מסמך מוצר

שם הפרוייקט: Draw & Drive

שמות הסטודנטים: שני קורלנד, שמואל סילמן ויותם גואטה.

תיאור כללי:

משחק 2D multiplayer מאתגר ומשעשע לכמה חברים מול אותו המסך.

המשחק מורכב מ-2 קבוצות בנות 3 שחקנים שמשחקות אחת נגד השנייה. כל קבוצה מקבלת תמונה שהיא צריכה לצייר, באמצעות 3 מכוניות בצבעי היסוד.

ניקוד הקבוצות נקבע בכל סיבוב ע"פ הדיוק של הציור שציירו ביחס לציור הנדרש. הקבוצה המנצחת היא הקבוצה שתקבל את מירב הנקודות.

תמונת המשחק שהשחקנים צריכים לצבוע תבחר אקראית מגלריית התמונות של השחקנים.

תיאור מפורט:

כל קבוצה מורכבת מ-3 שחקנים, שכל אחד שולט במכונית אחרת מצבעי היסוד. מטרת כל קבוצה היא לצייר את התמונה המוצגת על קנבס דו-ממדי ריק באמצעות המכוניות.

השחקנים יוכלו לשלוט במכוניות באופן הבא:

- **נסיעה קדימה/אחורה** – מחשב: למעלה למטה בחיצים/מקשים W,S
  טלפון: לחצן קדימה אחורה
  אפקט: תצייר את הקנבס שמתחת למכונית בצבע שלה, או תערבב את הצבע שלה עם הצבע שכבר על הקנבס.
- **פניית המכונית** – מחשב: ימינה שמאלה בחיצים/מקשים A,D
  טלפון: הטעית הטלפון לצדדים.
  אפקט: תשנה את הכיוון של המכונית בהתאם לכיוון הרצוי.

שליטה מהפלאפון:

שחקני הקבוצות יוכלו לשלוט במכוניות דרך הסמארטפון (ע"י הטיית הטלפון ולחיצה על 2 פדלים), תוך כדי, צפייה במסך המשחק דרך המחשב עם החברים לצוות.

כוחות בונוס (power-ups):

לאורך המשחק יופיעו על הקנבס power-ups בצורה אקראית שהמכוניות יוכלו לאסוף. כל power-up יפעיל אפקט אחר במשחק, ויהיה תקף לזמן קצוב.

ה-power-ups:

- **Size Up**: המכונית שתאסוף את ה-power-up תשנה את צורתה לצורה עבה יותר ובכך הצביעה על הקנבס תהיה עבה יותר לזמן מוגבל
- **Size Down**: המכונית שתאסוף את ה-power-up תשנה את צורתה לצורה דקה יותר ובכך הצביעה על הקנבס תהיה דקה יותר לזמן מוגבל
- **Speed Up:** המכונית שתאסוף את ה-power-up תגביר את מהירותה לזמן מוגבל
- **Speed Down:** המכונית שתאסוף את ה-power-up תאט לזמן מוגבל
- **Change Color**: ה-power-up יחליף את צבעי המכוניות של השחקן לצבע אחר לזמן מוגבל ואז יחזיר אותו לצבעו המקורי.

כסף ושדרוג המכוניות:

לכל שחקן במשחק יש משתמש שנשמר בענן, במשתמש יישמרו נתוני השחקן כגון:

- שם משתמש וסיסמא
- שם השחקן
- תמונת פרופיל
- הישגים, כסף
- המכוניות/השדרוגים שרכש
- רשימת חברים
- תמונות שצילם למשחק

כל שחקן יוכל לרכוש שדרוגים ומכוניות עם יכולות שונות:

- עובי/מהירויות בסיס שונות
- יכולות פנייה שונות (steer)
- skin-ים שונים
- רכבים שונים

גלריית השחקנים

הציורים שהקבוצות יציירו יהיו ציורים שצולמו ע"י השחקנים מהקבוצות היריבות. כל שחקן יוכל להעלות תמונות למשחק, שיהפכו לציורים מפושטים ומאוזנים מבחינת צבעי היסוד.

<u>מהלך המשחק:</u>

המשחק מתחיל ע"י בניית הקבוצה שאיתה השחקן ישחק. אחד שחקני הקבוצה יצור lobby דרך המחשב, וכולם יוכלו להתחבר לסיסמת ה-lobby (מהאפליקציה או מהמחשב שלהם). לאחר מכן, הקבוצה תוכל לחפש קבוצה אחרת לשחק מולה..

בכניסה למשחק, ייבחר ציור אקראי מהגלריות של כל השחקנים, אותו שתי הקבוצות יתחרו לצייר.

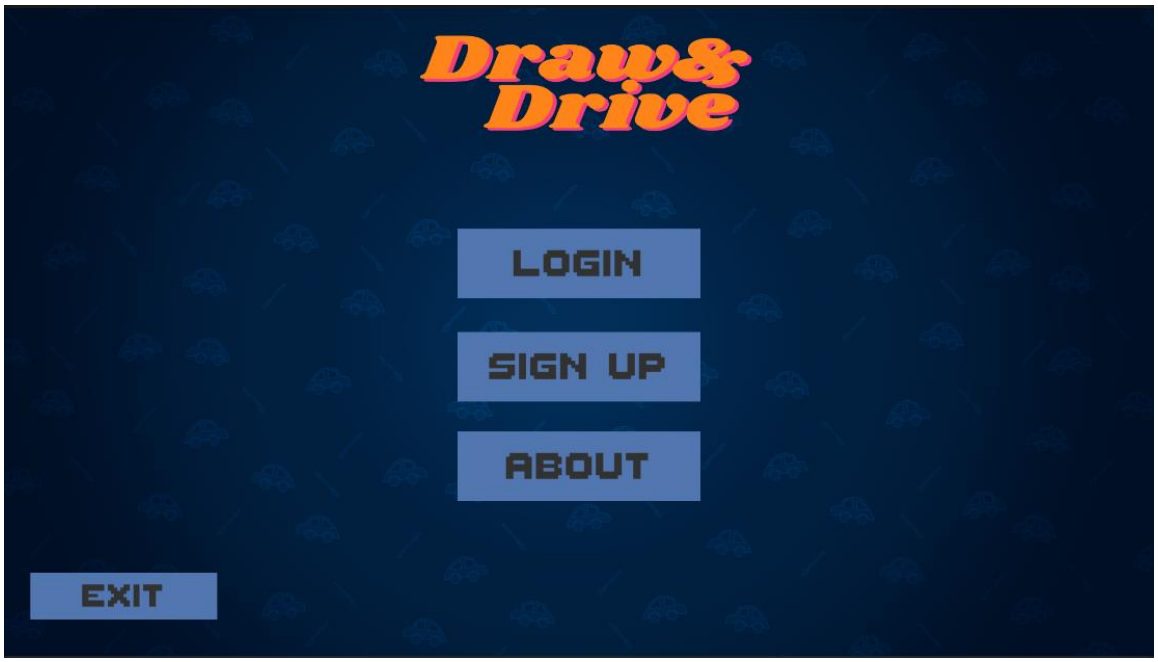
לכל קבוצה יהיה זמן קצוב לצייר את התמונה הנדרשת, ולבסוף תוכרע הקבוצה הזוכה

ע"פ הדמיון בין הקנבס לתמונה המקורית.

