משחק הקלדה אינטרנטי בזמן אמת

Space Typing Online

מכללה: מכללת תל חי

מגיש: כינאן שמס

מנחה: ד"ר הורוביץ מיכל

שנה: 2025

תוכן עניינים

פרק 1 – מבוא

1.1 תיאור הפרויקט

Space Typing Online הוא משחק הקלדה אינטרנטי בזמן אמת, שנועד לשפר את מהירות ודיוק ההקלדה של המשתמשים באמצעות חוויית משחק מהנה ודינמית. המשחק מתרחש בסביבה חללית צבעונית, שבה מילים "נופלות" מהחלק העליון של המסך, והשחקן נדרש להקליד אותן במהירות לפני שיפגעו בקרקע.

ייחודו של המשחק הוא בשילוב בין חוויית משחק אינטראקטיבית לבין אלגוריתמיקה חכמה. מאחורי הקלעים פועל מבנה נתונים מסוג **Trie**, המאפשר לזהות ולחפש מילים לפי תחילית באופן מיידי, ובכך מעניק תגובה מהירה וחלקה להקלדת המשתמש.

המשחק כולל שני מצבים עיקריים: **מצב אונליין**, שבו שני שחקנים מתחרים בזמן אמת דרך שרת `Flask–SocketIO`, ו־**מצב אופליין**, שבו המשתמש משחק לבדו ללא צורך בחיבור לאינטרנט. שני המצבים חולקים את אותו מנוע לוגי, אך כל אחד מהם מציע חוויה שונה — תחרותית לעומת אישית.

1.2 טכנולוגיות וכלים

רכיב	שימוש
Python	מימוש צד השרת והלוגיקה המרכזית של המשחק
Flask-SocketIO	תקשורת בזמן אמת בין שחקנים
JavaScript	ניהול לוגיקת המשחק בצד הלקוח
HTML / CSS	בניית ממשק המשתמש
JSON	ניהול נתוני משתמשים ומחסן מילים

פרק 2 – ממשק המשתמש (UI)

2.1 מבוא כללי לממשק

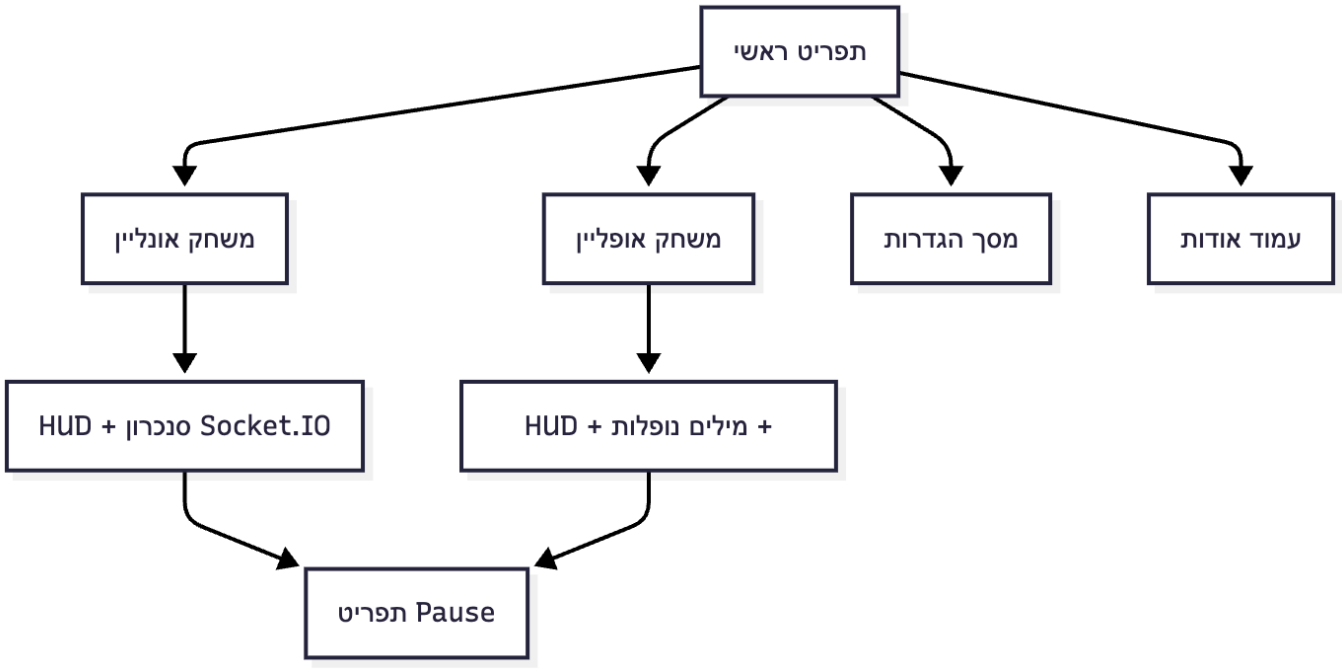
Space Typing Online מציג חוויית משחק ויזואלית דינמית השואבת השראה מעולם החלל והניאון. כבר במסך הפתיחה מופיע רקע מלא כוכבים וגרפיקה זוהרת, היוצרת תחושת עומק ותנועה. בתפריט הראשי השחקן יכול לבחור בין משחק אונליין, משחק אופליין, עמוד ההגדרות ועמוד ה־About, כאשר כל מעבר בין מסכים מלווה באנימציות עדינות ובאפקטי אור המדמים חללית עתידנית.

במהלך המשחק, על המסך מופיעים מילים "נופלות" מלמעלה כלפי הקרקע. כל מילה מלווה באפקטים חזותיים: לייזר נורה מהמקלדת כאשר ההקלדה נכונה, ופיצוץ קטן מתרחש כאשר מילה מושלמת. החלק העליון של המסך מציג **HUD** ברור הכולל את הניקוד, מספר החיים, רמת הקושי, מהירות ההקלדה (**WPM**) ואחוז הדיוק. במצב **Pause**, מוצג חלון חצי־שקוף המאפשר לבחור בין Resume, Quit או מעבר להגדרות – תוך שמירה על עיצוב אחיד ואלגנטי.

