Bulut Bilişimde Sanallaştırma Teknolojilerine Giriş Dersi Proje Raporu

Hacer Sueda EFE
211307060
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi - Kocaeli Üniversitesi
İmzit, Kocaeli /Türkiye
suedaefe@gmail.com

Github : https://github.com/suedaefe/BulutBilisim
Dockerhub : https://hub.docker.com/repository/docker/sueda06/snakegame/general
Google Drive : https://drive.google.com/drive/folders/16QkSmdlcdQ5t9VLeyHqzkE4qBu8DpmzT?usp=sharing
AWS : http://13.51.109.159/nowlogin.html

I. ÖZET

Bu rapor geliştirilen web tabanlı Snake oyunu için geliştirme aşamalarını, bu oyunun phpmyadmin ile bir mysql veritabanına bağlanmasını bu veritabanına veri gönderip veritabanından veri alma aşamalarını (kullanıcı kayıt ve girişi, en yüksek skor tutulması, leaderboard gibi özellikler); bu uygulamanın Docker kullanılarak konteynerlaştırılmasını ve oluşturulan bu konteynerların AWS bulut sağlayıcısına yüklenerek sanallaştırılmasını içerir.

II. ABSTRACT

This report contains the development stages for the web-based Snake game, connecting this game to a mysql database via phpmyadmin, sending data to this database and receiving data from the database (features such as user registration and login, keeping the highest score, leaderboard); It involves containerizing this application using Docker and virtualizing these created containers by uploading them to the AWS cloud provider.

III. GİRİŞ

A. Bulut Bilişim ve Sanallaştırma Teknolojileri Nedir?

Bulut bilişim, internet üzerinden bilgi işleme kaynaklarına (örneğin, sunucular, depolama, ağlar, yazılımlar) erişim sağlayan bir hizmet modelidir. Bu kaynaklar, kullanıcılara herhangi bir yerden ve herhangi bir cihazdan erişilebilen sanal ortamlarda barındırılır.
Hizmet modelleri kısaca aşağıda açıklanmıştır :

- IaaS : Fiziksel donanım depolama ve ağ kaynakları gibi temel altyapı kaynakalrı sağlar.
- PaaS : Uygulama geliştirme için platform araçları sağlar
- SaaS : Kullanıcılara internet üzerinden uygulama yazılımı sunar.

Avantajları:

- Maliyet esnekliği
- Kaynakların esnek ve ölçeklenebilir olması
- Erişim kolaylığı
- Güvenlik
- Hızlı dağıtım ve güncelleme imkanı

Sanallaştırma, fiziksel donanımın ve kaynakların sanal ortamlarda çalıştırılmasını sağlayan bir teknoloji konseptidir. Bu, bir bilgisayarın, sunucunun veya depolama cihazının birden çok sanal örneğini çalıştırmak anlamına gelir.

Avantajları :

- Kaynak verimliliği
- İzolasyon
- Güvenlik
- Yedekleme ve kurtarma
- Hızlı dağıtım

Sanal Makineler, fiziksel bir bilgisayar üzerinde çalışan sanal bir işletim sistemi ve uygulama yazılımlarının kombinasyonudur.

Konteynerlar, sanal makinelerden daha hafif ve hızlıdır, uygulamaları izole bir şekilde çalıştırarak taşınabilirliği artırır.

Avantajları :

- Donanımın daha etkin kullanımı
- Hızlı dağıtım ve ölçeklendirme
- Yedekleme ve kurtarma
- Çeviklik ve kaynak yönetimi esnekliği

B. *Projenin Amacı*

Proje konusu olarak klasik bir oyun olan Snake'in seçilmesinin ana sebebi kullanıcılara dijital bağlamda bir nostalji yaşatmaktı. Snake oyunu çıktığı zamanlarda bir rekabet ortamı yaratılması pek mümkün değildi bu şekilde web tabanlı bir Snake oyunu geliştirmek ve oyuncuların oyun içi performanslarını kaydedip leaderboard gibi özellikler sunmak kullanıcılar arasında bir rekabet ortamı oluşturur ve bu nostaljik oyunu oynarken daha keyifli bir deneyim yaşatır diye düşünüldü. Oyunun ayrıca eğitici boyutu da bulunmaktadır, kullanıcıları eğlendirirken aynı zamanda refleks becerileri geliştirme ve strateji oluşturma yeteneklerini artırır bu yüzden çocukların gelişimi için de çok faydalı bir oyundur. Kullanıcıların dikkatlerini artırmalarına ve hızlı kararlar almalarına olanak tanır.

IV. KULLANILAN TEKNOLOJİLER

Web Geliştirme : Oyun web için geliştirildiği için temel olarak HTML, CSS ve JavaScript kullanılarak geliştirilmiştir.

Veritabanı : MySQL veritabanı PHP ve PHPMyAdmin aracılığıyla kullanılarak oyuncuların giriş bilgileri ve en yüksek skorları saklamak için kullanılmıştır.

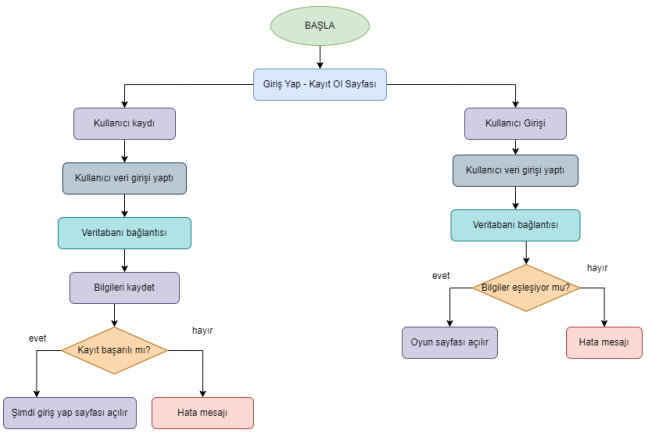
Web Sunucu : XAMPP ile bir lokal web sunucu (apache) oluşturulmuş, projenin geliştirme aşamasında kullanılmıştır.

