YAZILIM GELİŞTİRME LABAROTUVARI PROJESİ

Aslı Bozkurt, Mohammad Nazir Sharifi, Judy Ndotoni Nkwama
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Umuttepe Kampüsü,41380, Kocaeli-Türkiye
{201307025, 191307094, 191307093}@kocaeli.edu.tr

Abstract*— 4*4 Square Puzzle Game project has been developed. The user interface(GUI) was designed with React.js for the game. The game has been developed as web-based. Linked List Data Structure is used to match puzzle pieces. The purpose of the Linked List is to hold the address of the next node in addition to the data that each element (node) should hold. In other words, this is how we determine the correctness of the locations of the puzzle pieces. It selects any picture from your files on the desktop and divides it into 16 pieces. With the mix button in the GUI, the puzzle pieces are mixed. The user will continue to shuffle the puzzle until they manage to match at least one correct puzzle piece. The game starts with a 4*4 puzzle as level 1. Then if the user solves the first level, they move on the second level. 6*6 for second level, 8*8 for third level… etc. Each time the user solves a level of the puzzle they have the option to finish the game and save their score or they can continue to the next level. There is a leaderboard in the GUI. Users can see the score, time and moves. They can print the scoreboard as a .Txt file. The game has a scoring and move system. You can complete your puzzle pieces in the light and dark theme in the game. The game has been deployed both frontend and backend via Heroku.

***Keywords*—puzzle, linked data structure, gui, .txt file, score, move, shuffle button**

Özet*— 4*4 Kare Puzzle Oyun projesi geliştirilmiştir. Oyun için React.js ile kullanıcı arayüzü(GUI) tasarlanmıştır. Oyun, web tabanlı olarak geliştirilmiştir. Yapboz parçalarının eşleştirilmesi için Bağlı Liste Veri Yapısı kullanılmıştır. Bağlı Veri Yapısı (Linked List) kullanım amacı ise her bir eleman(node) tutması gereken dataya ilaveten kendinden sonraki gelecek olan node'un adresini tutar. Yani puzzle parçalarının yerlerinin doğruluğunu bu şekilde tespit ediyoruz Masaüstünde bulunan dosyalarınızdan herhangibir resmi seçip 16 parçaya ayırmaktadır.. GUI 'de bulunan karıştır butonu ile yapboz parçaları karıştırılır. Kullanıcı en az bir doğru yapboz parçasını eşleştirmeyi başarana dek yapbozu karıştırmaya devam edecektir. Oyun 1. seviye olarak 4x4'lük bir bulmaca ile başlar. Daha sonra kullanıcı 1. seviyeyi çözerse 2. seviyeye geçer. 2. seviye için 6x6, 3. seviye için 8x8... vb. Kullanıcı bulmacanın bir seviyesini her çözdüğünde, oyunu bitirme ve puanını kaydetme seçeneğine sahiptir veya bir sonraki seviyeye devam edebilirler. GUI'de skor tablosu bulunmaktadır. Kullanıcılar skor, süre ve hamlelerini görebilmektedir. Skor tablosunu .Txt uzantılı dosya halinde çıktı alabilirler. Oyunda puanlama ve hamle sistemi vardır. Oyunda açık ve koyu temada puzzle parçalarınızı tamamlayabilirsiniz. Oyun, Heroku üzerinden hem frontendi ve backendi deploy edilmiştir.

