פרויקט מערכת משחק X-O (איקס-עיגול) מבוסס JWT ו-WebSockets

1.10.24
תומר שגיא

תיאור כללי

הפרויקט מתמקד ביצירת מערכת משחק X-O (איקס-עיגול) המאפשרת למשתמשים להתחבר, להירשם, לשחק נגד המחשב או נגד משתמשים אחרים בזמן אמת, ולצפות בתוצאות המשחקים הקודמים שלהם. הפרויקט יחולק לשני חלקים: ממשק משתמש (פרונטאנד) ושרת (בקאנד). המטרה היא לעבוד בשיתוף פעולה בין שני מפתחים, תוך יישום טכניקות של למידה עצמאית, עבודה משותפת על הקוד ושימוש נכון ב-GIT לניהול גרסאות.

חלוקת עבודה בין המפתחים

1. **מפתח ראשון**: אחראי על פיתוח ה-**Backend** (בקאנד) של המערכת.
2. **מפתח שני**: אחראי על פיתוח ה-**Frontend** (פרונטאנד) של המערכת.

שיתוף פעולה

- שני המפתחים יעבדו בשיתוף פעולה מלא, תוך שימוש ב-GIT לניהול גרסאות, ביצוע commit-ים תכופים, ופתרון קונפליקטים (merge conflicts) ביחד.
- יידרש תיאום שוטף בנוגע לממשקי API (כמו פורמט JSON, שדות, וטיפוסי נתונים) כדי להבטיח חיבור תקין בין ה-Backend וה-Frontend.
- במקרה של קונפליקטים בקוד (merge conflicts), המפתחים יפתרו אותם יחד תוך שיח על השינויים והבנה של הקוד.

פירוט התפקידים:

מפתח Backend:

האחריות של מפתח זה היא לפתח את השרת באמצעות **Node.js**, לנהל את הנתונים באמצעות **Express** ולטפל באימות משתמשים באמצעות **JWT**. בנוסף, המפתח יגדיר WebSockets לניהול המהלכים בזמן אמת במשחק.

משימות עיקריות למפתח ה-Backend:

1. **יצירת API לאימות משתמשים**:
   - יש ליצור מערכת רישום והתחברות עם JWT.
   - לאחר התחברות מוצלחת, יש להחזיר טוקן JWT המאוחסן בקובץ cookie מאובטח.
2. **ניהול API של משחקים**:

- קריאה להתחלת משחק חדש.
- קריאה להחזרת כל המשחקים שהמשתמש שיחק בהם בעבר (ניצחון/הפסד/תיקו).

3. **שימוש ב-WebSockets לניהול המהלכים בזמן אמת**:
   - ניהול מהלכי המשחק בזמן אמת בין המשתמש למחשב, או בין שני משתמשים דרך WebSockets.
   - השרת יקבע אם המשחק הסתיים ויחזיר את תוצאת המשחק דרך WebSocket. הלקוח לא יגדיר בעצמו אם המשחק הסתיים.
   - ניהול לוגיקת המשחק באופן מרכזי בשרת, כך שהשרת הוא זה שקובע אם המשחק נגמר ומחזיר את התוצאה הסופית ללקוח.
4. **אחסון נתונים**:
   - הנתונים (משתמשים ומשחקים) יישמרו בקבצי JSON.

חתימות API כלליות (יישום ישיר על ידי המפתח):

- **POST /auth/register** – רישום משתמש חדש.
- **POST /auth/login** – התחברות משתמש קיים.
- **GET /games** – החזרת כל המשחקים שהמשתמש שיחק בהם.
- **POST /games/start** – התחלת משחק חדש.
- **WebSocket /games/{gameId}/moves** – שליחת מהלכים בזמן אמת וקבלת תגובה עם תוצאת המשחק כשהוא מסתיים.

מפתח Frontend:

האחריות של מפתח זה היא לפתח את ממשק המשתמש של המשחק, הכולל רישום והתחברות, לוח המשחק עצמו ותצוגת המשחקים הקודמים. הפרונטאנד ייבנה ב-**TypeScript**, תוך שימוש ב-DOM לניהול ממשק המשתמש.

משימות עיקריות למפתח ה-Frontend:

1. **ממשק רישום והתחברות**:
   - יצירת טופס רישום והתחברות המשתמש באמצעות JWT.
   - שמירת ה-JWT ב-cookie מאובטח לצורך הזדהות במשחקים.
2. **ממשק ניהול משחקים**:
   - לוח משחק (3x3 X-O) המאפשר לשחקנים לבצע מהלכים בזמן אמת מול המחשב או מול משתמש אחר.
   - שימוש ב-WebSocket לניהול המהלכים בזמן אמת.
   - תצוגה של כל המשחקים שהמשתמש שיחק בהם בעבר עם סטטוס (ניצחון, הפסד, תיקו).
3. **שימוש ב-REST API**:
   - חיבור למערכת ה-API שפותחה על ידי מפתח ה-Backend לצורך ניהול משחקים, רישום והתחברות.