Docker : Proje Docker teknolojisi kullanılarak konteynerlaştırılmıştır.

AWS Cloud : Proje AWS Cloud üzerinde bir sanal sunucuya yüklenmiştir.

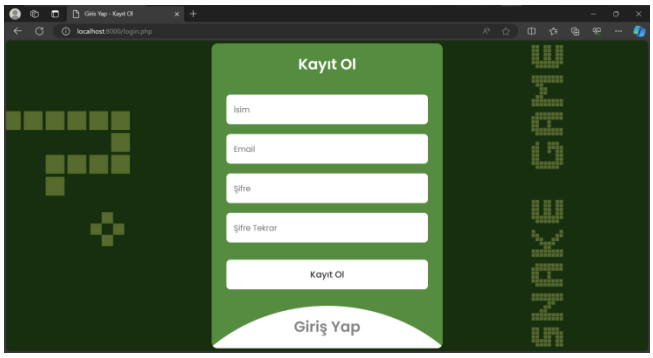
A. *Web Geliştirme, Veritabanı ve Web Sunucu*

A.1 Giriş Yap-Kayıt Ol Sayfası

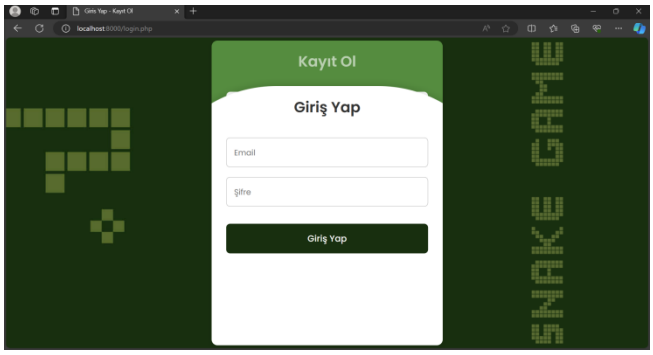


Şekil 1 - Akış Diyagramı

Giriş yap ve kayıt ol formlarının ikisi de aynı sayfada bulunmaktadır. Giriş yap butonuna tıklandığında alt kısımdaki form açılarak giriş yap kısmı görüntülenmektedir. CSS kısmında gerekli ayarlamalar yapılmış, renk ve form alanlarının border radius padding gibi elementleri ayarlanmıştır.



Şekil 2 - Kayıt Ol



Şekil 3 - Giriş Yap

Bunların çalışma prensibi için iki adet script mevcuttur.

```
<script>
    const wrapper = document.querySelector(".wrapper"),
        signupHeader = document.querySelector(".signup header"),
        loginHeader = document.querySelector(".login header");
    loginHeader.addEventListener("click", () => {
        wrapper.classList.add("active");
    });
    signupHeader.addEventListener("click", () => {
        wrapper.classList.remove("active");
    });
</script>
```

Şekil 4 - Script

```
JS script.js > switchers.forEach() callback
1  const switchers = [...document.querySelectorAll('.switcher')]
2
3  switchers.forEach(item => {
4      item.addEventListener('click', function () {
5          switchers.forEach(item => item.parentElement.classList.remove('is-active'))
6          this.parentElement.classList.add('is-active')
7      })
8  })
```

Şekil 5 - Script

Kullanıcı kaydı almak ve giriş yapmak gibi işlemleri gerçekleştirmek için ise php kullanılmıştır. Ama bundan önce xamppden phpmyadmine lokalde giriş yaptıktan sonra bir veritabanı ve tablo oluşturulmuş olması gerekir. Bu aşamada proje daha henüz konteynerlaştırılmadığı için projeyi test etme amaçlı phpmyadmini kullanabilmek için

xampp kullanılmış, veritabanı bu şekilde lokalde çalıştırılmıştır. Bu projede kullanıcılar için kullanılan tablo aşağıdaki şekildedir.

```
CREATE TABLE user (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE,
    password_hash VARCHAR(255)
);
```

Şekil 6 - Kullanıcı Tablosu

Kayıt sayfasının bulunduğu dosyanın üst kısmına veritabanı bağlantısı için gerekli php kodları yazıldı, mysqli kullanıldı. Ayrıca giriş durumunda session start ile oturum başlatıldı kullanıcı bilgilerinin alınıp oyun.php sayfasına yönlendirme yapılması komutu verildi.

```
<?php
$connect = mysqli_connect(
    'db',       // service name
    'php_docker', // username
    'password', // password
    'php_docker' // db table
);

if (!$connect) {
    die("Connection failed: " . mysqli_connect_error());
}

$is_invalid = false;

if ($_SERVER["REQUEST_METHOD"] === "POST") {

    $email = $connect->real_escape_string($_POST["email"]);

    $query = "SELECT * FROM user2 WHERE email = '$email'";
    $result = mysqli_query($connect, $query);

    if ($result) {
        $user2 = $result->fetch_assoc();

        if ($user2 && password_verify($_POST["password"], $user2["password_hash"])) {

            session_start();

            session_regenerate_id();

            $_SESSION["user_id"] = $user2["id"];

            header("Location: oyun.php");
            exit;
        }
    }

    $is_invalid = true;
}
?>
```

Şekil 7 - Veritabanı Bağlantısı ve Giriş Yap Php

Kayıt için ise ayrı bir php dosyası oluşturuldu bu dosyada veritabanına veri gönderme işlemleri yapıldı. Bu dosyaya bu sayfadan yönlendirme yapılması için ise form action özelliği kullanıldı bilgilerin alındığı form şekildeki şekilde ayarlandı.

```
<form action="signup.php" method="post" id="signup"
novalidate>
```