הממשק מחולק לשני מצבים עיקריים:

- **Offline Mode** – השחקן משחק לבדו ומנסה לשפר את ביצועיו ללא הגבלת זמן.
- **Online Mode** – שני שחקנים מתחרים בזמן אמת, כאשר הנתונים מסונכרנים דרך השרת ומוצגים לשני הצדדים במקביל.

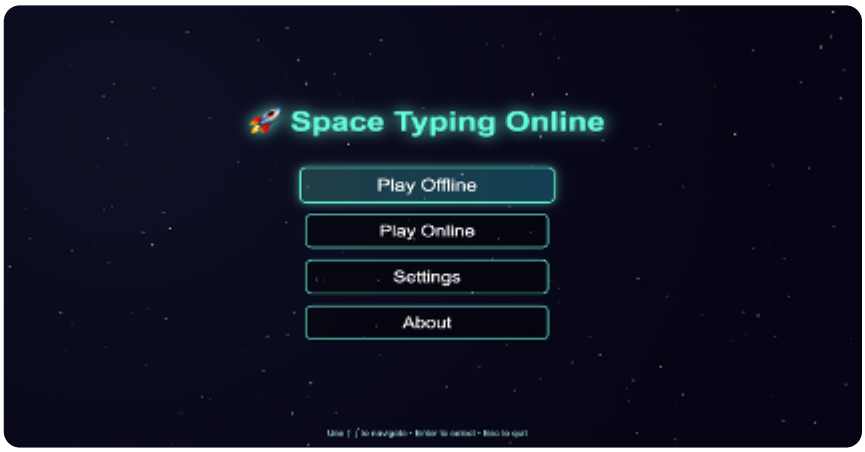
העיצוב של שני המצבים זהה, אך מצב האונליין מוסיף **HUD** נוסף המציג את מצב היריב בזמן אמת, כולל ניקוד, חיים ופס זמן. כל רכיבי ה־UI פועלים בשילוב עם אנימציות חלקות, צבעים בהירים על רקע כהה ואפקטי צליל המעצימים את תחושת הזרימה והקצב.



תמונה 2.1 – מבוא כללי למשחק Space Typing Online

2.2 התפריט הראשי (Main Menu)

במסך הראשי של המשחק מופיעות כל האפשרויות המרכזיות: **Settings**, **Play Online**, **Play Offline**, ו־**About**. התפריט הראשי נועד לאפשר ניווט נוח בין מצבי המשחק השונים ולשמור על חוויית משתמש אינטואיטיבית וברורה.



2.3 מצב אופליין (Play Offline)

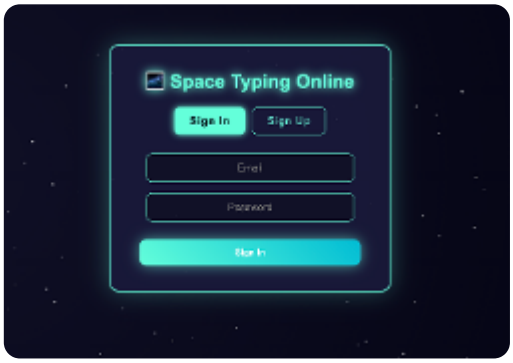
במצב זה השחקן משחק נגד הזמן ללא חיבור לשרת. המילים נשלפות ממחסן המילים המקומי (JSON), והזיהוי שלהן מתבצע בעזרת מבנה הנתונים Trie. המסך מציג את המילים "הנופלות", את שורת ההקלדה, את הניקוד ואת סרגל החיים.

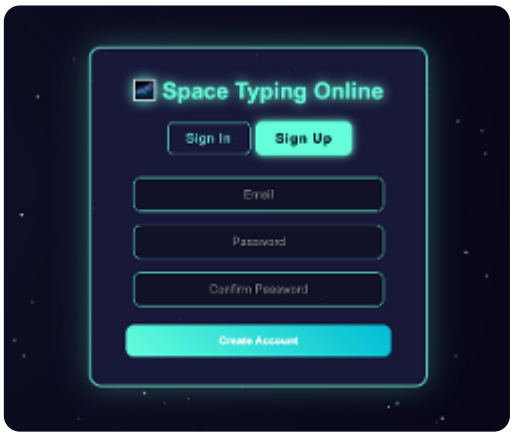
2.4 מצב אונליין (Play Online)

במצב אונליין, שני שחקנים מתחרים בזמן אמת על מהירות ודיוק ההקלדה. השרת מסנכרן בין השחקנים ומציג עדכונים על הניקוד, ההקלדות, והזמן הנותר לסיום המשחק.

2.4.1 התחברות והרשמה

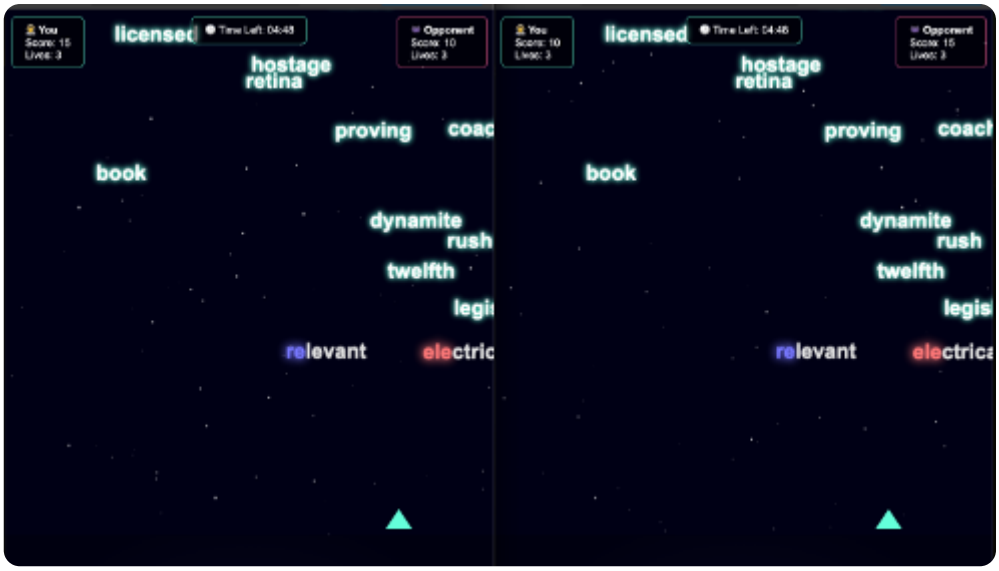
כדי לשחק אונליין, על המשתמש להתחבר באמצעות מסך **Sign In** או להירשם דרך **Sign Up**. הנתונים נשמרים בקובץ `users.json` בצורה מאובטחת, ומאפשרים ניהול משתמשים קבוע.