***AnahtarKelimeler*— puzzle, bağlı veri yapısı, kullanıcı arayüzü, .txt dosyası, skor, hamle, karıştır butonu**

I. GİRİŞ

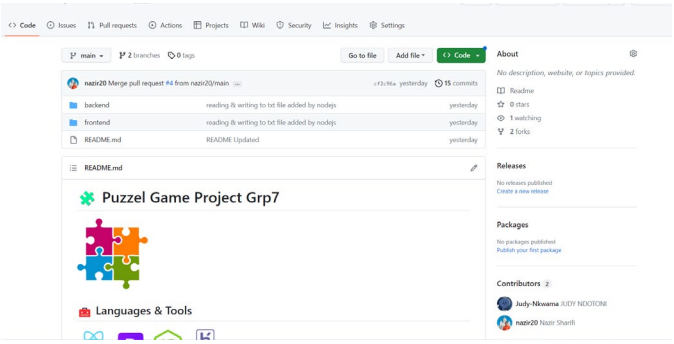
Proje kapsamında 16 parçadan oluşan kare puzzle oyunu tasarlanmıştır. Kullanıcı istediği resmi yükleyerek kendi puzzleını oluşturabilir. Oyunda en az bir adet doğru puzzle parçası denk gelene kadar kullanıcı karıştır butonuyla oyuna başlamaktadır. Oyun sırasında yapılan doğru hamle için +5 puan ve her hatalı hamle için -10 puan almaktadır. Kullanıcı, skorunu sayfanın sol kısmında güncel olarak uygulama tarafından txt formatında otomatik kaydeder. Eğer ki kullanıcı oyuna devam etmek isterse bir üst levele geçiş yapabiliyor. 6*6, 8*8 gibi üst seviyelere geçebildiği gibi direkt yapmış olduğu skor uygulama tarafından otomatik kaydedilip oyunu sonlandırabilir.

II. PROBLEM

İsterlerde bölünmüş puzzle parçalarının doğru yerleşimleri için bağlı veri yapısı(linked list) kullanılması ve yapılan hamle sonucunda kullanıcıların oyun bitiminde puanlarını txt formatında skor tablosu adı altında en iyiden en aza doğru sıralandırılması istenilmiştir. İsterlere ek olarak oyuna level atlama opsiyonunu geliştirdik. Kullanıcı oyunu bitirebilir veya bir üst seviyeye geçebilir. Default olarak 4*4 olan oyun ilerledikçe 6*6, 8*8 gibi daha da resimler bölünmektedir.

III. PROJE PLANLAMA

Projede planlama olarak mail ve çevrimiçi uygulamalar ile haberleşme sağlandı. Google Meet ve Discord uygulamaları kullanıldı. Proje geliştime süreci ve tamamlanması Github üzerinden devam etmiştir.



Resim- 1

IV. KULLANILAN TEKNOLOJİLER

Yazılımı geliştireceğimiz teknolojilere ise gruptaki çoğu kişinin hâkim olduğu ve proje içinde en iyi şekilde geliştirme sağlayacak dil seçimi yaptık. Projenin geliştirilmesinde kullanılan teknolojiler aşağıdaki gibidir.

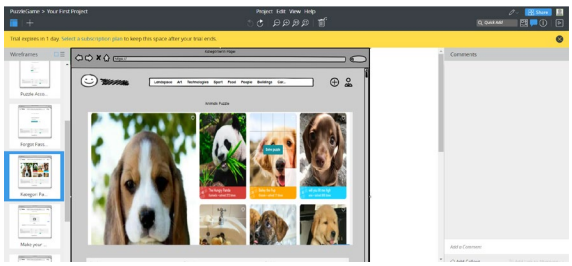
- NODE.JS,EXPRESS.JS,(BACK-END)
- HTML-CSS-BOOTSTRAP-REACT.JS(FRONT-END)

1) NODE.JS,EXPRESS.JS(BACK-END):Express.js modülü Node.js tabanlı bir web uygulama sunucu çatısıdır. Express.js'in sunduğu sınırsız http yardımcı araçları ve katmanlar sayesinde sağlam bir API oluşturmak oldukça hızlı ve kolaydır. Express.js dinamik web siteleri geliştirmek için kullanılan açık kaynak kodlu yazılım demetidir.

2) HTML-CSS-BOOTSTRAP-REACT.JS(FRONT-END): Bu teknolojiyi kullanmamızın amacı uygulamanın temel hatlarını oluşturmaktı. GUI için React.js kullanılmıştır.

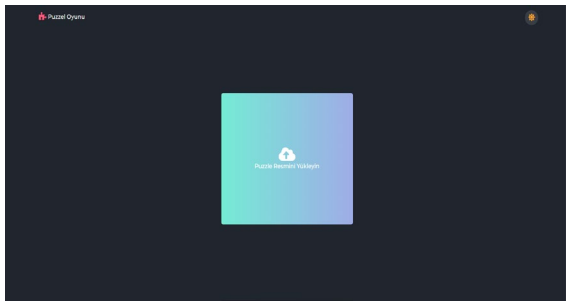
V. UYGULAMA SAYFALARI

İlk olarak projeye başlamadan önce projenin sayfalarının nasıl olacağına dair bir plan belirleme amacıyla ilk adımda WireFrameler oluşturduk. Daha sonrasında da bu WireFramelere göre projemizi geliştirmeye başladık. WireFramleri tasarlarken BALSEMIQ internet sitesinden yararlandık. Bu site sayesinde kolay ve etkileyici sayfalar tasarladık.