Şekil 8 - Kayıt Ol için Form Action

Bu php dosyasında ise yine veritabanı bağlantısı yapıldı ve bazı kısıtlar belirlendi, şifrenin 8 karakterden az olmaması en az bir harf içermesi, emailin uygun formatta olması gibi.

```
<?php

if (empty($_POST["name"])) {
    die("isim girmek zorunludur");
}

if ( ! filter_var($_POST["email"], FILTER_VALIDATE_EMAIL)) {
    die("gecerli email girin");
}
```

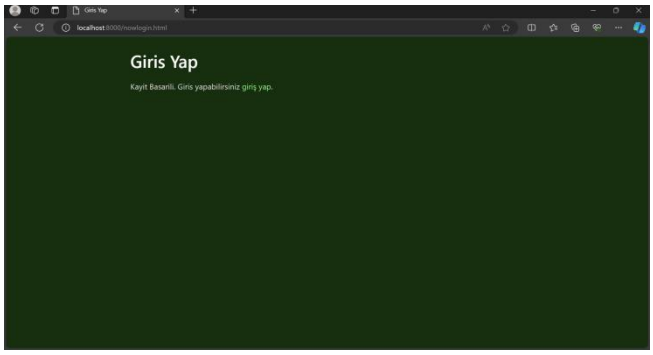
Şekil 9 - Geçerli email vb. için kısıtlar

Bu alınan bilgiler için bir sql insert sorgusu yazıldı php post özelliği ile de şekil 10'da gösterildiği gibi veritabanındaki tabloya bu bilgilerin kaydı yapıldı.

```
$sql = "INSERT INTO user2 (name, email, password_hash)
        VALUES (?, ?, ?)";

$stmt = $connect->stmt_init();

if ( ! $stmt->prepare($sql)) {
    die("SQL error: " . $mysqli->error);
}

$stmt->bind_param("sss",
                  $_POST["name"],
                  $_POST["email"],
                  $password_hash);

if ($stmt->execute()) {

    header("Location: nowlogin.html");
    exit;
```

Şekil 10 - Kayıt Ol Php

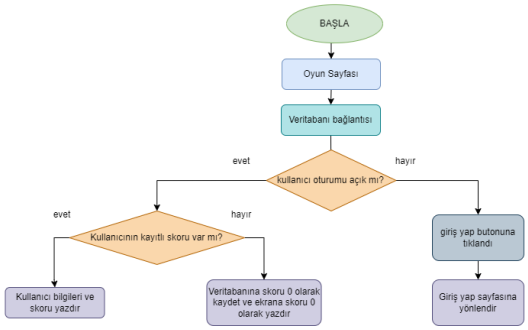
Kaydın başarılı olduğu durumda kullanıcı bir sayfaya yönlendirilmektedir bu sayfadan butona tıklanarak giriş yap sayfasına yönlendirilmektedir.



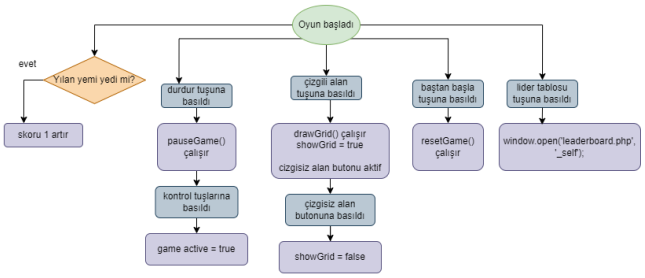
Şekil 11 - Kayıt Başarılı

Bir adet çıkış yapmak için de bir php dosyası oluşturuldu, burada da session destroy komutu yazıldı çıkıştan sonra header(location : login.php) ile de giriş yap sayfasına yönlendirilmesi sağlandı.

A.2 Oyun Sayfası

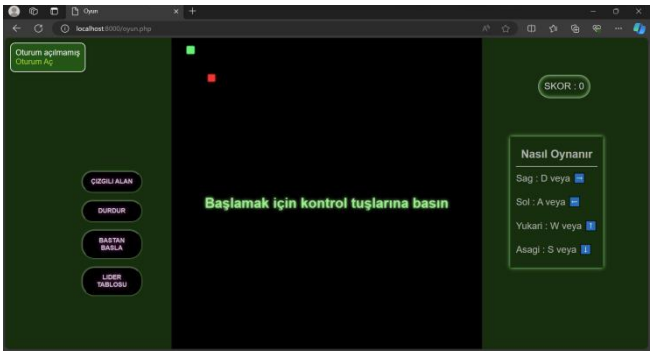


Şekil 12 - Akış Diyagramı



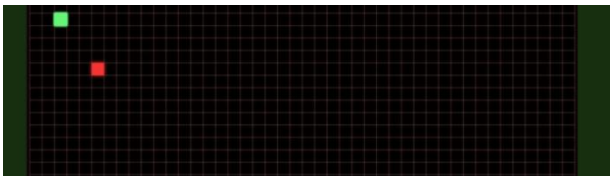
Şekil 13 - Akış Diyagramı

Oyun sayfasına gelindiğinde ortada oyun alanı için bir panel karşımıza çıkıyor. Panelin sağ tarafında oyunun nasıl oynanacağıyla ilgili bir bölüm ve skoru tutan bir bölüm bulunmakta. Sol tarafta ise üstteki kutucukta kullanıcı giriş yapmışsa kullanıcı bilgileri yer alıyor giriş yapmamışsa giriş yapmak için olan sayfaya yönlendiren bir buton mevcut. Ayrıca oyunu ilk defa oynayanlar olursa diye veya çocuklar için oynamayı kolaylaştırmak amacıyla panel zeminini kareli kağıt gibi çizgili hale getiren bir buton, oyunu durdurmaya yarayan bir buton, baştan başlamak için ve lider tablosuna gitmek için de ayrıca butonlar bulunuyor. Giriş yapmadan da oyunun oynanması mümkün, fakat en yüksek skor tutulmadığı için lider tablosuna kayıt söz konusu değil.