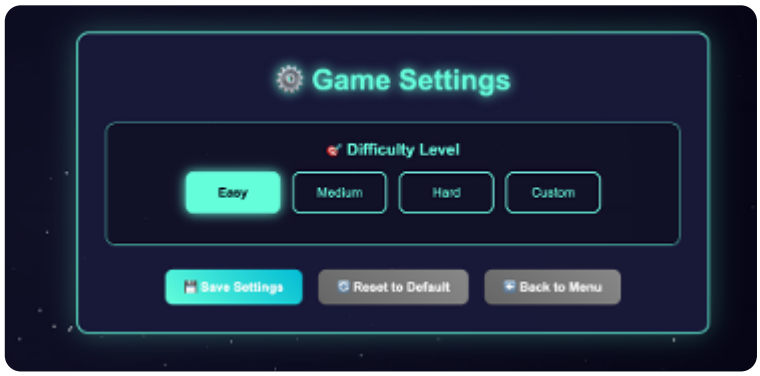
2.4.2 משחק אונליין בפעולה (Play Online)

לאחר ההתחברות, שני השחקנים נכנסים לאותו חדר משחק. המילים נשלחות מהשרת בו־זמנית לשניהם, וכל שחקן מתחרה על מי יסיים יותר מילים בזמן נתון. בסיום המשחק השרת מציג את התוצאות הסופיות של שני הצדדים.



2.5 מסך הגדרות (Settings)

מסך ההגדרות מאפשר לשחקן להתאים את רמת הקושי ואת חוויית המשחק האישית שלו. במרכז המסך מופיעה האפשרות Difficulty Level, הכוללת ארבע רמות קושי: Easy, Medium, Hard, ו־Custom. השחקן יכול לבחור את הרמה הרצויה בהתאם לניסיונו וליכולת ההקלדה שלו



2.6 מסך אודות (About)

מסך זה משמש כמדריך אינטראקטיבי לשיפור מיומנויות ההקלדה. הוא מציג הסבר חזותי על מיקום האצבעות במקלדת וטיפים מעשיים להקלדה מהירה ומדויקת, כגון שמירה על תנוחה נכונה ותרגול עקבי. בנוסף מופיעה בו גם רשימת כללים בסיסיים של המשחק ותיאור קצר על מטרותיו.

פרק 3 – ארכיטקטורת המערכת (משחק אונליין)

המשחק **Space Typing Online** בנוי בארכיטקטורה של **Client–Server**, המאפשרת פעולה בזמן אמת בין שני שחקנים דרך האינטרנט. צד הלקוח, שנכתב ב־HTML, CSS ו־JavaScript, אחראי להצגת הגרפיקה, קבלת הקלט מהמקלדת וניהול מצב המשחק המקומי. בצד השני, שרת ה־**Flask** בשפת **Python** מטפל בכל הלוגיקה העסקית: יצירת חדרי משחק, ניהול מילים נופלות, חישוב ניקוד, סנכרון בין השחקנים והעברת הודעות בעזרת **Socket.IO**.

כאשר שחקן מתחבר, הלקוח שולח בקשה לשרת להצטרף לחדר. השרת מוסיף את המשתמש למבנה נתונים פנימי, מאתחל את מצב המשחק ושולח את הנתונים הגרפיים לכל המשתתפים. כל תו שהשחקן מקליד נשלח מיידית דרך **Socket.IO** אל השרת, שמעבד את הקלט, בודק את ההתאמה למילה הרלוונטית, ומחזיר תשובה לכל הלקוחות המחוברים. כך נשמרת חוויית משחק חלקה ומסונכרנת, שבה כל פעולה מצד אחד מתעדכנת מייד בצד השני.

הארכיטקטורה מתוכננת כך שכל רכיב עצמאי אך גם מחובר היטב לאחרים:

- **Client** – אחראי להצגה ולשליחת פעולות.
- **Server** – מטפל בלוגיקה ובמצב המשחק המשותף.
- **Socket.IO** – הגשר ביניהם, המאפשר תקשורת דו־כיוונית בזמן אמת.

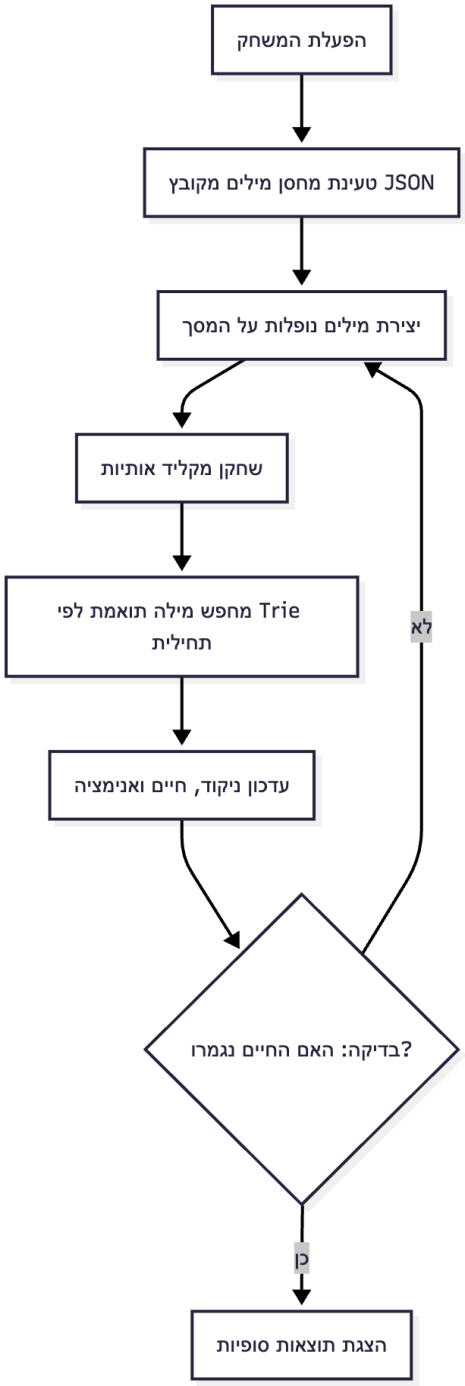
השילוב הזה מבטיח שהמשחק יישאר קל משקל, גמיש וניתן להרחבה — בין אם לשני שחקנים, רמות קושי חדשות או אפילו שינוי שפת המשחק.