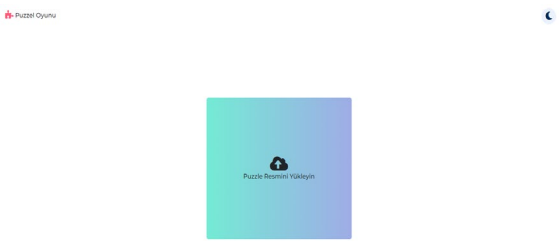


Resim- 2

1) UYGULAMA SAYFA TASARIMLARI:

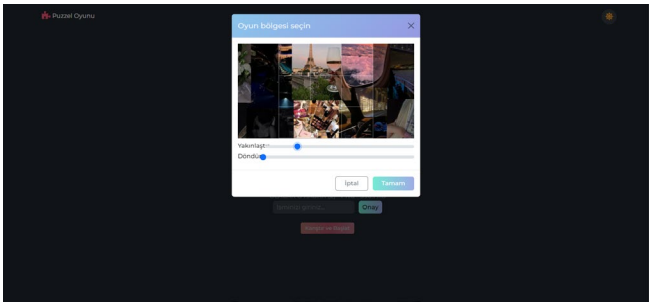


Resim- 3



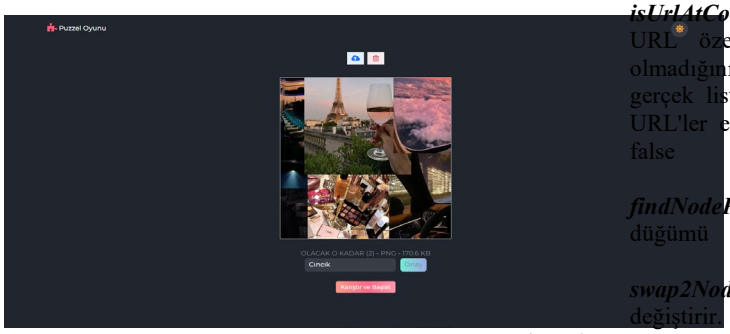
Resim- 4

- Web sitesinde görüldüğü üzere bu sayfa ile karşılaşıyorsunuz. Kullanıcının isteğine göre açık tema ve koyu tema tercih edebilir. Burada dosya yoluyla özelleştirmek istediği puzzle oyunu için resmi yüklemektedir.



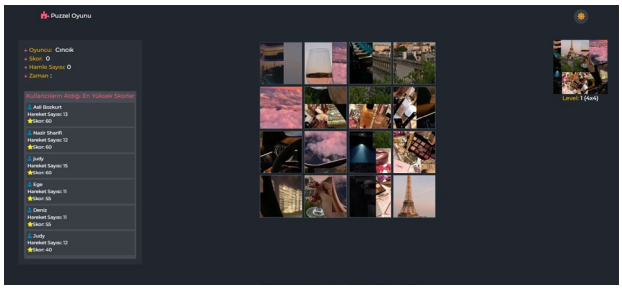
Resim- 5

- Default olarak 16 parçaya bölünecek olan puzzle için boyut ayarlaması ve döndürülmesi bu sayfada yapılmaktadır.



Resim- 6

- Bu kısımda oyundaki skoru kayıt altına alabilmek için kullanıcıdan bir isim almaktadır. Eğer ki farklı bir resim kullanmak istenilirse sil butonuna basarak tekrardan resim dosyası yükleyebilmektedir. Sonraki aşamada karıştır ve kaydet butonuna basarak bir sonraki sayfaya geçiş yapmaktadır.