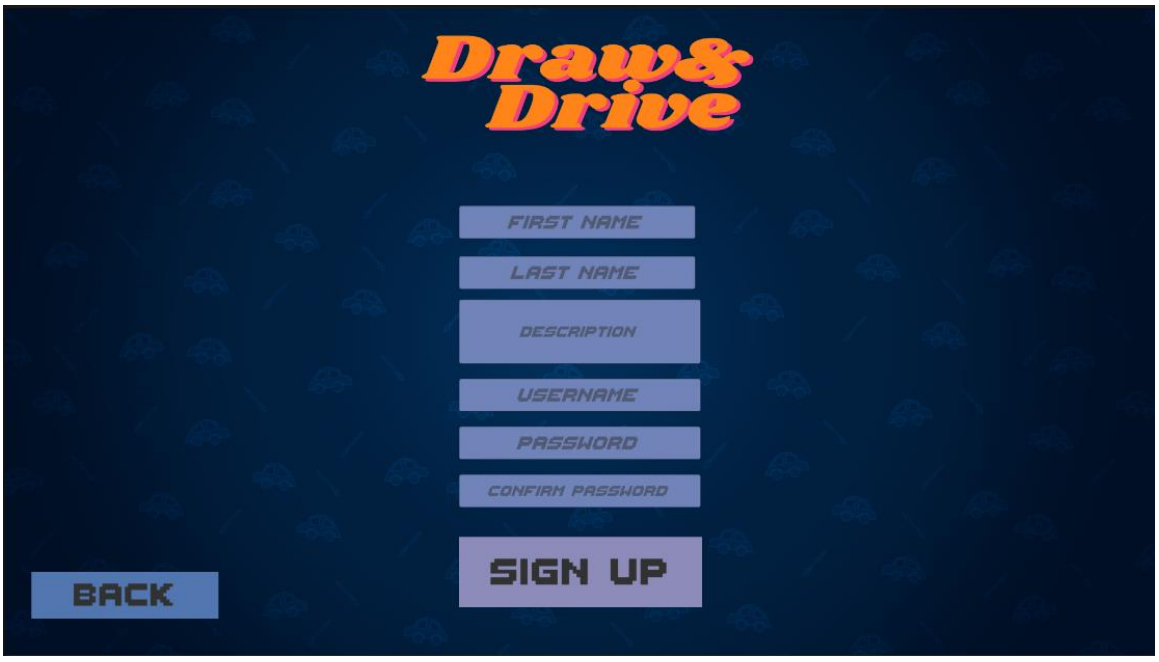
תיאור מסכים במחשב:

בכניסה למשחק מהמחשב, תחילה ייפתח מסך הכניסה (או המסך הראשי בהתחברות אוטומטית).

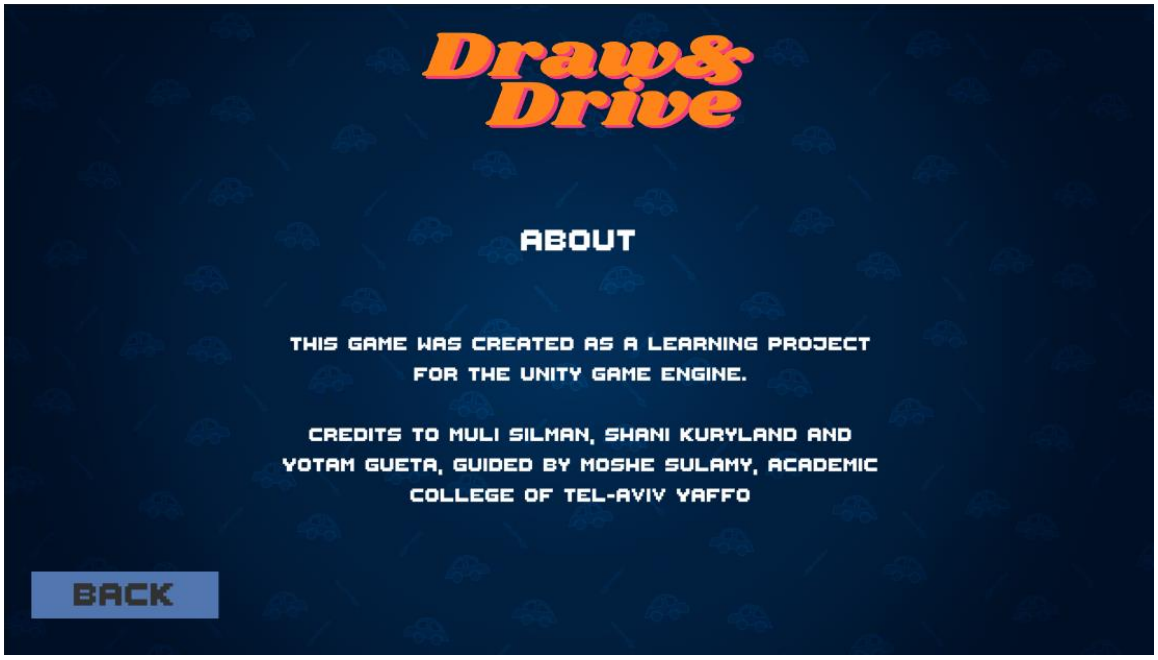
ממסך הכניסה ניתן יהיה להיכנס ל-3 מסכים.



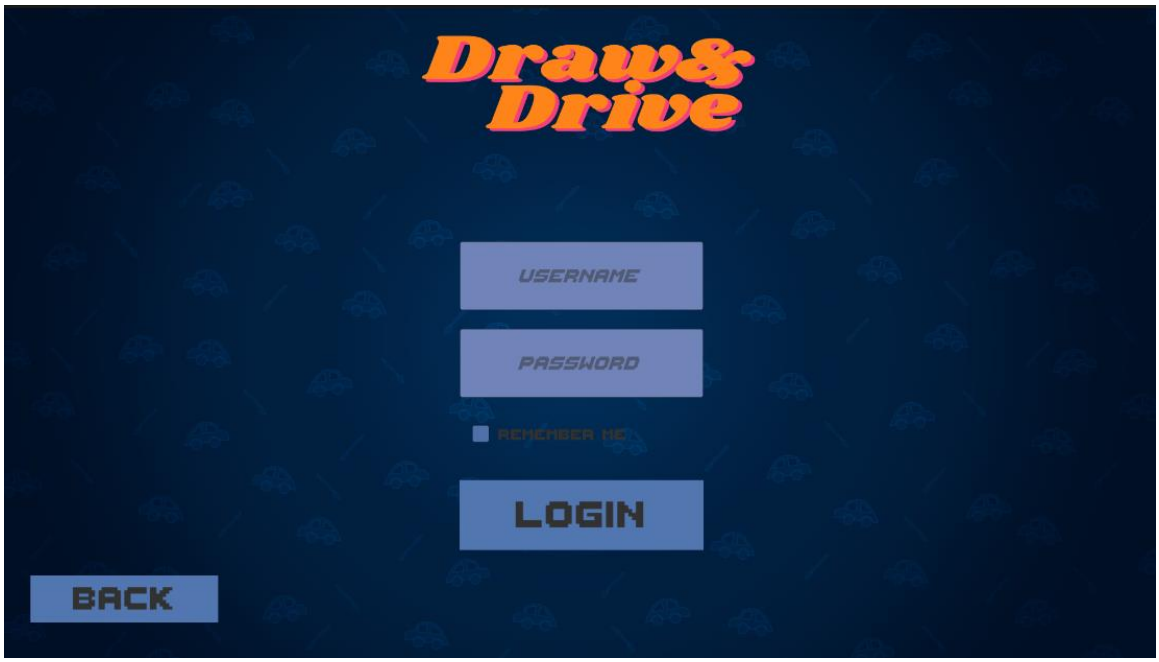
- מסך ה-Signup שיאפשר לרשום משתמש חדש למשחק.