3.1 דיאגרמת זרימה כללית

פרק זה מציג את אופן פעולת המערכת בשני המצבים: אופליין ואונליין. כל אחד מהם משתמש במנגנון דומה של טיפול בהקלדות, אך שונה באופן ניהול הנתונים. באופליין הכול מתבצע במחשב המשתמש, ואילו באונליין השרת מתאם בין שני שחקנים בזמן אמת.

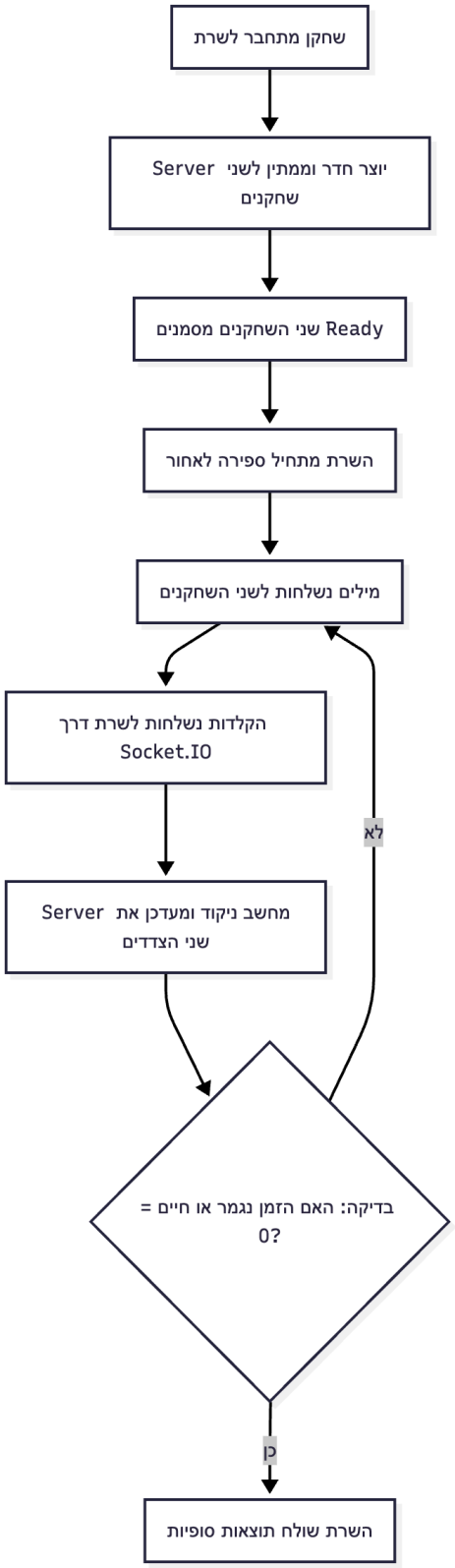
3.1.1 מצב אופליין

במצב האופליין המערכת פועלת באופן עצמאי על מכשיר המשתמש. המילים נטענות מקובץ JSON מקומי, ואלגוריתם מבוסס Trie אחראי על בחירת המילה הרלוונטית בהתאם לקלט שהוקלד. המשחק מסתיים כאשר כל החיים נגמרים.



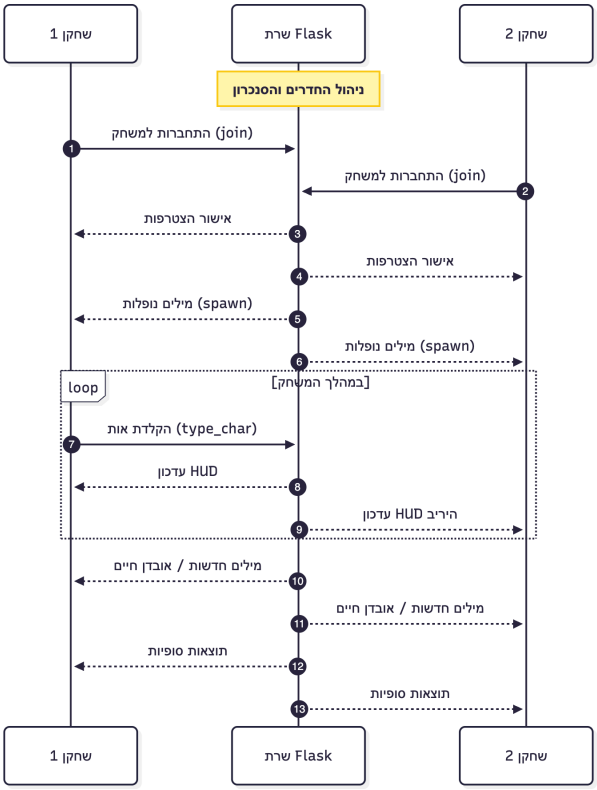
3.1.2 מצב אונליין

במצב אונליין, המשחק מבוסס על תקשורת דו-כיוונית בין השרת ללקוחות (השחקנים). השרת יוצר חדר חדש, ממתין לשני שחקנים, ושולח אליהם את אותן המילים. כל הקלדה נשלחת לשרת, אשר מחשב את הניקוד ומעדכן את שני הצדדים בזמן אמת.



3.2 דיאגרמת Sequence – Online Mode

הדיאגרמה הבאה מציגה את רצף הפעולות המתרחש במהלך משחק אונליין: החל מחיבור המשתמשים לשרת, דרך שליחת המילים בזמן אמת ועד הצגת התוצאות בסוף המשחק.