Şekil 14 - Oyun Sayfası (oturum açılmadı)

Şekil 15 - Oturum Açıldıktan Sonra Kullanıcı Bilgileri



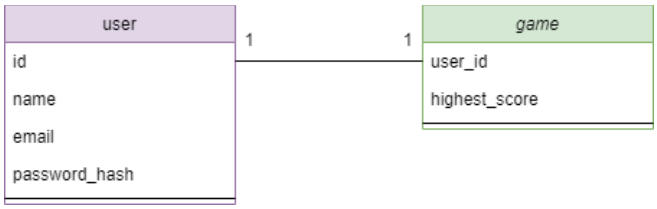
Şekil 16 - Kareli Oyun Alanı

Oyun kısmının kodlarına gelmeden önce oyunda her kullanıcının en yüksek skorunu veritabanında tutmak için ilk oluşturulan user tablosundaki kullanıcı id'sine foreign key ile bağlı yeni bir tablo oluşturmak gerekli bu projede şekil 17deki şekilde bir tablo kullanıldı

```
CREATE TABLE game (
    user_id INT,
    highest_score INT,
    FOREIGN KEY (user_id) REFERENCES user(id)
);
```

Şekil 17 - Skor Tutmak İçin Tablo

Sonuç olarak tablolar bire bir ilişkilidir.



Şekil 18 - Skor Tutmak İçin Tablo

Oyun kodunun en başında veritabanı bağlantısı kodları yine yazıldı ardından oturum başlatıldı. Eğer kullanıcı ilk defa kayıt olmuşsa normal şartlarda game tablosunda herhangi bir kaydı bulunmuyor dolayısıyla bu durumda bilgi gösterilmesi gereken alanda bir sürü hata mesajlarıyla karşılaşıldı çünkü önce oyunun oynanması oyun game over olduktan sonra skorun kullanıcı id ile birlikte veritabanına eklenmesi şeklinde işliyordu kod, bu sorunu ortadan kaldırmak için kullanıcı oturumu başlatıldıktan sonra game tablosunda bu kullanıcının kaydı var mı yok mu kontrol eden bir php kodu yazıldı, eğer kaydı yoksa kayıt oluşturup skoru default olarak 0 ataması sağlandı. Bu veritabanında user tablosuna after insert trigger oluşturarak da yapılacabilecek bir işlemdi ama ben bu şekilde yapmayı tercih ettim. Yani sql ile de çözümlenebilecek bir sorun.

```
// Oturumu başlat
session_start();

// Oturumda kullanıcı bilgileri var mı kontrol et
if (isset($_SESSION['user_id'])) {
    // Kullanıcının oturum bilgilerini al
    $userId = $_SESSION['user_id'];
    echo '<script>const userId = ' . $userId . ';</script>';


    // Kullanıcının "game" tablosunda kaydı var mı kontrol et
    $checkQuery = "SELECT user_id FROM game WHERE user_id = $userId";
    $checkResult = mysqli_query($connect, $checkQuery);

    if (mysqli_num_rows($checkResult) == 0) {
        // Eğer kayıt yoksa, yeni bir kayıt ekle
        $insertQuery = "INSERT INTO game (user_id, highest_score) VALUES ($userId, 0)";
        $insertResult = mysqli_query($connect, $insertQuery);

        if (!$insertResult) {
            echo "Oyun tablosuna kayıt eklenirken bir hata oluştu: " . mysqli_error($connect);
        }
    }
```

Şekil 19 - Skor Kaydı Yoksa Otomatik 0 Yaz

Daha sonra kullanıcının veritabanındaki isim user id ve en yüksek skoru alınarak ekranın sol üst köşesindeki alana yazdırıldı. Bunun için sql sorgusu yazılırken iki tablodan da veri alınması gerektiği için inner join işlemi yapıldı.

```php
<?php
// Oturumda kullanıcı bilgileri var mı kontrol et
if (isset($_SESSION['user_id'])) {
    // Kullanıcının oturum bilgilerini al
    $userId = $_SESSION['user_id'];

    // Kullanıcı bilgilerini sorgula
    $query = "SELECT id, name, highest_score FROM user2 u
              INNER JOIN game g on g.user_id=u.id
              WHERE id = $userId";
    $result = mysqli_query($connect, $query);

    // Sorgu sonuçlarını kontrol et
    if ($result) {
        // Veritabanından alınan verileri kullanarak HTML çıktısı oluştur
        $row = mysqli_fetch_assoc($result);
        echo "İsim: " . $row['name'] . "<br>";
        echo "ID: " . $row['id'] . "<br><br>";
        echo "En yüksek: " . $row['highest_score'] . "<br><br>";


        echo '<p><a href="logout.php">Çıkış Yap</a></p>';
    } else {
```

Şekil 20 - Kullanıcı Bilgilerini Yazdır

Skoru veritabanına kaydetmek için ayrı bir php doyası bulunmakta dolayısıyla oyunun script kısmından bir verinin alınıp php dosyasına gönderilebilmesi için AJAX kullanıldı. Alınan skor score.php adındaki bu skor kaydetmek için oluşturulmuş olan php'ye gönderildi. Şekil 21'deki fonksiyon oluşturuldu ve bu fonksiyon game over yazısı kullanıcıya verildikten sonra çağrıldı.

```javascript
function saveScore(skor) {

    const uid = userId;
    const xhttp = new XMLHttpRequest();
    const url = 'score.php';
    xhttp.open("POST", url, true);
    xhttp.setRequestHeader("Content-Type", "application/json");
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState === 4 && xhttp.status === 200) {
            console.log(xhttp.responseText);
        }
    };

    const data = {
        uid: uid,
        skor: skor
    };

    xhttp.send(JSON.stringify(data));
```

Şekil 21 - Skoru js'den php'ye Gönder (AJAX)