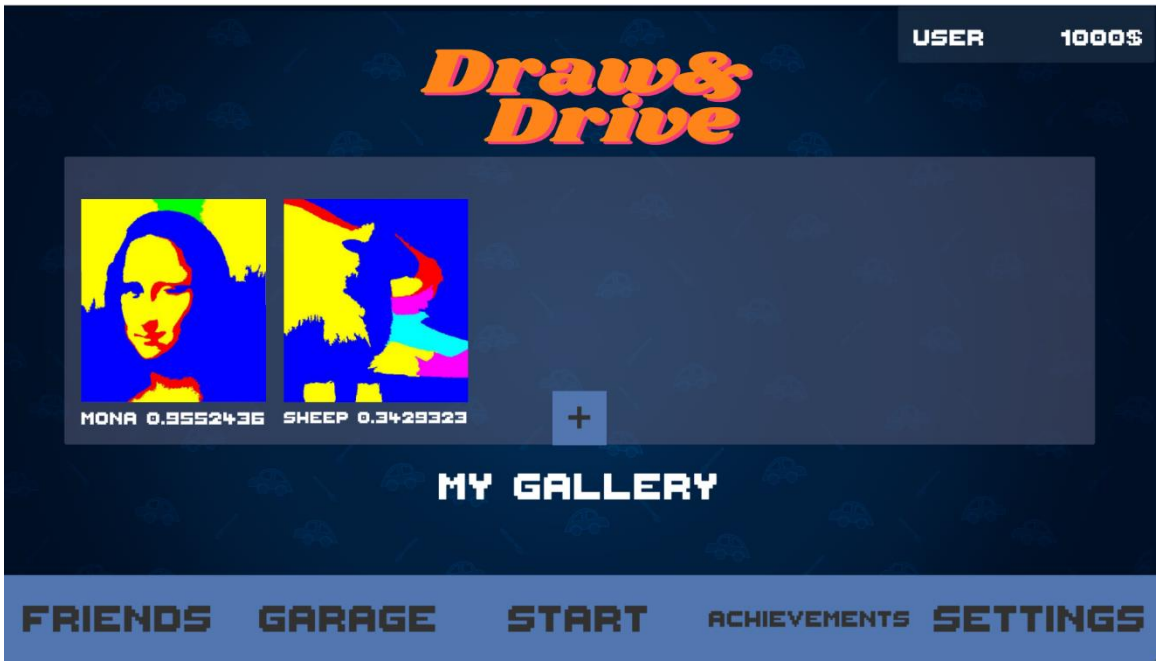
- מסך ה-About שיאפשר לקבל מידע נוסף על המשחק (שמות המפתחים, קרדיטים וכו').



- מסך ה-Login שיאפשר להתחבר למשחק עם משתמש קיים.



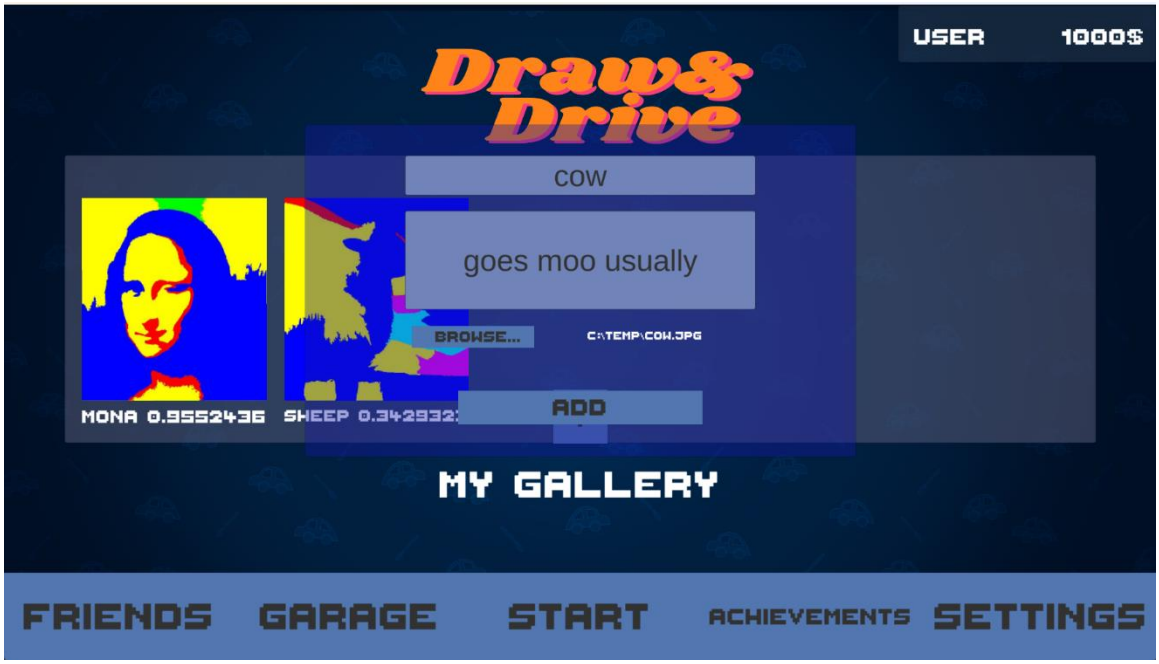
ברגע שמשתמש יתחבר לאפליקציה, המסך הראשי ייפתח.



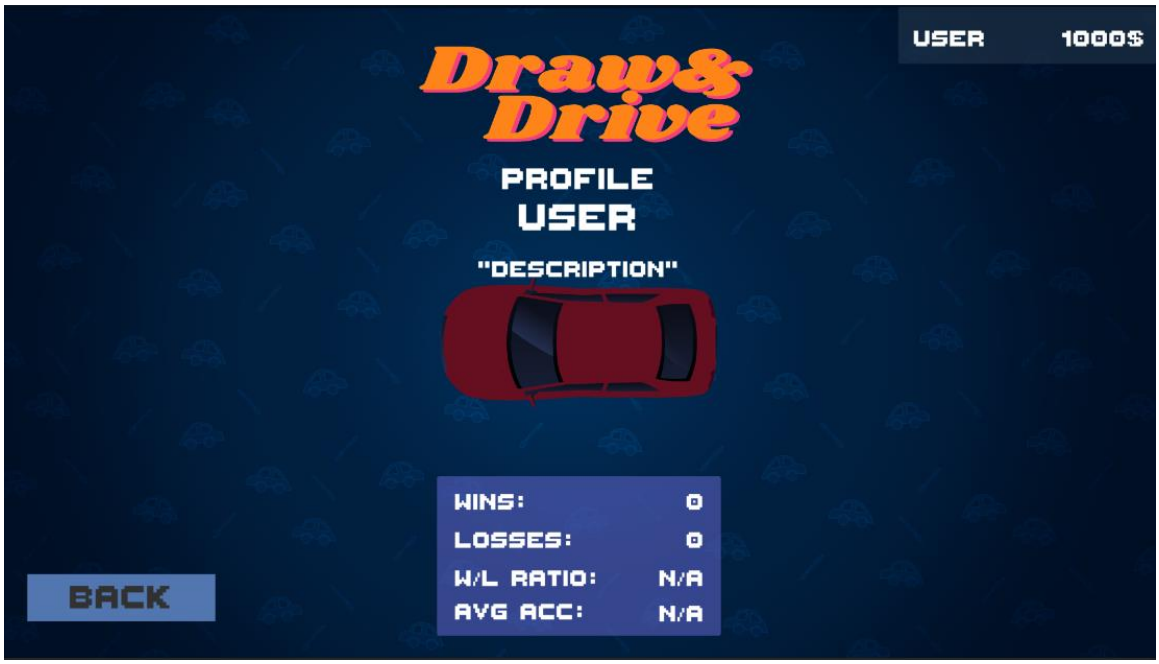
במסך הראשי, המשתמש יוכל:

- לגלול בגלריית הציורים שלו
- להיכנס להישגי המשתמש (pop-up – Achievements
  שיופיע במרכז המסך בו יופיעו ההישגים השונים של המשתמש –
  לכל הישג תמונת נושא, תיאור ותאריך ההישג).
- להיכנס להגדרות המשחק (pop-up שיופיע במרכז המסך ויאפשר עריכת הגדרות המשחק, כניסה לפרופיל המשתמש או יציאה מהמשתמש, וצפייה/עריכה של מקשי השליטה במכונית)
- צפייה ברשימת החברים (pop-up שיעלה מהכפתור של "friends", ובתוכו המשתמש יוכל להוסיף/למחוק חברים, או להיכנס לפרופיל שלהם)
- להיכנס ל"מוסך" של המשתמש (חנות)
- להתחיל משחק חדש
- להצטרף למשחק של חבר (ע"י הכנסת קוד משחק)
- להעלות ציורים חדשים לגלריה:

ניתן לחפש בקבצי המחשב תמונה להעלות עם שם ותאור שתעבור אלגוריתם שמפשט אותה ובכך ייווצר תמונה שגם קל יותר לצייר וגם עולה פחות מקום בענן.