במשחק Space Typing Online, מצב האונליין כולל גם לוגיקה פנימית מורכבת וגם תקשורת רציפה בזמן אמת בין השחקנים לשרת. כדי להבין את מנגנון הפעולה במלואו, נדרש להבחין בין שני היבטים שונים של אותו תהליך. הראשון הוא תהליך הזרימה הלוגי של המשחק, כלומר סדר הפעולות והשלבים שהמערכת מבצעת – החל מההתחברות, דרך שליחת מילים ועד סיום המשחק. השני הוא תהליך התקשורת (Sequence), המתאר כיצד הנתונים וההודעות מועברים הלוך ושוב בין השרת לשני השחקנים בכל רגע נתון. הדיאגרמה הראשונה מציגה את הזרימה הכללית של המשחק, ואילו הדיאגרמה השנייה מדגישה את אופי התקשורת והסנכרון בזמן אמת. שילוב שתי הדיאגרמות מאפשר להבין את מבנה המערכת בצורה מלאה – גם מבחינת הפעולות הפנימיות וגם מבחינת האינטראקציה בין רכיביה.

3.3 מבנה Client–Server במשחק אונליין

במשחק Space Typing Online, צד הלקוח הוא השחקן, וצד השרת הוא **המוח שמנהל את המשחק. כאשר שחקן מתחבר, השרת יוצר חדר, מחכה לשחקן נוסף, ושולח לשניהם מילים לשדה המשחק. כל הקלדה נשלחת לשרת, שמעבד את הנתונים, מחשב ניקוד, ומחזיר עדכון לשני הצדדים בזמן אמת דרך Socket.IO. כך שני השחקנים רואים בדיוק אותו מצב משחק בכל רגע. במצב אופליין, לעומת זאת, אין שרת. כאן המשחק פועל כולו מקומית: כל מילה נבדקת באמצעות עץ Trie, שמזהה במהירות את המילה לפי התחילית ומעדכן את הניקוד מיד. שני המצבים מבוססים על עיקרון דומה —תגובה מיידית למשתמש — אך ההבדל הוא במי שמבצע את החישוב: בשרת באונליין, ובמחשב השחקן באופליין.

פרק 4 – מבנה הנתונים Trie

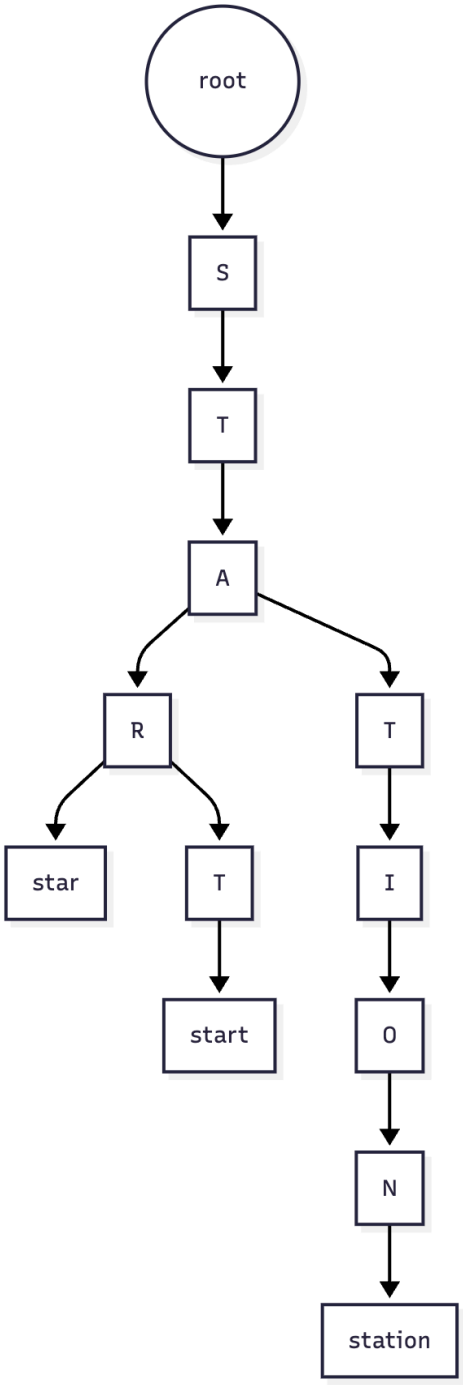
4.1 הסבר על עקרון פעולת Trie

במצב האופליין של המשחק **Space Typing Online**, נעשה שימוש במבנה הנתונים **Trie** כדי לזהות מילים בצורה מהירה ויעילה בזמן אמת. Trie הוא עץ מיוחד שבו כל צומת מייצג אות, והענפים יוצרים יחד מילים שלמות.

בכל פעם שהשחקן מקליד אות, המשחק עובר בעץ לפי האותיות שהוקלדו עד כה, וכך מסוגל לדעת מייד אילו מילים אפשריות מתחילות באותה התחילית. השיטה הזו מאפשרת למערכת לזהות התאמות עוד לפני שהשחקן השלים את כל המילה, מה שמעניק תחושת זרימה ותגובה מיידית.

לדוגמה, אם קיימות במאגר המילים **"start"**, **"star"** ו־**"station"**, והשחקן מקליד **"st"**, העץ מאפשר למצוא בבת אחת את כל המילים שמתחילות באותן שתי אותיות, מבלי לעבור על כל הרשימה. לכן Trie מתאים במיוחד למשחק הקלדה מהיר, שבו יש צורך בתגובה מידית לכל הקשה.

במשחק אונליין אין שימוש ב־**Trie**, מפני שכל הנתונים עוברים דרך השרת, והוספת Trie הייתה יוצרת בלבול בין השחקנים — כל אחד היה עשוי לקבל תוצאות שונות בזמן אמת. לכן Trie משמש אך ורק באופליין, שבו הכול מתבצע מקומית במחשב השחקן.



4.2 למה נבחר Trie

הבחירה במבנה Trie נבעה מצורך ב**חיפוש מהיר ומדויק** של מילים בזמן אמת. במבני נתונים רגילים כמו רשימות או מערכים, החיפוש דורש מעבר על כל האיברים, מה שגורם לעומס רב בזמן משחק. לעומת זאת, Trie מאפשר גישה ישירה לכל אות בנתיב וכך מאתר את המילה הרלוונטית באופן מיידי. בנוסף, הוא חוסך מקום בזיכרון בזכות שיתוף אותיות משותפות בין מילים בעלות קידומות דומות.