Skoru kaydeden php kodunda ise veritabanı bağlantı kodları yazıldıktan sonra şekil 22'deki gibi json_decode kullanılarak javascriptten gelen veriler input alındı ve uid ve skor adındaki değişkenlere atandı.

```php
$d = json_decode(file_get_contents("php://input"),
true);
$uid = $d['uid'];
$skor = $d['skor'];
```

Şekil 22 - Ajaxtan Gelen Verileri php'de Al

Daha sonra mevcut skor varsa yeni skor mevcuttan büyük mü kontrol ediliyor değilse işlem yapılmıyor büyükse yeni skorun yerine sql update komutuyla tablodaki skorun güncellenmesi sağlanıyor.

```php
$Scoresql = "SELECT * FROM game WHERE user_id = '$uid'";
$result = $connect->query($Scoresql);

if ($result->num_rows > 0) {
    // Kullanıcının daha önce skoru var, yeni skor eski skordan büyük mü kontrol et
    $row = $result->fetch_assoc();
    $Scoreo = $row['highest_score'];

    if ($skor > $Scoreo) {
        // Yeni skor eski skordan büyük, güncelleme işlemi yap
        $ScoreUpdatesql = "UPDATE game SET highest_score = '$skor' WHERE user_id = '$uid'";
        if ($connect->query($ScoreUpdatesql) === TRUE) {
            echo "Skor guncellendi";
        } else {
            echo "Hata: " . $connect->error;
        }
    } else {
        echo "Yeni skor eski skordan kucuk veya esit";
    }
```

Şekil 23 - Yeni Skor Mevcuttan Büyük mü

Oyun panelini çizgili hale getirmek için şekil 24'de verilen fonksiyon yazıldı. Yılanın oluşturulması ve pozisyonunun güncellenmesi için gerekli kodlar da şekil 25'de verilmiştir. Şekil 26'da ise klavye kontrolleriyle iligli örnek bir kod verilmiştir.

```javascript
const drawGrid = () => {
  ctx.beginPath();
  for (let i = 0; i <= grid_line_len; i += cellSize) {
    ctx.moveTo(i + pGrid, pGrid);
    ctx.lineTo(i + pGrid, grid_line_len + pGrid);
  }
  for (let i = 0; i <= grid_line_len; i += cellSize) {
    ctx.moveTo(pGrid, i + pGrid);
    ctx.lineTo(grid_line_len + pGrid, i + pGrid);
  }
  ctx.closePath();
  ctx.strokeStyle = canvasStrokeColor;
  ctx.stroke();
};
```

Şekil 24 - Çizgili Panel

```javascript
const drawSnake = () => {
  //loop through our snakeparts array
  snakeParts.forEach((part) => {
    part.draw();
  });

  snakeParts.push(new Tail(head.x, head.y));

  if (snakeParts.length > tailLength) {
    snakeParts.shift(); //remove furthest item f
  }
  head.color = randomColor();
  head.draw();
};

const updateSnakePosition = () => {
  head.x += head.vX;
  head.y += head.vY;
};
```

Şekil 25 - Yılanı Oluştur ve Pozisyonunu Güncelle

```javascript
const changeDir = (e) => {
  let key = e.keyCode;

  if (key == 68 || key == 39) {
    if (head.vX === -1) return;
    head.vX = 1;
    head.vY = 0;
    gameActive = true;
    return;
  }
```

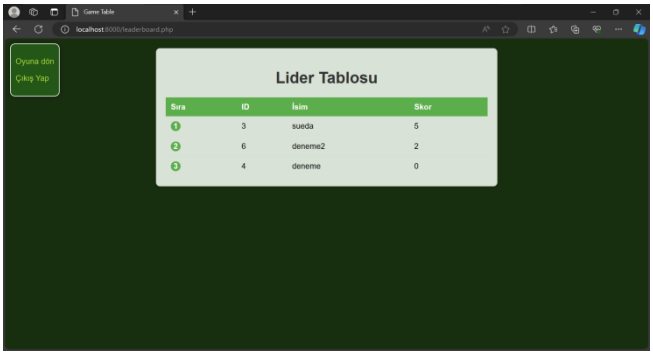
Şekil 26 - Klavye Kontrolleri

```
const foodCollision = () => {
  let foodCollision = false;
  snakeParts.forEach((part) => {
    if (part.x == food.x && part.y == food.y) {
      foodCollision = true;
    }
  });
  if (foodCollision) {
    food.x = Math.floor(Math.random() * cellCount);
    food.y = Math.floor(Math.random() * cellCount);
    score++;
    tailLength++;
  }
};
```

Şekil 27 - Yılan yemi yedi mi

A.3 Lider Tablosu

Lider tablosu için klasik bir tablo görünümü kullanıldı. Kişinin sırası ID'si adı ve skoru yazdırıldı.



Şekil 28 - Lider Tablosu

Tablonun kodunun başında tekrar mysqli ile veritabanına bağlantı kodları yazıldıktan sonra inner joinle iki tabloyu birleştiren bir sql sorgusu yazılarak istenen veriler çekildi. Bu da mysqli query olarak bir response değişkenine atandı. Daha sonra tabloda tbody etiketi içerisinde mysqli fetch assoc ile response içerisindeki veriler çekildi ve şekildeki gibi php etiketleri açılarak bu çekilen veriler echo ile yazdırıldı.

```
<tbody>
    <?php
    $rank = 1;
    while ($row = mysqli_fetch_assoc($response)) :
    ?>
        <tr>
            <td><div class="rank"><?php echo $rank; ?></div></td>
            <td><?php echo $row['id']; ?></td>
            <td><?php echo $row['name']; ?></td>
            <td><?php echo $row['highest_score']; ?></td>
        </tr>
    <?php
        $rank++;
    endwhile; ?>
</tbody>
```

Şekil 29 - Lider Tablosu Verisini Tabloya Ekle

B. Docker

Bu aşamadan Docker desktop uygulaması kullanıldı. Proje dosyalarının olduğu yere bir yml uzantılı dosya oluşturuldu kod şekil 30'da verildi. Bu dosya Dockerda bir konteyner içerisine gerekli imageleri oluşturmak için kullanıldı. Www php image'ı için, db mysql veritabanı image'ı için, phpmyadmin ise phpmyadmin image'ı için gerekli bölümler. İlgili bölümlerde hangi portlarda çalışması isteniyorsa bu portlar girilmeli, eğer kullanmak istenen port doluysa hata verecektir bu hatayla projenin geliştirme aşamasında karşılaşıldı çözüm olarak farklı bir port numarası denendi ve işe yaradı.