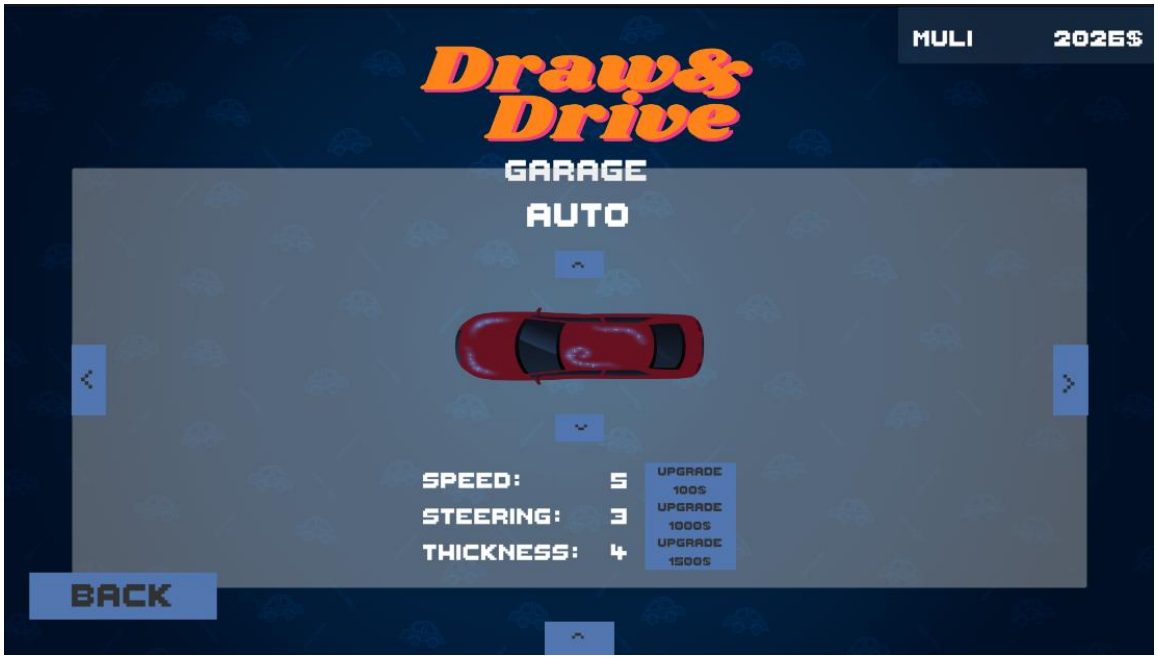


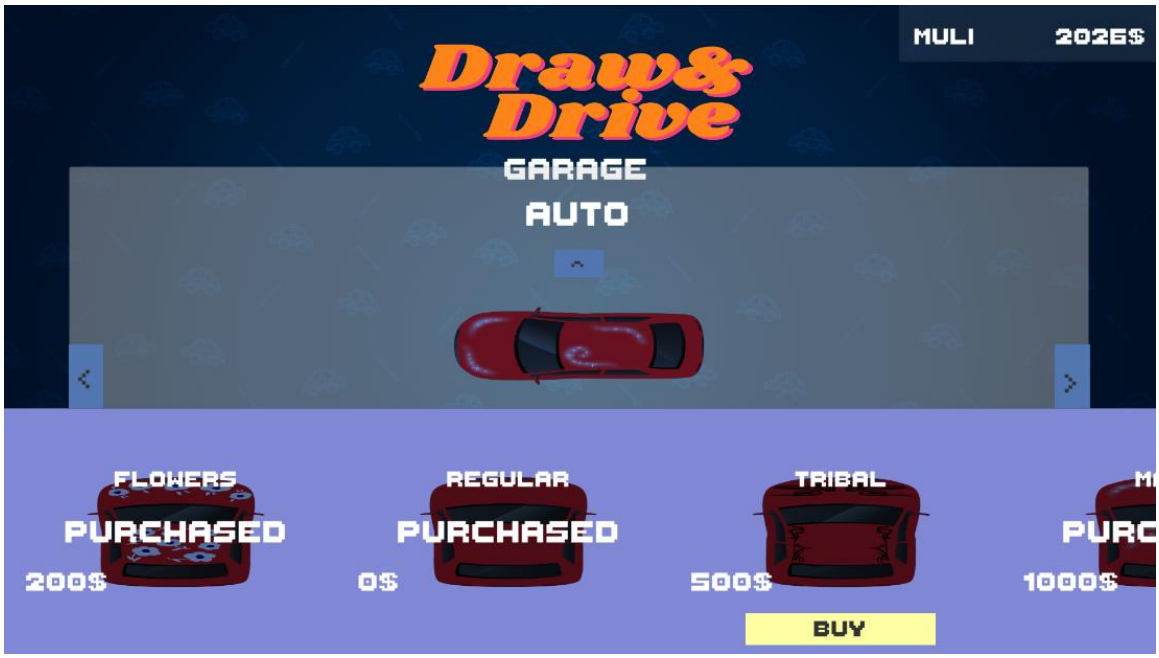
במסך פרופיל (משתמש/חבר), המשתמש יוכל לצפות ברכב הנוכחי שלו, בשם והרמה שלו, ובתיאור כללי לבחירתו. בנוסף במרכז המשחק המשתמש יוכל לגלול בחלון מינימלי של ההישגים, ובצד המסך יוכל לצפות בנתונים הסטטיסטים (אחוז הדיוק הממוצע, יחס הניצחונות וכו')



במסך המוסך, המשתמש יוכל:

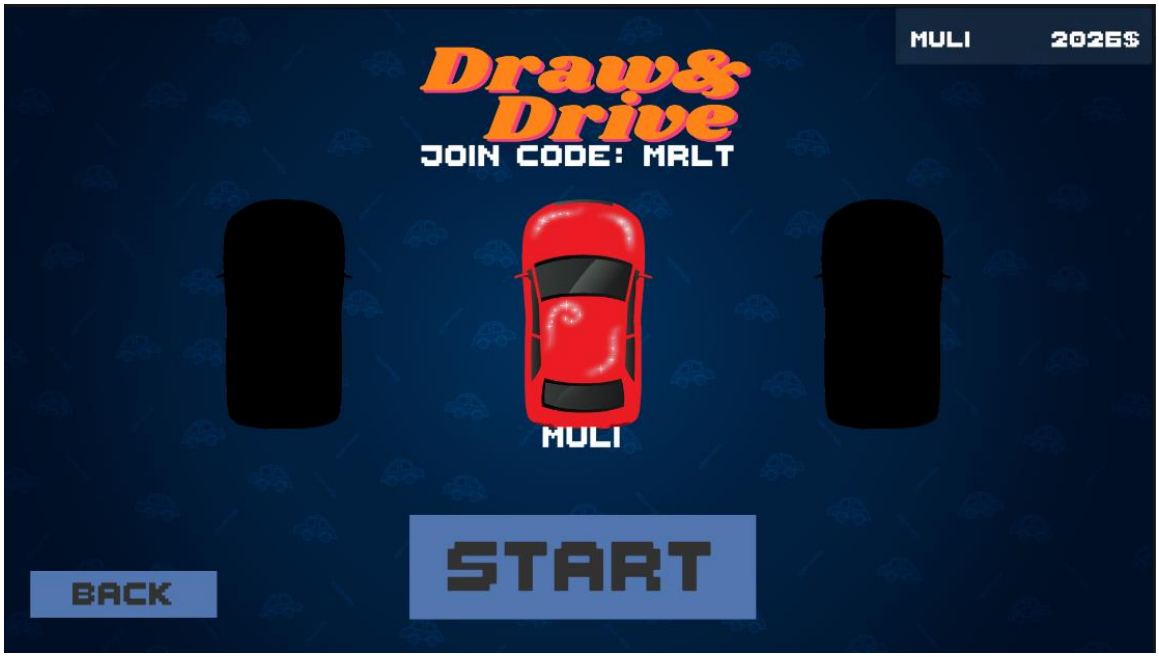
- לצפות במכונית הנוכחית
- להחליף את המכונית הנוכחית ע"י לחיצה על החצים ימינה/שמאלה
- לשדרג את המכונית הנוכחית
- לרכוש מכוניות נוספות
- לרכוש skin-ים