Resim- 7

Bu sayfadaki ss için görüldüğü üzere 16 parçalık puzzleın sol en üst kısmı doğru yerleşimde olduğu için sabit kalmıştır. Puzzle parçalarının doğru yerleşimleri için linked list yani bağlı veri yapısından yararlanılmıştır. 16 yapboz bloğunun bir parçasına karşılık gelen bir bağlantılı liste düğümü oluşturuyoruz. Her düğümün iki özelliği vardır:

I. Temsil ettiği ana resmin parçasının url'si olan url(veri)
II. Bir sonraki düğüme (sonraki resim parçası) bağlantı.

Kullanıcı hareketinin doğru olup olmadığını kontrol etmek için, her bir parça doğru yerde olacak şekilde, resmin 16 parçasından oluşan bir Bağlantılı liste oluşturuyoruz. Daha sonra, her kullanıcı hareketinden sonra güncellenecek başka bir bağlantılı liste oluştururuz. Kullanıcı iki düğümü değiştirdiğinde. Güncellenmiş bağlantılı listenin durumunu, oyun başlamadan önce yakalanan sıralı listeyle karşılaştırıyoruz. Bu sonuca göre hareketin doğru olup olmadığına, kullanıcının oyunu kazanıp kazanmadığına vb. karar veririz.

_PuzzelBlock: sınıfı, tek bir veri parametresi alan bir yapılandırıcı ile tanımlanır. İki özellik içeren bir nesneyi başlatır: data ve next. Data özelliği gerçek verileri depolar, next özelliği ise başlangıçta null olarak ayarlanır.

LinkedList: sınıfı, baş, kuyruk, aktif düğüm, karışık liste, skor ve hamle sayısı gibi birkaç özelliği null veya 0 olarak ayarlayan bir yapılandırıcı ile tanımlanır.

LinkedList: sınıfı, birçok yöntem içerir, bunlar arasında:

addElement(veri): Bu yöntem, bir öğeyi bağlı listeye ekler. Bir veri parametresi alır ve bu veri ile yeni bir _PuzzelBlock nesnesi oluşturur. Bağlı liste boşsa, yeni öğe liste başı ve kuyruğu olur. Aksi takdirde, listenin sonuna eklenir.

isUrlAtCoorectPlace(nodeToCheckUrl): Bu yöntem, belirli bir URL özelliğine sahip bir düğümün doğru konumda olup olmadığını kontrol eder. nodeToCheckUrl parametresini alır ve gerçek liste ve karışık liste konumundaki URL'leri karşılaştırır. URL'ler eşleşirse, yöntem true değerini döndürür, aksi takdirde false değerini döndürür.

findNodePerUrl(url): Bu yöntem bir URL alır ve o URL'yi içeren düğümü döndürür.

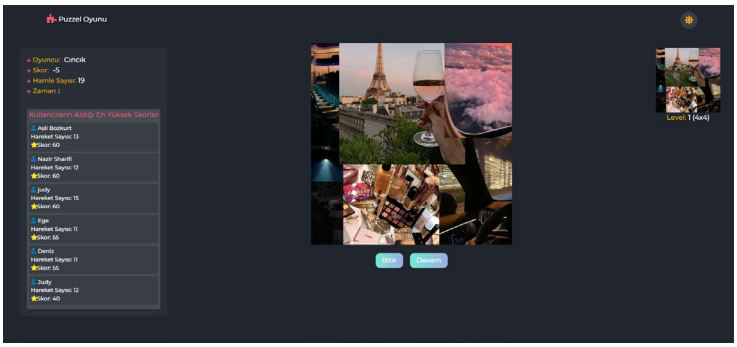
swap2Nodes(node1, node2): Bu yöntem, iki düğümün verilerini değiştirir. İki düğümü parametre olarak alır ve veri özelliklerini değiştirir.

swap2NodesPerUrl(url1, url2): Bu yöntem, iki URL alır ve ilgili düğümlerin verilerini değiştirir. Karışık listedeki düğümlerin verilerini değiştirmek için swap2Nodes yöntemini çağırır.

mixElemnts(): Bu yöntem, düğümleri alfabetik olarak sıralayarak listedeki öğeleri karıştırır. Her düğümün URL'sini bir sonraki düğümün URL'siyle karşılaştırarak bunu yapar ve bir sonraki düğümün URL'si alfabetik olarak önce geliyorsa, düğümleri değiştirir.

genMixedList(): Bu yöntem, mevcut listenin karışık bir sürümünü oluşturur. Mevcut listeyi kopyalar ve mixElemnts yöntemini kullanarak öğelerini karıştırır.

map(callback): Bu yöntem, Array.map yöntemini taklit eder. Bağlı listeyi dolaşır ve her düğümün verisine verilen