Ayrıca dbnin volumes kısmına yazılan bölüm veritabanı içindir, docker kapatılıp tekrar açıldığında veritabanına kaydedilen bilgiler kaybedilmekte, bunu önlemek için proje dosyalarına db isminde bir klasör oluşturuldu bu kalsörün içerisine de veritabanından dışarıya aktardığımız sql dosyası kondu, bu şekilde veriler kaybedilmedi.

```
docker-compose.yml
version: '3'
services:
  www:
    image: php:apache
    volumes:
      - "./:/var/www/html"
    ports:
      - 8000:80
      - 443:443
  db:
    image: mysql:latest
    environment:
      - MYSQL_DATABASE=php_docker
      - MYSQL_USER=php_docker
      - MYSQL_PASSWORD=password
      - MYSQL_ALLOW_EMPTY_PASSWORD=1
    volumes:
      - "./db:/docker-entrypoint-initdb.d"
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - 8001:80
    environment:
      - PMA_HOST=db
      - PMA_PORT=3306
```

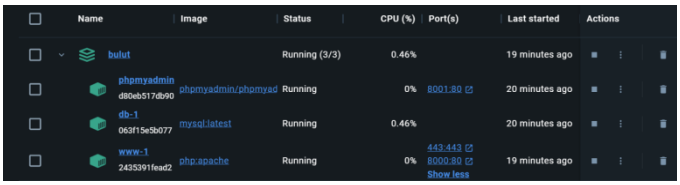
Şekil 30 - Docker-compose.yml

Bu yml dosyası hazırlandıktan sonra projenin terminal bölümüne yml dosyasının adını ve daha sonra bu dosyanın adının yanına up komutunu yazarak konteyner oluşturulur. Şekil 31'de görüldüğü üzre oluşturulduğunda terminalde bu ekranla karşılaşılır.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\bulut> docker-compose up
[+] Running 4/4
 ✓ Network bulut_default          Created
 ✓ Container bulut-db-1           Created
 ✓ Container bulut-www-1          Created
 ✓ Container bulut-phpmyadmin-1   Created
```

Şekil 31 - Docker-compose up

Docker desktopa gidildiğinde de bunların oluşturulduğu görülebilir.



Şekil 32 - Docker Desktop

Daha sonra desktop uygulamasında verilen php:apahce image'ının port linkine tıklayarak veya tarayıcıya bu portu girerek ve yanına /sayfa adı yazarak sayfayı görüntülemek mümkün. Fakat bu aşamada mysqli eklenti hatasıyla karşılaşıldı :

Fatal error: Uncaught Error: Call to undefined function mysqli_connect() in /var/www/html/oyun.php:3 Stack trace: #0 {main} thrown in /var/www/html/oyun.php on line 3

Bu hatayı gidermek için Docker desktopta php:apache image'ını terminalde açarak buraya " docker-php-ext-install mysqli " komutu yazılarak çalıştırıldı ve image baştan başlatıldı, bu şekilde sorun kalktı.

Oluşturulan imageler Docker Hub'a da yüklendi. Bunun için DockerHub'da bir repository oluşturuldu ardından bilgisayarın komut istemine aşağıdaki komutlar yazıldı. Bu komutlarla image tagler oluşturuldu ve DockerHub'a pushlandı. sueda06/snakegame repository adı tag'in ardından gelen kısım ise image'ın tag'idir.

1) docker tag mysql:latest sueda06/snakegame:latest

2) docker login

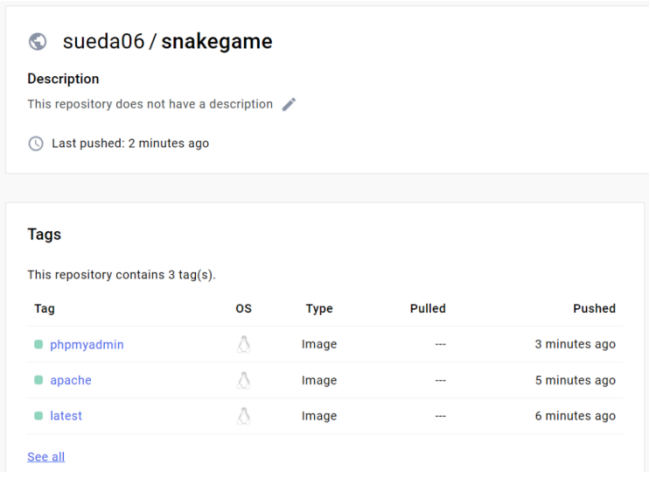
3) docker push sueda06/snakegame:latest

4) docker tag php:apache sueda06/snakegame:apache

5) docker push sueda06/snakegame:apache