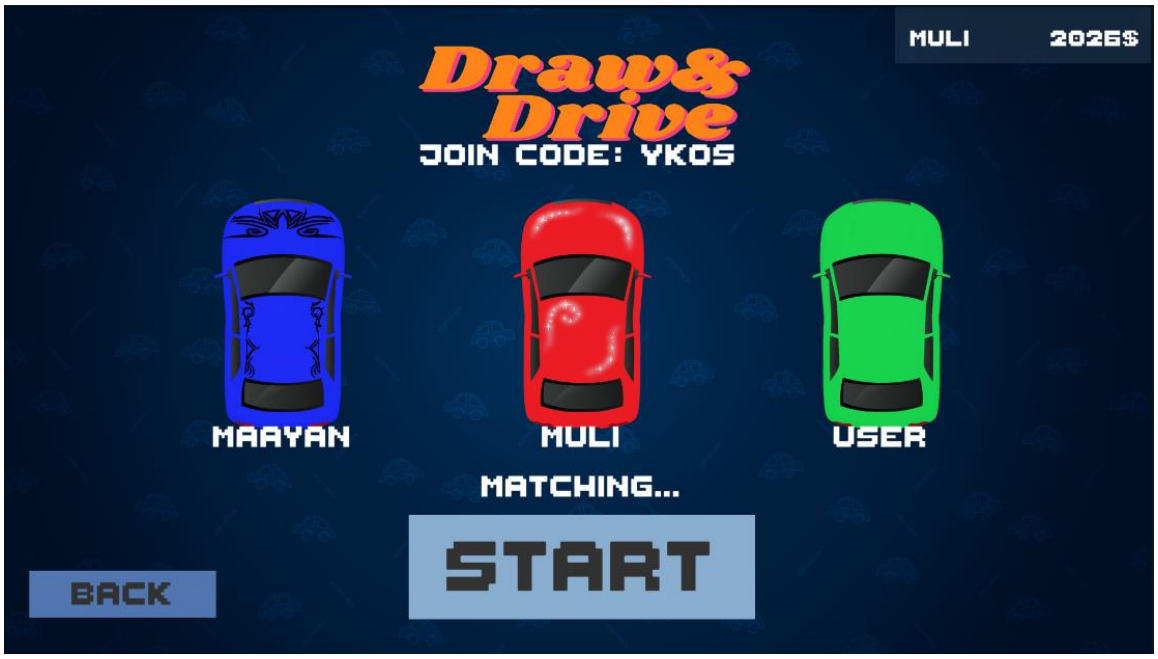




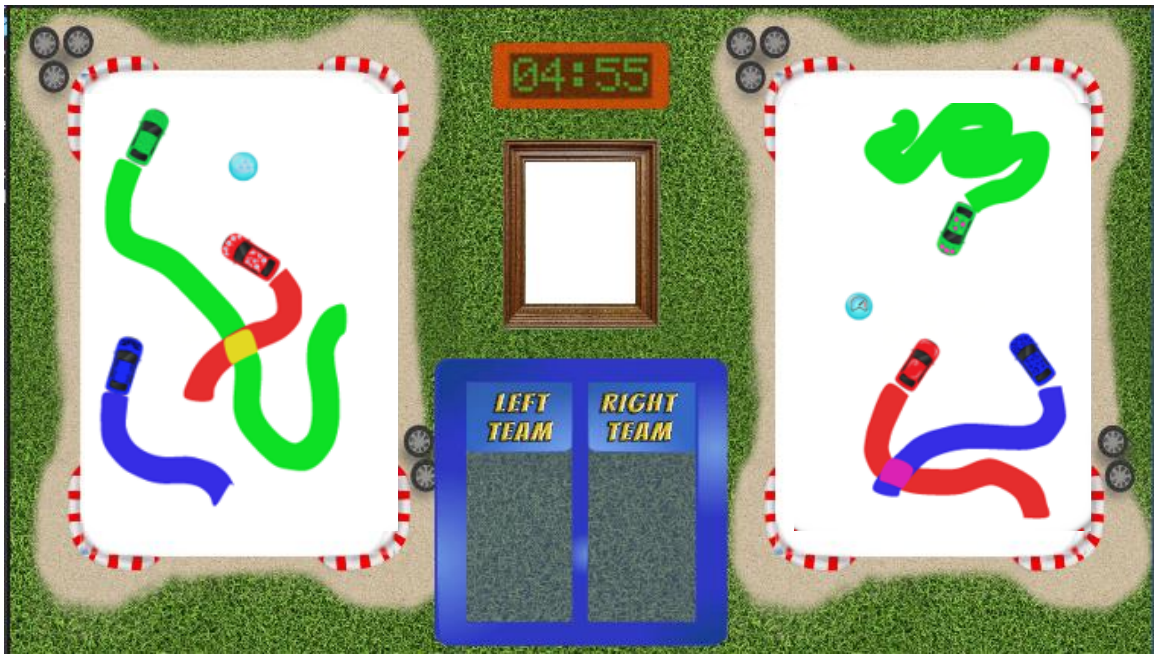
במסך המשחק החדש, המשתמש יראה את המכונית שלו במרכז המסך, ואת סיסמת המשחק לצורך הצטרפות. כשכל החברים הצטרפו המשתמש יוכל להתחיל את המשחק ולהיכנס לתהליך ה-matchmaking, שימצא קבוצה יריבה.

ניתן גם להתחבר לשחקן דרך הכנסת קוד הכניסה בSTART ובכך להצטרף למשחק עם אנשים שאתה מכיר. התחברות לקוד יכולה להתבצע ממחשבים נוספים או באמצעות התחברות דרך הטלפון.
כשתהליך ה-matchmaking יסתיים, המשתמש יעבור עם קבוצתו (והקבוצה היריבה) למסך המשחק.