4.3 מימוש ב־Python

הקוד הבא מציג את המימוש של Trie בשפת Python כפי שיושם בפרויקט. כל צומת בעץ מייצג אות אחת, ולכל אות יש רשימה של המשכים אפשריים (children). עם סיום הוספת מילה, הצומת האחרון מסומן כסוף מילה מלאה.

```
class TrieNode:
    def __init__(self):
        self.children = {}            # רשימת הבנים – אותיות אפשריות הלאה
        self.is_end_of_word = False   # האם זו סוף של מילה שלמה
        self.word_text = None         # המילה עצמה (אם זו סוף מילה)
        self.prefix_count = 0         # כמה מילים מתחילות מכאן
```

```
class Trie:
    def __init__(self):
        self.root = TrieNode()  # נקודת ההתחלה של כל המילים – השורש

    def insert(self, word):
        """הוספת מילה לעץ – עובר אות־אות ובונה מסלול"""
        node = self.root
        for ch in word.lower():
            # אם האות לא קיימת – צור צומת חדש
            if ch not in node.children:
                node.children[ch] = TrieNode()
            node = node.children[ch]
            node.prefix_count += 1  # עדכון כמה מילים מתחילות באות זו
        # סימון סוף מילה
        node.is_end_of_word = True
        node.word_text = word

    def find_best_match(self, prefix):
        """מוצא את המילה הקרובה ביותר לקרקע שמתחילה עם הקידומת"""
        matches = self.find_words_starting_with(prefix)
        if not matches:
            return None
        # מיין לפי מיקום Y (גובה) לקרקע והחזר את הקרובה ביותר
        return sorted(matches, key=lambda w: w.y, reverse=True)[0]
```

הסבר:
הקוד ממש עץ Trie פשוט: כל צומת מייצג אות אחת, וכל מסלול יוצר מילה שלמה. בעת הוספת מילה, התוכנית עוברת אות־אות ובונה את המסלול המתאים. כך ניתן לזהות מילים לפי התחילית שלהן מבלי לבדוק את כל המאגר.

4.4 שימוש במשחק (Offline Mode)

במצב האופליין של המשחק, **Trie** משמש כרכיב מרכזי בזיהוי המילים בזמן אמת. כאשר מילים נופלות על המסך והשחקן מתחיל להקליד, כל אות שהוקלדה נבדקת מול **Trie** כדי לבדוק אם קיימת מילה שמתחילה באותן אותיות.

אם מתגלה התאמה, המשחק יודע שהשחקן נמצא "על המסלול הנכון", וברגע שהמילה מושלמת — הניקוד מתעדכן מיידית. **Trie** מאפשר לבצע את כל הבדיקות האלה ללא שרת ובלי לעבור על כל רשימת המילים בכל פעם.

בזכות המבנה ההיררכי שלו, כל חיפוש נעשה בזמן ליניארי ביחס לאורך המילה בלבד, מה שמבטיח ביצועים חלקים גם כשהמשחק כולל מאות מילים פעילות. בנוסף, **Trie** תורם גם לשימור תחושת הזרימה במשחק: אין השהיות, אין עיכובים, והתגובה להקלדה היא מיידית — דבר חיוני במשחק שמבוסס כולו על מהירות הקלדה.

לסיכום, השימוש ב־**Trie** מהווה נדבך חשוב במנגנון האופליין, ומדגים כיצד שילוב נכון של מבני נתונים יכול לשפר את חוויית המשתמש ואת ביצועי המשחק.

פרק 5 – ניהול המילים והקלדה

5.1 מנגנון בחירת מילים ("מחסן מילים")

במשחק קיים מנגנון טעינה חכם של מאגר מילים, הנקרא "מחסן מילים" (Word Bank). המחסן מכיל קובץ בשם wordcache_en.json`` שבו מאוחסנות עשרת אלפים המילים הנפוצות ביותר באנגלית. המילים מחולקות לפי אורך (3 עד 12 תווים), כך שניתן להתאים את רמת הקושי למהירות המשחק: מילים קצרות בשלבים קלים ומילים ארוכות בשלבים מתקדמים. הטעינה נעשית בתחילת המשחק, לפני הופעת המילים על המסך. כך ניתן לוודא שהמערכת תוכל לבחור בכל רגע מילה חדשה מבלי ליצור השהיה בזמן אמת

```
async function loadWords() {
    // טעינת מילים מקובץ JSON-המקומי
    const response = await fetch('/assets/wordcache_en.json');
    const data = await response.json();

    // אם המילים מאורגנות לפי "buckets" (אורך מילה)
```

```
    if (data.buckets) {
        for (const bucketWords of Object.values(data.buckets)) {
            WORDS.push(...bucketWords); // הוספה לרשימה הראשית
        }
    }
}
```

הסבר:
הפונקציה שולפת את המילים מקובץ ה־JSON ומוסיפה אותן למערך `WORDS`. השימוש ב־`buckets` מאפשר שליטה על חלוקת המילים לפי אורך או רמת קושי. כך המשחק יודע מראש אילו מילים זמינות, מה שמונע עיכובים בזמן אמת.

5.2 טיפול בהקלדה – צד שרת וצד לקוח

כאשר השחקן מקליד, כל אות נבדקת בזמן אמת. בצד הלקוח (הדפדפן), כל הקשה נשלחת מיד לשרת באמצעות **Socket.IO**, שמעבד את הנתון ומחזיר עדכון על מצב המילה והניקוד.

```
// שליחת כל הקשה לשרת בזמן אמת
window.addEventListener("keydown", (e) => {
    const ch = e.key.toLowerCase();
    if (ch.length === 1 && ch >= "a" && ch <= "z") {
        socket.emit("type_char", { ch });
    }
});
```

הסבר:
כל הקשה במקלדת נלכדת על ידי אירוע keydown. אם האות תקינה (A–Z), היא נשלחת לשרת באירוע type_char. כך נוצרת תקשורת רציפה בזמן אמת בין שני הצדדים.

