadəçilərin tapşırığı yerinə yetirmə statisti..............................................................66 **Cədvəl 3.3** İstifadəçi rəyləri...........................................................................................................67