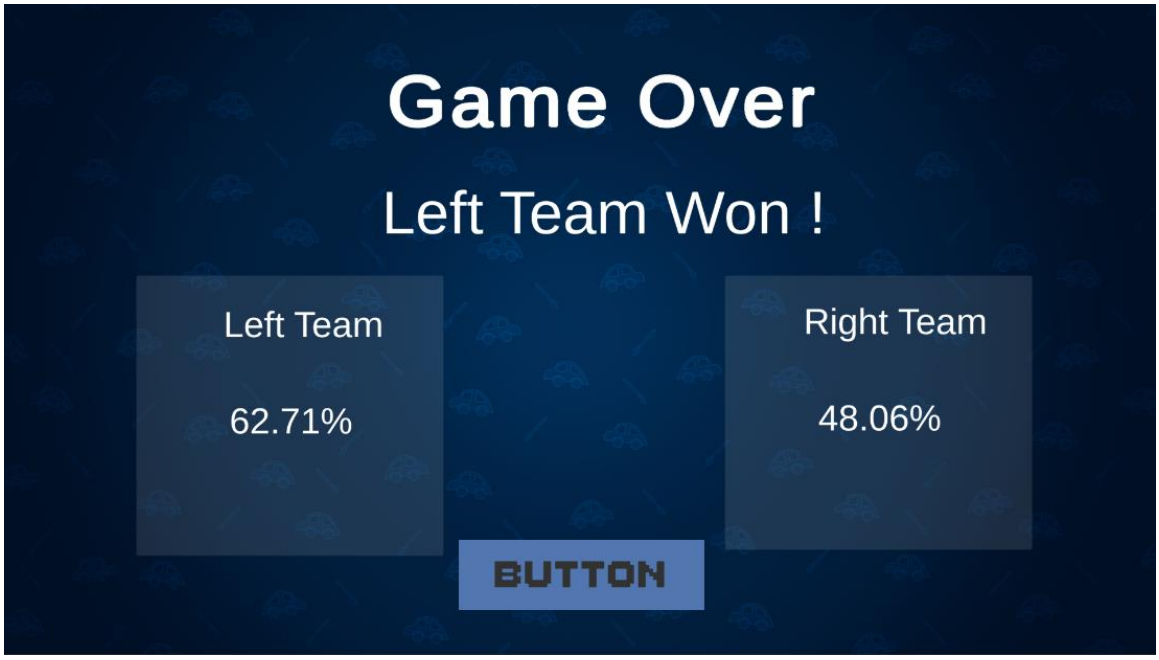




במסך המשחק, יופיעו שני קנבסים בשני צידי המסך, כאשר ביניהם תופיע התמונה הרצויה, טיימר המשחק ו-power-ups אקטיביים של כל קבוצה. מכוניות הקבוצות יתחילו בסידור משולש סימטרי על גבי הקנבס שלהם, תתחיל ספירה לאחור והמכוניות יוזנקו לצייר על הקנבס.
במהלך המשחק, ה-power-ups השונים יופיעו בצורה אקראית על גבי הקנבס, ובאיסוף ה-power-ups יופיע אפקט מתאים על כל המסך וה-power-up יוצג ויופעל לפרק זמן קצוב (יהיה אפקט על התמונה של ה-power-ups שיסמל את הזמן עד שייגמרו).



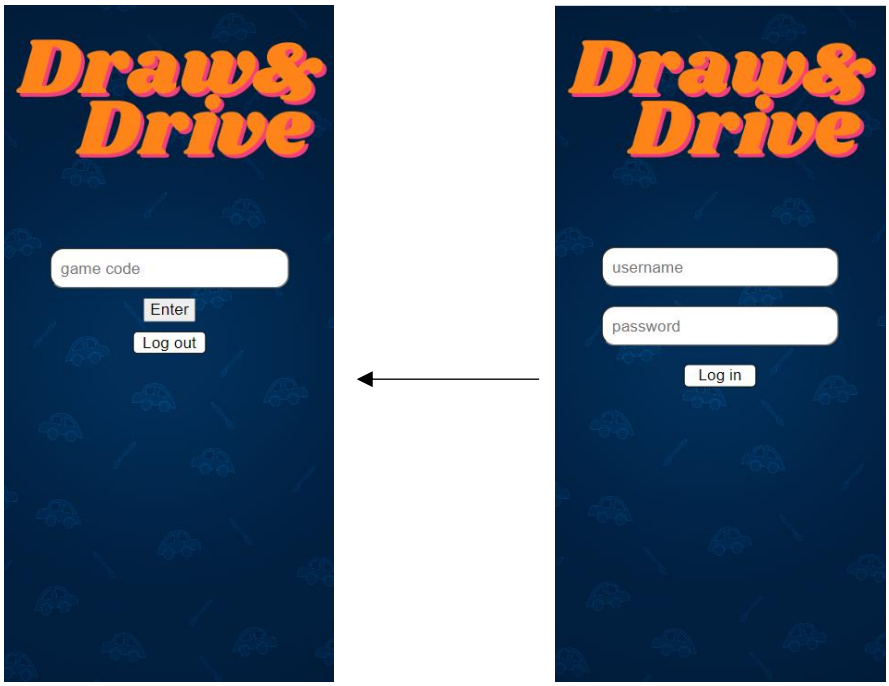
במסך סיום המשחק, יופיע אחוז הדיוק של כל קבוצה ביחס לתמונה המקורית.



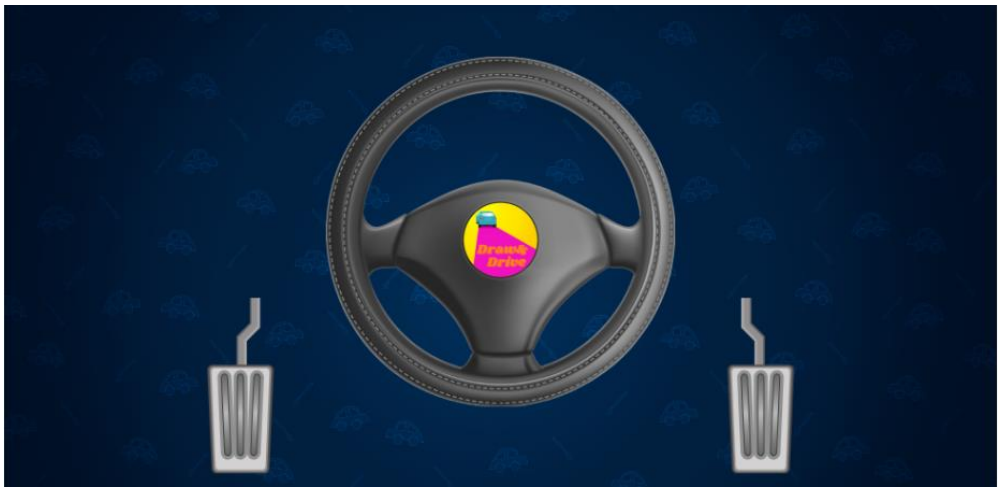
תיאור מסכים בסמארטפון:

בכניסה לאפליקציה, תחילה ייפתח מסך ההתחברות.

במסך ההתחברות, המשתמש יוכל להתחבר דרך האפליקציה, לאחר מכן יידרש להכניס את קוד המשחק שעליו הוא רוצה להצטרף, ולבסוף יעבור למסך השליטה מרחוק.



במסך השליטה מרחוק, בצד ימין של המסך יופיע פדל גז- בעת לחיצה עליו הרכב יתחיל לנסוע בעת עזיבה הרכב ייעצר, ובצד שמאל של המסך יופיע פדל נסיעה אחורה- בעת לחיצה עליו הרכב יתחיל לנסוע לאחור, בעת עזיבה הרכב ייעצר. אי לחיצה או לחיצה על 2 הפדלים במקביל תביא לעצירת המכונית.
בכדי שהמכונית תפנה, המשתמש יוכל להטות את הסמארטפון בהתאם לכיוון הרצוי.



ארכיטקטורה:

ארכיטקטורת המשחק בנויה מכמה רכיבים מרכזיים:

- מחשבי PC (Unity, C#)
- Google Cloud Platform (Python, fastapi)
- MongoDB
- מכשירי Mobile (javascript/html/css)

במחשב ה-PC רץ המשחק שפותח במנוע Unity + C# ומתקשר עם ה-Google Cloud Platform

ב-Google Cloud Platform רצה מכונה וירטואלית אחת תחת שרתי גוגל, שמנגישה את שרת ה-FastApi (שנכתב ב-Python) ב-HTTP ו-WS. בכדי שהשרת יונגש ב-HTTPS (SSL), רץ ב-GCP גורם נוסף, ה – Load Balancer. ה-Load Balancer מתנהג כמעין proxy בין המחשבים/הטלפונים השונים בעולם והמכונה הווירטואלית שלנו. ה-Load Balancer מוגדר לעבוד עם SSL Certificate לצורך תקשורת HTTPS (בפורט 443), ובקבלת כל בקשת HTTPS הוא מתרגם אותה לבקשות HTTP פשוטות מול המכונה הווירטואלית.

בנוסף הוגדר domain במיוחד ל-ip הסטטי של ה-load balancer בשם

unity-https-drawndrive.com עליו גם מוגדר ה-SSL Certificate

שרת ה-MongoDB רץ תחת cloud.mongodb.com והוא מנגיש 2 collection-ים שונים:

- Players – אוסף כל השחקנים שרשומים למשחק, כולל נתוניהם (התמונות שלהם, המכוניות/שדרוגים שרכשו וכו')
- Cars – אוסף כל המכוניות שניתן לרכוש במשחק (כולל נתוני שדרוגים ו-skin-ים, מחירים וכו')

בנוסף ב-MongoDB נעשה שימוש ב-GridFS לצורך שמירת קבצים גדולים – קבצי התמונות של השחקנים.