6) docker tag phpmyadmin/phpmyadmin sueda06/snakegame:phpmyadmin

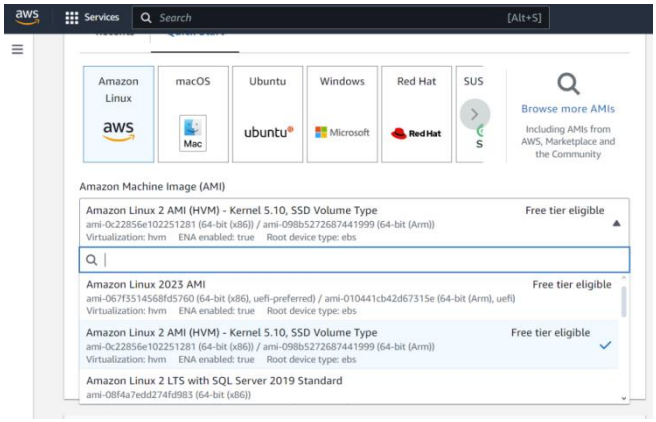
7) docker push sueda06/snakegame:phpmyadmin



Şekil 33 - DockerHub

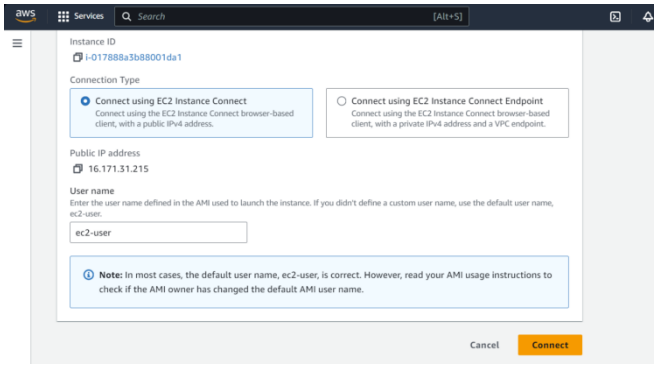
C. AWS Bulut

Bulut sağlayıcısı olarak AWS seçildi. AWS'ye kayıt olduktan sonra arama kısmına EC2 yazıldı, buradan launch instance seçeneğine tıklandı. Instance'a bir isim verilidkten sonra sanal makine için seçenekler seçildi bu proje için Kernel 5.10 Linux kullanıldı.



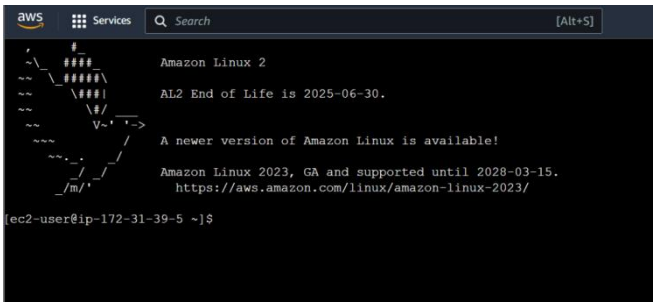
Şekil 34 - Sanal Makine

Daha sonra bir key pair oluşturuldu, tipi RSA uzantısı .pma seçildi. Bu key bilgisayara otomatik olarak indirilmekte. Network settings tarafında https işlemlerinin hepsine izin verdikten sonra launch instance seçeneğine tıklandı. Daha sonra çıkan sayfalarda verilen idlere tıklanarak connect yazısıyla karşılaşılan bir sayfaya ulaşıldı, burada connecte tıklayarak şekildeki sayfaya ulaşıldı. Burada ec-2 kullanarak bağlantı oluşturulduktan sonra konsol ekranıyla karşılaşılır.



Şekil 35 - Connect Instance

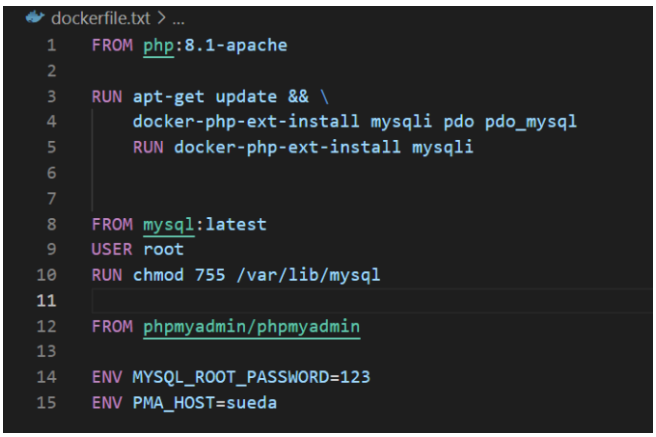
Konsol ekranı şekil 36'da verilmiştir. Burada sırasıyla aşağıda verilen komutlar yazılmıştır.



Şekil 36 - AWS Linux Konsol

sudo-amazon-linux-extras install docker
y
sudo service docker start
sudo usermod -a -G docker ec2-user
mkdir downloads
cd downloads/

Daha sonra amazondan alınan .pem uzantılı key dosyası proje dosyalarımızın olduğu konuma eklenmiştir. Ayrıca bir dockerfile.txt dosyası oluşturulmuştur.



Şekil 37 - Dockerfile.txt

Daha sonra proje dosyalarının olduğu terminalde aşağıdaki komutlar sırayla yazılmıştır.

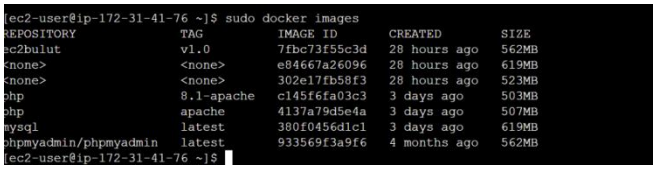
icacls bulutKey.pem /inheritance:r /grant:r "sueda:(F)"

bulutKey keyin adı sueda da bilgisayardaki kullanıcı adıdır

scp -i bulutKey.pem dockerfile.txt login.php logout.php style.css signup.php -r db ec2-user@13.51.109.159:/home/ec2-user/downloads

Dockerfile.txt nin ardından yazılanlar projenin sayfalarının ismidir, aws'ye gönderilmek istenen proje dosyalarının adı bu şekilde yazılmadılır. @den sonra gelen kısım awsdeki terminalin sol altında verilen public id'dir. Id'den sonra : nın ardından gelen kısım için aws'nin terminaline pwd yazılarak çıkan adres alınmıştır. En sonki komuttan sonra yes no şeklinde bir soru çıkmakta buna yes denmesi gerekir. Bunun ardından dosyaların başarılı bir şekilde aws'ye gönderilmiş olması gerekir. Kontrol etmek için aws terminaline ls yazılırsa gelen dosya varsa cevap olarak gösterilecektir.

Tekrar aws terminaline gelerek :
sudo docker build -t ec2-flask:v1.0 -f dockerfile .
Komutu yazılmalıdır. Bu komut imageleri oluşturur ve gereken dosyaları indirir, sudo docker images yazılırsa imageler görüntülenebilir.