פרוטוקול WebSocket לניהול מהלכי המשחק:

1. **שליחת מהלך (Client → Server)**:
   - הלקוח ישלח הודעת WebSocket עם פרטי המהלך שנעשה על הלוח.

הודעת המהלך תכיל:

```
{
  "gameId": "12345",
  "player": "X",
  "position": [0, 1]
}
```

- `gameId`: מזהה המשחק הנוכחי.
- `player`: השחקן שעושה את המהלך (X או O).
- `position`: המיקום (x, y) על הלוח שבו נעשה המהלך.

2. **קבלת תגובה מהשרת (Server → Client)**:
   - השרת יחזיר תגובה לכל מהלך שהתקבל, כולל סטטוס אם המשחק הסתיים.
   - במידה והמשחק הסתיים, תוצאת המשחק תישלח בחזרה ללקוח.

תגובת WebSocket מהשרת יכולה להיראות כך:

```
{
  "gameId": "12345",
  "move": {
    "player": "X",
    "position": [0, 1]
  },
  "status": "ongoing"
}
```

או אם המשחק נגמר:

```
{
  "gameId": "12345",
  "move": {
    "player": "X",
    "position": [0, 1]
  },
  "status": "finished",
  "result": "X wins"
}
```

- `status`: יכול להיות `ongoing` (המשחק נמשך) או `finished` (המשחק הסתיים).
- `result`: במקרה שהמשחק הסתיים, השרת מחזיר את תוצאת המשחק (לדוגמה, "X wins"," "O wins" או "draw").

3. **עדכון בזמן אמת**:
   - כל מהלך שהתקבל ישודר לכל השחקנים המחוברים למשחק בזמן אמת.
   - לאחר סיום המשחק, התוצאה תישלח לכל השחקנים המחוברים.

עבודה משותפת ב-GIT:

1. המפתחים יעבדו באותו מאגר קוד (repository) ויבצעו commit-ים תכופים לצורך שמירת התקדמות העבודה.
2. עליהם לבצע שינויים בסניפים (branches) נפרדים, ולאחר מכן למזג (merge) אותם אל הסניף הראשי (main branch).
3. במידה ויתקיימו קונפליקטים במיזוג (merge conflicts), עליהם לפתור אותם ביחד בצורה משותפת, תוך שמירה על תקשורת פתוחה והבנת השינויים שבוצעו.
4. כל שלב משמעותי בפרויקט ילווה בהודעה ברורה ב-commit log, כך שניתן יהיה לעקוב אחרי כל שלב בעבודה.

סיכום: הפרויקט יאפשר למפתחים לעבוד בשיתוף פעולה על מערכת משחק X-O עם דגש על עבודה עם JWT, WebSockets, REST API ושימוש בטכניקות עבודה משותפת ב-GIT. העבודה המשותפת תכלול למידה עצמאית של טכנולוגיות חדשות, פתרון בעיות בזמן אמת ושיתוף פעולה במיזוג שינויים.

בונוס: אפשרות לאתגר משתמשים אחרים או לצפות במשחקים

במסגרת הפרויקט, תהיה אפשרות למשתמשים לאתגר אחד את השני למשחקים בזמן אמת, וכן לצפות במשחקים חיים. הנספח הזה מתאר כיצד הפונקציונליות הזו תתבצע:

1. אתגר משתמש אחר למשחק

- כל משתמש יוכל לראות רשימת משתמשים מחוברים.
- הוא יוכל לשלוח הזמנה לשחק מול משתמש אחר באמצעות WebSocket.

הפרוטוקול לאתגר משתמש:

```
{
  "action": "challenge",
  "challenger": "user1",
  "opponent": "user2"
}
```

אם המשתמש השני מאשר את האתגר, תתחיל משחק חדש (לקרוא לאותה לוגיקה של התחלת משחק):

```
{
  "action": "challengeAccepted",
  "gameId": "67890",
  "players": ["user1", "user2"]
}
```

2. צפייה במשחק של משתמשים אחרים

- כל משתמש יוכל לבחור לצפות במשחק אחר שמתקיים בזמן אמת.
- הוא יתחבר ל-WebSocket של אותו משחק ויקבל עדכונים על כל מהלך בזמן אמת.

הפרוטוקול לצפייה במשחק:

```
{
  "action": "watch",
  "gameId": "67890"
}
```

- כל מהלך שהמשתמשים במשחק יבצעו, יישלח לצופים