במכשירי ה-Mobile רצה האפליקציה האינטרנטית המונגשת באמצעות דומיין המתקשר מול ה-Google Cloud Platform בדומה למחשבי ה-PC.

מבנה הקוד:

קוד שרת

כפי שהזכרנו קודם, השרת נכתב בפייתון בתשתית fastapi. אפשר לחלק את קוד השרת ל-2 חלקים עיקריים: http ו-websocket.

http:

החלק העיקרי בשרת הוא החלק שמטפל בכל בקשות ה-http למיניהן.
מטרת בקשות ה-http היא להנגיש ולאפשר עריכה של נתוני המשחק לאפליקציות, כמו קבלת נתוני משתמש, העלאת תמונות, רכישת מכוניות וכו'.

כמעט כל בקשות ה-http מצפות לשם משתמש וסיסמא תקינים לצורך שליפת הנתונים הרלוונטיים ולצורך פרטיות (משתמשים לא יוכלו לדעת כמה כסף יש לאחרים לדוגמה).

להלן מצורפת דוגמה לבקשת GET לצורך צפיה במכוניות השחקן:

```
SamuelSill
@app.get("/players/cars",
         status_code=status.HTTP_404_NOT_FOUND)
def get_player_cars(username: str,
                    password: str,
                    response: Response):
    if (player_found := player(username, password)) is None:
        return "Player Not Found!"

    response.status_code = status.HTTP_200_OK
    return player_found["cars"]
```

תוכלו לראות בתחילת הפעולה שנעשה שימוש בפעולת player – פעולה פשוטה שמאפשרת שליפת נתוני משתמש מסוים מה-MongoDB:

```
SamuelSill
def player(username: str,
           password: str) -> Optional[dict[str, ...]]:
    return players_collection.find_one({
        "username": username,
        "password": password
    })


SamuelSill
def car(car_id: str) -> Optional[dict[str, ...]]:
    return cars_collection.find_one({
        "id": car_id
    })
```

להלן דוגמה נוספת לעריכת נתוני המשתמש – בקשת PUT לצורך בחירה במכונית מבין המכוניות של המשתמש.

```
SamuelSill
@app.put("/players/cars",
         status_code=status.HTTP_404_NOT_FOUND)
def select_car(username: str,
               password: str,
               car_index: int,
               response: Response):
    if (player_found := player(username, password)) is None:
        return "Player Not Found!"

    if car_index < 0 or car_index ≥ len(player_found["cars"]):
        response.status_code = status.HTTP_409_CONFLICT
        return "Invalid Car Index!"

    players_collection.update_one({"username": username}, {
        "$set": {
            "selected_car": car_index
        }
    })

    response.status_code = status.HTTP_200_OK
    return "Car Selected"
```

תוכלו לראות שבתחילת הפעולה נבצע בדיקות נכונות – שהשחקן אכן קיים, ושאינדקס המכונית שנבחרה הגיוני. רק לאחר בדיקות אלו נעדכן את ה-MongoDB ונחזיר תשובת הצלחה.

websocket:

בהמשך הקוד של השרת, תוכלו למצוא 2 פעולות שונות לטיפול בהתחברויות של websockets, שמטרתן לנהל את תהליך ה-matchmaking: endpoint אחד ליצירת lobby ו-endpoint נוסף להתחברות ל-lobby.

ביצירת lobby, האפליקציה תתחבר לפעולה המתאימה בשרת, ובהנחה שנתוני המשתמש שהועברו תקינים, יווצר קוד משחק אקראי חדש ל-lobby, והוא יוחזר לאפליקציה.
לאחר מכן בהצטרפות כל משתמש חדש ל-lobby (או עזיבה), השרת יודיע לאפליקציה על כך.

בהצטרפות ל-lobby, השרת יוודא שה-lobby אכן קיים ויש בו מקום, ושנתוני המשתמש תקינים, ולאחר מכן השרת ישלח לאפליקציה את נתוני כל המשתמשים שנמצאים ב-lobby (וכן יעדכן על עזיבה/הצטרפות של משתמשים).

כשה-host ילחץ על כפתור ה-start, השרת יקבל התראה על כך ויבדוק אם יש קבוצה בהמתנה כרגע, ואם כן הוא יצמיד ביניהם.

ה-host של הקבוצה הראשונה שהמתינה יהפוך ל-server ב-unity multiplayer כפי שמתואר בהמשך.

קוד תפריטים + צד לקוח

תפריטי המשחק עוצבו ותוכננו באמצעות תשתית ה-UI של Unity.

נעשה שימוש במגוון סוגים שונים של אובייקטי unity כמו:

- כפתורים עם אפקט ל-Hover, Click
- תיבות טקסט
- Scroll Rect לצורך רשימת אלמנטים (כמו רשימת חברים, גלריה, מכוניות וכו')
- וכו'

האפליקציה ב-unity תוכננה כך שבהתחברות, כלל הנתונים של השחקן נשלפים מהשרת בבת אחת, ובכך חווית הגלישה בתפריטים יותר מהירה, שכן כל הנתונים כבר נמצאים על המחשב.

```
IEnumerator LoadUserData(Action loginSuccessfulCallback)
{
    yield return GetUserMoney();
    yield return GetUserPaintings();
    yield return GetFriends();
    yield return GetAchievements();
    yield return GetUserOwnedCars();
    yield return GetUserSelectedCar();
    yield return GetAllCars();

    PerformAction(loginSuccessfulCallback);
}
```

כל תקשורת ה-http (לרבות ההתחברות ושליפת כל הנתונים) באה לידי ביטוי במודול ServerSession.

מודול זה תוכנן ב-Design Pattern של Singleton, כך שכל המערכת יכולה לגשת לפעולות הסטטיות שלו ולשלוף את אותם נתונים.

אחריות המודול היא לייחצן את כל הנתונים שהשרת מנהל בצד הלקוח לשאר המודולים של המשחק. הדבר בא לידי ביטוי בצורת properties:

```
// Properties
3 references
public static string Username => _loggedUsername;
3 references
public static int Money => _userMoney;
1 reference
public static List<string> Friends => _userFriends;
1 reference
public static List<Painting> Paintings => _userPaintings;
1 reference
public static List<Achievement> Achievements => _userAchievements;
5 references
public static List<PlayerCar> OwnedCars => _ownedCars;
28 references
public static PlayerCar CurrentCar => _ownedCars[_selectedCarIndex];
2 references
public static int CurrentCarIndex => _selectedCarIndex;
2 references
public static List<Car> GameCars => _cars;
13 references
public static Car CurrentGameCar => GameCars.Find(car => car.id == CurrentCar.id);
3 references
public static string CurrentSkin => CurrentCar.skins[CurrentCar.selected_skin];
2 references
public static int SpeedUpgradeCost => CurrentCar.upgrades.speed == CurrentGameCar.u
```

בנוסף לכך שניתן לשלוף נתונים מהמודול לקריאה, ניתן להפעיל פעולות סטטיות שונות לעריכת נתוני המשתמש בשרת (כמו העלאת תמונות, רכישת מכוניות וכד')

קוד משחק

קוד שרת-לקוח:

שרת המשחק משתמש בunity- netcode.

בעזרת התסריט -NetworkManager, לכל לקוח שמתחבר לקבל player שעליו שולט, שמוגדר כprefab של מכונית.

רוב הפעולות של עדכון מצב המשחק ויצירה או מחיקה של גופים, נעשים ע"י השרת בעזרת בקשת ServerRPC ולאחר מיכן עדכון הלקוחות ע"י בקשת ClientRPC.