Şekil 38 - AWS Imageler

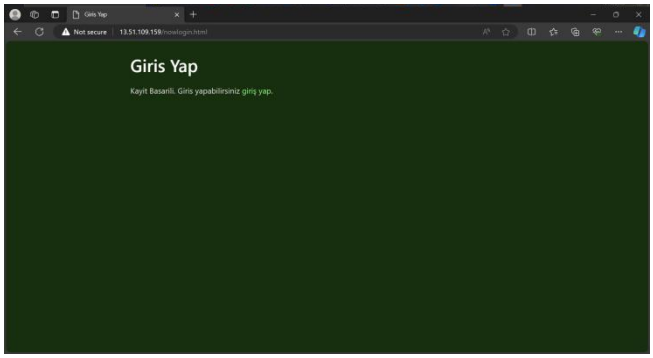
Daha sonra sırasıyla aşağıdaki komutlar yazılmalıdır.

sudo docker run -d -p 8001:80 --name phpmyadmin -e PMA_HOST=db -e PMA_PORT=3306 phpmyadmin/phpmyadmin

sudo docker run -d -p 3306:3306 -e MYSQL_DATABASE=kullanicilar -e MYSQL_USER=sueda_docker -e MYSQL_PASSWORD=my-secret-pw -e MYSQL_ALLOW_EMPTY_PASSWORD=1 mysql:latest

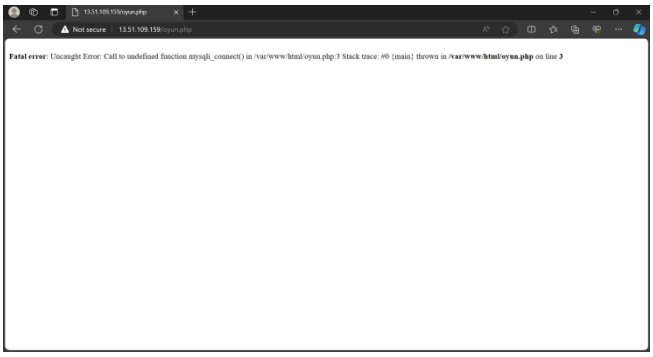
sudo docker run -d -p 80:80 -p 443:443 --name my-php -v "$(pwd):/var/www/html" php:apache

Bunun ardından proje bulutta sanallaştırılmış olur. Verilen public id'nin yanına sayfanın adı yazıldığında çalışır.



Şekil 39 - Proje Bulutta Çalışır Durumda

Fakat mysqli eklenti hatasıyla burada da karşılaşıldı. Mysqli kullanılan sayfalar çalıştırılamadı, bir çözüm henüz bulunamadı. Sadece html kodu bulunan bir sayfa test edildiğinde şekilde de görüldüğü gibi uygulamaya buluttan erişilebilmektedir. Mysqli olan sayfalarda şekil 40'da verilen hata mevcuttur.



Şekil 40 - Mysqli eklenti hatası

KARŞILAŞILAN PROBLEMLER

Projenin geliştirme aşamasında proje ilk başta xampp kullanılarak veritabanına bağlanmıştı, bu aşamada oyunun front ve backend kısımları tamamlanmıştı her şey çalışmaktaydı fakat proje Docker'a yüklenip konteynerlaştırıldığında oyun panelinde sıkıntılar çıktı, kodda htmlspecialschars komutları kullanılmıştı bu komutlar oyunu bozmaktaydı, oyun panelini çalışmaz bir hale getirmişti dolayısıyla bu hatayı gidermek için o tarz özel komutlar yerine direkt olarak normal php kullanıldı, oyun kodlarının bir kısmı baştan yazıldı.

Proje Docker'a yüklendikten sonra raporda bahsedilen mysqli hatasıyla da karşılaşılmıştı bu hata raporda bahsedildiği gibi Docker terminale bir php mysqli extention'ı eklenerek çözüldü fakat AWS'ye yüklendikten sonra yine aynı hatayla karşılaşıldı ve bu hata giderilemedi, proje Docker'da düzgün bir şekilde çalışmakta falan AWS'de bu hata sebebiyle mysqli kullanılan php sayfaları error mesajı yüzünden görüntülenememektedir.

SONUÇ

Proje, klasik Snake oyununu web tabanlı bir platforma taşıyarak nostaljik bir deneyim sunma amacını başarıyla gerçekleştirmiştir. Web geliştirme teknolojileri, veritabanı yönetimi ve sanallaştırma teknolojileri kullanılarak oluşturulan bu proje, kullanıcılar arasında rekabeti teşvik etmiş, eğitici bir boyut eklemiştir.

Uygulama kolay kullanılabilir kullanıcı dostu bir arayüz ile tasarlanmıştır.

KAYNAKÇA

[1] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html
[2] https://hub.docker.com/_/php
[3] https://www.youtube.com/watch?v=qNIniDftAcU
[4] https://www.youtube.com/watch?v=2Bxh5FNGznQ&t
[5] https://www.youtube.com/watch?v=bhqW5V5CarM
https://www.youtube.com/watch?v=VlAACb6aOvw