בצד השרת, השרת מקבל את האות ובודק אם היא תואמת למילה הפעילה של השחקן:

```
def type_char(self, sid: str, ch: str) -> dict:
    p = self.players.get(sid)
    if not p or p.lives <= 0:
        return {"type": "noop"}

    ch = ch.lower()
    if len(ch) != 1 or ch not in string.ascii_lowercase:
        return {"type": "noop"}

    # בדיקה אם האות מתאימה למילה הנוכחית
    if p.current_word_id and p.current_word_id in self.words:
        w = self.words[p.current_word_id]
        if w.owner_sid == sid and w.remaining and w.remaining[0] == ch:
            w.typed += ch
            w.remaining = w.remaining[1:]
            p.score += SCORE_PER_CHAR
            return {"type": "progress", "word": w.to_public(self.players)}
```

הסבר:
השרת מזהה אם השחקן עדיין במשחק ואם יש לו מילה פעילה. במידה שהאות תואמת לאות הבאה במילה, הניקוד עולה והמילה מתקדמת. השרת מחזיר עדכון מידי לשני השחקנים כדי להציג את ההתקדמות בזמן אמת.

5.3 חיפוש מילה – משחק אופליין

במצב **אופליין**, אין שרת שמעבד את ההקלדה. במקום זאת, המשחק משתמש במבנה ה־**Trie** שהוצג בפרק הקודם כדי לזהות בזמן אמת את המילה שהשחקן מנסה להקליד.

```
function findTarget(ch) {
    const testPrefix = state.currentTypedPrefix + ch;
    const best = findBestMatch(testPrefix);
    if (best) {
        state.currentId = best.id;
        state.currentTypedPrefix = testPrefix;
        best.typed = testPrefix;
        best.remaining = best.text.slice(testPrefix.length);
        return best;
    }
    return null;
}
```

הסבר:
בכל הקלדה, המשחק מצרף את האות החדשה לתחילית הנוכחית (`prefix`) ומחפש ב־**Trie** את המילה המתאימה ביותר. אם נמצאה התאמה – היא "ננעלת" על המסך, והמשתמש ממשיך להקליד עד שהמילה מושלמת. כך **Trie** מאפשר מנגנון חיפוש מהיר גם בלי שרת, עם זמן תגובה מיידי.

5.4 מניעת כפילויות מילים

כדי לשמור על חוויית משחק מאתגרת, נדרש לוודא שאותה מילה לא תופיע פעמיים במהלך סשן אחד. לשם כך, המערכת שומרת רשימות של מילים שכבר הופיעו (`usedWords`) ושל מילים שהושלמו (`completedWords`). לפני יצירת מילה חדשה, מתבצעת בדיקה שמונעת כפילויות.

```
def _choose_unique_text(self):
    """בחר מילה חדשה שלא הופיעה כבר במשחק"""
    if not self.used_words:
        self.load_word_bank()
    for _ in range(100):
        word = random.choice(self.word_bank)
        if word not in self.used_words:
            return word
    self.used_words.clear()
    return random.choice(self.word_bank)
```

הסבר:
הפונקציה מוודאת שכל מילה חדשה היא ייחודית. אם כל המילים כבר נוצלו, הרשימה מתאפסת ומתחילים מחדש. כך נשמרת זרימה טבעית של מילים חדשות לאורך כל המשחק, גם לאחר זמן ממושך.

פרק 6 – בעיות וקשיים בפרויקט

אחד האתגרים המרכזיים שפגשתי במהלך הפיתוח היה יצירת **מחסן מילים אמין** שישמש את המשחק. בהתחלה ניסיתי להשתמש בספריית **NLTK** של פייתון, הכוללת מאגר גדול של מילים באנגלית. המטרה שלי הייתה להימנע ממחסן מילים ידני – לא רציתי ליצור רשימה בעצמי, אלא להשתמש במקור מוכן ואוטומטי שיחסוך זמן ויהיה מגוון.

עם זאת, לאחר בדיקה התברר ש־**NLTK** מכילה מילים רבות שאינן אמיתיות, כמו צירופי אותיות רנדומליים וחלקי מילים חסרי משמעות. ניסיתי לסנן את המילים בעזרת הצעה שקיבלתי מ־AI — לבחור רק מילים שמכילות לפחות אחת מהאותיות הקוליות **a, e, i, o, u** — אך גם פתרון זה לא הצליח לסנן את כל המילים הבעייתיות.

בסופו של דבר, בעזרת המרצה, עברתי להשתמש במאגר מבוסס **Google Word Bank**, קובץ JSON מסודר הכולל רק מילים תקניות ונפוצות. המעבר הזה פתר את הבעיה לחלוטין ושיפר את איכות המשחק.

הלקח המרכזי היה להבין שלא תמיד הספרייה הנוחה ביותר היא גם המתאימה ביותר, ולעיתים יש צורך לשלב ניסיון, חשיבה ביקורתית והכוונה מקצועית כדי להגיע לפתרון הנכון.

פרק 7 – שיפורים ועדכונים עתידיים

7.1 שיתוף קוד לוגיקה בין משחק האונליין והאופליין

נכון לעכשיו קיימות שתי גרסאות נפרדות של המשחק – אונליין ואופליין. עם זאת, שתי הגרסאות חולקות בסיס לוגי כמעט זהה הכולל ניהול מילים, ניקוד, חיים ומעקב אחר מצב המשחק. בגרסה עתידית המטרה היא **לאחד את הקוד** כך ששני המצבים יתבססו על אותה תשתית, כאשר רק מנגנון התקשורת ישתנה – במצב אונליין המשחק יעבוד מול שרת `Flask-SocketIO`, ובאופליין הקוד ירוץ כולו מקומית.

איחוד זה יפשט את התחזוקה, יבטל כפילויות בקוד, ויאפשר עדכון אחיד של פיצ'רים לשני המצבים בו־זמנית. המטרה היא להפוך את המשחק לגרסה אחת מאוחדת, שבה המשתמש בוחר את סוג המשחק מהתפריט, אך מאחורי הקלעים שני המצבים חולקים את אותה מערכת לוגית אחת.

7.2 ניהול רמות קושי באונליין

במצב האופליין כבר קיימות רמות קושי שונות, אך במצב האונליין הן טרם יושמו. בעתיד ניתן יהיה להרחיב את מנגנון הקושי גם לרב־משתתפים, כך שכל חדר יקבל הגדרות מותאמות אישית לפי רמת השחקנים. השרת יוכל לשלוט בפרמטרים כמו מהירות תנועת המילים (`word_speed`) ותדירות הופעתן (`spawn_every`), בהתאם לביצועי השחקנים.