לדוגמא: צביעת הקנבס לפי מיקום השחקן נעשתה ע"י הפעלת

הפונקציה BrushAreaWithColorOnServer() שקורא לשרת לבצע צביעה בלקוחות ואצלו

```
1 reference
private void CmdBrushAreaWithColorOnServer(Vector2 pixelUV, Color color, int size)
{
    BrushAreaWithColorOnServerRpc(pixelUV, color, size);
}

[ServerRpc(RequireOwnership = false)]
1 reference
private void BrushAreaWithColorOnServerRpc(Vector2 pixelUV, Color color, int size)
{
    BrushAreaWithColorOnClientRpc(pixelUV, color, size);
    BrushAreaWithColor(pixelUV, color, size);
}

[ClientRpc]
1 reference
private void BrushAreaWithColorOnClientRpc(Vector2 pixelUV, Color color, int size)
{
    BrushAreaWithColor(pixelUV, color, size);
}
```

קוד unity:

קוד הצביעה נעשתה בתסריט PlayerBrush שמחפש לכל החלקים שצובעים (הגלגלים) את מקום הנגיעה שלהם עם הקנבס ומשנה אותו למיקום שלו בעולם, ואז שולח לעדכון הצבע שלהם לשרת לפני שנאמר בקוד השרת-לקוח.

```
private Vector2 WorldToPixelUV(Vector3 worldPosition, int textureWidth, int textureHeight)
{
    Vector3 localPosition = pallet.transform.InverseTransformPoint(worldPosition);
    Vector2 pixelUV = new Vector2(localPosition.x + 0.5f, localPosition.y + 0.5f);
    pixelUV.x *= textureWidth;
    pixelUV.y *= textureHeight;
    return pixelUV;
}
```

```
private void FixedUpdate()
{
    if (PlayerOptions.PositionNetworkSpawned < NetworkManegerUI.NUMBER_OF_PLAYERS || !TimerStarter.GameStarted)
    {
        return;
    }
    foreach (Transform objectChild in objectChildren)
    {
        Vector3 playerPosition = objectChild.position;

        if (pallet != null)
        {
            Renderer rend = pallet.GetComponent<Renderer>();
            MeshCollider meshCollider = pallet.GetComponent<MeshCollider>();

            if (rend == null || rend.sharedMaterial == null || rend.sharedMaterial.mainTexture == null || meshCollider == null)
                return;

            Texture2D tex = rend.sharedMaterial.mainTexture as Texture2D;
            Vector2 pixelUV = WorldToPixelUV(playerPosition, tex.width, tex.height);

            CmdBrushAreaWithColorOnServer(pixelUV, PlayerCustomisation.getColor(), PlayerCustomisation.BrushSize);
            BrushAreaWithColor(pixelUV, PlayerCustomisation.getColor(), PlayerCustomisation.BrushSize);
        }
    }
}
```

הpower-ups נבנו ע"י קוד מרכזי של PowerUp שממנו כל ה- power-ups יורשים את תכונות כמו הפעלת אנימציה בעת לקיחה, העלמות בעת לקיחה של שחקו ועוד..
את התכונות השונות בין power-ups כמו סוג היכול שניתנת מלקיחתה, מומשו ע"י פעולות אבסטרקטיות שכל יורש צריך לממש.

```
protected abstract void PlayParticalEffect();
6 references
protected abstract void ApplyPowerUpEffectOnPlayer(Collider2D player);
6 references
protected abstract void RemovePowerUpEffectOnPlayer(Collider2D player);
6 references
protected abstract float GetDuration();
6 references
protected abstract string GetMessage();
```

```
private IEnumerator PowerUpEffect(Collider2D player)
{
    baseDuration = GetDuration();
    message = GetMessage();
    sprite = GetSprite();

    PlayParticalEffect();

    ApplyPowerUpEffectOnPlayer(player);

    SendNotification();

    // Hide this object
    GetComponent<SpriteRenderer>().enabled = false;
    GetComponent<Collider2D>().enabled = false;

    // Hide all his childern
    Transform[] objectChildren = gameObject.GetComponentsInChildren<Transform>();
    for (int i = 0; i < objectChildren.Length; i++)
    {
        objectChildren[i].GetComponent<SpriteRenderer>().enabled = false;
    }

    yield return new WaitForSeconds(baseDuration);

    RemovePowerUpEffectOnPlayer(player);

    RemoveNotification();

    DespawnTimerServerRpc(gameObject.GetComponent<NetworkObject>());
}
```

קוד טלפון

השליטה מהטלפון מונגשת ע"י שרת הווב שרץ על google cloud platform.
בכתובת - https://unity-https-drawndrive.com/

קוד צד לקוח:

הקוד לסמארטפון מומש בHTML, CSS, Java Script.
צד הלקוח מתבצע תחילה בקריאות HTTP פשוטות (על אף שהן https במימוש עם דומיין אשר היה הכרחי לשם הwebSocket וגם לשם גישה לנתוני הorientation). ולאחר עמוד הלובי מתקיים מימוש צד לקוח של הWebSocket.

```
const api_login_request = () =>{
    console.log(username.value);
    console.log(password.value);
    const name = username.value ;
    const pass = password.value;
    fetch(`${api_url}/players/login?username=${name}&password=${pass}`, {
        method: 'GET',

    })
    .then(res => {
        if(res.status !== 200){
            alert("The username or password is incorrect");
            throw new Error('Login failed');
        }
        else{
        console.log('Success:', res);
        localStorage.setItem('username', username.value)
        localStorage.setItem("password", password.value)
        // Redirect to lobby.html after successful login
        window.location.href = '/lobby.html';
    }
    })
    .catch((error) => {
```

```
const api_gamecode_request = (gamePin,username,password) =>{
  try{
    const ws_url = (codePin) => `${socker_api_url}/games/ws/${codePin}/${username}/${password}/true`;
    const webHook = ws_url(gamePin);
    console.log(gamePin,webHook);
    var ws = new WebSocket(webHook);

  //--------------------Send data to socket----------------------------------------
    function send_data_to_socket(){···
  //-------------------------------------------------------------------------------

    ws.addEventListener("error", (event) => { //maybe check if the gamecode exists
      console.log("WebSocket error: ", event);
      window.location.href = '/lobby.html';
    });

    ws.onopen = function() {
    // Web Socket is connected, send data using send()
      console.log("connected");
      setInterval(send_data_to_socket, intervalSeconds);
    };
```

קוד השליטה:

הקוד ממומש גם הוא בJava Script, וממש 2 פדלים- פדל גז (כשלוחצים נוסעים, עוזבים- הרכב נעצר) ופדל רברס (כל עוד לחוץ מתבצעת נסיעה לאחור) ובנוסף מבצע שימוש ב Event Orientation המזהה שינוי בהטיית הטלפון ובכך נותן את האפשרות לשלוט במכונית דרך הסמארטפון בתור הגה.

```
function handleOrientation(event) {
    const max_orientation = 15.0;
    if (event.beta < -15) {
        orientation = -1.0;
    }
    else if (event.beta > 15) {
        orientation = 1.0;
    }
    else {
        orientation = event.beta / 15;
    }

    orientation = orientation.toFixed(2);
}

window.addEventListener("deviceorientation", handleOrientation);
```

```
function pedal_press() {
    event.preventDefault();
    document.getElementById("pedal").style.height = "30%";
    pedal_pressed = true;
}
function pedal_release() {
    event.preventDefault();
    document.getElementById("pedal").style.height = "50%";
    pedal_pressed = false;
}
```

```
function send_data_to_socket(){
  let drive;
  if (reverse_pressed === pedal_pressed) {
      drive = 'stop';
  } else if (pedal_pressed) {
      drive = 'forward';
  } else {
      drive = 'reverse';
  }
  ws.send(JSON.stringify({id:"MobileControls", direction: orientation, drive: drive}));
}
```

```
ws.onopen = function() {
// Web Socket is connected, send data using send()
  console.log("connected");
  setInterval(send_data_to_socket, intervalSeconds);
};
```