Resim- 8

- Puzzle tamamlandığında sol kısımda txt formatında skor tablosunda kullanıcılar ve puanları yer almaktadır. Txt formatında tutmak için yapılan code işlemleri sırasıyla:

1)En Yüksek Skorları enyüksekscore.txt dosyasında tutmak

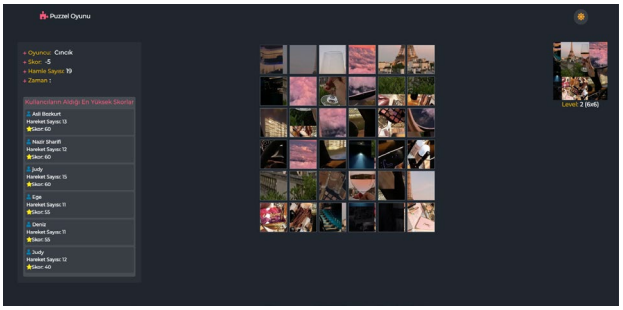
Puzzle oyunumuzda kullanıcın adını, aldığı skoru ve kaç hareket ile puzzle'i çözdüğünü tutabilmek için enyuksekscore.txt dosyasını oluşturduk. Bu işlemi yapabilmek için kullandığımız tekonoloji ise yine JavaScript dilinin bize sunduğu backend framework'u olan Node.js ve Express.js'i kullandık.

Kullanıcı puzzle oyunu için sisteme resim yükledikten sonra, kullanıcından kayıt tutabilmek adına kullanıcı'dan bir isim isteniyor.

Daha sonra, uygulamanın içeresinde tandımladığımız Context API kullanarak bu ismi bir state'te tutuyoruz. Kullanıcı oyuna başlar ve parçaları doğru yere yerleştirmeye çalışıyor. Bu esnada kullanıcın kaç hareket ettiği ve aldığı puan anlık olarak hesaplanıp kullanıcı'ya
gösteriliyor. Tüm parçalar doğru yere yerleştirildikten sonra, oyun sona erer. Oyun bittiğinde yazdığımız backend'e bir POST REQUEST
gerçekleşiyor ve /api/data dizine kullanıcının verileri gönderiliyor. Burada ise Nodejs'in varsayılan olarak tanımladığı file-system modulunu kullanarak kullanıcının verilerini enyüksekscore.txt dosyasına yazdırıyoruz. Backend'te yazdığımız mantığa göre, eğer kullanıcın enyüksekscore.txt dosyasında önceden kayıdı varsa ki bunu kullanıcın seçtiği isim ile kontrol ediyoruz. Eğer kayıdı varsa, yazdığımız mantığa göre bu sefer sistem kullanıcın eski ve yeni aldığı skoru karşılaştırıyor. Yüksek olan hangisi ise, o enyüksekscore.txt dosyasında yazdırılacak.

2) GUI'de En Yüksek Skorları Göstermek

Projenin isterlerinde de yazıldığı gibi oyun her açıldığında, tasarladığınız arayüzün bir kısmında en yüksek puanlar azalan sırada gösterilmelidir. Bu işlemi gerçekleştirilebilmek için yazdığımız backend koduna yani /api endPiont'ine bir GET REQUEST gönderiliyor. Kullanıcı, Puzzle Oyunu için sisteme resim yükledikten sonra hemen hemen /api endPoint'ine bir GET ReQUEST gerçekleştiriliyor. /api endPiont'inde ise Nodejs'in varsayılan olarak tanımladığı file-system modulunu kullanarak enyüksekscore.txt dosyasında var olan tüm kayıtları bir dizi haline RESPONSE olarak geri gönderiyoruz. Daha sonra frontend'te React.js'in useState, useEffect Hooklarını kullanarak bu kayıtları önce en yüksek alınan skorlar şeklinde sıralayıp kullanıcıya bir tablo halinde gösteriyoruz.



Resim- 9

- Devam et butonuna tıkladığımızda bir üst level olan 6*6 ‘lık versiyonuyla puzzle oyununa devam etmektedir. Eğer ki bitirsek oyunda yaptığımız skor otomatik kaydedilip tekrar resim yükleme sayfasına gitmektedir.

REFERENCES

https://www.helpfulgames.com/