גישה זו תאפשר בניית מנגנון **Adaptive Difficulty** – מערכת שמזהה בזמן אמת את רמת הביצועים של כל שחקן ומתאימה לו את רמת האתגר. כך המשחק ישמור על איזון, וימנע מצבים שבהם שחקן מנוסה מתקדם בקלות מדי לעומת שחקן חדש.

7.3 אפשרות לעצי Trie גם באונליין

מבנה הנתונים **Trie** שימש בהצלחה רבה במצב האופליין, שם הוא מאפשר חיפוש מילים מהיר מאוד לפי תחיליות ההקלדה. עם זאת, במצב אונליין לא שולב **Trie** בשלב זה, משום שהשימוש בו בכל לקוח בנפרד היה עלול ליצור **חוסר סנכרון** בין השחקנים.

כל לקוח היה עשוי לזהות מילה שונה כ"נכונה" עוד לפני שהשרת היה מאשר אותה, וכך היה נוצר פער בין המסכים של המשתתפים. בעתיד, ניתן לשלב **Trie** כך שהשרת יבצע את החיפוש בצורה מהירה ומדויקת יותר, בעוד הלקוחות ימשיכו לפעול על פי התשובה המרכזית שמגיעה מהשרת.

כך ניתן יהיה ליהנות מהיתרונות של **Trie** – מהירות, יעילות ואופטימיזציה – מבלי לפגוע באחידות חוויית המשחק.

7.4 פלקסיביליות (גמישות) של המערכת

7.4.1 שינוי מחסן מילים בקלות והתאמה לשפות נוספות

המערכת נבנתה כך שמאגר המילים נטען מקובץ חיצוני (`wordcache_en.json`) ולא משולב ישירות בקוד. שיטה זו מאפשרת להחליף את המאגר בקלות – למשל לעבור ממילים באנגלית למילים בעברית או לערבית – פשוט על ידי החלפת קובץ אחד, ללא שינוי לוגי או תכנותי.

המערכת תומכת גם בשפות שבהן הכיוון הוא מימין לשמאל, כך שניתן לשנות את שפת המשחק במלואה, כולל הממשק והקלדת המשתמשים. גישה זו מעניקה למערכת גמישות מרבית ומאפשרת התאמה לקהלים ולמדינות שונות ללא שינוי מבני.

7.4.2 הרחבה למספר משתמשים (יותר מ־2)

השרת בנוי במבנה "חדרים" (**Rooms**), כאשר כל חדר הוא משחק עצמאי שבו משתתפים שני שחקנים. בשל הארכיטקטורה הזו, ניתן להרחיב את המשחק לתמוך ביותר משני שחקנים על ידי שינוי מספר המשתתפים שמוקצים לכל חדר והוספת סנכרון תצוגתי לכל המשתתפים.

מאחר שהשרת כבר יודע לשדר הודעות לכל הלקוחות המחוברים לחדר, התוספת הזו תדרוש שינוי מינימלי בלבד בקוד – בעיקר בעדכון ממשק המשתמש ובניהול הניקוד. כך ניתן להפוך את המשחק מרב־משתתפים פשוט לתחרות קבוצתית מלאה, ללא צורך בשכתוב משמעותי של המערכת.

פרק 8 – סיכום ומסקנות

8.1 הישגי הפרויקט

במהלך פיתוח המשחק **Space Typing Online**, הושגו יעדים טכנולוגיים ופדגוגיים משמעותיים. נבנו בהצלחה שני מצבי משחק — אונליין ואופליין — הפועלים בצורה יציבה, תוך שימוש במבנה נתונים מסוג **Trie** לשיפור ביצועי האופליין ובמערכת **Socket.IO** לניהול התקשורת באונליין.

הפרויקט מדגים שילוב בין לוגיקה מתמטית, תכנות בצד לקוח ושרת, וחשיבה מערכתית שלמה. בנוסף, יושמו מנגנוני ניהול מילים, ניקוד, חיים ורמות קושי, תוך הקפדה על קוד קריא, גמיש וניתן להרחבה עתידית.

8.2 מסקנות כלליות ותרומה אישית

העבודה על הפרויקט אפשרה לי ליישם ידע שנרכש בקורסים שונים — מתכנות מונחה עצמים ועד תכנות שרתים וניהול נתונים. למדתי כיצד לתכנן מערכת מודולרית הניתנת לתחזוקה, איך להתמודד עם בעיות אמיתיות (כמו בחירת מאגר מילים אמין), ואיך ליצור חיבור נכון בין צד הלקוח והשרת.

הפרויקט גם חיזק את היכולת שלי לשלב חשיבה תכנונית עם יצירתיות, ולהפוך רעיון מופשט למשחק אינטראקטיבי אמיתי.

8.3 שיפורים ועדכונים עתידיים (סקירה כללית על ההתקדמות הצפויה)

הפרויקט הנוכחי מהווה בסיס יציב להמשך פיתוח והרחבה. בעתיד מתוכננת הרחבה לגרסה אחודה אחת של המשחק, שתכלול את מצבי האונליין והאופליין במערכת אחת גמישה, תוך שילוב מבנה הנתונים **Trie** בצד השרת לשיפור ביצועים והוספת מנגנון רמות קושי דינמיות שיתאימו את המשחק לרמת המשתמש.

בנוסף, תיבחן אפשרות להרחיב את התמיכה לשפות נוספות ולמספר רב יותר של שחקנים, ולשפר את חוויית המשתמש באמצעות עיצוב חזותי ואנימציות מתקדמות. החזון לעתיד הוא ליצור פלטפורמת משחק חכמה, אחידה ורב־לשונית שיכולה להמשיך להתפתח בקלות מבלי לשנות את מבנה הקוד הקיים.

8.4 קישורים לפרויקט

GitHub Repository:

https://github.com/kinan138/my_project_game

8.5 תודות

ברצוני להודות למרצה והמנחה שלי, **ד"ר הורוביץ מיכל**, על הליווי הצמוד, ההכוונה המקצועית והעזרה בבחירת פתרונות נכונים לאורך הדרך.